

THE DL_POLY_2 REFERENCE MANUAL

T.R. Forester and W. Smith,
CCLRC, Daresbury Laboratory,
Daresbury,
Warrington WA4 4AD,
England

Version 2.13 July 2001

ABOUT DL_POLY

DL_POLY is a parallel molecular dynamics simulation package developed at Daresbury Laboratory by W. Smith and T.R. Forester under the auspices of the Engineering and Physical Sciences Research Council (EPSRC) for the EPSRC's Collaborative Computational Project for the Computer Simulation of Condensed Phases (CCP5) and the Advanced Research Computing Group (ARCG) at Daresbury Laboratory. The package is the property of Daresbury Laboratory.

DL_POLY is issued free **under licence** to academic institutions pursuing scientific research of a non-commercial nature. Commercial organisations may be permitted a licence to use the package after negotiation with the owners. Daresbury Laboratory is the sole centre for distribution of the package. Under no account is it to be redistributed to third parties without consent of the owners.

The purpose of the DL_POLY package is to provide software for academic research that is inexpensive, accessible and free of commercial considerations. Users have direct access to source code for modification and inspection. In the spirit of the enterprise, contributions in the form of working code are welcome, provided the code is compatible with DL_POLY in regard to its interfaces and programming style and it is adequately documented.

DISCLAIMER

Neither the CCLRC, EPSRC, CCP5 nor any of the authors of the DL_POLY_2 package or its derivatives guarantee that the package is free from error. Neither do they accept responsibility for any loss or damage that results from its use.

ACKNOWLEDGEMENTS

DL_POLY_2 was developed under the auspices of the Central Laboratory of the Research Councils, the Engineering and Physical Sciences Research Council, and the former Science and Engineering Research Council, under grants from the Computational Science Initiative and the Science and Materials Computing Committee. The package is the property of the Council for the Central Laboratory of the Research Councils of the United Kingdom.

Advice, assistance and encouragement in the development of DL_POLY_2 has been given by many people. We gratefully acknowledge the following:

D. Tildesley, M.J. Gillan, J. Goodfellow, D. Fincham, M. Rodger, W.C. Mackrodt, J.H.R. Clarke, D. Brown, S. Price, P.J. Durham, P. Sherwood, G.D. Price, S.C. Potter, S. Melchionna, F. Müller-Plathe, G. Ciccotti, M.W. Smith, A. Simpson, J. Geronowicz, the HPCI Materials Consortium, the Edinburgh Parallel Computing Centre, the HPCI Centre at Southampton and the CCP5 community. We also thank users of previous versions of DL_POLY who have passed on helpful comments. P.M. Rodger and A. Smondyrev have been particularly helpful in this respect.

Manual Notation

In the DL_POLY Manual and Reference Manual specific fonts are used to convey specific meanings:

1. *directories* - indicates unix file directories
2. ROUTINES - indicates subroutines, functions and programs.
3. *macros* - indicates a macro (file of unix commands)
4. **directive** - indicates directives or keywords
5. **variables** - indicates named variables and parameters
6. FILE - indicates filenames.

Contents

1	Introduction	15
1.1	The DL_POLY Package	17
1.2	Functionality	18
1.2.1	Molecular Systems	18
1.2.2	Force Field	18
1.2.3	Boundary Conditions	19
1.2.4	Java Graphical User Interface	19
1.2.5	Algorithms	20
1.3	Programming Style	21
1.3.1	Programming Language	21
1.3.2	Memory Management	21
1.3.3	Target Computers	21
1.3.4	Version Control System (CVS)	22
1.3.5	Required Program Libraries	22
1.3.6	Internal Data Transfer	22
1.3.7	Internal Documentation	23
1.3.8	Subroutine/Function Calling Sequences	23
1.3.9	FORTTRAN Parameters	23
1.3.10	Arithmetic Precision	23
1.3.11	Units	24
1.3.12	Error Messages	24
1.4	The DL_POLY_2 Directory Structure	25
1.4.1	The <i>source</i> Sub-directory	25
1.4.2	The <i>utility</i> Sub-directory	25
1.4.3	The <i>data</i> Sub-directory	25
1.4.4	The <i>bench</i> Sub-directory	25
1.4.5	The <i>execute</i> Sub-directory	26
1.4.6	The <i>build</i> Sub-directory	26
1.4.7	The <i>public</i> Sub-directory	26
1.4.8	The <i>respa</i> Sub-directory	26
1.4.9	The <i>sdk</i> Sub-directory	26
1.4.10	The <i>java</i> Sub-directory	27
1.5	Obtaining the Source Code	27

1.6	DLPROTEIN	28
1.7	Other Information	28
2	DL_POLY_2 Force Fields and Algorithms	30
2.1	The DL_POLY_2 Force Field	32
2.2	The Intramolecular Potential Functions	34
2.2.1	Bond Potentials	34
2.2.2	Distance Restraints	36
2.2.3	Valence Angle Potentials	36
2.2.4	Angular Restraints	38
2.2.5	Dihedral Angle Potentials	39
2.2.6	Improper Dihedral Angle Potentials	42
2.2.7	Inversion Angle Potentials	42
2.2.8	Tethering Forces	45
2.2.9	Frozen Atoms	46
2.3	The Intermolecular Potential Functions	46
2.3.1	Short Ranged (van der Waals) Potentials	46
2.3.2	Metal Potentials	49
2.3.3	Three Body Potentials	50
2.3.4	Four Body Potentials	51
2.3.5	External Fields	52
2.4	Long Ranged Electrostatic (Coulombic) Potentials	53
2.4.1	Atomistic and Charge Group Implementation	54
2.4.2	Direct Coulomb Sum	54
2.4.3	Truncated and Shifted Coulomb Sum	55
2.4.4	Coulomb Sum with Distance Dependent Dielectric	56
2.4.5	Ewald Sum	57
2.4.6	Smooth Particle Mesh Ewald	59
2.4.7	Hautman Klein Ewald (HKE)	61
2.4.8	Reaction Field	63
2.4.9	Dynamical Shell Model	64
2.5	Integration algorithms	66
2.5.1	Bond Constraints	67
2.5.2	Potential of Mean Force (PMF) Constraints and the Evaluation of Free Energy	69
2.5.3	Thermostats	70
2.5.4	Gaussian Constraints	71
2.5.5	Barostats	72
2.5.6	Rigid Bodies and Rotational Integration Algorithms	75
2.5.7	The DL_POLY_2 Multiple Timestep Algorithm	80
2.5.8	The DL_POLY_2 RESPA Multiple Timestep Implementation	81
2.6	DL_POLY Parallelisation	82
2.6.1	The Replicated Data Strategy	82
2.6.2	Distributing the Intramolecular Bonded Terms	83

2.6.3	Distributing the Nonbonded Terms	83
2.6.4	Modifications for the Ewald Sum	85
2.6.5	Three Body Forces	85
2.6.6	Metal Potentials	85
2.6.7	Summing the Atomic Forces	86
2.6.8	The RD-SHAKE and Parallel QSHAKE Algorithms	86
3	DL_POLY_2 Construction and Execution	88
3.1	Constructing DL_POLY_2 : an Overview	90
3.1.1	Constructing the Standard Version	90
3.1.2	Constructing Nonstandard Versions	91
3.2	Compiling and Running DL_POLY_2	95
3.2.1	Compiling the Source Code	95
3.2.2	Compiling Older Versions with the Utility Program PARSET	99
3.2.3	Running DL_POLY_2	100
3.2.4	Restarting DL_POLY_2	101
3.3	A Guide to Preparing Input Files	101
3.3.1	Inorganic Materials	102
3.3.2	Macromolecules	102
3.3.3	Adding Solvent to a Structure	103
3.3.4	Analysing Results	103
3.3.5	Choosing Ewald Sum Variables	104
3.4	DL_POLY_2 Error Processing	107
3.4.1	The DL_POLY_2 Internal Error Facility	107
4	DL_POLY_2 Data Files	108
4.1	The INPUT files	110
4.1.1	The CONTROL File	110
4.1.2	The CONFIG File	119
4.1.3	The FIELD File	122
4.1.4	The REVOLD File	137
4.1.5	The TABLE File	138
4.2	The OUTPUT Files	140
4.2.1	The HISTORY File	140
4.2.2	The OUTPUT File	143
4.2.3	The REVCON File	146
4.2.4	The REVIVE File	146
4.2.5	The RDFDAT File	146
4.2.6	The ZDNDAT File	147
4.2.7	The STATIS File	147

5	The DL_POLY_2 Java Graphical User Interface	150
5.1	The DL_POLY_2 Java Graphical User Interface	152
5.2	Compiling the Java GUI	152
5.3	Starting the Java GUI	153
5.4	The Graphics Window	155
5.5	The Monitor Window	156
5.6	The GUI Application Menus	156
5.7	File Menu	156
5.7.1	Quit	157
5.7.2	View File	157
5.7.3	Delete File	157
5.7.4	Defaults	158
5.7.5	Reset	159
5.8	FileMaker Menu	159
5.8.1	CONTROL	159
5.8.2	CONFIG	160
5.8.3	FIELD	164
5.8.4	Display	169
5.8.5	Tools	169
5.9	Execute Menu	172
5.9.1	Run DL_POLY	172
5.9.2	Store/Fetch Data	174
5.10	Analysis Menu	175
5.10.1	Structure	175
5.10.2	Dynamics	179
5.10.3	van Hove	181
5.10.4	Display	184
5.10.5	Tools	185
5.11	Information Menu	186
5.12	The GUI Graph Plotter Window	186
6	DL_POLY_2 Examples	189
6.1	DL_POLY Examples	191
6.1.1	Test Cases	191
6.1.2	Benchmark Cases	193
7	DL_POLY_2 Utilities	195
7.1	Initialisation Utilities	197
7.1.1	BUCKY	197
7.1.2	CSD2EDT	199
7.1.3	EIGHTFOLD	201
7.1.4	FFGEN	203
7.1.5	FILLEDIT	210
7.1.6	GENLAT	212

7.1.7	GENLAT.TO	214
7.1.8	HADD1_1	216
7.1.9	HADD1_2	217
7.1.10	HADD1_3	218
7.1.11	HADD2_1	219
7.1.12	HADD2_2	221
7.1.13	HADD3_1	222
7.1.14	HADD4_0	224
7.1.15	NCONNECT	225
7.1.16	NFOLD	226
7.1.17	PDB2EDT	228
7.1.18	POLYGROW	231
7.1.19	POMEDT	234
7.1.20	PROSEQ	235
7.1.21	RESIZE	236
7.1.22	SOLVADD	238
7.1.23	WATERADD	240
7.2	Analysis Utilities	242
7.2.1	ACF3D	242
7.2.2	CORREL1	244
7.2.3	CORREL2	246
7.2.4	CURDEN	248
7.2.5	DENCOR	254
7.2.6	DIPOLE	259
7.2.7	GDIFF	261
7.2.8	GSELF	264
7.2.9	MSDBLK	267
7.2.10	MSDPRG	270
7.2.11	SLOWMO	273
7.2.12	SMOOTH	276
7.2.13	SPECTRUM	278
7.2.14	STATBLOCK	280
7.2.15	VACF	282
7.3	Miscellaneous Utilities	285
7.3.1	BESTFIT	285
7.3.2	CFGCHK	287
7.3.3	CFGCMP	289
7.3.4	DLPXYZ	291
7.3.5	FFT	293
7.3.6	GEARPC	294
7.3.7	HISXYZ	296
7.3.8	JACOBI	298
7.3.9	Useful Macros	300
7.3.10	PARSET	303

7.3.11	ROTATE	305
7.3.12	SOLVE	306
7.3.13	SORT	307
7.3.14	SYMINV	308
7.3.15	TABCHK	310
8	DL_POLY_2 Subroutines and Functions	312
8.1	Subroutine and Function Specifications	314
8.1.1	DL_PARAMS.INC: The DL_POLY Parameters (Include) File	314
8.1.2	ANGFRC	318
8.1.3	BNDFRC	321
8.1.4	BSPCOE	324
8.1.5	BSPGEN	326
8.1.6	CFGSCAN	328
8.1.7	CONSCAN	330
8.1.8	CORSHL	332
8.1.9	COUL0	333
8.1.10	COUL0_SW	335
8.1.11	COUL0NEU	337
8.1.12	COUL1	339
8.1.13	COUL2	341
8.1.14	COUL2_SW	343
8.1.15	COUL2NEU	345
8.1.16	COUL3	347
8.1.17	COUL3_SW	349
8.1.18	COUL3NEU	351
8.1.19	COUL4	353
8.1.20	CPY_RTC	355
8.1.21	CRECV	356
8.1.22	CSEND	357
8.1.23	DBLSTR	358
8.1.24	DCELL	359
8.1.25	DENLOC	360
8.1.26	DIFFSN0	362
8.1.27	DIFFSN1	364
8.1.28	DIHFRC	366
8.1.29	DLPFFT3	369
8.1.30	DLPOLY	371
8.1.31	DLPOLY_RESPA	374
8.1.32	DUNI	376
8.1.33	ERFCGEN	377
8.1.34	ERROR	379
8.1.35	EWALD1	380
8.1.36	EWALD2	383

8.1.37	EWALD2_SW	385
8.1.38	EWALD2NEU	387
8.1.39	EWALD3	389
8.1.40	EWALD4	391
8.1.41	EWALD_SPME	394
8.1.42	EWCORR	397
8.1.43	EWCORR_NEU	399
8.1.44	EXCLUDE	401
8.1.45	EXCLUDE_ATOM	404
8.1.46	EXCLUDE_LINK	406
8.1.47	EXCLUDE_RESPA	408
8.1.48	EXCLUDENEU	410
8.1.49	EXITCOMMS	412
8.1.50	EXTNFLD	413
8.1.51	FBPFRF	415
8.1.52	FCAP	417
8.1.53	FLDSCAN	419
8.1.54	FORCES	421
8.1.55	FORCESNEU	425
8.1.56	FORCESNEU_RESPA	428
8.1.57	FORGEN	431
8.1.58	FORTAB	433
8.1.59	FREEZE	435
8.1.60	GAUSS	436
8.1.61	GDSUM	438
8.1.62	GETREC	439
8.1.63	GIMAX	440
8.1.64	GISUM	441
8.1.65	GSTATE	442
8.1.66	GSYNC	443
8.1.67	HALF_STEP	444
8.1.68	HKEWALD1	450
8.1.69	HKEWALD2	453
8.1.70	HKEWALD3	455
8.1.71	HKEWALD4	457
8.1.72	HKGEN	459
8.1.73	IMAGES	461
8.1.74	INITCOMMS	463
8.1.75	INTLIST	464
8.1.76	INTSTR	467
8.1.77	INVERT	468
8.1.78	JACOBI	469
8.1.79	KINSTR	471
8.1.80	LOWCASE	473

8.1.81	LRCMETAL	474
8.1.82	LRCORRECT	476
8.1.83	MACHINE	478
8.1.84	MERGE	479
8.1.85	MERGE1	481
8.1.86	MERGE4	483
8.1.87	METGEN	485
8.1.88	MULTIPLE	486
8.1.89	MULTIPLE_NSQ	491
8.1.90	MULTIPLENEU	494
8.1.91	MYNODE	498
8.1.92	NEUTLST	499
8.1.93	NEUTLST_RESPA	501
8.1.94	NODEDIM	503
8.1.95	NPT_B0	504
8.1.96	NPT_B1	507
8.1.97	NPT_B3	511
8.1.98	NPT_H0	515
8.1.99	NPT_H1	518
8.1.100	NPT_H3	522
8.1.101	NPTQ_B1	526
8.1.102	NPTQ_B2	531
8.1.103	NPTQ_B3	536
8.1.104	NPTQ_B4	541
8.1.105	NPTQ_H1	546
8.1.106	NPTQ_H2	551
8.1.107	NPTQ_H3	556
8.1.108	NPTQ_H4	561
8.1.109	NST_B0	566
8.1.110	NST_H0	569
8.1.111	NUMNODES	572
8.1.112	NVE_0	573
8.1.113	NVE_1	575
8.1.114	NVT_B0	578
8.1.115	NVEQ_1	580
8.1.116	NVEQ_2	584
8.1.117	NVT_B0	588
8.1.118	NVT_B1	590
8.1.119	NVT_E0	593
8.1.120	NVT_E1	595
8.1.121	NVT_H0	598
8.1.122	NVT_H1	600
8.1.123	NVT_H1	603
8.1.124	NVTQ_B1	606

8.1.125 NVTQ_B2	611
8.1.126 NVTQ_H1	616
8.1.127 NVTQ_H2	621
8.1.128 PAIR_FORCES_SW	626
8.1.129 PARLINK	629
8.1.130 PARLINKNEU	631
8.1.131 PARLST	633
8.1.132 PARLST_NSQ	635
8.1.133 PARLST_RESPA	637
8.1.134 PARNEULST	639
8.1.135 PARNEULST_RESPA	641
8.1.136 PARSET	644
8.1.137 PASSCON	645
8.1.138 PASSPMF	647
8.1.139 PASSQUAT	649
8.1.140 PMF_1	651
8.1.141 PMF_SHAKE	654
8.1.142 PRIMLST	656
8.1.143 PRNEULST	658
8.1.144 QSHAKE	660
8.1.145 QUATBOOK	663
8.1.146 QUATQNCH	666
8.1.147 QUENCH	668
8.1.148 RDF0	670
8.1.149 RDF0_SW	672
8.1.150 RDF0NEU	674
8.1.151 RDF1	676
8.1.152 RDSHAKE_1	678
8.1.153 RESPA_R	681
8.1.154 RESPA_V	684
8.1.155 RESULT	686
8.1.156 REVIVE	689
8.1.157 SCDENS	691
8.1.158 SCL_CSUM	693
8.1.159 SET_BLOCK	694
8.1.160 SHELLSORT	695
8.1.161 SHLFRC	696
8.1.162 SHLMERGE	698
8.1.163 SHLQNCH	700
8.1.164 SHMOVE	702
8.1.165 SIMDEF	704
8.1.166 SIMDEF_RESPA	707
8.1.167 SPL_CEXP	710
8.1.168 SPLICE	711

8.1.169 SRFRCE	713
8.1.170 SRFRCE_SW	715
8.1.171 SRFRCE_NEU	717
8.1.172 STATIC	719
8.1.173 STRIP	722
8.1.174 STRUCOPT	723
8.1.175 SUTTCHEM	726
8.1.176 SYMINV	728
8.1.177 SYSBOOK	730
8.1.178 SYSDEF	734
8.1.179 SYSDEF_RESPA	738
8.1.180 SYSGEN	741
8.1.181 SYSGEN	747
8.1.182 SYSGEN_RESPA	749
8.1.183 SYSINIT	754
8.1.184 SYSTEMP	757
8.1.185 TETHFRC	760
8.1.186 THBFRC	762
8.1.187 TIMCHK	764
8.1.188 TRAJECT	765
8.1.189 VERTEST	767
8.1.190 VERTEST1	769
8.1.191 VSCALE	771
8.1.192 VSCALEG	773
8.1.193 WARNING	776
8.1.194 XSCALE	777
8.1.195 ZDEN0	779
8.1.196 ZDEN1	781
8.1.197 Communications Routines	783
A The DL_POLY_2 Makefile	795
B Periodic Boundary Conditions in DL_POLY	806
C DL_POLY Error Messages and User Action	812
D DL_POLY Arrays	855
E DL_POLY Subroutine Dependencies	864

Chapter 1

Introduction

Scope of Chapter

This chapter describes the concept, design and directory structure of DL_POLY_2 and how to obtain a copy of the source code.

1.1 The DL_POLY Package

DL_POLY_2 (DL_POLY [1]) is a package of subroutines, programs and data files, designed to facilitate molecular dynamics simulations of macromolecules, polymers, ionic systems and solutions on a distributed memory parallel computer. The first version was written on behalf of CCP5 [2] in the eighteen months preceding September 1993 when the (beta) release appeared. The principal authors at that stage were Bill Smith and Tim Forester at Daresbury Laboratory. For want of a more imaginative acronym the package was named DL_POLY. The entire DL_POLY project was given a major boost by the provision of a grant in January 1993 from the SERC's Computational Science Initiative. An additional grant was made by the EPSRC in June 1994 to continue developments in the period 1995-97. In April 1995 the project came under the Council for the Central Laboratory of the Research Councils.

There were many reasons for writing the package. The first reason being to exploit parallel computing for molecular simulation in the UK. There were several worthy packages available to the academic community including GROMOS [3], AMBER [4] and X-PLOR [5], but none of these were designed (as far as we were able to tell) with parallel computers in mind, (though parallel versions of these programs now exist). We decided then that the effort involved in devising a new code which had parallelism built into it from the start would be well spent. This we thought would allow us to adapt the code to new architectures with greater ease. The second reason was to bring to fruition the high level of parallel programming expertise at Daresbury Laboratory, and generally within the CCP5 community, in the area of molecular dynamics. We therefore thought it worthwhile to tap into this expertise to produce a new package. The third reason was that we thought it would be valuable to produce a package that was entirely free from commercial constraints, by which we mean the source code would be available to academic institutions for adaptation and extension. There would be no restriction preventing users from examining (and verifying!) the source code. DL_POLY was not to be a "black box".

Of course, in a rapidly developing subject area, we cannot guarantee we have everything a potential user may need. However since this is very much a project endorsed by CCP5, which serves the UK (and World-Wide) academic community, we are also hopeful that our users will be inclined to contribute any extensions of the package they devise and so accelerate its development. As it stands, the package has some nice features we believe. While it is designed for distributed memory parallel machines, we have taken care to ensure that it can, with minimum modification, be run on the popular workstations. Scaling up a simulation from a small workstation to a massively parallel machine remains straightforward.

Users are reminded that we are interested in hearing what other features could be usefully incorporated. We obviously have ideas of our own and CCP5 strongly influences developments, but other input would be welcome nevertheless. We also request that our users respect the integrity of DL_POLY_2 source and not pass it on to third parties. We require that all users of the package register with us, not least because we need to keep everyone abreast of new developments and discovered bugs. We have developed various forms of licence, which we hope will ward off litigation (from both sides), without denying access to genuine scientific users.

In the next section we outline the capabilities of DL_POLY_2 as briefly as possible. This is followed by a description of the DL_POLY_2 directory structure and how to obtain the source code from Daresbury Laboratory. Much more information is to be found later in this manual. Access to all this information and more can be gained via our WWW page: http://www.dl.ac.uk/TCSC/Software/DL_POLY/main.html

1.2 Functionality

The following is a list of the features DL_POLY_2 . It is worth reminding users that DL_POLY_2 represents a *package* rather than a single program, so users should consider piecing together their own program with the desired functionality. We will however, supply a consolidated program in the distributed source.

1.2.1 Molecular Systems

DL_POLY_2 will simulate the following molecular species:

1. Simple atomic systems and mixtures e.g. Ne, Ar, Kr, etc.
2. Simple unpolarisable point ions e.g. NaCl, KCl, etc.
3. Polarisable point ions and molecules e.g. MgO, H₂O etc.D
4. Simple rigid molecules e.g. CCl₄, SF₆, Benzene, etc.
5. Rigid molecular ions with point charges e.g. KNO₃, (NH₄)₂SO₄, etc.
6. Polymers with rigid bonds e.g. C_nH_{2n+2}
7. Polymers with rigid bonds and point charges e.g. proteins
8. Macromolecules and biological systems
9. Molecules with flexible bonds
10. Silicate glasses and zeolites
11. Simple metals e.g. Al, Ni, Cu etc.

1.2.2 Force Field

The DL_POLY_2 force field includes the following features:

1. All common forms of non-bonded atom-atom potential;
2. Atom-atom (site-site) Coulombic potentials;
3. Valence angle potentials;

4. Dihedral angle potentials;
5. Inversion potentials;
6. Improper dihedral angle potentials;
7. 3-body valence angle and hydrogen bond potentials;
8. 4-body inversion potentials;
9. Sutton-Chen density dependent potentials (for metals) [6].

The parameters describing these potentials may be obtained, for example, from the GROMOS [3], Dreiding [7] or AMBER [4] forcefield, which share functional forms. It is relatively easy to adapt DL_POLY_2 to user specific force fields.

1.2.3 Boundary Conditions

DL_POLY_2 will accommodate the following boundary conditions:

1. None e.g. isolated polymer in space.
2. Cubic periodic boundaries.
3. Orthorhombic periodic boundaries.
4. Parallelepiped periodic boundaries.
5. Truncated octahedral periodic boundaries.
6. Rhombic dodecahedral periodic boundaries.
7. Slab (x,y periodic, z nonperiodic).
8. Hexagonal prism periodic boundaries.

These are describe in detail in Appendix B.

1.2.4 Java Graphical User Interface

DL_POLY_2 version 2.13 introduced a new Graphical User Interface (GUI) for the package, written in the Java programming language from Sun. This incorporated much of the functionality of the earlier DL_POLY_2 GUI which was based on the commercial software of Molecular Simulations Inc.(MSI - now part of Accelrys) The main advantage of this was the free availability of the Java programming environment. Also in its favour was its suitability for building graphical user interfaces and the portability of the compiled GUI, which may be run without recompiling on any Java supported machine. This is now an integral component of the DL_POLY_2 package and is available on exactly the same terms. (See section 5.1.)

1.2.5 Algorithms

1.2.5.1 Parallel Algorithms

DL_POLY_2 exclusively employs the **Replicated Data** parallelisation strategy [8, 9] (see section 2.6.1).

1.2.5.2 Molecular Dynamics Algorithms

The DL_POLY_2 MD algorithms are all couched in the form of the Verlet Leapfrog integration algorithm [10]. NVE, NVT, NPT and $N\sigma$ T ensembles are available, with a selection of thermostats and barostats. A parallel version of the SHAKE algorithm [11] called RD-SHAKE is used for bond constraints [9]. Fincham's implicit quaternion algorithm (FIQA) [12] is available for rigid molecular species. Rigid molecular species linked by rigid bonds are handled with an algorithm of our own devising, called the QSHAKE algorithm [13].

The following MD algorithms are available:

1. Verlet leapfrog;
2. Verlet leapfrog with RD-SHAKE;
3. Rigid molecules with FIQA and RD-SHAKE;
4. Linked rigid molecules with QSHAKE [13];
5. Berendsen constant T algorithm with Verlet or RD-SHAKE [14];
6. Evans constant T algorithm with Verlet or RD-SHAKE [15];
7. Hoover constant T algorithm with Verlet or RD-SHAKE [16];
8. Berendsen constant T algorithm with FIQA and RD-SHAKE [14];
9. Berendsen constant T algorithm with QSHAKE [14];
10. Hoover constant T algorithm with FIQA and RD-SHAKE [16];
11. Hoover constant T algorithm with QSHAKE [16];
12. Berendsen constant T,P algorithm with RD-SHAKE [14];
13. Berendsen constant T,σ algorithm with RD-SHAKE [14];
14. Hoover constant T,P algorithm with RD-SHAKE [16];
15. Hoover constant T,σ algorithm with RD-SHAKE [16];
16. Berendsen constant T,P algorithm with FIQA and RD-SHAKE [14];
17. Berendsen constant T,P algorithm with QSHAKE [14];

18. Berendsen constant $T, \underline{\sigma}$ algorithm with FIQA and RD-SHAKE [14];
19. Berendsen constant $T, \underline{\sigma}$ algorithm with QSHAKE [14];
20. Hoover constant T, P algorithm with FIQA and RD-SHAKE [16];
21. Hoover constant T, P algorithm with QSHAKE [16];
22. Hoover constant $T, \underline{\sigma}$ algorithm with FIQA and RD-SHAKE [16];
23. Hoover constant $T, \underline{\sigma}$ algorithm with QSHAKE [16];

A variant of DL_POLY_2 that handles the time reversible, multiple timestep RESPA algorithm [17, 18] is also available for atomic systems and rigid ion systems [19], but it is not applicable to systems which have rigid body molecules or constraints.

1.3 Programming Style

The programming style of DL_POLY_2 is intended to be as uniform as possible. The following stylistic rules apply throughout. Potential contributors of code are requested to note the stylistic convention.

1.3.1 Programming Language

Versions of DL_POLY_2 prior to 2.11 are written exclusively in FORTRAN 77. Versions of DL_POLY_2 from 2.11 contain extensions written in FORTRAN 90.

1.3.2 Memory Management

Since version 2.11 of DL_POLY_2, the major array dimensions are calculated at the time of execution and the associated arrays created through the dynamic array allocation features of FORTRAN 90. In versions prior to 2.11, array dimensions are fixed at compilation time and defined by FORTRAN PARAMETER statements

1.3.3 Target Computers

DL_POLY_2 is targeted towards distributed memory parallel computers. However, versions of the program for serial computers are easily produced. To facilitate this all machine specific calls are located in dedicated FORTRAN routines, to permit substitution by appropriate alternatives, or even deletion.

DL_POLY_2 will run on a wide selection of computers. This includes most single processor workstations for which it requires a FORTRAN 90 compiler and (preferably) a UNIX environment. It has also been compiled for a Windows PC using the Compaq Visual FORTRAN compiler.

The list of parallel machines DL_POLY_2 has been run on includes Cray T3E and T3D, IBM SP/2 and SP/3, Intel iPSC and also Sun and Silicon Graphics multiprocessor

machines. Porting of DL_POLY_2 to these and other machines requires MPI message passing tools, (with the exception of Intel distributed memory machines, which have native message passing libraries). Note that, though designed primarily for distributed memory computers, it has been successfully run on shared memory (symmetric multiprocessor) machines as a distributed application under MPI. Note that it has not been run as a shared memory application.

1.3.4 Version Control System (CVS)

DL_POLY_2 was developed with the aid of the CVS version control system. We strongly recommend that users of DL_POLY_2 adopt this system for local development of the DL_POLY_2 code, particularly where several users access the same source code. For information on CVS please contact:

info - cvs - request@gnu.org

or visit the web site:

<http://www.cyclic.com/>

1.3.5 Required Program Libraries

DL_POLY_2 is, for the most part, self contained and does not require access to many additional program libraries. The required program libraries for parallel execution are standard MPI. The other exception is the need for 3D Fast Fourier Transform (FFT) routines in the Smooth Particle Mesh Ewald (SPME) method introduced in Version 2.12. For applications on Cray T3E/D and IBM SP/2 computers the required routines are readily available in the scientific subroutine libraries (routines ccfft3d and dcft3 respectively). For other machines an internal 3D FFT routine (dlpfft3) is available as a default, though users may prefer the public domain FFT routine fftwnd-fft for optimal performance.

1.3.6 Internal Data Transfer

As a general policy, there should be **NO** COMMON blocks in DL_POLY_2. All nonlocal and large local arrays are passed as subroutine arguments. It follows that there should be no BLOCK DATA.

However the introduction of the FORTRAN 90 dynamic array allocation features in DL_POLY_2 version 2.11 has required the creation of a named COMMON block (/params/) to facilitate the passing of the array dimensions between subroutines. (Formerly the dimensions had been defined as FORTRAN parameters and incorporated in each subroutine through the include file DL_PARAMS.INC.) The COMMON block /params/ is defined in the new version of the include file DL_PARAMS.INC and is included in almost all subroutines.

Additional COMMON blocks are also required to facilitate inter-node communication via MPI.

1.3.7 Internal Documentation

All subroutines are supplied with a header block of FORTRAN COMMENT records giving:

1. The name of the author and/or modifying author
2. The version number or date of production
3. A brief description of the function of the subroutine
4. A copyright statement
5. A CVS revision number and associated data.

Elsewhere FORTRAN COMMENT cards are used liberally.

1.3.8 Subroutine/Function Calling Sequences

The variables in the subroutine arguments are specified in the order:

1. logical and logical arrays
2. character and character arrays
3. integer
4. real and complex
5. integer arrays
6. real and complex arrays

This is admittedly arbitrary, but it really does help with error detection.

1.3.9 FORTRAN Parameters

All parameters defined by the FORTRAN parameter statements are specified in the include file: `DL_PARAMS.INC`, which is included at compilation time in all subroutines requiring the parameters. All parameters specified in `DL_PARAMS.INC` are described by one or more comment cards.

Note that since the implementation of FORTRAN 90 memory management in Version 2.11, the `DL_PARAMS.INC` file also contains the COMMON block `/params/` described above.

1.3.10 Arithmetic Precision

All real variables and parameters are specified in 64-bit precision (i.e `real*8`).

1.3.11 Units

Internally all DL_POLY_2 subroutines and functions assume the use of the following defined *molecular units*:

1. The unit of time (t_o) is 1×10^{-12} seconds (i.e. picoseconds).
2. The unit of length (ℓ_o) is 1×10^{-10} metres (i.e. Ångströms).
3. The unit of mass (m_o) is $1.6605402 \times 10^{-27}$ kilograms (i.e. atomic mass units).
4. The unit of charge (q_o) is $1.60217733 \times 10^{-19}$ coulombs (i.e. unit of proton charge).
5. The unit of energy ($E_o = m_o(\ell_o/t_o)^2$) is $1.6605402 \times 10^{-23}$ Joules (10 J mol^{-1}).
6. The unit of pressure ($\mathcal{P}_o = E_o\ell_o^{-3}$) is 1.6605402×10^7 Pascal (166.05402 bar).
7. Planck's constant ($\hbar$) which is $6.350780668 \times E_ot_o$.

In addition the following conversion factors are used:

The coulombic conversion factor (γ_o) is:

$$\gamma_o = \frac{1}{E_o} \left[\frac{q_o^2}{4\pi\epsilon_o\ell_o} \right] = 138935.4835$$

such that:

$$U_{MKS} = E_o\gamma_o U_{Internal}$$

Where U represents the configuration energy.

The Boltzmann factor (k_B) is $0.831451115 E_o K^{-1}$, such that:

$$T = E_{kin}/k_B$$

represents the conversion from kinetic energy (in internal units) to temperature (in Kelvin).

Note: In the DL_POLY_2 OUTPUT file, the print out of pressure is in units of kbars at all times. The unit of energy is either DL_POLY_2 units specified above, or in other units specified by the user at run time. The default is DL_POLY units.

1.3.12 Error Messages

All errors detected by DL_POLY_2 during run time initiate a call to the subroutine ERROR, which prints an error message in the standard output file and terminates the program. All terminations of the program are global (i.e. every node of the parallel computer will be informed of the termination condition and stop executing.)

In addition to terminal error messages, DL_POLY_2 will sometimes print warning messages. These indicate that the code has detected something that is unusual or inconsistent. The detection is non-fatal, but the user should make sure that the warning does represent a harmless condition.

1.4 The DL_POLY_2 Directory Structure

The entire DL_POLY_2 package is stored in a Unix directory structure. The topmost directory is named *dl_poly_nn*, where *nn* is a generation number. Beneath this directory are several sub-directories named: *source*; *utility*; *data*; *bench*; *execute*; *build*, *public*, *respa*, *java* and *sdk*. Briefly, the content of each sub-directory is as follows:

sub-directory	contents
<i>source</i>	primary subroutines for the DL_POLY_2 package
<i>utility</i>	subroutines, programs and example data for all utilities
<i>data</i>	example input and output files for DL_POLY_2
<i>bench</i>	large test cases suitable for benchmarking
<i>execute</i>	the DL_POLY_2 run-time directory
<i>build</i>	makefiles to assemble and compile DL_POLY_2 programs
<i>public</i>	directory of routines donated by DL_POLY_2 users
<i>sdk</i>	directory of routines used to construct the DL_POLY_2 GUI
<i>java</i>	directory of Java and FORTRAN routines for the Java GUI.
<i>respa</i>	directory of routines used to construct the DL_POLY_2 RESPA program

A more detailed description of each sub-directory follows.

1.4.1 The *source* Sub-directory

In this sub-directory all the essential source code for DL_POLY_2 , excluding the utility software. In keeping with the ‘package’ concept of DL_POLY_2 , it does not contain any complete programs; these are assembled at compile time using an appropriate makefile. The subroutines in this sub-directory are documented in chapter 8.

1.4.2 The *utility* Sub-directory

This sub-directory stores all the utility subroutines, functions and programs in DL_POLY_2 , together with examples of data. The various routines in this sub-directory are documented in chapter 7 of this manual. Users who devise their own utilities are advised to store them in the *utility* sub-directory.

1.4.3 The *data* Sub-directory

This sub-directory contains examples of input and output files for testing the released version of DL_POLY_2 . The examples of input data are copied into the *execute* sub-directory when a program is being tested. The test cases are documented in chapter 6.

1.4.4 The *bench* Sub-directory

This directory contains examples of input and output data for DL_POLY_2 that are suitable for benchmarking DL_POLY_2 on large scale computers. These are described in chapter 6.

1.4.5 The *execute* Sub-directory

In the supplied version of DL_POLY_2, this sub-directory contains only a few macros for copying and storing data from and to the *data* sub-directory and for submitting programs for execution. (These are described in section 7.3.9.) However when a DL_POLY_2 program is assembled using its makefile, it will be placed in this sub-directory and will subsequently be executed from here. The output from the job will also appear here, so users will find it convenient to use this sub-directory if they wish to use DL_POLY_2 as intended. (The experienced user is not absolutely required to use DL_POLY_2 this way however.)

1.4.6 The *build* Sub-directory

This sub-directory contains the standard makefiles for the creation (i.e. compilation and linking) of the DL_POLY_2 simulation programs. The makefiles supplied select the appropriate subroutines from the *source* sub-directory and deposit the executable program in the *execute* directory. The user is advised to copy the appropriate makefile into the *source* directory, in case any modifications are required. The copy in the *build* sub-directory will then serve as a backup.

1.4.7 The *public* Sub-directory

This sub-directory contains assorted routines donated by DL_POLY users. Potential users should note that these routines are **unsupported** and come **without any guarantee or liability whatsoever**. They should be regarded as potentially useful resources to be hacked into shape as needed by the user. This directory is available from the CCP5 Program Library by direct FTP(see below).

1.4.8 The *respa* Sub-directory

This sub-directory first appeared in Version 2.12, when the RESPA subroutines were separated from the *source* directory to make construction of the RESPA program simpler. The *respa* sub-directory consists of the subroutines unique to the RESPA application. The RESPA makefile (Makefile_respa in the *build sub-directory*) has been designed to gather the additional subroutines required from the *source* sub-directory at compile time. It is not necessary for the user to copy them over.

1.4.9 The *sdk* Sub-directory

The DL_POLY_2 SDK Graphical User Interface is based on the CERIOUS² Visualiser from Accelrys/MSI. The subroutines *written by Daresbury Laboratory* are to be found in this sub-directory. However, alone they are insufficient to create a working GUI, but DL_POLY_2 users who have access to Accelrys/MSI's SDK (Software Development Kit) will be able to use these subroutines as a starting place for constructing, and modifying, the DL_POLY_2 GUI. For DL_POLY_2 users without this resource, a compiled version of the GUI is available for Silicon Graphics computers running the CERIOUS² Visualiser (version 3.5). See the document "The DL_POLY_2 Graphical User Interface", by W. Smith [20].

1.4.10 The *java* Sub-directory

The DL_POLY_2 Java Graphical User Interface (GUI) is based on the Java language developed by Sun. The Java source code for this GUI is to be found in this sub-directory, along with a few FORTRAN sub-sub-directories which contain some additional capabilities accessible from the GUI. These sources are complete and sufficient to create a working GUI, provided the user has installed the Java Development Kit, (1.3 or above) which is available free from Sun at

[http : //java.sun.com](http://java.sun.com).

The GUI, once compiled, may be executed on any machine where Java is installed, though note the FORTRAN components will need to be recompiled if the machine is changed. (See section 5.1.)

1.5 Obtaining the Source Code

To obtain a copy of DL_POLY_2 it is first necessary to obtain a licence from Daresbury Laboratory. A copy of the licence form may be obtained in two ways: either by selecting the licence button on the World Wide Web page:

(http://www.dl.ac.uk/TCSC/Software/DL_POLY/main.html)

and downloading and printing the file; or by using FTP to copy the postscript file from the CCP5 Program Library at Daresbury Laboratory in the following manner:

1. move to the desired directory on YOUR machine,
2. type: **ftp ftp.dl.ac.uk**
3. enter userid: **anonymous**
4. enter passwd: (use your name and site)
5. change to the CCP5 directory: **cd ccp5**
6. change to the DL_POLY directory: **cd DL_POLY**
7. type: **binary**
8. type: **get LICENCE.ps.Z** or **get GROUP.LICENCE.ps.Z**
9. type: **quit**

The licence file will need to be uncompressed (using the unix *uncompress* command) before printing. Note that there are two versions of the licence available; one for single academic users (with perhaps one or two postgraduate students); and one for academic groups (with perhaps several research staff, including postdoctoral and permanent staff.) Choose the one most suitable for you.

Once you have obtained the licence form you should sign it and return it to the following address.

Dr. W. Smith
DL_POLY Program Library
Computational Science and Engineering Department
CCLRC Daresbury Laboratory
Daresbury
Warrington WA4 4AD
England

Please return the licence **by post please**; FAXes are not acceptable to our Contracts Department. When the signed licence has been received DL_POLY_2 source code will be sent by ftp. We will need to contact you about this procedure, so **please supply your e-mail address**. Please note we cannot create accounts on any of our machines for this purpose.

The DL_POLY_2 User Manual is freely available via World Wide Web or ftp, in the same manner as the licence form. The much larger DL_POLY_2 Reference Manual will be available by the above ftp procedure only.

Daresbury Laboratory is the sole centre for the distribution of DL_POLY_2 and copies obtained from elsewhere will be regarded as illegal and will not be supported.

1.6 DLPROTEIN

DLPROTEIN is a further development of DL_POLY that was written by S. Melchionna and S. Cozzini (with assistance from M. Venturoli and Antonella Luise) specifically to handle protein and nucleic acid simulations. In agreement with the authors the package is also available from Daresbury Laboratory under the same terms as DL_POLY. Though having much in common with DL_POLY it represents a major rewrite, with many features not present in the original code (and with many original features missing). It is therefore documented separately from DL_POLY and users should consult the appropriate manual: "The DLPROTEIN User Manual" by S. Melchionna and S. Cozzini [21], which also available from Daresbury Laboratory.

Our thanks go to S. Melchionna and S. Cozzini and their collaborators for generously making DLPROTEIN available to us and for extending that availability to DL_POLY_2 users.

1.7 Other Information

The DL_POLY_2 web page: http://ftp.dl.ac.uk/TCS/Software/DL_POLY/main.html provides additional information in the form of

1. Access to all documentation (including licences);
2. Frequently asked questions;
3. Bug reports.

Daresbury Laboratory also maintains two DL_POLY_2 associated electronic mailing lists:

1. *dl_poly_news* - to which all registered DL_POLY_2 users are automatically subscribed. It is via this list that error reports and announcements of new versions are made. If you are a DL_POLY_2 user, but not on this list you may request to be added. Contact w.smith@dl.ac.uk.
2. *dl_poly_mail* - is a group list which is available to DL_POLY_2 users by request. Its purpose is to allow DL_POLY_2 users to broadcast information and queries to each other. To subscribe to this list send a mail message to majordomo@dl.ac.uk with the one-line message:

subscribe dl_poly_mail

Subsequent messages may be broadcast by e-mailing to the address: dl.poly_mail@dl.ac.uk. Note that this is a vetted list, so electronic spam is not possible.

Chapter 2

DL_POLY_2 Force Fields and Algorithms

Scope of Chapter

This chapter describes the interaction potentials and simulation algorithms coded into DL_POLY_2 .

2.1 The DL_POLY_2 Force Field

The force field is the set of functions needed to define the interactions in a molecular system. These may have a wide variety of analytical forms, with some basis in chemical physics, which must be parameterised to give the correct energy and forces. A huge variety of forms is possible and for this reason the DL_POLY_2 force field is designed to be adaptable. While it is not supplied with its own force field parameters, many of the functions familiar to GROMOS, [3] Dreiding [7] and AMBER [4] users have been coded in the package, as well as less familiar forms. In addition DL_POLY_2 retains the possibility of the user defining additional potentials.

In DL_POLY_2 the total configuration energy of a molecular system may be written as:

$$\begin{aligned}
 U(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_N) = & \sum_{i_{bond}=1}^{N_{bond}} U_{bond}(i_{bond}, \mathbf{r}_a, \mathbf{r}_b) \\
 & + \sum_{i_{angle}=1}^{N_{angle}} U_{angle}(i_{angle}, \mathbf{r}_a, \mathbf{r}_b, \mathbf{r}_c) \\
 & + \sum_{i_{dihed}=1}^{N_{dihed}} U_{dihed}(i_{dihed}, \mathbf{r}_a, \mathbf{r}_b, \mathbf{r}_c, \mathbf{r}_d) \\
 & + \sum_{i_{inv}=1}^{N_{inv}} U_{inv}(i_{inv}, \mathbf{r}_a, \mathbf{r}_b, \mathbf{r}_c, \mathbf{r}_d) \\
 & + \sum_{i=1}^{N-1} \sum_{j>i}^N U_{pair}(i, j, |\mathbf{r}_i - \mathbf{r}_j|) \\
 & + \sum_{i=1}^{N-2} \sum_{j>i}^{N-1} \sum_{k>j}^N U_{3\_body}(i, j, k, \mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k) \\
 & + \sum_{i=1}^{N-3} \sum_{j>i}^{N-2} \sum_{k>j}^{N-1} \sum_{n>k}^N U_{4\_body}(i, j, k, n, \mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k, \mathbf{r}_n) \\
 & + \sum_{i=1}^N U_{extn}(i, \mathbf{r}_i, \mathbf{v}_i)
 \end{aligned} \tag{2.1}$$

where U_{bond} , U_{angle} , U_{dihed} , U_{inv} , U_{pair} , U_{3_body} and U_{4_body} are empirical interaction functions representing chemical bonds, valence angles, dihedral angles, inversion angles, pair-body, three-body and four-body forces respectively. The first four are regarded by DL_POLY_2 as *intra*-molecular interactions and the next three as *inter*-molecular interactions. The final term U_{extn} represents an *external field* potential. The position vectors $\mathbf{r}_a, \mathbf{r}_b, \mathbf{r}_c$ and $\mathbf{r}_d$ refer to the positions of the atoms specifically involved in a given interaction. (Almost universally, it is the *differences* in position that determine the interaction.) The numbers N_{bond} , N_{angle} , N_{dihed} and N_{inv} refer to the total numbers of these respective interactions present in the simulated system, and the indices i_{bond} , i_{angle} , i_{inv} and i_{dihed} uniquely specify an individual interaction of each type. It is important to note that there is

no global specification of the intramolecular interactions in DL_POLY_2 - all bonds, valence angles and dihedrals must be individually cited.

The indices i , j (and k , n) appearing in the pair-body (and three or four-body) terms indicate the atoms involved in the interaction. There is normally a very large number of these and they are therefore specified according to atom *types* rather than indices. In DL_POLY_2 it is assumed that the pair-body terms arise from van der Waals and/or electrostatic (Coulombic) forces. The former are regarded as short ranged interactions and the latter as long ranged. Long range forces require special techniques to evaluate accurately (see section 2.4.) In DL_POLY_2 the three-body terms are restricted to valence angle and H-bond forms. The nonbonded, three-body and four-body interactions are globally specified according to the *types* of atoms involved. DL_POLY_2 also has the ability to handle metals via density dependent functions (see below). Though essentially many-body potentials they are handled in DL_POLY_2 as special forms of pair potential.

In DL_POLY_2 the intramolecular bonded terms are handled using bookkeeping arrays, which specify the atoms involved in a particular interaction and point to the appropriate arrays of parameters that define the potential. The calculation of bonded forces therefore follows the simple scheme:

1. Every atom in the simulated system is assigned a unique index number from 1 to N ;
2. Every intramolecular bonded term U_{type} in the system has a unique index number i_{type} : from 1 to N_{type} where *type* represents a bond, angle or dihedral.
3. A pointer array $key_{type}(n_{type}, i_{type})$ carries the indices of the specific atoms involved in the potential term labelled i_{type} . The dimension n_{type} will be 2, 3 or 4, if the term represents a bond, valence angle, dihedral/inversion.
4. The array $key_{type}(n_{type}, i_{type})$ is used to identify the atoms in a bonded term and the appropriate form of interaction and thus to calculate the energy and forces.

DL_POLY_2 calculates the nonbonded pair interactions using a Verlet neighbour list [10] which is reconstructed at intervals during the simulation. This list records the indices of all ‘secondary’ atoms within a certain radius of each ‘primary’ atom; the radius being the cut-off radius (r_{cut}) normally applied to the nonbonded potential function, plus an additional increment (Δr_{cut}). The neighbour list removes the need to scan over all atoms in the simulation at every timestep. The larger radius ($r_{cut} + \Delta r_{cut}$) means the same list can be used for several timesteps without requiring an update. The frequency at which the list must be updated depends on the thickness of the region Δr_{cut} . DL_POLY_2 has two methods for constructing the neighbour list: the first is based on the Brode-Ahlrichs scheme [22] and is used when r_{cut} is large in comparison with the simulation cell; the second uses the link-cell algorithm [23] when r_{cut} is relatively small. The potential energy and forces arising from the nonbonded interactions are calculated using interpolation tables.

A complication in the construction of the Verlet neighbour list for macromolecules is the concept of *excluded atoms*, which arises from the need to exclude certain atom pairs from the overall list. Which atom pairs need to be excluded is dependent on the precise

nature of the force field model, but as a minimum atom pairs linked via extensible bonds or constraints and atoms (grouped in pairs) linked via valence angles are probable candidates. The assumption behind this requirement is that atoms that are formally bonded in a chemical sense, should not participate in nonbonded interactions. (However this is not a universal requirement of all force fields.) The same considerations are needed in dealing with charged excluded atoms. DL_POLY_2 has several subroutines available for constructing the Verlet neighbour list, while taking care of the excluded atoms (see chapters 3 and 8 for further information.)

Three- and four-body nonbonded forces are assumed to be short ranged and therefore calculated using the link-cell algorithm [23]. They ignore the possibility of there being any excluded interactions involving the atoms concerned.

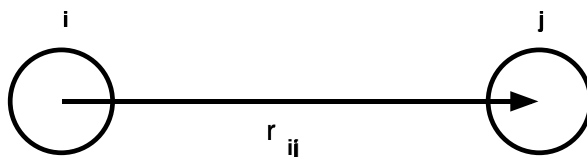
Throughout this section the description of the force field assumes the simulated system is described as an assembly of atoms. This is for convenience only and readers should understand that DL_POLY_2 does recognise molecular entities, defined either through constraint bonds or rigid bodies. In the case of rigid bodies, the atomic forces are resolved into molecular forces and torques. These matters are discussed in greater detail later in sections 2.5.1 and 2.5.6).

Note that the subroutines mentioned in the subsections of this chapter are described in greater detail in chapter 8 of the Reference Manual.

2.2 The Intramolecular Potential Functions

In this section we catalogue and describe the forms of potential function available in DL_POLY_2. The **key words** required to select potential forms are given in brackets () before each definition. The derivations of the atomic forces, virial and stress tensor are also outlined.

2.2.1 Bond Potentials



The interatomic bond vector.

The bond potentials describe *explicit* bonds between specified atoms. They are functions of the interatomic distance only. The potential functions available are as follows.

1. Harmonic bond: (**harm**)

$$U(r_{ij}) = \frac{1}{2}k(r_{ij} - r_o)^2; \quad (2.2)$$

2. Morse potential: (**mors**)

$$U(r_{ij}) = E_o[\{1 - \exp(-k(r_{ij} - r_o))\}^2 - 1]; \quad (2.3)$$

3. 12-6 potential bond: (**12-6**)

$$U(r_{ij}) = \left(\frac{A}{r_{ij}^{12}} \right) - \left(\frac{B}{r_{ij}^6} \right); \quad (2.4)$$

4. Restrained harmonic: (**rhrm**)

$$U(r_{ij}) = \frac{1}{2}k(r_{ij} - r_o)^2 \quad |r_{ij} - r_o| \leq r_c; \quad (2.5)$$

$$U(r_{ij}) = \frac{1}{2}kr_c^2 + kr_c(|r_{ij} - r_o| - r_c) \quad |r_{ij} - r_o| > r_c; \quad (2.6)$$

5. Quartic potential: (**quar**)

$$U(r_{ij}) = \frac{k}{2}(r_{ij} - r_o)^2 + \frac{k'}{3}(r_{ij} - r_o)^3 + \frac{k''}{4}(r_{ij} - r_o)^4. \quad (2.7)$$

In these formulae r_{ij} is the distance between atoms labelled i and j :

$$r_{ij} = |\underline{r}_j - \underline{r}_i|, \quad (2.8)$$

where $\underline{r}_\ell$ is the position vector of an atom labelled ℓ .¹

The force on the atom j arising from a bond potential is obtained using the general formula:

$$\underline{f}_j = -\frac{1}{r_{ij}} \left[\frac{\partial}{\partial r_{ij}} U(r_{ij}) \right] \underline{r}_{ij}, \quad (2.9)$$

The force $\underline{f}_i$ acting on atom i is the negative of this.

The contribution to be added to the atomic virial is given by

$$\mathcal{W} = -\underline{r}_{ij} \cdot \underline{f}_j, \quad (2.10)$$

with only *one* such contribution from each bond.

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^\alpha f_j^\beta, \quad (2.11)$$

where α and β indicate the x, y, z components. The atomic stress tensor derived in this way is symmetric.

In DL_POLY_2 bond forces are handled by the routine BNDFRG.

¹Note: some DL_POLY_2 routines may use the convention that $\underline{r}_{ij} = \underline{r}_i - \underline{r}_j$.

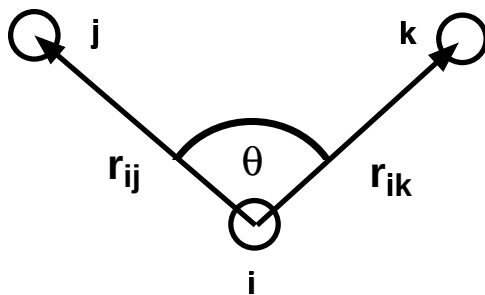
2.2.2 Distance Restraints

In DL_POLY_2 distance restraints, in which the separation between two atoms, is maintained around some preset value r_0 is handled as a special case of bond potentials. As a consequence distance restraints may be applied only between atoms in the same molecule. Unlike with application of the “pure” bond potentials, the electrostatic and van der Waals interactions between the pair of atoms are still evaluated when distance restraints are applied. All the potential forms of the previous section are available as distance restraints, although they have different key words:

1. Harmonic potential: (**-hrm**)
2. Morse potential: (**-mrs**)
3. 12-6 potential bond: (**-126**)
4. Restrained harmonic: (**-rhm**)
5. Quartic potential: (**-qur**)

In DL_POLY_2 distance restraints are handled by the routine BNDFRC.

2.2.3 Valence Angle Potentials



The valence angle and associated vectors

The valence angle potentials describe the bond bending terms between the specified atoms. They should not be confused with the three body potentials described later, which are defined by atom types rather than indices.

1. Harmonic: (**harm**)

$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2; \quad (2.12)$$

2. Quartic: (**quar**)

$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 + \frac{k'}{3}(\theta_{jik} - \theta_0)^3 + \frac{k''}{4}(\theta_{jik} - \theta_0)^4; \quad (2.13)$$

3. Truncated harmonic: (**thrm**)

$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]; \quad (2.14)$$

4. Screened harmonic: (**shrm**)

$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]; \quad (2.15)$$

5. Screened Vessal[24]: (**bvs1**)

$$U(\theta_{jik}) = \frac{k}{8(\theta_{jik} - \pi)^2} \left\{ [(\theta_0 - \pi)^2 - (\theta_{jik} - \pi)^2]^2 \right\} \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]; \quad (2.16)$$

6. Truncated Vessal[25]: (**bvs2**)

$$U(\theta_{jik}) = k[\theta_{jik}^a(\theta_{jik} - \theta_0)^2(\theta_{jik} + \theta_0 - 2\pi)^2 - \frac{a}{2}\pi^{a-1}(\theta_{jik} - \theta_0)^2(\pi - \theta_0)^3] \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]. \quad (2.17)$$

7. Harmonic cosine: (**hcos**)

$$U(\theta_{jik}) = \frac{k}{2}(\cos(\theta_{jik}) - \cos(\theta_0))^2 \quad (2.18)$$

8. Cosine: (**cos**)

$$U(\theta_{jik}) = A[1 + \cos(m\theta_{jik} - \delta)] \quad (2.19)$$

In these formulae θ_{jik} is the angle between bond vectors $\underline{r}_{ij}$ and $\underline{r}_{ik}$:

$$\theta_{jik} = \cos^{-1} \left\{ \frac{\underline{r}_{ij} \cdot \underline{r}_{ik}}{r_{ij}r_{ik}} \right\} \quad (2.20)$$

In DL_POLY_2 the most general form for the valence angle potentials can be written as:

$$U(\theta_{jik}, r_{ij}, r_{ik}) = A(\theta_{jik})S(r_{ij})S(r_{ik}) \quad (2.21)$$

where $A(\theta)$ is a purely angular function and $S(r)$ is a screening or truncation function. All the function arguments are scalars. With this reduction the force on an atom derived from the valence angle potential is given by:

$$f_\ell^\alpha = -\frac{\partial}{\partial r_\ell^\alpha} U(\theta_{jik}, r_{ij}, r_{ik}), \quad (2.22)$$

with atomic label ℓ being one of i, j, k and α indicating the x, y, z component. The derivative is

$$\begin{aligned} -\frac{\partial}{\partial r_\ell^\alpha} U(\theta_{jik}, r_{ij}, r_{ik}) &= -S(r_{ij})S(r_{ik})\frac{\partial}{\partial r_\ell^\alpha} A(\theta_{jik}) \\ &\quad -A(\theta_{jik})S(r_{ik})(\delta_{\ell j} - \delta_{\ell i})\frac{r_{ij}^\alpha}{r_{ij}}\frac{\partial}{\partial r_{ij}} S(r_{ij}) \\ &\quad -A(\theta_{jik})S(r_{ij})(\delta_{\ell k} - \delta_{\ell i})\frac{r_{ik}^\alpha}{r_{ik}}\frac{\partial}{\partial r_{ik}} S(r_{ik}), \end{aligned} \quad (2.23)$$

with $\delta_{ab} = 1$ if $a = b$ and $\delta_{ab} = 0$ if $a \neq b$. In the absence of screening terms $S(r)$, this formula reduces to:

$$-\frac{\partial}{\partial r_\ell^\alpha} U(\theta_{jik}, r_{ij}, r_{ik}) = -\frac{\partial}{\partial r_\ell^\alpha} A(\theta_{jik}) \quad (2.24)$$

The derivative of the angular function is

$$-\frac{\partial}{\partial r_\ell^\alpha} A(\theta_{jik}) = \left\{ \frac{1}{\sin(\theta_{jik})} \right\} \frac{\partial}{\partial \theta_{jik}} A(\theta_{jik}) \frac{\partial}{\partial r_\ell^\alpha} \left\{ \frac{\underline{r}_{ij} \cdot \underline{r}_{ik}}{r_{ij}r_{ik}} \right\}, \quad (2.25)$$

with

$$\begin{aligned} \frac{\partial}{\partial r_j^\alpha} \left\{ \frac{\underline{r}_{ij} \cdot \underline{r}_{ik}}{r_{ij}r_{ik}} \right\} &= (\delta_{\ell j} - \delta_{\ell i})\frac{r_{ik}^\alpha}{r_{ij}r_{ik}} + (\delta_{\ell k} - \delta_{\ell i})\frac{r_{ij}^\alpha}{r_{ij}r_{ik}} - \\ &\quad \cos(\theta_{jik}) \left\{ (\delta_{\ell j} - \delta_{\ell i})\frac{r_{ij}^\alpha}{r_{ij}^2} + (\delta_{\ell k} - \delta_{\ell i})\frac{r_{ik}^\alpha}{r_{ik}^2} \right\} \end{aligned} \quad (2.26)$$

The atomic forces are then completely specified by the derivatives of the particular functions $A(\theta)$ and $S(r)$.

The contribution to be added to the atomic virial is given by

$$\mathcal{W} = -(\underline{r}_{ij} \cdot \underline{f}_j + \underline{r}_{ik} \cdot \underline{f}_k) \quad (2.27)$$

It is worth noting that in the absence of screening terms $S(r)$, the virial is zero [26].

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^\alpha f_j^\beta + r_{ik}^\alpha f_k^\beta \quad (2.28)$$

and the stress tensor is symmetric.

In DL_POLY_2 valence forces are handled by the routine ANGFRG.

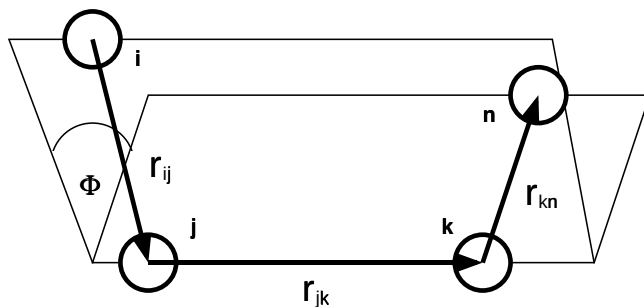
2.2.4 Angular Restraints

In DL_POLY_2 angle restraints, in which the angle subtended by a triplet of atoms, is maintained around some preset value θ_0 is handled as a special case of angle potentials. As a consequence angle restraints may be applied only between atoms in the same molecule. Unlike with application of the “pure” angle potentials, the electrostatic and van der Waals interactions between the pair of atoms are still evaluated when distance restraints are applied. All the potential forms of the previous section are available as angular restraints, although they have different key words:

1. Harmonic: (**-hrm**)
2. Quartic: (**-qur**)
3. Truncated harmonic: (**-thm**)
4. Screened harmonic: (**-shm**)
5. Screened Vessal[24]: (**-bv1**)
6. Truncated Vessal[25]: (**-bv2**)
7. Harmonic cosine: (**-hcs**)
8. Cosine : (**-cos**)

In DL_POLY_2 angular restraints are handled by the routine ANGFRG.

2.2.5 Dihedral Angle Potentials



The dihedral angle and associated vectors

The dihedral angle potentials describe the interaction arising from torsional forces in molecules. (They are sometimes referred to as torsion potentials.) They require the specification of four atomic positions. The potential functions available in DL_POLY_2 are as follows.

1. Cosine potential: (**cos**)

$$U(\phi_{ijkn}) = A[1 + \cos(m\phi_{ijkn} - \delta)] \quad (2.29)$$

2. Harmonic: (**harm**)

$$U(\phi_{ijkn}) = \frac{1}{2}k(\phi_{ijkn} - \phi_0)^2 \quad (2.30)$$

3. Harmonic cosine: (**hcos**)

$$U(\phi_{ijkn}) = \frac{k}{2}(\cos(\phi_{ijkn}) - \cos(\phi_0))^2 \quad (2.31)$$

4. Triple cosine: (**cos3**)

$$U(\phi) = \frac{1}{2}A_1(1 + \cos(\phi)) + \frac{1}{2}A_2(1 - \cos(2\phi)) + \frac{1}{2}A_3(1 + \cos(3\phi)) \quad (2.32)$$

In these formulae ϕ_{ijkn} is the dihedral angle defined by

$$\phi_{ijkn} = \cos^{-1}\{B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn})\}, \quad (2.33)$$

with

$$B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn}) = \left\{ \frac{(\underline{r}_{ij} \times \underline{r}_{jk}) \cdot (\underline{r}_{jk} \times \underline{r}_{kn})}{|\underline{r}_{ij} \times \underline{r}_{jk}| |\underline{r}_{jk} \times \underline{r}_{kn}|} \right\}. \quad (2.34)$$

With this definition, the sign of the dihedral angle is positive if the vector product $(\underline{r}_{ij} \times \underline{r}_{jk}) \times (\underline{r}_{jk} \times \underline{r}_{kn})$ is in the same direction as the bond vector $\underline{r}_{jk}$ and negative if in the opposite direction.

The force on an atom arising from the dihedral potential is given by

$$f_\ell^\alpha = -\frac{\partial}{\partial r_\ell^\alpha} U(\phi_{ijkn}), \quad (2.35)$$

with ℓ being one of i, j, k, n and α one of x, y, z . This may be expanded into

$$-\frac{\partial}{\partial r_\ell^\alpha} U(\phi_{ijkn}) = \left\{ \frac{1}{\sin(\phi_{ijkn})} \right\} \frac{\partial}{\partial \phi_{ijkn}} U(\phi_{ijkn}) \frac{\partial}{\partial r_\ell^\alpha} B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn}). \quad (2.36)$$

The derivative of the function $B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn})$ is

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn}) &= \frac{1}{|\underline{r}_{ij} \times \underline{r}_{jk}| |\underline{r}_{jk} \times \underline{r}_{kn}|} \frac{\partial}{\partial r_\ell^\alpha} \{(\underline{r}_{ij} \times \underline{r}_{jk}) \cdot (\underline{r}_{jk} \times \underline{r}_{kn})\} \\ &- \frac{\cos(\phi_{ijkn})}{2} \left\{ \frac{1}{|\underline{r}_{ij} \times \underline{r}_{jk}|^2} \frac{\partial}{\partial r_\ell^\alpha} |\underline{r}_{ij} \times \underline{r}_{jk}|^2 + \frac{1}{|\underline{r}_{jk} \times \underline{r}_{kn}|^2} \frac{\partial}{\partial r_\ell^\alpha} |\underline{r}_{jk} \times \underline{r}_{kn}|^2 \right\}, \end{aligned} \quad (2.37)$$

with

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} \{(\underline{r}_{ij} \times \underline{r}_{jk}) \cdot (\underline{r}_{jk} \times \underline{r}_{kn})\} &= r_{ij}^\alpha ([\underline{r}_{jk} \underline{r}_{jk}]_\alpha (\delta_{\ell k} - \delta_{\ell n}) + [\underline{r}_{jk} \underline{r}_{kn}]_\alpha (\delta_{\ell k} - \delta_{\ell j})) + \\ &r_{jk}^\alpha ([\underline{r}_{ij} \underline{r}_{jk}]_\alpha (\delta_{\ell n} - \delta_{\ell k}) + [\underline{r}_{jk} \underline{r}_{kn}]_\alpha (\delta_{\ell j} - \delta_{\ell i})) + \\ &r_{kn}^\alpha ([\underline{r}_{ij} \underline{r}_{jk}]_\alpha (\delta_{\ell k} - \delta_{\ell j}) + [\underline{r}_{jk} \underline{r}_{jk}]_\alpha (\delta_{\ell i} - \delta_{\ell j})) + \\ &2r_{jk}^\alpha [\underline{r}_{ij} \underline{r}_{kn}]_\alpha (\delta_{\ell j} - \delta_{\ell k}), \end{aligned} \quad (2.38)$$

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} |\underline{r}_{ij} \times \underline{r}_{jk}|^2 &= 2r_{ij}^\alpha ([\underline{r}_{jk} \underline{r}_{jk}]_\alpha (\delta_{\ell j} - \delta_{\ell i}) + [\underline{r}_{ij} \underline{r}_{jk}]_\alpha (\delta_{\ell j} - \delta_{\ell k})) + \\ &2r_{jk}^\alpha ([\underline{r}_{ij} \underline{r}_{ij}]_\alpha (\delta_{\ell k} - \delta_{\ell j}) + [\underline{r}_{ij} \underline{r}_{jk}]_\alpha (\delta_{\ell i} - \delta_{\ell j})), \end{aligned} \quad (2.39)$$

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} |\underline{r}_{jk} \times \underline{r}_{kn}|^2 &= 2r_{kn}^\alpha ([\underline{r}_{jk}\underline{r}_{jk}]_\alpha (\delta_{\ell n} - \delta_{\ell k}) + [\underline{r}_{jk}\underline{r}_{kn}]_\alpha (\delta_{\ell j} - \delta_{\ell k})) + \\ &2r_{jk}^\alpha ([\underline{r}_{kn}\underline{r}_{kn}]_\alpha (\delta_{\ell k} - \delta_{\ell j}) + [\underline{r}_{jk}\underline{r}_{kn}]_\alpha (\delta_{\ell k} - \delta_{\ell n})). \end{aligned} \quad (2.40)$$

Where we have used the the following definition:

$$[\underline{a} \ \underline{b}]_\alpha = \sum_\beta (1 - \delta_{\alpha\beta}) a^\beta b^\beta. \quad (2.41)$$

Formally, the contribution to be added to the atomic virial is given by

$$\mathcal{W} = - \sum_{i=1}^4 \underline{r}_i \cdot \underline{f}_i \quad (2.42)$$

However it is possible to show (by tedious algebra using the above formulae, or more elegantly by thermodynamic arguments [26],) that the dihedral makes *no* contribution to the atomic virial.

The contribution to be added to the atomic stress tensor is given by

$$\begin{aligned} \sigma^{\alpha\beta} &= r_{ij}^\alpha p_i^\beta + r_{jk}^\alpha p_{jk}^\beta + r_{kn}^\alpha p_n^\beta \\ &\quad - \frac{\cos(\phi_{ijkn})}{2} \left\{ r_{ij}^\alpha g_i^\beta + r_{jk}^\alpha g_k^\beta + r_{jk}^\alpha h_j^\beta + r_{kn}^\alpha h_n^\beta \right\}, \end{aligned} \quad (2.43)$$

with

$$p_i^\alpha = (r_{jk}^\alpha [\underline{r}_{jk}\underline{r}_{kn}]_\alpha - r_{kn}^\alpha [\underline{r}_{jk}\underline{r}_{jk}]_\alpha) / (|\underline{r}_{ij} \times \underline{r}_{jk}| |\underline{r}_{jk} \times \underline{r}_{kn}|) \quad (2.44)$$

$$p_n^\alpha = (r_{jk}^\alpha [\underline{r}_{ij}\underline{r}_{jk}]_\alpha - r_{ij}^\alpha [\underline{r}_{jk}\underline{r}_{jk}]_\alpha) / (|\underline{r}_{ij} \times \underline{r}_{jk}| |\underline{r}_{jk} \times \underline{r}_{kn}|) \quad (2.45)$$

$$p_{jk}^\alpha = (r_{ij}^\alpha [\underline{r}_{jk}\underline{r}_{kn}]_\alpha + r_{kn}^\alpha [\underline{r}_{ij}\underline{r}_{jk}]_\alpha - 2r_{jk}^\alpha [\underline{r}_{ij}\underline{r}_{kn}]_\alpha) / (|\underline{r}_{ij} \times \underline{r}_{jk}| |\underline{r}_{jk} \times \underline{r}_{kn}|) \quad (2.46)$$

$$g_i^\alpha = 2(r_{ij}^\alpha [\underline{r}_{jk}\underline{r}_{jk}]_\alpha - r_{jk}^\alpha [\underline{r}_{ij}\underline{r}_{jk}]_\alpha) / |\underline{r}_{ij} \times \underline{r}_{jk}|^2 \quad (2.47)$$

$$g_k^\alpha = 2(r_{jk}^\alpha [\underline{r}_{ij}\underline{r}_{ij}]_\alpha - r_{ij}^\alpha [\underline{r}_{ij}\underline{r}_{jk}]_\alpha) / |\underline{r}_{ij} \times \underline{r}_{jk}|^2 \quad (2.48)$$

$$h_j^\alpha = 2(r_{jk}^\alpha [\underline{r}_{kn}\underline{r}_{kn}]_\alpha - r_{kn}^\alpha [\underline{r}_{jk}\underline{r}_{kn}]_\alpha) / |\underline{r}_{jk} \times \underline{r}_{kn}|^2 \quad (2.49)$$

$$h_n^\alpha = 2(r_{kn}^\alpha [\underline{r}_{kn}\underline{r}_{kn}]_\alpha - r_{jk}^\alpha [\underline{r}_{jk}\underline{r}_{kn}]_\alpha) / |\underline{r}_{jk} \times \underline{r}_{kn}|^2 \quad (2.50)$$

The sum of the diagonal elements of the stress tensor is zero (since the virial is zero) and the matrix is symmetric.

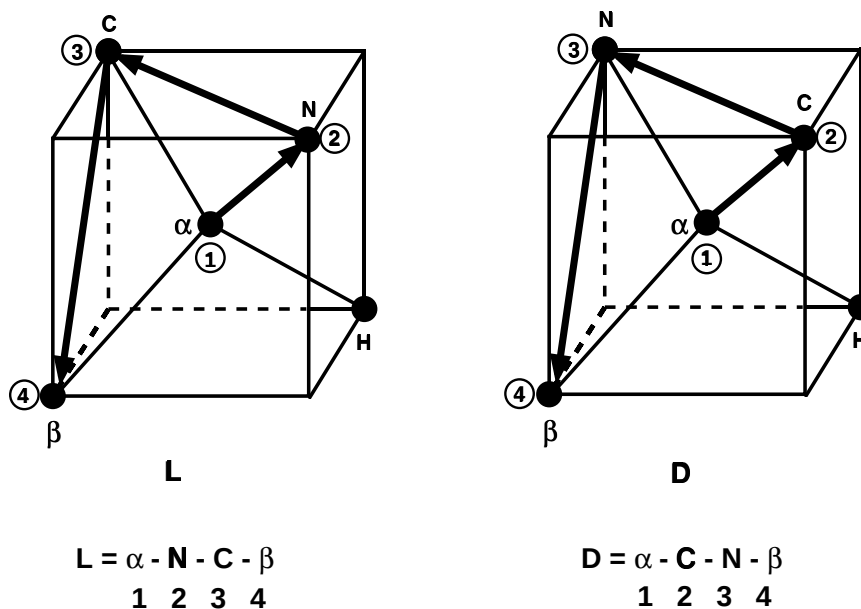
Lastly, it should be noted that the above description does not take into account the possible inclusion of distance-dependent 1-4 interactions, as permitted by some force fields. Such interactions are permissible in DL_POLY_2 and are described in the section on pair potentials below. DL_POLY_2 also permits scaling of the 1-4 interactions by a numerical factor. 1-4 interactions do, of course, contribute to the atomic virial.

In DL_POLY_2 dihedral forces are handled by the routine DIHFRG.

2.2.6 Improper Dihedral Angle Potentials

Improper dihedrals are used to restrict the geometry of molecules and as such need not have a simple relation to conventional chemical bonding. DL_POLY_2 makes no distinction between dihedral angle functions and improper dihedrals (both are calculated by the same subroutines) and all the comments made in the preceeding section apply.

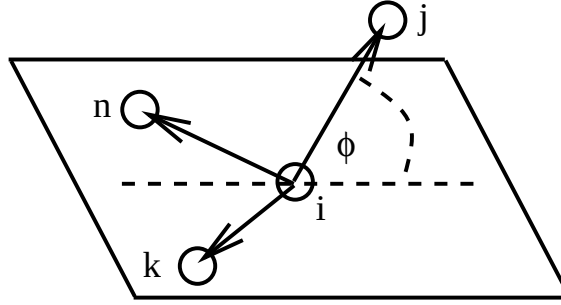
An important example of the use of the improper dihedral is to conserve the structure of chiral centres in molecules modelled by united-atom centres. For example α -amino acids such as alanine ($\text{CH}_3\text{CH}(\text{NH}_2)\text{COOH}$), in which it is common to represent the CH_3 and CH groups as single centres. Conservation of the chirality of the α carbon is achieved by defining a harmonic improper dihedral angle potential with an equilibrium angle of 35.264° . The angle is defined by vectors $\mathbf{r}_{12}$, $\mathbf{r}_{23}$ and $\mathbf{r}_{34}$, where the atoms 1,2,3 and 4 are shown in the following figure. The figure defines the D and L enantiomers consistent with the international (IUPAC) convention. When defining the dihedral, the atom indices are entered in DL_POLY_2 in the order 1-2-3-4.



The L and D enantiomers and defining vectors

In DL_POLY_2 improper dihedral forces are handled by the routine DIHFRC.

2.2.7 Inversion Angle Potentials



The inversion angle and associated vectors

The inversion angle potentials describe the interaction arising from a particular geometry of three atoms around a central atom. The best known example of this is the arrangement of hydrogen atoms around nitrogen in ammonia to form a trigonal pyramid. The hydrogens can ‘flip’ like an inverting umbrella to an alternative structure, which in this case is identical, but in principle causes a change in chirality. The force restraining the ammonia to one structure can be described as an inversion potential (though it is usually augmented by valence angle potentials also). The inversion angle is defined in the figure above - **note that the inversion angle potential is a sum of the three possible inversion angle terms**. It resembles a dihedral potential in that it requires the specification of four atomic positions.

The potential functions available in DL_POLY_2 are as follows.

1. Harmonic: (**harm**)

$$U(\phi_{ijkn}) = \frac{1}{2}k(\phi_{ijkn} - \phi_0)^2 \quad (2.51)$$

2. Harmonic cosine: (**hcos**)

$$U(\phi_{ijkn}) = \frac{k}{2}(\cos(\phi_{ijkn}) - \cos(\phi_0))^2 \quad (2.52)$$

3. Planar potential: (**plan**)

$$U(\phi_{ijkn}) = A[1 - \cos(\phi_{ijkn})] \quad (2.53)$$

In these formulae ϕ_{ijkn} is the inversion angle defined by

$$\phi_{ijkn} = \cos^{-1} \left\{ \frac{\underline{r}_{ij} \cdot \underline{w}_{kn}}{r_{ij}w_{kn}} \right\}, \quad (2.54)$$

with

$$\underline{w}_{kn} = (\underline{r}_{ij} \cdot \underline{\hat{u}}_{kn})\underline{\hat{u}}_{kn} + (\underline{r}_{ij} \cdot \underline{\hat{v}}_{kn})\underline{\hat{v}}_{kn} \quad (2.55)$$

and the unit vectors

$$\begin{aligned} \underline{\hat{u}}_{kn} &= (\underline{\hat{r}}_{ik} + \underline{\hat{r}}_{in})/|\underline{\hat{r}}_{ik} + \underline{\hat{r}}_{in}| \\ \underline{\hat{v}}_{kn} &= (\underline{\hat{r}}_{ik} - \underline{\hat{r}}_{in})/|\underline{\hat{r}}_{ik} - \underline{\hat{r}}_{in}|. \end{aligned} \quad (2.56)$$

As usual, $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$ etc. and the hat $\hat{\underline{r}}$ indicates a *unit* vector in the direction of $\underline{r}$. The total inversion potential requires the calculation of three such angles, the formula being derived from the above using the cyclic permutation of the indices $j \rightarrow k \rightarrow n \rightarrow j$ etc.

Equivalently, the angle ϕ_{ijkn} may be written as

$$\phi_{ijkn} = \cos^{-1} \left\{ \frac{[(\underline{r}_{ij} \cdot \hat{\underline{u}}_{kn})^2 + (\underline{r}_{ij} \cdot \hat{\underline{v}}_{kn})^2]^{1/2}}{r_{ij}} \right\} \quad (2.57)$$

Formally, the force on an atom arising from the inversion potential is given by

$$f_\ell^\alpha = -\frac{\partial}{\partial r_\ell^\alpha} U(\phi_{ijkn}), \quad (2.58)$$

with ℓ being one of i, j, k, n and α one of x, y, z . This may be expanded into

$$\begin{aligned} -\frac{\partial}{\partial r_\ell^\alpha} U(\phi_{ijkn}) &= \left\{ \frac{1}{\sin(\phi_{ijkn})} \right\} \frac{\partial}{\partial \phi_{ijkn}} U(\phi_{ijkn}) \times \\ &\quad \frac{\partial}{\partial r_\ell^\alpha} \left\{ \frac{[(\underline{r}_{ij} \cdot \hat{\underline{u}}_{kn})^2 + (\underline{r}_{ij} \cdot \hat{\underline{v}}_{kn})^2]^{1/2}}{r_{ij}} \right\}. \end{aligned} \quad (2.59)$$

Following through the (extremely tedious!) differentiation gives the result:

$$\begin{aligned} f_\ell^\alpha &= \left\{ \frac{1}{\sin(\phi_{ijkn})} \right\} \frac{\partial}{\partial \phi_{ijkn}} U(\phi_{ijkn}) \times \\ &\quad \left\{ -(\delta_{\ell j} - \delta_{\ell i}) \frac{\cos(\phi_{ijkn})}{r_{ij}^2} r_{ij}^\alpha + \frac{1}{r_{ij} w_{kn}} \left[(\delta_{\ell j} - \delta_{\ell i}) \{ (\underline{r}_{ij} \cdot \hat{\underline{u}}_{kn}) \hat{u}_{kn}^\alpha + (\underline{r}_{ij} \cdot \hat{\underline{v}}_{kn}) \hat{v}_{kn}^\alpha \} \right. \right. \\ &\quad + (\delta_{\ell k} - \delta_{\ell i}) \frac{\underline{r}_{ij} \cdot \hat{\underline{u}}_{kn}}{u_{kn} r_{ik}} \left\{ r_{ij}^\alpha - (\underline{r}_{ij} \cdot \hat{\underline{u}}_{kn}) \hat{u}_{kn}^\alpha - (\underline{r}_{ij} \cdot \underline{r}_{ik} - (\underline{r}_{ij} \cdot \hat{\underline{u}}_{kn})(\underline{r}_{ik} \cdot \hat{\underline{u}}_{kn})) \frac{r_{ik}^\alpha}{r_{ik}^2} \right\} \\ &\quad + (\delta_{\ell k} - \delta_{\ell i}) \frac{\underline{r}_{ij} \cdot \hat{\underline{v}}_{kn}}{v_{kn} r_{ik}} \left\{ r_{ij}^\alpha - (\underline{r}_{ij} \cdot \hat{\underline{v}}_{kn}) \hat{v}_{kn}^\alpha - (\underline{r}_{ij} \cdot \underline{r}_{ik} - (\underline{r}_{ij} \cdot \hat{\underline{v}}_{kn})(\underline{r}_{ik} \cdot \hat{\underline{v}}_{kn})) \frac{r_{ik}^\alpha}{r_{ik}^2} \right\} \\ &\quad + (\delta_{\ell n} - \delta_{\ell i}) \frac{\underline{r}_{ij} \cdot \hat{\underline{u}}_{kn}}{u_{kn} r_{in}} \left\{ r_{ij}^\alpha - (\underline{r}_{ij} \cdot \hat{\underline{u}}_{kn}) \hat{u}_{kn}^\alpha - (\underline{r}_{ij} \cdot \underline{r}_{in} - (\underline{r}_{ij} \cdot \hat{\underline{u}}_{kn})(\underline{r}_{in} \cdot \hat{\underline{u}}_{kn})) \frac{r_{in}^\alpha}{r_{in}^2} \right\} \\ &\quad \left. \left. + (\delta_{\ell n} - \delta_{\ell i}) \frac{\underline{r}_{ij} \cdot \hat{\underline{v}}_{kn}}{v_{kn} r_{in}} \left\{ r_{ij}^\alpha - (\underline{r}_{ij} \cdot \hat{\underline{v}}_{kn}) \hat{v}_{kn}^\alpha - (\underline{r}_{ij} \cdot \underline{r}_{in} - (\underline{r}_{ij} \cdot \hat{\underline{v}}_{kn})(\underline{r}_{in} \cdot \hat{\underline{v}}_{kn})) \frac{r_{in}^\alpha}{r_{in}^2} \right\} \right] \right\} \end{aligned} \quad (2.60)$$

This general formula applies to all atoms $\ell = i, j, k, n$. It must be remembered however, that these formulae apply to just one of the three contributing terms (i.e. one angle ϕ) of the full inversion potential: specifically the inversion angle pertaining to the out-of-plane vector $\underline{r}_{ij}$. The contributions arising from the other vectors $\underline{r}_{ik}$ and $\underline{r}_{in}$ are obtained by the cyclic permutation of the indices in the manner described above. All these force contributions must be added to the final atomic forces.

Formally, the contribution to be added to the atomic virial is given by

$$\mathcal{W} = -\sum_{i=1}^4 \underline{r}_i \cdot \underline{f}_i \quad (2.61)$$

However it is possible to show by thermodynamic arguments (*cf* [26],) or simply from the fact that the sum of forces on atoms j, k and n is equal and opposite to the force on atom i , that the inversion potential makes *no* contribution to the atomic virial.

If the force components f_ℓ^α for atoms $\ell = i, j, k, n$ are calculated using the above formulae, it is easily seen that the contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^\alpha f_j^\beta + r_{ik}^\alpha f_k^\beta + r_{in}^\alpha f_n^\beta \quad (2.62)$$

The sum of the diagonal elements of the stress tensor is zero (since the virial is zero) and the matrix is symmetric.

In DL_POLY_2 inversion forces are handled by the routine INVFRG.

2.2.8 Tethering Forces

DL_POLY_2 also allows atomic sites to be tethered to a fixed point in space, $\underline{r}_0$ taken as their position at the beginning of the simulation. This is also known as position restraining. The specification, which comes as part of the molecular description, requires a tether potential type and the associated interaction parameters.

Note, firstly, that application of tethering potentials means that momentum will no longer be a conserved quantity of the simulation. Secondly, in constant pressure simulations, where the MD cell changes size or shape, the reference position is scaled with the cell vectors.

The potential functions available in DL_POLY_2 are as follows, in each case r_{i0} is the distance of the atom from its position at $t = 0$:

1. harmonic potential: (**harm**)

$$U(r_{i0}) = \frac{1}{2}k(r_{i0})^2; \quad (2.63)$$

2. restrained harmonic :(**rharm**)

$$U(r_{i0}) = \frac{1}{2}k(r_{i0})^2 \quad r_{i0} \leq r_c; \quad (2.64)$$

$$U(r_{i0}) = \frac{1}{2}kr_c^2 + kr_c(r_{i0} - r_c) \quad r_{i0} > r_c; \quad (2.65)$$

3. Quartic potential: (**quar**)

$$U(r_{i0}) = \frac{k}{2}(r_{i0})^2 + \frac{k'}{3}(r_{i0})^3 + \frac{k''}{4}(r_{i0})^4. \quad (2.66)$$

The force on the atom i arising from a tether potential is obtained using the general formula:

$$\underline{f}_i = -\frac{1}{r_{i0}} \left[\frac{\partial}{\partial r_{i0}} U(r_{i0}) \right] \underline{r}_{i0}, \quad (2.67)$$

The contribution to be added to the atomic virial is given by

$$\mathcal{W} = \underline{r}_{i0} \cdot \underline{f}_i, \quad (2.68)$$

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = -r_{i0}^{\alpha} f_i^{\beta}, \quad (2.69)$$

where α and β indicate the x, y, z components. The atomic stress tensor derived in this way is symmetric.

In DL_POLY_2 bond forces are handled by the routine TETHFRC.

2.2.9 Frozen Atoms

DL_POLY_2 also allows atoms to be completely immobilised (*i.e.* “frozen” at a fixed point in the MD cell). This is achieved by setting all forces and velocities associated with that atom to zero during each MD timestep. Frozen atoms are signalled by assigning an atom a non-zero value for the freeze parameter in the FIELD file. DL_POLY_2 does not calculate contributions to the virial or the stress tensor arising from the constraints required to freeze atomic positions. In DL_POLY_2 the frozen atom option cannot be used for sites in a rigid body. As with the tethering potential, the reference position is scaled with the cell vectors in constant pressure simulations.

In DL_POLY_2 the frozen atom option is handled by the subroutine FREEZE.

2.3 The Intermolecular Potential Functions

In this section we outline the pair-body, three-body and four-body potential functions available in DL_POLY_2. An important distinction between these and intramolecular (bond) forces in DL_POLY_2 is that they are specified by *atom types* rather than atom indices.

2.3.1 Short Ranged (van der Waals) Potentials

The short ranged pair forces available in DL_POLY_2 are as follows.

1. 12 - 6 potential: **(12-6)**

$$U(r_{ij}) = \left(\frac{A}{r_{ij}^{12}} \right) - \left(\frac{B}{r_{ij}^6} \right); \quad (2.70)$$

2. Lennard-Jones: **(lj)**

$$U(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^6 \right]; \quad (2.71)$$

3. n - m potential [27]: **(nm)**

$$U(r_{ij}) = \frac{E_o}{(n-m)} \left[m \left(\frac{r_o}{r_{ij}} \right)^n - n \left(\frac{r_o}{r_{ij}} \right)^m \right]; \quad (2.72)$$

4. Buckingham potential: (**buck**)

$$U(r_{ij}) = A \exp\left(-\frac{r_{ij}}{\rho}\right) - \frac{C}{r_{ij}^6}; \quad (2.73)$$

5. Born-Huggins-Meyer potential: (**bhm**)

$$U(r_{ij}) = A \exp[B(\sigma - r_{ij})] - \frac{C}{r_{ij}^6} - \frac{D}{r_{ij}^8}; \quad (2.74)$$

6. Hydrogen-bond (12 - 10) potential: (**hbnd**)

$$U(r_{ij}) = \left(\frac{A}{r_{ij}^{12}}\right) - \left(\frac{B}{r_{ij}^{10}}\right); \quad (2.75)$$

7. Shifted force n - m potential [27]: (**snm**)

$$\begin{aligned} U(r_{ij}) = & \frac{\alpha E_o}{(n-m)} \left[m\beta^n \left\{ \left(\frac{r_o}{r_{ij}}\right)^n - \left(\frac{1}{\gamma}\right)^n \right\} - n\beta^m \left\{ \left(\frac{r_o}{r_{ij}}\right)^m - \left(\frac{1}{\gamma}\right)^m \right\} \right] \\ & + \frac{nm\alpha E_o}{(n-m)} \left(\frac{r_{ij} - \gamma r_o}{\gamma r_o} \right) \left\{ \left(\frac{\beta}{\gamma}\right)^n - \left(\frac{\beta}{\gamma}\right)^m \right\} \end{aligned} \quad (2.76)$$

with

$$\gamma = \frac{r_{cut}}{r_o} \quad (2.77)$$

$$\beta = \gamma \left(\frac{\gamma^{m+1} - 1}{\gamma^{n+1} - 1} \right)^{\frac{1}{n-m}} \quad (2.78)$$

$$\alpha = \frac{(n-m)}{[n\beta^m(1 + (m/\gamma - m - 1)/\gamma^m) - m\beta^n(1 + (n/\gamma - n - 1)/\gamma^n)]} \quad (2.79)$$

This peculiar form has the advantage over the standard shifted n-m potential in that both E_o and r_o (well depth and location of minimum) retain their original values after the shifting process.

8. Tabulation: (**tab**). The potential is defined numerically only.

The parameters defining these potentials are supplied to DL_POLY_2 at run time (see the description of the FIELD file in section 4.1.3). Each atom type in the system is specified by a unique eight-character label defined by the user. The pair potential is then defined internally by the combination of two atom labels.

As well as the numerical parameters defining the potentials, DL_POLY_2 must also be provided with a cutoff radius r_{cut} , which sets a range limit on the computation of the interaction. Together with the parameters, the cutoff is used by the subroutine FORGEN (or FORGEN_RSQ) to construct an interpolation array **vvv** for the potential function over

the range 0 to r_{cut} . A second array **ggg** is also calculated, which is related to the potential via the formula:

$$G(r_{ij}) = -r_{ij} \frac{\partial}{\partial r_{ij}} U(r_{ij}), \quad (2.80)$$

and is used in the calculation of the forces. Both arrays are tabulated in units of energy. The use of interpolation arrays, rather than the explicit formulae, makes the routines for calculating the potential energy and atomic forces very general, and enables the use of user defined pair potential functions. DL_POLY_2 also allows the user to read in the interpolation arrays directly from a file (see the description of the FORTAB routine (chapter 8) and the TABLE file (section 4.1.5). This is particularly useful if the pair potential function has no simple analytical description (e.g. spline potentials).

The force on an atom j derived from one of these potentials is formally calculated with the standard formula:

$$\underline{f}_j = -\frac{1}{r_{ij}} \left[\frac{\partial}{\partial r_{ij}} U(r_{ij}) \right] \underline{r}_{ij}, \quad (2.81)$$

where $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$. The force on atom i is the negative of this.

The contribution to be added to the atomic virial (for each pair interaction) is

$$\mathcal{W} = -\underline{r}_{ij} \cdot \underline{f}_j. \quad (2.82)$$

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta}, \quad (2.83)$$

where α and β indicate the x, y, z components. The atomic stress tensor derived from the pair forces is symmetric.

Since the calculation of pair potentials assumes a spherical cutoff (r_{cut}) it is necessary to apply a *long range correction* to the system potential energy and virial. Explicit formulae are needed for each case and are derived as follows. For two atom types a and b , the correction for the potential energy is calculated via the integral

$$U_{corr}^{ab} = 2\pi \frac{N_a N_b}{V} \int_{r_{cut}}^{\infty} g_{ab}(r) U_{ab}(r) r^2 dr \quad (2.84)$$

where N_a, N_b are the numbers of atoms of types a and b , V is the system volume and $g_{ab}(r)$ and $U_{ab}(r)$ are the appropriate pair correlation function and pair potential respectively. It is usual to assume $g_{ab}(r) = 1$ for $r > r_{cut}$. DL_POLY_2 sometimes makes the additional assumption that the repulsive part of the short ranged potential is negligible beyond r_{cut} .

The correction for the system virial is

$$\mathcal{W}_{corr}^{ab} = -2\pi \frac{N_a N_b}{V} \int_{r_{cut}}^{\infty} g_{ab}(r) \frac{\partial}{\partial r} U_{ab}(r) r^3 dr, \quad (2.85)$$

where the same approximations are applied. Note that these formulae are based on the assumption that the system is reasonably isotropic beyond the cutoff.

In DL_POLY_2 the short ranged forces are calculated by one of the routines SRFRCE, SRFRCE_RSQ, and SRFRCE_NEU. The long range corrections are calculated by routine LR-CORRECT. The calculation makes use of the Verlet neighbour list described above.

2.3.2 Metal Potentials

DL_POLY_2 includes density dependent potentials suitable for calculating the properties of metals. The basic model is due to Finnis and Sinclair [28] as implemented by Sutton and Chen [6]. The form of the potential is: (**stch**)

$$U_{sc} = \epsilon \left[\sum_{i < j} \left(\frac{a}{r_{ij}} \right)^n - C \sum_i \rho_i^{1/2} \right], \quad (2.86)$$

where the *local density* ρ_i is given by

$$\rho_i = \sum_j \left(\frac{a}{r_{ij}} \right)^m \quad (2.87)$$

The Sutton-Chen potential has the advantage that it is decomposable into pair contributions and thus falls within the general tabulation scheme of DL_POLY_2, where it is treated as a short ranged interaction. The same form of potential may be used in alloys, through the appropriate choice of parameters ϵ and a . The parameters n and m however must be the same for all component elements. DL_POLY_2 calculates this potential in two stages: the first calculates the local density ρ_i for each atom; and the second calculates the potential energy and forces. Interpolation arrays are used in both these stages.

The total force $\underline{f}_j^{tot}$ on an atom j derived from this potential is calculated in the standard way:

$$\underline{f}_j^{tot} = -\nabla_j U_{sc}, \quad (2.88)$$

which gives

$$\underline{f}_j^{tot} = -\epsilon \sum_{i \neq j} \left[n \left(\frac{a}{r_{ij}} \right)^n - \frac{Cm}{2} (\rho_j^{-1/2} + \rho_i^{-1/2}) \left(\frac{a}{r_{ij}} \right)^m \right] \left(\frac{1}{r_{ij}^2} \right) \underline{r}_{ij}, \quad (2.89)$$

which is recognisable as a sum of pair forces, for example the force on atom j due to the atom i :

$$\underline{f}_j = -\epsilon \left[n \left(\frac{a}{r_{ij}} \right)^n - \frac{Cm}{2} (\rho_j^{-1/2} + \rho_i^{-1/2}) \left(\frac{a}{r_{ij}} \right)^m \right] \left(\frac{1}{r_{ij}^2} \right) \underline{r}_{ij}, \quad (2.90)$$

where $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$. The force on atom i is the negative of this.

With the pair forces thus defined the contribution to be added to the atomic virial *from each atom pair* is then

$$\mathcal{W} = -\underline{r}_{ij} \cdot \underline{f}_j. \quad (2.91)$$

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^\alpha f_j^\beta, \quad (2.92)$$

where α and β indicate the x, y, z components. The atomic stress tensor is symmetric.

The long range correction for the system potential is in two parts. Firstly by analogy with the short ranged potentials the correction to the local density is obtained by

$$\rho_i = \rho_i^o + 4\pi\bar{\rho} \int_{r_{cut}}^{\infty} \left(\frac{a}{r}\right)^m r^2 dr, \quad (2.93)$$

where ρ_i^o is the uncorrected local density and $\bar{\rho}$ is the *mean particle density*. Evaluating the integral yields

$$\delta\rho = 4\pi \frac{\bar{\rho}a^3}{(m-3)} \left(\frac{a}{r_{cut}}\right)^{m-3} \quad (2.94)$$

which is the local density correction and is identical for all atoms. The correction is applied immediately after the local density is calculated. The density term of the Sutton Chen potential needs no further correction. The pair term correction is obtained by analogy with the short ranged potentials and is

$$U_{corr} = 2\pi \frac{N\epsilon\bar{\rho}a^3}{(n-3)} \left(\frac{a}{r_{cut}}\right)^{n-3}. \quad (2.95)$$

The correction to the local density having already been applied.

To estimate the virial correction we assume the corrected local densities are constants (i.e. independent of distance - at least beyond the range r_{cut}). This allows the virial correction to be computed by the methods used in the short ranged potentials. The result is:

$$\mathcal{W}_{corr} = -2\pi\bar{\rho}a^3 \left\{ \frac{nN\epsilon}{(n-3)} \left(\frac{a}{r_{cut}}\right)^{n-3} - \frac{m\epsilon^2C^2}{(m-3)} \left(\frac{a}{r_{cut}}\right)^{m-3} \sum_i^N \rho_i^{-1/2} \right\} \quad (2.96)$$

This correction may be used as it stands, or with the further approximation:

$$\sum_i^N \rho_i^{-1/2} = \frac{N}{\langle \rho_i^{1/2} \rangle} \quad (2.97)$$

where $\langle \rho_i^{1/2} \rangle$ is regarded as a constant of the system.

In DL_POLY_2 the metal forces are handled by the routine SUTTCHEM. The local density is calculated by routines SCDENS and DENLOC. The long range corrections are calculated by LRCMETAL.

2.3.3 Three Body Potentials

The three-body potentials in DL_POLY_2 are mostly valence angle forms. (They are primarily included to permit simulation of amorphous materials e.g. silicate glasses.) However, these have been extended to include the Dreiding [7] hydrogen bond. The potential forms available are as follows.

1. Truncated harmonic: (**thrm**)

$$U(\theta_{jik}) = \frac{k}{2} (\theta_{jik} - \theta_0)^2 \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]; \quad (2.98)$$

2. Screened Harmonic: (**shrm**)

$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]; \quad (2.99)$$

3. Screened Vessal[24]: (**bvs1**)

$$U(\theta_{jik}) = \frac{k}{8(\theta_{jik} - \pi)^2} \left\{ [(\theta_0 - \pi)^2 - (\theta_{jik} - \pi)^2]^2 \right\} \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]; \quad (2.100)$$

4. Truncated Vessal[25]: (**bvs2**)

$$U(\theta_{jik}) = k[\theta_{jik}^a(\theta_{jik} - \theta_0)^2(\theta_{jik} + \theta_0 - 2\pi)^2 - \frac{a}{2}\pi^{a-1}(\theta_{jik} - \theta_0)^2(\pi - \theta_0)^3] \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]. \quad (2.101)$$

5. Dreiding hydrogen bond [7]: (**hbnd**)

$$U(\theta_{jik}) = D_{hb} \cos^4(\theta_{jik}) [5(R_{hb}/r_{jk})^{12} - 6(R_{hb}/r_{jk})^{10}] \quad (2.102)$$

Note that for the hydrogen bond, the hydrogen atom *must* be the central atom. Several of these functions are identical to those appearing in the *intra*-molecular valence angle descriptions above. There are significant differences in implementation however, arising from the fact that the three-body potentials are regarded as *inter*-molecular. Firstly, the atoms involved are defined by atom types, not specific indices. Secondly, there are *no* excluded atoms arising from the three body terms. (The inclusion of pair potentials may in fact be essential to maintain the structure of the system.)

The three body potentials are very short ranged, typically of order 3 Å. This property, plus the fact that three body potentials scale as N^3 , where N is the number of particles, makes it essential that these terms are calculated by the link-cell method [29].

The calculation of the forces, virial and stress tensor as described in the section valence angle potentials above.

DL_POLY_2 applies no long range corrections to the three body potentials. The three body forces are calculated by the routine THBFRG.

2.3.4 Four Body Potentials

The four-body potentials in DL_POLY_2 are entirely inversion angle forms, primarily included to permit simulation of amorphous materials (particularly borate glasses). The potential forms available in DL_POLY_2 are as follows.

1. Harmonic: (**harm**)

$$U(\phi_{ijkn}) = \frac{1}{2}k(\phi_{ijkn} - \phi_0)^2 \quad (2.103)$$

2. Harmonic cosine: (**hcos**)

$$U(\phi_{ijkn}) = \frac{k}{2}(\cos(\phi_{ijkn}) - \cos(\phi_0))^2 \quad (2.104)$$

3. Planar potential: (**plan**)

$$U(\phi_{ijkn}) = A[1 - \cos(\phi_{ijkn})] \quad (2.105)$$

These functions are identical to those appearing in the *intra*-molecular inversion angle descriptions above. There are significant differences in implementation however, arising from the fact that the four-body potentials are regarded as *inter*-molecular. Firstly, the atoms involved are defined by atom types, not specific indices. Secondly, there are *no* excluded atoms arising from the four-body terms. (The inclusion of other potentials, for example pair potentials, may in fact be essential to maintain the structure of the system.)

The four body potentials are very short ranged, typically of order 3 Å. This property, plus the fact that four body potentials scale as N^4 , where N is the number of particles, makes it essential that these terms are calculated by the link-cell method [29].

The calculation of the forces, virial and stress tensor described in the section on inversion angle potentials above.

DL_POLY_2 applies no long range corrections to the four body potentials. The four-body forces are calculated by the routine FBPFRC.

2.3.5 External Fields

In addition to the molecular force field, DL_POLY_2 allows the use of an *external* force field. Examples of field available include:

1. Electric field: (**elec**)

$$\underline{F}_i = \underline{F}_i + q_i \cdot \underline{H} \quad (2.106)$$

2. Oscillating shear: (**oshm**)

$$\underline{F}_x = A \cos(2n\pi \cdot z / L_z) \quad (2.107)$$

3. Continuous shear: (**shrx**)

$$\underline{v}_x = \frac{1}{2} A \frac{|z|}{z} \quad : |z| > z_0 \quad (2.108)$$

4. Gravitational field: (**grav**)

$$\underline{F}_i = \underline{F}_i + m_i \cdot \underline{H} \quad (2.109)$$

5. Magnetic field: (**magn**)

$$\underline{F}_i = \underline{F}_i + q_i \cdot (\underline{v}_i \wedge \underline{H}) \quad (2.110)$$

6. Containing sphere: (**sphr**)

$$\underline{F} = A(R_0 - r)^{-n} \quad : r > R_{cut} \quad (2.111)$$

7. Repulsive wall: (**z***bnd*)

$$\underline{F} = A(z_o - z) \quad : z > z_o \quad (2.112)$$

It is recommended that the use of an external field should be accompanied by a thermostat (this does not apply to examples 6 and 7, since these are conservative fields). The user is advised to be careful with units!

In DL_POLY_2 external field forces are handled by the routine EXTNFLD.

2.4 Long Ranged Electrostatic (Coulombic) Potentials

DL_POLY_2 incorporates several techniques for dealing with long ranged electrostatic potentials.² These are as follows.

1. Atomistic and charge group implementation.
2. Direct Coulomb sum;
3. Truncated and shifted Coulomb sum;
4. Coulomb sum with distance dependent dielectric;
5. Ewald sum;
6. Smooth Particle Mesh Ewald (SPME);
7. Hautman Klein Ewald for systems with 2D periodicity;
8. Reaction field;
9. Dynamical shell model.

Some of these techniques can be combined. For example 1, 3 and 4 can be used in conjunction with 7. The Ewald sum, SPME and Hautman Klein Ewald are restricted to periodic (or pseudo-periodic) systems only, though DL_POLY_2 can handle a broad selection of periodic boundary conditions, including cubic, orthorhombic, parallelepiped, truncated octahedral, hexagonal prism and rhombic dodecahedral. The Ewald sum is the method of choice for periodic systems. The other techniques can be used with either periodic or non-periodic systems, though in the case of the direct Coulomb sum, there are likely to be problems with convergence.

DL_POLY_2 will correctly handle the electrostatics of both molecular and atomic species. However it is assumed that the system is electrically neutral. A warning message is printed if the system is found to be charged, but otherwise the simulation proceeds as normal. No correction for non-neutrality is applied.

²Unlike the other elements of the force field, the electrostatic forces are NOT specified in the input FIELD file, but by setting appropriate directives in the CONTROL file. See section 4.1.1.

2.4.1 Atomistic and Charge Group Implementation

The Ewald sum is an accurate method for summing long-ranged Coulomb potentials in periodic systems. This can be a very cpu intensive calculation and the use of more efficient, but less accurate methods, is common. Invariably this involves truncation of the potential at some finite distance r_{cut} . If an atomistic scheme is used for the truncation criterion there is no guarantee that the interaction sphere will be neutral and spurious “charging” effects will almost certainly be seen in a simulation. This arises because the potential being truncated is long-ranged ($1/r$ for charge-charge interactions). However if the cutoff scheme is based on *neutral* groups of atoms, then at worst, at long distance the interaction will be a dipole-dipole interaction and vary as $1/r^3$. The truncation effects at the cutoff are therefore much less severe than if an atomistic scheme is used. In DL_POLY_2 the interaction is evaluated between all atoms of both groups if any site of the first group is within the cutoff distance of any site of the second group. The groups are known interchangeably as “charge groups” or “neutral groups” in the documentation - which serves as a reminder that the advantages of using such a scheme are lost if the groups carry an overall charge. There is no formal requirement in DL_POLY_2 that the groups actually be electrically neutral.

The charge group scheme is more cpu intensive than a simple atomistic cutoff scheme as more computation is required to determine whether or not to include a set of interactions. However the size of the Verlet neighbourhood list (easily the largest array in DL_POLY_2) is considerably smaller with a charge group scheme than an atomistic scheme as only a list of interacting groups need be stored as opposed to a list of interacting atoms.

2.4.2 Direct Coulomb Sum

Use of the direct Coulomb sum is sometimes necessary for accurate simulation of isolated (nonperiodic) systems. It is *not* recommended for periodic systems.

The interaction potential for two charged ions is

$$U(r_{ij}) = \frac{1}{4\pi\epsilon_0} \frac{q_i q_j}{r_{ij}} \quad (2.113)$$

with q_ℓ the charge on an atom labelled ℓ , and r_{ij} the magnitude of the separation vector $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$.

The force on an atom j derived from this force is

$$\underline{f}_j = \frac{1}{4\pi\epsilon_0} \frac{q_i q_j}{r_{ij}^3} \underline{r}_{ij} \quad (2.114)$$

with the force on atom i the negative of this.

The contribution to the atomic virial is

$$\mathcal{W} = -\frac{1}{4\pi\epsilon_0} \frac{q_i q_j}{r_{ij}} \quad (2.115)$$

which is simply the negative of the potential term.

The contribution to be added to the atomic stress tensor is

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta}, \quad (2.116)$$

where α, β are x, y, z components. The atomic stress tensor is symmetric.

In DL_POLY_2 these forces are handled by the routines COUL0 and COUL0NEU.

2.4.3 Truncated and Shifted Coulomb Sum

This form of the Coulomb sum has the advantage that it drastically reduces the range of electrostatic interactions, without giving rise to a violent step in the potential energy at the cutoff. Its main use is for preliminary preparation of systems and it is not recommended for realistic models.

The form of the potential function is

$$U(r_{ij}) = \frac{q_i q_j}{4\pi\epsilon_0} \left\{ \frac{1}{r_{ij}} - \frac{1}{r_{cut}} \right\} \quad (2.117)$$

with q_ℓ the charge on an atom labelled ℓ , r_{cut} the cutoff radius and r_{ij} the magnitude of the separation vector $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$.

The force on an atom j derived from this potential, within the radius r_{cut} , is

$$\underline{f}_j = \frac{1}{4\pi\epsilon_0} \frac{q_i q_j}{r_{ij}^3} \underline{r}_{ij} \quad (2.118)$$

with the force on atom i the negative of this.

The contribution to the atomic virial is

$$\mathcal{W} = -\underline{r}_{ij} \cdot \underline{f}_j \quad (2.119)$$

which is *not* the negative of the potential term in this case.

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta}, \quad (2.120)$$

where α, β are x, y, z components. The atomic stress tensor is symmetric.

In DL_POLY_2 these forces are handled by the routine COUL1.

A further refinement of this approach is to truncate the $1/r$ potential at r_{cut} and add a linear term to the potential in order to make both the energy and the force zero at the cutoff. The potential is thus

$$U(r_{ij}) = \frac{q_i q_j}{4\pi\epsilon_0} \left[\frac{1}{r_{ij}} + \frac{r_{ij}}{r_{cut}^2} - \frac{2}{r_{cut}} \right] \quad (2.121)$$

with the force on atom j given by

$$\underline{f}_j = \frac{q_i q_j}{4\pi\epsilon_0} \left[\frac{1}{r_{ij}^2} - \frac{1}{r_{cut}^2} \right] \underline{r}_{ij} \quad (2.122)$$

with the force on atom i the negative of this.

This removes the heating effects that arise from the discontinuity in the forces at the cutoff in the simple truncated and shifted potential. The physics of this potential however are little better. It is only recommended for very crude structure optimizations.

The contribution to the atomic virial is

$$\mathcal{W} = -\underline{r}_{ij} \cdot \underline{f}_j \quad (2.123)$$

which is *not* the negative of the potential term.

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta}, \quad (2.124)$$

where α, β are x, y, z components. The atomic stress tensor is symmetric.

In DL_POLY_2 these forces are handled by the routine COUL4.

2.4.4 Coulomb Sum with Distance Dependent Dielectric

This potential attempts to address the difficulties of applying the direct Coulomb sum, without the brutal truncation of the previous case. It hinges on the assumption that the electrostatic forces are effectively ‘screened’ in real systems - an effect which is approximated by introducing a dielectric term that increases with distance.

The interatomic potential for two charged ions is

$$U(r_{ij}) = \frac{1}{4\pi\epsilon_0\epsilon(r_{ij})} \frac{q_i q_j}{r_{ij}} \quad (2.125)$$

with q_ℓ the charge on an atom labelled ℓ , and r_{ij} the magnitude of the separation vector $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$. $\epsilon(r)$ is the distance dependent dielectric function. In DL_POLY_2 it is assumed that this function has the form

$$\epsilon(r) = \epsilon r \quad (2.126)$$

where ϵ is a constant. Inclusion of this term effectively accelerates the rate of convergence of the Coulomb sum.

The force on an atom j derived from this potential is

$$\underline{f}_j = \frac{1}{2\pi\epsilon_0\epsilon} \frac{q_i q_j}{r_{ij}^4} \underline{r}_{ij} \quad (2.127)$$

with the force on atom i the negative of this.

The contribution to the atomic virial is

$$\mathcal{W} = -\underline{r}_{ij} \cdot \underline{f}_j \quad (2.128)$$

which is -2 times the potential term.

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta}, \quad (2.129)$$

where α, β are x, y, z components. The atomic stress tensor is symmetric.

In DL_POLY_2 these forces are handled by the routines COUL2 and COUL2NEU.

2.4.5 Ewald Sum

The Ewald sum [10] is the best technique for calculating electrostatic interactions in a periodic (or pseudo-periodic) system.

The basic model for a neutral periodic system is a system of charged point ions mutually interacting via the Coulomb potential. The Ewald method makes two amendments to this simple model. Firstly each ion is effectively neutralised (at long range) by the superposition of a spherical gaussian cloud of opposite charge centred on the ion. The combined assembly of point ions and gaussian charges becomes the *Real Space* part of the Ewald sum, which is now short ranged and treatable by the methods described above (section 2.1).³ The second modification is to superimpose a second set of gaussian charges, this time with the same charges as the original point ions and again centred on the point ions (so nullifying the effect of the first set of gaussians). The potential due to these gaussians is obtained from Poisson's equation and is solved as a Fourier series in *Reciprocal Space*. The complete Ewald sum requires an additional correction, known as the self energy correction, which arises from a gaussian acting on its own site, and is constant. Ewald's method therefore replaces a potentially infinite sum in real space by two finite sums: one in real space and one in reciprocal space; and the self energy correction.

For molecular systems, as opposed to systems comprised simply of point ions, additional modifications are necessary to correct for the excluded (intra-molecular) Coulombic interactions. In the real space sum these are simply omitted. In reciprocal space however, the effects of individual gaussian charges cannot easily be extracted, and the correction is made in real space. It amounts to removing terms corresponding to the potential energy of an ion ℓ due to the gaussian charge on a neighbouring charge m (or *vice versa*). This correction appears as the final term in the full Ewald formula below. The distinction between the error function *erf* and the more usual complementary error function *erfc* found in the real space sum, should be noted.

The total electrostatic energy is given by the following formula.

$$\begin{aligned}
 U_c = & \frac{1}{2V_o\epsilon_0} \sum_{\underline{k} \neq \underline{0}}^{\infty} \frac{\exp(-k^2/4\alpha^2)}{k^2} \left| \sum_j^N q_j \exp(-i\underline{k} \cdot \underline{r}_j) \right|^2 + \frac{1}{4\pi\epsilon_0} \sum_{n < j}^{N^*} \frac{q_j q_n}{r_{nj}} \text{erfc}(\alpha r_{nj}) \\
 & - \frac{1}{4\pi\epsilon_0} \sum_{\text{molecules}} \sum_{\ell \leq m}^{M^*} q_\ell q_m \left\{ \delta_{\ell m} \frac{\alpha}{\sqrt{\pi}} + \frac{\text{erf}(\alpha r_{\ell m})}{r_{\ell m}^{1-\delta_{\ell m}}} \right\}, \quad (2.130)
 \end{aligned}$$

where N is the number of ions in the system and N^* the same number discounting any excluded (intramolecular) interactions. M^* represents the number of excluded atoms in a given molecule and includes the atomic self correction. V_o is the simulation cell volume and $\underline{k}$ is a reciprocal lattice vector defined by

$$\underline{k} = \ell \underline{u} + m \underline{v} + n \underline{w} \quad (2.131)$$

³Strictly speaking, the real space sum ranges over all periodic images of the simulation cell, but in the DL_POLY_2 implementation, the parameters are chosen to restrict the sum to the simulation cell and its nearest neighbours i.e. the *minimum images* of the cell contents.

where ℓ, m, n are integers and $\underline{u}, \underline{v}, \underline{w}$ are the *reciprocal space* basis vectors. Both V_o and $\underline{u}, \underline{v}, \underline{w}$ are derived from the vectors $(\underline{a}, \underline{b}, \underline{c})$ defining the simulation cell. Thus

$$V_o = |\underline{a} \cdot \underline{b} \times \underline{c}| \quad (2.132)$$

and

$$\begin{aligned} \underline{u} &= 2\pi \frac{\underline{b} \times \underline{c}}{\underline{a} \cdot \underline{b} \times \underline{c}} \\ \underline{v} &= 2\pi \frac{\underline{c} \times \underline{a}}{\underline{a} \cdot \underline{b} \times \underline{c}} \\ \underline{w} &= 2\pi \frac{\underline{a} \times \underline{b}}{\underline{a} \cdot \underline{b} \times \underline{c}}. \end{aligned} \quad (2.133)$$

With these definitions, the Ewald formula above is applicable to general periodic systems. (A small additional modification is necessary for rhombic dodecahedral and truncated octahedral simulation cells [30].)

In practice the convergence of the Ewald sum is controlled by three variables: the real space cutoff r_{cut} ; the convergence parameter α and the largest reciprocal space vector $\underline{k}_{max}$ used in the reciprocal space sum. These are discussed more fully in section 3.3.5. DL_POLY_2 can provide estimates if requested (see CONTROL file description 4.1.1).

The force on an atom j is obtained by differentiation and is

$$\begin{aligned} \underline{f}_j &= -\frac{q_j}{V_o \epsilon_0} \sum_{\underline{k} \neq \underline{0}}^{\infty} i \underline{k} \exp(\underline{k} \cdot \underline{r}_j) \frac{\exp(-k^2/4\alpha^2)}{k^2} \sum_n^N q_n \exp(-i \underline{k} \cdot \underline{r}_n) \\ &\quad + \frac{q_j}{4\pi \epsilon_0} \sum_n^{N^*} \frac{q_n}{r_{nj}^3} \left\{ \operatorname{erfc}(\alpha r_{nj}) + \frac{2\alpha r_{nj}}{\sqrt{\pi}} \exp(-\alpha^2 r_{nj}^2) \right\} \underline{r}_{nj} \\ &\quad - \frac{q_j}{4\pi \epsilon_0} \sum_{\ell}^{M^*} \frac{q_{\ell}}{r_{\ell j}^3} \left\{ \operatorname{erf}(\alpha r_{\ell j}) - \frac{2\alpha r_{\ell j}}{\sqrt{\pi}} \exp(-\alpha^2 r_{\ell j}^2) \right\} \underline{r}_{\ell j} \end{aligned} \quad (2.134)$$

The electrostatic contribution to the system virial can be obtained as the negative of the Coulombic energy. However in DL_POLY_2 this formal equality can be used as a check on the convergence of the Ewald sum. The actual electrostatic virial is obtained during the calculation of the diagonal of the stress tensor.

The electrostatic contribution to the stress tensor is given by

$$\begin{aligned} \underline{\underline{\sigma}} &= \frac{1}{2V_o \epsilon_0} \sum_{\underline{k} \neq \underline{0}}^{\infty} \left\{ \underline{\underline{1}} - 2 \left(\frac{1}{4\alpha^2} + \frac{1}{k^2} \right) \underline{\underline{K}} \right\} \frac{\exp(-k^2/4\alpha^2)}{k^2} \left| \sum_j^N q_j \exp(-i \underline{k} \cdot \underline{r}_j) \right|^2 \\ &\quad + \frac{1}{4\pi \epsilon_0} \sum_{j < n}^{N^*} \frac{q_j q_n}{r_{nj}^3} \left\{ \operatorname{erfc}(\alpha r_{nj}) + \frac{2\alpha r_{nj}}{\sqrt{\pi}} \exp(-\alpha^2 r_{nj}^2) \right\} \underline{\underline{R}}_{nj} \\ &\quad - \frac{1}{4\pi \epsilon_0} \sum_{j < \ell}^{M^*} \frac{q_j q_{\ell}}{r_{\ell j}^3} \left\{ \operatorname{erf}(\alpha r_{\ell j}) - \frac{2\alpha r_{\ell j}}{\sqrt{\pi}} \exp(-\alpha^2 r_{\ell j}^2) \right\} \underline{\underline{R}}_{\ell j}, \end{aligned} \quad (2.135)$$

where matrices $\underline{\underline{\mathbf{K}}}$ and $\underline{\underline{\mathbf{R}}}_{\ell j}$ are defined as follows.

$$K^{\alpha\beta} = k^\alpha k^\beta \quad (2.136)$$

$$R_{\ell j}^{\alpha\beta} = r_{\ell j}^\alpha r_{\ell j}^\beta \quad (2.137)$$

In DL_POLY_2 the full Ewald sum is handled by several routines: EWALD1 and EWALD1A handle the reciprocal space terms; EWALD2, EWALD2_2PT, EWALD2_RSQ and EWALD4, EWALD4_2PT handle the real space terms (with the same Verlet neighbour list routines that are used to calculate the short range forces); and EWALD3 calculates the self interaction corrections. It should be noted that the Ewald potential and force interpolation arrays in DL_POLY_2 are `erc` and `fer` respectively.

2.4.6 Smooth Particle Mesh Ewald

As its name implies the Smooth Particle Mesh Ewald (SPME) method is a modification of the standard Ewald method. DL_POLY_2 implements the SPME method of Essmann *et al.* [31]. Formally this method is capable of treating van der Waals forces also, but in DL_POLY_2 it is confined to electrostatic forces only. The main difference from the standard Ewald method is in its treatment of the the reciprocal space terms. By means of an interpolation procedure involving (complex) B-splines, the sum in reciprocal space is represented on a three dimensional rectangular grid. In this form the Fast Fourier Transform (FFT) may be used to perform the primary mathematical operation, which is a 3D convolution. The efficiency of these procedures greatly reduces the cost of the reciprocal space sum when the range of $\underline{k}$ vectors is large. The method (briefly) is as follows (for full details see [31]):

1. Interpolation of the $\exp(-i\underline{k} \cdot \underline{r}_j)$ terms (given here for one dimension):

$$\exp(2\pi i u_j k / L) \approx b(k) \sum_{\ell=-\infty}^{\infty} M_n(u_j - \ell) \exp(2\pi i k \ell / K) \quad (2.138)$$

in which k is the integer index of the $\underline{k}$ vector in a principal direction, K is the total number of grid points in the same direction and u_j is the fractional coordinate of ion j scaled by a factor K (i.e. $u_j = K s_j^x$). Note that the definition of the B-splines implies a dependence on the integer K , which limits the formally infinite sum over ℓ . The coefficients $M_n(u)$ are B-splines of order n and the factor $b(k)$ is a constant computable from the formula:

$$b(k) = \exp(2\pi i(n-1)k/K) \left[\sum_{\ell=0}^{n-2} M_n(\ell+1) \exp(2\pi i k \ell / K) \right]^{-1} \quad (2.139)$$

2. Approximation of the structure factor $S(\underline{k})$:

$$S(\underline{k}) \approx b_1(k_1) b_2(k_2) b_3(k_3) Q^\dagger(k_1, k_2, k_3) \quad (2.140)$$

where $Q^\dagger(k_1, k_2, k_3)$ is the discrete Fourier transform of the *charge array* $Q(\ell_1, \ell_2, \ell_3)$ defined as

$$Q(\ell_1, \ell_2, \ell_3) = \sum_{j=1}^N q_j \sum_{n_1, n_2, n_3} M_n(u_{1j} - \ell_1 - n_1 L_1) M_n(u_{2j} - \ell_2 - n_2 L_2) M_n(u_{3j} - \ell_3 - n_3 L_3) \quad (2.141)$$

(in which the sums over $n_{1,2,3}$ etc are required to capture contributions from all relevant periodic cell images, which in practice means the nearest images.)

3. Approximating the reciprocal space energy U_{recip} :

$$U_{recip} = \frac{1}{2V_o \epsilon_0} \sum_{k_1, k_2, k_3} G^\dagger(k_1, k_2, k_3) Q(k_1, k_2, k_3) \quad (2.142)$$

in which $G^\dagger$ is the discrete Fourier transform of the function

$$G(k_1, k_2, k_3) = \frac{\exp(-k^2/4\alpha^2)}{k^2} B(k_1, k_2, k_3) (Q^\dagger(k_1, k_2, k_3))^* \quad (2.143)$$

and where

$$B(k_1, k_2, k_3) = |b_1(k_1)|^2 |b_2(k_2)|^2 |b_3(k_3)|^2 \quad (2.144)$$

and $(Q^\dagger(k_1, k_2, k_3))^*$ is the complex conjugate of $Q^\dagger(k_1, k_2, k_3)$. The function $G(k_1, k_2, k_3)$ is thus a relatively simple product of the gaussian screening term appearing in the conventional Ewald sum, the function $B(k_1, k_2, k_3)$ and the discrete Fourier transform of $Q(k_1, k_2, k_3)$

4. Calculating the atomic forces, which are given formally by:

$$f_j^\alpha = -\frac{\partial U_{recip}}{\partial r_j^\alpha} = -\frac{1}{V_o \epsilon_0} \sum_{k_1, k_2, k_3} G^\dagger(k_1, k_2, k_3) \frac{\partial Q(k_1, k_2, k_3)}{\partial r_j^\alpha} \quad (2.145)$$

Fortunately, due to the recursive properties of the B-splines, these formulae are easily evaluated.

The virial and the stress tensor are calculated in the same manner as for the conventional Ewald sum.

The DL_POLY_2 subroutines required to calculate the SPME contributions are: BSPGEN, which calculates the B-splines; BSPCOE, which calculates B-spline coefficients; SPL_CEXP, which calculates the FFT and B-spline complex exponentials; EWALD_SPME, which calculates the reciprocal space contributions; SPME_FOR, which calculates the reciprocal space forces; and DLPFFT3, which calculates the 3D complex fast Fourier transform (default code only, Cray, SGI, IBM SP machines have their own FFT routines, selected at compile time and the FFTW public FFT is also an option). These subroutines calculate the reciprocal space components of the Ewald sum only, the real-space calculations are performed by EWALD2, EWALD3 and EWALD 4, as for the normal Ewald sum. In addition there are a few minor utility routines : CPY_RTC copies a real array to a complex array; ELE_PRD is an element-for-element product of two arrays; SCL_CSUM is a scalar sum of elements of a complex array; and SET_BLOCK initialises an array to a present value (usually zero).

2.4.7 Hautman Klein Ewald (HKE)

The method of Hautman and Klein is an adaptation of the Ewald method for systems which are periodic in two dimensions only [32]. (DL_POLY_2 assumes this periodicity is in the XY plane.)

The HKE method gives the following formula for the electrostatic energy of a system of N (nonbonded) ions that is overall charge neutral⁴:

$$\begin{aligned}
 U_c = & \frac{1}{4\epsilon_0 A} \sum_{n=0}^{n_{max}} a_n \sum_{i,j}^N q_i q_j z_{ij}^{2n} \sum_{\underline{g} \neq \underline{0}} f_n(g; \alpha) g^{2n-1} \exp(i\underline{g} \cdot s_{ij}) + \\
 & \frac{1}{8\pi\epsilon_0} \sum_{ij} ' q_i q_j \sum_{\underline{L}} \left(\frac{1}{r_{ij,L}} - \sum_n^{n_{max}} a_n z_{ij}^n \frac{h_n(s_{ij,L}; \alpha)}{s_{ij,L}^{2n+1}} \right) + \\
 & \frac{1}{8\pi\epsilon_0} \sum_i q_i^2 \sum_{\underline{L}} \frac{(1 - h_0(L; \alpha))}{L} - \frac{\alpha}{\epsilon_0 \pi^{3/2}} \sum_i q_i^2
 \end{aligned} \tag{2.146}$$

Note that the prime (') on the sum over i, j in real space indicates that the condition $i \neq j$ operates when $\underline{L} = \underline{0}$. Also in this formula A is the system area (in the XY plane), $\underline{L}$ is a 2D lattice vector representing the 2D periodicity of the system, s_{ij} is the in-plane (XY) component of the interparticle distance r_{ij} and $\underline{g}$ is a reciprocal lattice vector. Thus

$$\underline{L} = \ell_1 \underline{a} + \ell_2 \underline{b}, \tag{2.147}$$

where ℓ_1, ℓ_2 are integers and vectors $\underline{a}$ and $\underline{b}$ are the lattice basis vectors. The reciprocal lattice vectors are:

$$\underline{g} = n_1 \underline{u} + n_2 \underline{v} \tag{2.148}$$

where n_1, n_2 are integers $\underline{u}, \underline{v}$ are reciprocal space vectors (defined in terms of the vectors $\underline{a}$ and $\underline{b}$):

$$\begin{aligned}
 \underline{u} &= 2\pi(b_y, -b_x)^\dagger / (a_x b_y - a_y b_x) \\
 \underline{v} &= 2\pi(-a_y, a_x)^\dagger / (a_x b_y - a_y b_x).
 \end{aligned} \tag{2.149}$$

The functions $h_n(s; \alpha)$ and $f_n(s; \alpha)$ are the HKE convergence functions, in real and reciprocal space respectively. (C.f. the complementary error and gaussian functions of the original Ewald method.) However they occur to higher orders here, as indicated by the sum over subscript n , which corresponds to terms in a Taylor expansion of r^{-1} in s , the in-plane distance [32]. Usually this sum is truncated at $n_{max} = 1$, but in DL_POLY_2 can go as high as $n_{max} = 3$. In the HKE method the convergence functions are defined as follows:

$$h_n(s; \alpha) / s^{2n+1} = \frac{1}{a_n (2n)!} \nabla^{2n} (h_0(s; \alpha) / s) \tag{2.150}$$

⁴The reader is warned that for the purpose of compatibility with other DL_POLY_2 Ewald routines we have defined $\alpha = 0.5/\alpha_{HK}$, where α_{HK} is the α parameter defined by Hautman and Klein in [32].

with

$$h_0(s; \alpha) = \operatorname{erf}(\alpha s) \quad (2.151)$$

and

$$f_n(g; \alpha) = \frac{1}{a_n(2n)!} f_0(g; \alpha) \quad (2.152)$$

with

$$f_0(g; \alpha) = \operatorname{erfc}(g/2\alpha). \quad (2.153)$$

In DL_POLY_2 the $h_n(s; \alpha)/s^{2n+1}$ functions are derived by a recursion algorithm, while the $f_n(g; \alpha)$ functions are obtained by direct evaluation. The coefficients a_n are given by

$$a_n = (-1)^n (2n)! / (2^{2n} (n!)^2). \quad (2.154)$$

As pointed out by Hautman and Klein, the equation (2.146) allows separation of the z_{ij}^{2n} components via the binomial expansion, which greatly simplifies the double sum over atoms in reciprocal space. Thus the reciprocal space part of equation (2.146) becomes

$$U_{\text{recip}} = \frac{1}{4\epsilon_0 A} \sum_{n=0}^{n_{\text{max}}} a_n \sum_{\underline{g} \neq \underline{0}} f_n(g; \alpha) g^{2n-1} \sum_{p=0}^{2n} (-1)^p C_p^{2n} Z_p(\underline{g}) Z_{2n-p}^*(\underline{g}) \quad (2.155)$$

with C_p^{2n} a binomial coefficient and

$$Z_p(\underline{g}) = \sum_{j=1}^N q_j z_j^p \exp(i \underline{g} \cdot \underline{s}_j) \quad (2.156)$$

The force on an ion is obtained by the usual differentiation, however in this case the z components have different expressions from the x and y .

$$\begin{aligned} -\frac{\partial U_c}{\partial u_j} &= \frac{1}{4\epsilon_0 A} \sum_{\underline{g} \neq \underline{0}} \sum_{n=0}^{n_{\text{max}}} a_n f_n(g; \alpha) g^{2n-1} \sum_{p=0}^{2n} (-1)^p C_p^{2n} \left(Z_p(\underline{g}) \frac{\partial Z_{2n-p}^*(\underline{g})}{\partial u_j} + Z_{2n-p}^*(\underline{g}) \frac{\partial Z_p(\underline{g})}{\partial u_j} \right) \\ &+ \frac{q_j}{4\pi\epsilon_0} \sum_{n=0}^{n_{\text{max}}} \sum_{\underline{L}} \sum_{ij}^N 'a_n q_i \frac{\partial}{\partial u_j} \left(z_{ij,L}^{2n} \frac{h_n(s_{ij,L}; \alpha)}{s_{ij,L}^{2n+1}} \right) \end{aligned} \quad (2.157)$$

where u_j is one of x_j, y_j, z_j and (noting for brevity that x and y derivatives are similar)

$$\begin{aligned} \frac{\partial Z_p(\underline{g})}{\partial x_j} &= i g_x q_j z_j^p \exp(i \underline{g} \cdot \underline{s}_j) \\ \frac{\partial Z_p(\underline{g})}{\partial z_j} &= p q_j z_j^{p-1} \exp(i \underline{g} \cdot \underline{s}_j) \end{aligned} \quad (2.158)$$

and

$$\begin{aligned} \frac{\partial}{\partial x_j} \left(z_{ij,L}^{2n} \frac{h_n(s_{ij,L}; \alpha)}{s_{ij,L}^{2n+1}} \right) &= s_{ij,L}^x \frac{z_{ij,L}^{2n}}{s_{ij,L}} \frac{\partial}{\partial x_j} \left(\frac{h_n(s_{ij,L}; \alpha)}{s_{ij,L}^{2n+1}} \right) \\ \frac{\partial}{\partial z_j} \left(z_{ij,L}^{2n} \frac{h_n(s_{ij,L}; \alpha)}{s_{ij,L}^{2n+1}} \right) &= 2n z_{ij,L}^{2n-1} \frac{h_n(s_{ij,L}; \alpha)}{s_{ij,L}^{2n+1}}. \end{aligned} \quad (2.159)$$

In DL_POLY_2 the partial derivatives of $h_n(s_{ij,L}; \alpha)/s_{ij,L}^{2n+1}$ are calculated by a recursion algorithm. Note that when $n = 0$ there is no derivative w.r.t. z .

The virial and stress tensor terms in real space may be calculated directly from the pair forces and interatomic distances in the usual way, and need not be discussed further. The calculation of the reciprocal space contributions (the terms involving the $f_n(g; \alpha)$ functions) are more difficult. Firstly however we note that the reciprocal space contributions to σ_{xz} , σ_{yz} and σ_{zz} may be obtained directly from the force calculations thus:

$$\begin{aligned}\sigma_{xz}^{recip} &= \sum_j z_j f_j^x \\ \sigma_{yz}^{recip} &= \sum_j z_j f_j^y \\ \sigma_{zz}^{recip} &= \sum_j z_j f_j^z\end{aligned}\tag{2.160}$$

which renders the calculation of these components trivial. The remaining components are calculated from

$$\begin{aligned}\sigma_{uv}^{recip} &= U_{recip}\delta_{uv} + \frac{1}{4\epsilon_0 A} \sum_{n=0}^{n_{max}} a_n \sum_{\underline{g} \neq \underline{0}} g_u g_v \frac{g^{2n-2}}{a_n (2n)!} \\ &\quad \left(\frac{(2n-1)f_0(g; \alpha)}{g} - \frac{1}{\alpha\sqrt{\pi}} \exp(-g^2/4\alpha^2) \right) \\ &\quad \sum_{p=0}^{2n} (-1)^p C_p^{2n} Z_p(\underline{g}) Z_{2n-p}^*(\underline{g})\end{aligned}\tag{2.161}$$

where u, v are one or both of the components x, y . Note that, although it is possible to define these contributions to the stress tensor, it is not possible to calculate a pressure from them unless a finite, arbitrary boundary is imposed on the z direction (which is an assumption applied in DL_POLY_2, but without implications of periodicity in the z -direction). The x, y components define the surface tension however.

For bonded molecules, as with the standard 3D Ewald sum, it is necessary to extract contributions associated with the excluded atom pairs. In the DL_POLY_2 HKE implementation this amounts to an *a posteriori* subtraction of the corresponding coulomb terms.

In DL_POLY_2 the HKE method is handled by several subroutines: HKGEN constructs the $h_n(s; \alpha)$ convergence functions and their derivatives; HKEWALD1 calculates the reciprocal space terms; HKEWALD2 and HKEWALD3 calculate the real space terms and the bonded atom corrections respectively. HKEWALD4 calculates the primary interactions in the multiple timestep implementation.

2.4.8 Reaction Field

In the reaction field method it is assumed that any given molecule is surrounded by a spherical cavity of finite radius within which the electrostatic interactions are calculated explicitly. Outside the cavity the system is treated as a dielectric continuum. The occurrence

of any net dipole within the cavity induces a polarisation in the dielectric, which in turn interacts with the given molecule. The model allows the replacement of the infinite Coulomb sum by a finite sum plus the reaction field.

The reaction field model coded into DL_POLY_2 is the implementation of Neumann based on charge-charge interactions [33]. In this model, the total Coulombic potential is given by

$$U_c = \frac{1}{4\pi\epsilon_0} \sum_{j < n} q_j q_n \left[\frac{1}{r_{nj}} + \frac{B_0 r_{nj}^2}{2R_c^3} \right] \quad (2.162)$$

where the second term on the right is the reaction field correction to the explicit sum, with R_c the radius of the cavity. The constant B_0 is defined as

$$B_0 = \frac{2(\epsilon_1 - 1)}{(2\epsilon_1 + 1)}, \quad (2.163)$$

with ϵ_1 the dielectric constant outside the cavity. The effective pair potential is therefore

$$U(r_{nj}) = \frac{1}{4\pi\epsilon_0} q_j q_n \left[\frac{1}{r_{nj}} + \frac{B_0 r_{nj}^2}{2R_c^3} \right]. \quad (2.164)$$

This expression unfortunately leads to large fluctuations in the system Coulombic energy, due to the large ‘step’ in the function at the cavity boundary. In DL_POLY_2 this is countered by subtracting the value of the potential at the cavity boundary from each pair contribution. The term subtracted is

$$\frac{1}{4\pi\epsilon_0} \frac{q_j q_n}{R_c} \left[1 + \frac{B_0}{2} \right]. \quad (2.165)$$

The effective pair force on an atom j arising from another atom n within the cavity is given by

$$\underline{f}_j = \frac{q_j q_n}{4\pi\epsilon_0} \left[\frac{1}{r_{nj}^3} - \frac{B_0}{R_c^3} \right] \underline{r}_{nj}. \quad (2.166)$$

The contribution of each effective pair interaction to the atomic virial is

$$\mathcal{W} = -\underline{r}_{nj} \cdot \underline{f}_j \quad (2.167)$$

and the contribution to the atomic stress tensor is

$$\sigma^{\alpha\beta} = r_{nj}^\alpha f_j^\beta. \quad (2.168)$$

In DL_POLY_2 the reaction field is handled by the routines COUL3 and COUL3NEU.

2.4.9 Dynamical Shell Model

An atom or ion is polarisable if it develops a dipole moment when placed in an electric field. It is commonly expressed by the equation

$$\underline{\mu} = \alpha \underline{E}, \quad (2.169)$$

where $\underline{\mu}$ is the induced dipole and $\underline{E}$ is the electric field. The constant α is the polarisability.

The dynamical shell model is a method of incorporating polarisability into a molecular dynamics simulation. The method used in DL_POLY_2 is that devised by Fincham *et al* [34] and is known as the adiabatic shell model. An alternative model is presented in [35].

In the *static* shell model a polarisable atom is represented by a massive core and massless shell, connected by a harmonic spring, hereafter called the core-shell unit. The core and shell carry different electric charges, the sum of which equals the charge on the original atom. There is no electrostatic interaction (i.e. self interaction) between the core and shell of the same atom. Non-Coulombic interactions arise from the shell alone. The effect of an electric field is to separate the core and shell, giving rise to a *polarisation* dipole. The condition of static equilibrium gives the polarisability as:

$$\alpha = (2q_s^2 - q_c^2)/k \quad (2.170)$$

where q_s and q_c are the shell and core charges and k is the force constant of the harmonic spring.

In the adiabatic method, a fraction of the atomic mass is assigned to the shell to permit a dynamical description. The fraction of mass is chosen to ensure that the natural frequency of vibration ν of the harmonic spring (i.e.

$$\nu = \frac{1}{2\pi} \left[\frac{k}{x(1-x)m} \right]^{1/2}, \quad (2.171)$$

with m the atomic mass,) is well above the frequency of vibration of the whole atom in the bulk system. Dynamically the core-shell unit resembles a diatomic molecule with a harmonic bond, however the high vibrational frequency of the bond prevents effective exchange of kinetic energy between the core-shell unit and the remaining system. Therefore, from an initial condition in which the core-shell units have negligible internal vibrational energy, the units will remain close to this condition throughout the simulation. This is essential if the core shell unit is to maintain a net polarisation. (In practice there is a slow leakage of kinetic energy into the core-shell units, but this should not amount to more than a few percent of the total kinetic energy.)

The calculation of the virial and stress tensor in this model is based on that for a diatomic molecule with charged atoms. The electrostatic and short ranged forces are calculated as described above. The forces of the harmonic springs are calculated as described for intramolecular harmonic bonds. The relationship between the kinetic energy and the temperature is different however, as the core-shell unit is permitted only three translational degrees of freedom, and the degrees of freedom corresponding to rotation and vibration of the unit are discounted (the kinetic energy of these is regarded as zero).

In DL_POLY_2 the shell forces are handled by the routine SHLFRC. The kinetic energy is calculated by CORSHL and the routine SHQNCH performs the temperature scaling. The dynamical shell model is used in conjunction with the methods for long range forces described above.

2.5 Integration algorithms

DL_POLY integration algorithms are based around the Verlet leapfrog scheme, which is both symplectic (time reversible) and simple in nature [10]. It generates trajectories in the microcanonical (NVE) ensemble in which the total energy (kinetic plus potential energy) is conserved. If this property drifts or fluctuates significantly in the course of a simulation it indicates that the timestep is too large or the potential cutoffs too small (relative r.m.s. fluctuations in the total energy of 10^{-5} are typical with this algorithm).

The algorithm requires values of position ($\underline{r}$) and force ($\underline{f}$) at time t while the velocities ($\underline{v}$) are half a timestep behind. The first step is to advance the velocities to $t + (1/2)\Delta t$ by integration of the force:

$$\underline{v}(t + \frac{1}{2}\Delta t) \leftarrow \underline{v}(t - \frac{1}{2}\Delta t) + \Delta t \frac{\underline{f}(t)}{m} \quad (2.172)$$

where m is the mass of a site and Δt is the timestep.

The positions are then advanced using the new velocities:

$$\underline{r}(t + \Delta t) \leftarrow \underline{r}(t) + \Delta t \underline{v}(t + \frac{1}{2}\Delta t) \quad (2.173)$$

Molecular dynamics simulations normally require properties that depend on position and velocity *at the same time* (such as the sum of potential and kinetic energy). The velocity at time t is obtained from the average of the velocities half a timestep either side of time t :

$$\underline{v}(t) = \frac{1}{2} \left[\underline{v}(t - \frac{1}{2}\Delta t) + \underline{v}(t + \frac{1}{2}\Delta t) \right] \quad (2.174)$$

The instantaneous temperature, for example can then be obtained from the atomic velocities assuming the system has no net momentum:

$$\mathcal{T} = \frac{\sum_{i=1}^{\mathcal{N}} m_i v_i^2(t)}{k_B f} \quad (2.175)$$

where i labels particles (which can be atoms or rigid molecules), $\mathcal{N}$ the number of particles in the system, k_B Boltzmanns constant and f the number of degrees of freedom in the system ($3\mathcal{N} - 3$ if the system is periodic and without constraints).

The routine NVE_0 implements the Verlet leapfrog algorithm and calculates the instantaneous temperature. The conserved quantity is the total energy of the system

$$\mathcal{H}_{\text{NVE}} = U + KE \quad (2.176)$$

where U is the potential energy of the system and KE the kinetic energy at time t .

The full selection of integration algorithms within DL_POLY_2 is as follows:

NVE_0	Verlet leapfrog
NVE_1	Verlet leapfrog with RD-SHAKE

NVEQ_1	Rigid units with FIQA and RD-SHAKE
NVEQ_2	Linked rigid units with QSHAKE
NVT_B0	Constant T (Berendsen [14]) with Verlet leapfrog
NVT_B1	Constant T (Berendsen [14]) with RD-SHAKE
NVT_E0	Constant T (Evans [15]) with Verlet leapfrog
NVT_E1	Constant T (Evans [15]) with RD-SHAKE
NVT_H0	Constant T (Hoover [16]) with Verlet leapfrog
NVT_H1	Constant T (Hoover [16]) with RD-SHAKE
NVTQ_B1	Constant T (Berendsen [14]) with FIQA and RD-SHAKE
NVTQ_B2	Constant T (Berendsen [14]) with QSHAKE
NVTQ_H1	Constant T (Hoover [16]) with FIQA and RD-SHAKE
NVTQ_H2	Constant T (Hoover [16]) with QSHAKE
NPT_B0	Constant T,P (Berendsen [14]) with Verlet leapfrog
NPT_B1	Constant T,P (Berendsen [14]) with FIQA and RD-SHAKE
NPT_B3	Constant T, $\underline{\sigma}$ (Berendsen [14]) with RD-SHAKE
NPT_H0	Constant T,P+ (Hoover [16]) with Verlet leapfrog
NPT_H1	Constant T,P+ (Hoover [16]) with RD-SHAKE
NPT_H3	Constant T, $\underline{\sigma}$ (Hoover [16]) with RD-SHAKE
NPTQ_B1	Constant T,P (Berendsen [14]) with FIQA and RD-SHAKE
NPTQ_B2	Constant T,P (Berendsen [14]) with QSHAKE
NST_B0	Constant T, $\underline{\sigma}$ (Berendsen [14]) with Verlet leapfrog
NST_H0	Constant T, $\underline{\sigma}$ (Hoover [16]) with Verlet leapfrog
NPTQ_B3	Constant T, $\underline{\sigma}$ (Berendsen [14]) with FIQA and RD-SHAKE
NPTQ_B4	Constant T, $\underline{\sigma}$ (Berendsen [14]) with QSHAKE
NPTQ_H1	Constant T,P (Hoover [16]) with FIQA and RD-SHAKE
NPTQ_H2	Constant T,P (Hoover [16]) with QSHAKE
NPTQ_H3	Constant T, $\underline{\sigma}$ (Hoover [16]) with FIQA and RD-SHAKE
NPTQ_H4	Constant T, $\underline{\sigma}$ (Hoover [16]) with QSHAKE

In the above table the FIQA algorithm is Fincham's Implicit Quaternion Algorithm [12] and QSHAKE is the DL_POLY_2 algorithm combining rigid bonds and rigid bodies in the same molecule [13].

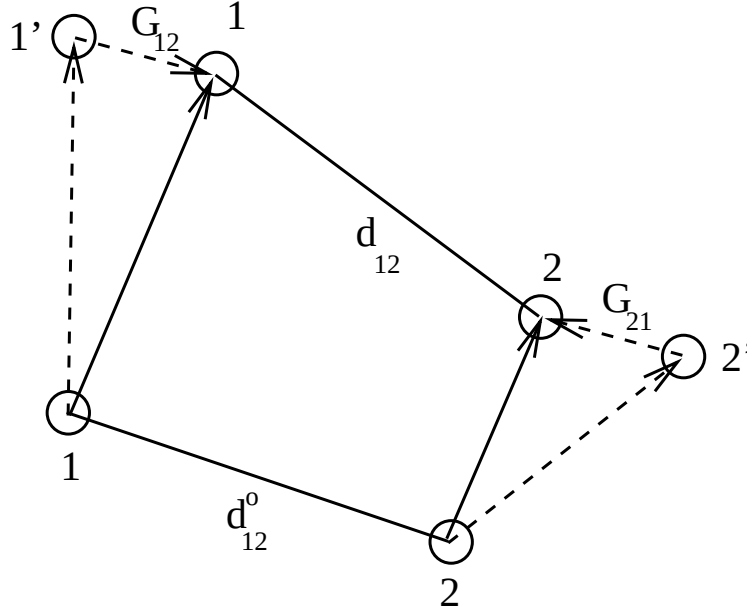
2.5.1 Bond Constraints

The SHAKE algorithm for bond constraints was devised by Ryckaert *et al.* [11] and is widely used in molecular simulation. It is a two stage algorithm based on the Verlet integration scheme [10]. DL_POLY_2 employs a leapfrog variant. In the first stage the Verlet leapfrog algorithm calculates the motion of the atoms in the system assuming a complete absence of the rigid bond forces. The positions of the atoms at the end of this stage do not conserve the distance constraint required by the rigid bond and a correction is necessary. In the second stage the deviation in the length of a given rigid bond is used retrospectively to compute the constraint force needed to conserve the bondlength. It is relatively simple to

show that the constraint force has the form

$$\underline{G}_{ij} \approx \frac{\mu_{ij}(d_{ij}^2 - d_{ij}'^2)}{2\Delta t^2 \underline{d}_{ij}^o \cdot \underline{d}_{ij}'} \underline{d}_{ij}^o \quad (2.177)$$

where: μ_{ij} is the reduced mass of the two atoms connected by the bond; $\underline{d}_{ij}^o$ and $\underline{d}_{ij}'$ are the original and intermediate bond vectors; d_{ij} is the constrained bondlength; and Δt is the Verlet integration time step. It should be noted that this formula is an approximation only.



The SHAKE algorithm calculates the constraint force $\underline{G}_{12} = -\underline{G}_{21}$ that conserves the bondlength d_{12} between atoms 1 and 2, following the initial movement to positions 1' and 2' under the unconstrained forces $\underline{F}_1$ and $\underline{F}_2$.

For a system of simple diatomic molecules, computation of the constraint force will, in principle, allow the correct atomic positions to be calculated in one pass. However in the general polyatomic case this correction is merely an interim adjustment, not only because the above formula is approximate, but the successive correction of other bonds in a molecule has the effect of perturbing previously corrected bonds. The SHAKE algorithm is therefore iterative, with the correction cycle being repeated for all bonds until each has converged to the correct length, within a given tolerance. The tolerance may be of the order 10^{-4} Å to 10^{-8} Å depending on the precision desired.

The procedure may be summarised as follows:

1. All atoms in the system are moved using the Verlet algorithm, assuming an absence of rigid bonds (constraint forces). (This is stage 1 of the SHAKE algorithm.)

2. The deviation in each bondlength is used to calculate the corresponding constraint force (2.177) that (retrospectively) ‘corrects’ the bond length.
3. After the correction (2.177) has been applied to all bonds, every bondlength is checked. If the largest deviation found exceeds the desired tolerance, the correction calculation is repeated.
4. Steps 2 and 3 are repeated until all bondlengths satisfy the convergence criterion (This iteration constitutes stage 2 of the SHAKE algorithm).

The parallel version of this algorithm, as implemented in DL_POLY_2, is known as RD_SHAKE [9] (see section 2.6.8). The subroutine NVE_1 implements the Verlet leapfrog algorithm with bond constraints. The routine RDSHAKE_1 is called to apply the SHAKE corrections to position.

It should be noted that the fully converged constraint forces G_{ij} make a contribution to the system virial and the stress tensor.

The contribution to be added to the atomic virial (for each constrained bond) is

$$\mathcal{W} = -\underline{d}_{ij} \cdot \underline{G}_{ij}. \quad (2.178)$$

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = d_{ij}^{\alpha} G_{ij}^{\beta}, \quad (2.179)$$

where α and β indicate the x, y, z components. The atomic stress tensor derived from the pair forces is symmetric.

2.5.2 Potential of Mean Force (PMF) Constraints and the Evaluation of Free Energy

A generalization of bond constraints can be made to constrain a system to some point along a reaction coordinate. A simple example of such a reaction coordinate would be the distance between two ions in solution. If a number of simulations are conducted with the system constrained to different points along the reaction coordinate then the mean constraint force may be plotted as a function of reaction coordinate and the function integrated to obtain the free energy for the overall process [36]. The PMF constraint force, virial and contributions to the stress tensor are obtained in a manner analogous to that for a bond constraint (see previous section). The only difference is that the constraint is now applied between the centres of two groups which need not be atoms alone. DL_POLY_2 reports the PMF constraint virial, $\mathcal{W}$, for each simulation. Users can convert this to the PMF constraint force from

$$G_{\text{PMF}} = -\mathcal{W}_{\text{PMF}}/d_{\text{PMF}}$$

where d_{PMF} is the constraint distance between the two groups used to define the reaction coordinate.

2.5.3 Thermostats

The system may be coupled to a heat bath to ensure that the average system temperature is maintained close to the requested temperature, T_{ext} . When this is done the equations of motion are modified and the system no longer samples the microcanonical ensemble. Instead trajectories in the canonical (NVT) ensemble, or something close to it are generated. DL_POLY_2 comes with three different thermostats: Nosé-Hoover [16], Berendsen [14], and Gaussian constraints [15]. Of these only the Nosé-Hoover algorithm generates trajectories in the canonical (NVT) ensemble. The other methods will produce properties that typically differ from canonical averages by $\mathcal{O}(1/N)$ [10]

2.5.3.1 Nosé- Hoover Thermostat

In the Nosé-Hoover algorithm [16] Newton's equations of motion are modified to read:

$$\begin{aligned} \frac{d\mathbf{r}(t)}{dt} &= \mathbf{v}(t) \\ \frac{d\mathbf{v}(t)}{dt} &= \frac{\mathbf{f}(t)}{m} - \chi(t)\mathbf{v}(t) \end{aligned} \tag{2.180}$$

$$\tag{2.181}$$

The friction coefficient, χ , is controlled by the first order differential equation

$$\frac{d\chi(t)}{dt} = \frac{1}{\tau_T^2} \left(\frac{\mathcal{T}}{T_{\text{ext}}} - 1 \right) \tag{2.182}$$

where τ_T is a specified time constant (normally in the range [0.5, 2] ps). In DL_POLY_2 χ is stored at half timesteps as it has dimensions of (1/time). The integration takes place as:

$$\begin{aligned} \chi(t + \frac{1}{2}\Delta t) &\leftarrow \chi(t - \frac{1}{2}\Delta t) + \frac{\Delta t}{\tau_T^2} \left(\frac{\mathcal{T}}{T_{\text{ext}}} - 1 \right) \\ \chi(t) &\leftarrow \frac{1}{2} \left[\chi(t - \frac{1}{2}\Delta t) + \chi(t + \frac{1}{2}\Delta t) \right] \\ \mathbf{v}(t + \frac{1}{2}\Delta t) &\leftarrow \mathbf{v}(t - \frac{1}{2}\Delta t) + \Delta t \left[\frac{\mathbf{f}(t)}{m} - \chi(t)\mathbf{v}(t) \right] \\ \mathbf{v}(t) &\leftarrow \frac{1}{2} \left[\mathbf{v}(t - \frac{1}{2}\Delta t) + \mathbf{v}(t + \frac{1}{2}\Delta t) \right] \\ \mathbf{r}(t + \Delta t) &\leftarrow \mathbf{r}(t) + \Delta t \mathbf{v}(t + \frac{1}{2}\Delta t) \end{aligned} \tag{2.183}$$

Since $\mathbf{v}(t)$ is required to calculate $\mathcal{T}$ and itself, the algorithm requires several iterations to obtain self consistency. In DL_POLY_2 the number of iterations is set to 3 (4 if the system has bond constraints). The iteration procedure is started with the standard Verlet leapfrog prediction of $\mathbf{v}(t)$ and $\mathcal{T}$. The conserved quantity is derived from the extended Hamiltonian

for the system which, to within a constant, is the Helmholtz free energy:

$$\mathcal{H}_{\text{NVT}} = \mathcal{H}_{\text{NVE}} + f k_B T_{\text{ext}} \left(\frac{\tau_T^2 \chi^2(t)}{2} + \int_o^t \chi(s) ds \right) \quad (2.184)$$

If bond constraints are present an extra iteration is required due to the call to the SHAKE routine. Note that the SHAKE corrections need only be applied during the first iteration as subsequent to this the velocities, relative to the bond c.o.m. velocity, will be orthogonal to the bond vectors. The velocity scaling imposed by the thermostat is isotropic so does not destroy this orthogonality. The algorithm is implemented in the DL_POLY routines NVT_H0 and NVT_H1, the latter being for systems with bond constraints.

2.5.3.2 Berendsen Thermostat

In the Berendsen algorithm the instantaneous temperature is pushed towards the desired temperature by scaling the velocities at each step:

$$\begin{aligned} \chi &\leftarrow \left[1 + \frac{\Delta t}{\tau_T} \left(\frac{T_{\text{ext}}}{\mathcal{T}} - 1 \right) \right]^{1/2} \\ \underline{v}(t + \frac{1}{2}\Delta t) &\leftarrow \left[\underline{v}(t - \frac{1}{2}\Delta t) + \Delta t \frac{\underline{f}(t)}{m} \right] \chi \\ \underline{v}(t) &\leftarrow \frac{1}{2} \left[\underline{v}(t - \frac{1}{2}\Delta t) + \underline{v}(t + \frac{1}{2}\Delta t) \right] \\ \underline{r}(t + \Delta t) &\leftarrow \underline{r}(t) + \Delta t \underline{v}(t + \frac{1}{2}\Delta t) \end{aligned} \quad (2.185)$$

As with the Nosé-Hoover thermostat iteration is required to obtain self consistency of χ , $\underline{v}(t)$ and $\mathcal{T}$, although it should be noted χ has different roles in the two thermostats. The Berendsen algorithm conserves total momentum but not energy.

As with the Nosé-Hoover algorithm the presence of constraint bonds requires an additional iteration with one application of SHAKE corrections. The algorithm is implemented in the DL_POLY routines NVT_B0 and NVT_B1, the latter being for systems with bond constraints.

2.5.4 Gaussian Constraints

Kinetic temperature can be made a constant of the equations of motion by imposing an additional constraint on the system. If one writes the equations of motions as :

$$\begin{aligned} \frac{d\underline{r}(t)}{dt} &= \underline{v}(t) \\ \frac{d\underline{v}(t)}{dt} &= \frac{\underline{f}(t)}{m} - \chi(t)\underline{v}(t) \end{aligned} \quad (2.186)$$

$$(2.187)$$

with the temperature constraint

$$\frac{d\mathcal{T}}{dt} \propto \frac{d}{dt} \left(\sum_i (m_i v_i)^2 \right) \propto \sum_i m_i^2 \underline{v}_i(t) \cdot \underline{f}_i(t) = 0 \quad (2.188)$$

then choosing

$$\chi = \frac{\sum_i m_i \underline{v}_i(t) \cdot \underline{f}_i(t)}{\sum_i m_i^2 v_i^2(t)} \quad (2.189)$$

minimizes the “least squares” differences between the Newtonian and constrained trajectories. Following Brown and Clarke [37] the algorithm is implemented by calculating $\eta = 1/(1 + \chi \Delta t/2)$

$$\begin{aligned} \eta &\leftarrow \sqrt{\frac{T_{\text{ext}}}{\mathcal{T}}} \\ \underline{v}(t + \frac{1}{2}\Delta t) &\leftarrow (2\eta - 1)\underline{v}(t - \frac{1}{2}\Delta t) + \eta \Delta t \frac{\underline{f}(t)}{m} \\ \underline{r}(t + \Delta t) &\leftarrow \underline{r}(t) + \Delta t \underline{v}(t + \frac{1}{2}\Delta t) \end{aligned} \quad (2.190)$$

where $\mathcal{T}$ is obtained from standard Verlet leapfrog integration. Only one iteration is needed (two if the system has bond constraints) to constrain the instantaneous temperature to exactly T_{ext} however energy is not conserved by this algorithm.

The algorithm is implemented in the DL_POLY routines `NVT_E0` and `NVT_E1`. The latter is for systems with bond constraints.

2.5.5 Barostats

The size and shape of the simulation cell may be dynamically adjusted by coupling the system to a barostat in order to obtain a desired average pressure (P_{ext}) and/or isotropic stress tensor ($\underline{\sigma}$). DL_POLY_2 has two such algorithms: a Hoover type barostat and the Berendsen barostat. Only the former has a well defined conserved quantity.

2.5.5.1 The Hoover Barostat

DL_POLY_2 uses the Melchionna modification of the Hoover algorithm [38] in which the equations of motion involve a Nosé - Hoover thermostat and a barostat in the same spirit.

Cell size variation

For isotropic fluctuations the equations of motion are:

$$\begin{aligned} \frac{d\underline{r}(t)}{dt} &= \underline{v}(t) + \eta(\underline{r}(t) - \underline{R}_0) \\ \frac{d\underline{v}(t)}{dt} &= \frac{\underline{f}(t)}{m} - [\chi(t) + \eta(t)] \underline{v}(t) \end{aligned}$$

$$\begin{aligned}
\frac{d\chi(t)}{dt} &= \frac{1}{\tau_T^2} \left(\frac{\mathcal{T}}{T_{\text{ext}}} - 1 \right) \\
\frac{d\eta(t)}{dt} &= \frac{1}{\mathcal{N}k_B T_{\text{ext}} \tau_P^2} V(t) (\mathcal{P} - P_{\text{ext}}) \\
\frac{dV(t)}{dt} &= [3\eta(t)] V(t)
\end{aligned} \tag{2.191}$$

where η is the barostat friction coefficient, R_0 the system centre of mass, τ_P a specified time constant for pressure fluctuations, $\mathcal{P}$ the instantaneous pressure and V the system volume.

The conserved quantity is, to within a constant, the Gibbs free energy of the system:

$$\mathcal{H}_{NPT} = \mathcal{H}_{NVT} + P_{\text{ext}} V(t) + \frac{3\mathcal{N}k_B T_{\text{ext}}}{2} \eta(t)^2 \tau_P^2 \tag{2.192}$$

The algorithm is readily implemented in the leapfrog scheme:

$$\begin{aligned}
\chi(t + \frac{1}{2}\Delta t) &\leftarrow \chi(t - \frac{1}{2}\Delta t) + \frac{\Delta t}{\tau_T^2} \left(\frac{\mathcal{T}}{T_{\text{ext}}} - 1 \right) \\
\chi(t) &\leftarrow \frac{1}{2} \left[\chi(t - \frac{1}{2}\Delta t) + \chi(t + \frac{1}{2}\Delta t) \right] \\
\eta(t + \frac{1}{2}\Delta t) &\leftarrow \eta(t - \frac{1}{2}\Delta t) + \frac{\Delta t V(t)}{\mathcal{N}k_B T_{\text{ext}} \tau_P^2} (\mathcal{P} - P_{\text{ext}}) \\
\eta(t) &\leftarrow \frac{1}{2} \left[\eta(t - \frac{1}{2}\Delta t) + \eta(t + \frac{1}{2}\Delta t) \right] \\
\underline{v}(t + \frac{1}{2}\Delta t) &\leftarrow \underline{v}(t - \frac{1}{2}\Delta t) + \Delta t \left[\frac{\underline{f}(t)}{m} - [\chi(t) + \eta(t)] \underline{v}(t) \right] \\
\underline{v}(t) &\leftarrow \frac{1}{2} \left[\underline{v}(t - \frac{1}{2}\Delta t) + \underline{v}(t + \frac{1}{2}\Delta t) \right] \\
\underline{r}(t + \Delta t) &\leftarrow \underline{r}(t) + \Delta t \left(\underline{v}(t + \frac{1}{2}\Delta t) + \eta(t + \frac{1}{2}\Delta t) \left[\underline{r}(t + \frac{1}{2}\Delta t) - \underline{R}_0 \right] \right) \\
\underline{r}(t + \frac{1}{2}\Delta t) &\leftarrow \frac{1}{2} [\underline{r}(t) + \underline{r}(t + \Delta t)]
\end{aligned} \tag{2.193}$$

Like the Nosé-Hoover thermostat, several iterations are required to obtain self consistency. DL_POLY_2 uses 4 iterations (5 if bond constraints are present) with the standard Verlet leapfrog predictions for the initial estimates of $\mathcal{T}$, $\mathcal{P}$, $\underline{v}(t)$ and $\underline{r}(t + \frac{1}{2}\Delta t)$. Note also that the change in box size requires the SHAKE algorithm to be called each iteration with the new cell vectors obtained from:

$$\begin{aligned}
V(t + \Delta t) &\leftarrow V(t) \exp \left[3\Delta t \eta(t + \frac{1}{2}\Delta t) \right] \\
\underline{\underline{\mathbf{H}}}(t + \Delta t) &\leftarrow \underline{\underline{\mathbf{H}}}(t) \exp \left[\Delta t \eta(t + \frac{1}{2}\Delta t) \right]
\end{aligned} \tag{2.194}$$

where $\underline{\underline{\mathbf{H}}}$ is the cell matrix whose columns are the three cell vectors $\underline{a}$, $\underline{b}$, $\underline{c}$.

The isotropic changes to cell volume are implemented in the DL_POLY routine NPT_H1 which allows for systems containing bond constraints.

Cell size and shape variation

The isotropic algorithm may be extended to allowing the cell shape to vary by defining $\underline{\eta}$ as a tensor, $\underline{\underline{\eta}}$. The equations of motion are implemented as:

$$\begin{aligned}
\chi(t + \frac{1}{2}\Delta t) &\leftarrow \chi(t - \frac{1}{2}\Delta t) + \frac{\Delta t}{\tau_T^2} \left(\frac{\mathcal{T}}{T_{\text{ext}}} - 1 \right) \\
\chi(t) &\leftarrow \frac{1}{2} \left[\chi(t - \frac{1}{2}\Delta t) + \chi(t + \frac{1}{2}\Delta t) \right] \\
\underline{\underline{\eta}}(t + \frac{1}{2}\Delta t) &\leftarrow \underline{\underline{\eta}}(t - \frac{1}{2}\Delta t) + \frac{\Delta t V(t)}{\mathcal{N} k_B T_{\text{ext}} \tau_P^2} (\underline{\underline{\sigma}} - P_{\text{ext}} \underline{\underline{\mathbf{1}}}) \\
\underline{\underline{\eta}}(t) &\leftarrow \frac{1}{2} \left[\underline{\underline{\eta}}(t - \frac{1}{2}\Delta t) + \underline{\underline{\eta}}(t + \frac{1}{2}\Delta t) \right] \\
\underline{\underline{v}}(t + \frac{1}{2}\Delta t) &\leftarrow \underline{\underline{v}}(t - \frac{1}{2}\Delta t) + \Delta t \left[\frac{\underline{\underline{f}}(t)}{m} - [\chi(t) \underline{\underline{\mathbf{1}}} + \underline{\underline{\eta}}(t)] \underline{\underline{v}}(t) \right] \\
\underline{\underline{v}}(t) &\leftarrow \frac{1}{2} \left[\underline{\underline{v}}(t - \frac{1}{2}\Delta t) + \underline{\underline{v}}(t + \frac{1}{2}\Delta t) \right] \\
\underline{\underline{r}}(t + \Delta t) &\leftarrow \underline{\underline{r}}(t) + \Delta t \left(\underline{\underline{v}}(t + \frac{1}{2}\Delta t) + \underline{\underline{\eta}}(t + \frac{1}{2}\Delta t) \left[\underline{\underline{r}}(t + \frac{1}{2}\Delta t) - \underline{\underline{R}}_0 \right] \right) \\
\underline{\underline{r}}(t + \frac{1}{2}\Delta t) &\leftarrow \frac{1}{2} [\underline{\underline{r}}(t) + \underline{\underline{r}}(t + \Delta t)]
\end{aligned} \tag{2.195}$$

where $\underline{\underline{\mathbf{1}}}$ is the identity matrix and $\underline{\underline{\sigma}}$ the pressure tensor. The new cell vectors are calculated from

$$\underline{\underline{\mathbf{H}}}(t + \Delta t) \leftarrow \underline{\underline{\mathbf{H}}}(t) \exp \left[\Delta t \underline{\underline{\eta}}(t + \frac{1}{2}\Delta t) \right] \tag{2.196}$$

DL_POLY_2 uses a power series expansion truncated at the quadratic term to approximate the exponential of the tensorial term. The new volume is found from

$$V(t + \Delta t) \leftarrow V(t) \exp \left[\Delta t \text{tr}(\underline{\underline{\eta}}) \right] \tag{2.197}$$

The conserved quantity is

$$\mathcal{H}_{NPT} = \mathcal{H}_{NVT} + P_{\text{ext}} V(t) + \frac{3\mathcal{N} k_B T_{\text{ext}}}{2} \text{tr}[\underline{\underline{\eta}}(t)]^2 \tau_P^2 \tag{2.198}$$

This algorithm is implemented in the routines NST_H0 (nonbonded systems) and NPT_H3 (with bond constraints).

2.5.5.2 Berendsen Barostat

With the Berendsen barostat the system is made to obey the equation of motion

$$\frac{d\mathcal{P}}{dt} = (P_{\text{ext}} - \mathcal{P})/\tau_P \quad (2.199)$$

Cell size variations

In the isotropic implementation, at each step the MD cell volume is scaled by by a factor η and the coordinates, and cell vectors, by $\eta^{1/3}$ where

$$\eta = 1 - \frac{\beta\Delta t}{\tau_P}(P_{\text{ext}} - \mathcal{P}) \quad (2.200)$$

and β is the isothermal compressibility of the system. The Berendsen thermostat is applied at the same time. In practice β is a specified constant which DL_POLY_2 takes to be the isothermal compressibility of liquid water. The exact value is not critical to the algorithm as it relies on the ratio τ_P/β . τ_P is specified by the user.

This algorithm is implemented in NPT_B1 with 4 or 5 iterations used to obtain self consistency in the $\underline{v}(t)$.

Cell size and shape variations

The extension of the isotropic algorithm to anisotropic cell variations is straightforward. The tensor $\underline{\eta}$ is defined by

$$\underline{\eta} = \underline{\mathbf{1}} - \frac{\beta\Delta t}{\tau_P}(P_{\text{ext}}\underline{\mathbf{1}} - \underline{\sigma}) \quad (2.201)$$

and the new cell vectors given by

$$\underline{\underline{\mathbf{H}}}(t + \Delta t) \leftarrow \underline{\underline{\mathbf{H}}}(t)\underline{\eta} \quad (2.202)$$

As in the isotropic case the Berendsen thermostat is applied simultaneously and 4 or 5 iterations are used to obtain convergence. The algorithm is implemented in NST_B0 (nonbonded systems) and NPT_B3 (with bond constraints).

2.5.6 Rigid Bodies and Rotational Integration Algorithms

2.5.6.1 Description of Rigid Body Units

A rigid body unit is a collection of point atoms whose local geometry is time invariant. One way to enforce this in a simulation is to impose a sufficient number of bond constraints between the atoms in the unit. However, in many cases this is may be either problematic or impossible. Examples in which it is impossible to specify sufficient bond constraints are

1. linear molecules with more than 2 atoms (e.g. CO₂)
2. planar molecules with more than three atoms (e.g. benzene).

Even when the structure *can* be defined by bond constraints the network of bonds produced may be problematic. Normally, they make the iterative SHAKE procedure slow, particularly if a ring of constraints is involved (as occurs when one defines water as a constrained triangle). It is also possible, inadvertently, to over constrain a molecule (e.g. by defining a methane tetrahedron to have 10 rather than 9 bond constraints) in which case the SHAKE procedure will become unstable. In addition, massless sites (e.g. charge sites) cannot be included in a simple constraint approach making modelling with potentials such as TIP4P water impossible. All these problems may be circumvented by defining rigid body units in terms of a center of mass (c.o.m) and an orientation and solving the resultant rigid body equations of motion.⁵

A rigid body has associated with it a rotational inertia matrix $\underline{\underline{\mathbf{I}}}$, whose components are given by $I_{\alpha\beta} = 1/2 \sum_{\text{sites}} m r_{\alpha} r_{\beta}$ where the r are the distance to the centre of mass and the m are the site masses. In DL_POLY_2 we define the local body frame to be that in which the rotational inertia tensor $\underline{\underline{\mathbf{I}}}$ is diagonal and the components satisfy $I_{xx} \geq I_{yy} \geq I_{zz}$. These three components are stored in the arrays `rotinx`, `rotiny` and `rotinz` for each unique type of rigid body in the system. The total mass of the rigid body unit, M , is stored in the array `gmass` and the location of sites with respect to the local body axes are stored in the arrays `gxx`, `gyy` and `gzz`.

The orientation of a local body frame with respect to the space fixed frame is described via a four dimensional unit vector, the quaternion $\underline{q} = [q_0, q_1, q_2, q_3]^T$. The Rotational matrix to transform from the local body frame to the space fixed frame is the unitary matrix

$$\underline{\underline{\mathbf{R}}} = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \quad (2.203)$$

so that if $\hat{\underline{d}}_{\alpha}$ is the position of a site in the local body frame with respect to its centre of mass, its position in the space fixed frame (w.r.t. its centre of mass) is given by

$$\underline{d}_{\alpha} = \underline{\underline{\mathbf{R}}} \hat{\underline{d}}_{\alpha} \quad (2.204)$$

2.5.6.2 Integration of the Rigid Body Equations of Motion

The net translational force acting upon the rigid unit is

$$\underline{F} = \sum_{\alpha} \underline{f}_{\alpha} \quad (2.205)$$

where $\underline{f}_{\alpha}$ is the force on a rigid unit site, and the sum includes all sites α in the body. The translational motion can be integrated by the standard leapfrog algorithm.

$$\underline{V}(t + \frac{\Delta t}{2}) = \underline{V}(t - \frac{\Delta t}{2}) + \Delta t M^{-1} \underline{F}(t) \quad (2.206)$$

⁵An alternative approach is to define “basic” and “secondary” particles. The basic particles are the minimum number needed to define a local body axis system. The remaining particle positions are expressed in terms of the c.o.m. and the basic particles. Ordinary bond constraints can then be applied to the basic particles provided the forces and torques arising from the secondary particles are transferred to the basic particles in a physically meaningful way.

$$\underline{X}(t + \Delta t) = \underline{X}(t) + \Delta t \underline{V}(t + \frac{\Delta t}{2}) \quad (2.207)$$

where M is the mass of the rigid unit, $\underline{V}$ is the rigid bodies c.o.m. velocity and $\underline{X}$ is the c.o.m. position. The cartesian components of these quantities are stored in the arrays: **gvxx**, **gvyy**, and **gvzz** for c.o.m. velocity; and **gcmx**, **gcmy**, and **gcmz** for c.o.m. position.

The torque acting upon the body in the space fixed frame is

$$\underline{\tau} = \sum_{\alpha} \underline{d}_{\alpha} \times \underline{f}_{\alpha}. \quad (2.208)$$

Transformed to the local body frame (and including the centrifugal terms) this is

$$\hat{\underline{\tau}} = \underline{\underline{\mathbf{R}}}^T \underline{\tau} + \underline{\eta}. \quad (2.209)$$

where

$$\eta_x = ((\hat{I})_{yy} - (\hat{I})_{zz})\hat{\omega}_y\hat{\omega}_z \quad (2.210)$$

plus cyclic permutations for y and z components. The angular velocity transformed to the local body frame, $\hat{\omega}$, can then be integrated using the leapfrog algorithm and the diagonal rotational inertia tensor.

$$\hat{\omega}(t + \frac{\Delta t}{2}) = \hat{\omega}(t - \frac{\Delta t}{2}) + \Delta t \hat{\underline{\mathbf{I}}}^{-1} \hat{\underline{\tau}}(t) \quad (2.211)$$

The new quaternions cannot be found so simply. **DL_POLY_2** uses Fincham's implicit quaternion algorithm (FIQA) to do this [12]. In this algorithm the new quaternions are found by solving the implicit equation

$$\underline{q}(t + \Delta t) = \underline{q}(t) + \frac{\Delta t}{2} \left(\underline{\underline{\mathbf{Q}}}[\underline{q}(t)]\hat{\underline{w}}(t) + \underline{\underline{\mathbf{Q}}}[\underline{q}(t + \Delta t)]\hat{\underline{w}}(t + \Delta t) \right) \quad (2.212)$$

where $\hat{\underline{w}} = [0, \hat{\omega}]^T$ and $\underline{\underline{\mathbf{Q}}}[\underline{q}]$ is

$$\underline{\underline{\mathbf{Q}}} = \frac{1}{2} \begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & -q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix} \quad (2.213)$$

The above equation is solved iteratively with

$$\underline{q}(t + \Delta t) = \underline{q}(t) + \Delta t \underline{\underline{\mathbf{Q}}}[\underline{q}(t)]\hat{\underline{w}}(t) \quad (2.214)$$

as the first guess. Typically no more than 3 or 4 iterations are needed for convergence. At each step the constraint

$$\|\underline{q}(t + \Delta t)\| = 1 \quad (2.215)$$

is imposed.

The quaternions are stored in the arrays **q0**, **q1**, **q2** and **q3**. The angular velocity (transformed to the body fixed frame) is stored in the arrays **omx**, **omy** and **omz**, while

the work arrays `opx`, `opy`, `opz`, `oqx`, `ogy`, `oqz` hold values of $\hat{\omega}(t)$ and $\hat{\omega}(t + \Delta t)$. The torques, $\hat{\tau}(t)$ are held in the work arrays `txq`, `tyq` and `tzq`.

The NVE algorithm is implemented in `NVEQ_1` which allows for a system containing a mixture of rigid bodies and atomistic species, provided the rigid bodies are not linked to other species by constraint bonds.

Thermostats and Barostats

It is straightforward to couple the rigid body equations of motion to a thermostat and/or barostat. The thermostat is coupled to both the translational and rotational degrees of freedom and so both the translational and rotational velocities are propagated in an analogous manner to the thermostated atomic velocities. The barostat, however, is coupled only to the translational degrees of freedom, not to the rotational ones. `DL_POLY_2` supports both Hoover type and Berendsen thermostats and barostats for systems containing rigid bodies. The Hoover thermostat is implemented in `NVTQ_H1`, the Hoover isotropic barostat (plus thermostat) in `NPTQ_H1` and the anisotropic barostat in `NPTQ_H3`. The analogous routines for the Berendsen algorithms are `NVTQ_B1`, `NPTQ_B1` and `NPTQ_B3`.

2.5.6.3 Linked Rigid Bodies

The above integration algorithm can be used for rigid bodies in a system containing “atomic” species (whose equations of motion are integrated with the standard leapfrog algorithm). These rigid bodies may even be linked to other species (including other rigid bodies) by extensible bonds. However if a rigid body is linked to an atom or another rigid body by a bond *constraint* the above algorithm is not adequate. The reason is that the constraint will introduce an additional force and torque onto the body that can only be found *after* the integration of the unconstrained unit. `DL_POLY_2` has a suite of integration algorithms to cope with this situation in which both the constraint conditions and the quaternion equations are solved simultaneously using an extension of the SHAKE algorithm called “QSHAKE” [13].

The QSHAKE algorithm proceeds as follows: Consider the figure in which two rigid bodies are linked by a constraint bond. We seek to choose $\underline{F}_{\text{const}}$ so that at the end of the integration step the two sites in the constraint bond are a distance r apart. The integration of the bodies as free units leaves the sites r' apart. Since the constraint force produces both a force and torque on the rigid units the correction to the constrained sites position must include both the translation and rotation of the body as a whole. The translational contribution is

$$\Delta \underline{x}_T = (\Delta t)^2 \underline{F}_{\text{const}} \left(\frac{1}{M} \right) \quad (2.216)$$

The torque induced by the constraint force is

$$\underline{\mathcal{I}}_{\text{const}} = \underline{d}_a \times \underline{F}_{\text{const}} \quad (2.217)$$

so the correction to the angular velocity of the body is

$$\Delta \underline{\omega} = \Delta t \underline{\mathbb{I}}^{-1} \underline{\mathcal{I}}_{\text{const}}. \quad (2.218)$$

The “rotational” correction to the position of the site is thus

$$\Delta \underline{x}_R = \Delta t (\Delta \underline{\omega} \times \underline{d}_a) \quad (2.219)$$

which in general will not be in the same direction as $\underline{F}_{\text{const}}$ and $\Delta \underline{x}_T$. If we denote the components of $\Delta \underline{x}_R$ parallel and perpendicular to $\underline{F}_{\text{const}}$ by $\Delta \underline{x}_R^{\parallel}$ and $\Delta \underline{x}_R^{\perp}$ the correction to the site position is

$$\Delta \underline{x} = (\Delta \underline{x}_T + \Delta \underline{x}_R^{\parallel}) + \Delta \underline{x}_R^{\perp} \quad (2.220)$$

Clearly the presence of the constraint force and torque will mean the positions and velocities of the remaining sites in the rigid body will also have to be corrected. We proceed by seeking an *approximation* to $\underline{F}_{\text{const}}$ then reintegrating the rigid body equations of motion with the improved total forces and torques on the body. A few iterations are normally sufficient to achieve convergence: If we assume the bond distance r is large in comparison to the correction then after the correction is made the bond length is

$$\begin{aligned} r^* &= \sqrt{\left(r' + \Delta x_{\parallel}\right)^2 + (\Delta x_{\perp})^2} \\ &= \sqrt{(r')^2 + 2r'\Delta x_{\parallel} + \Delta x_{\parallel}^2 + \Delta x_{\perp}^2} \\ &\approx \sqrt{(r')^2 + 2r'\Delta x_{\parallel}} \end{aligned} \quad (2.221)$$

$$(2.222)$$

where $\Delta x_{\parallel} = \sum_{\text{bodies}} \Delta x_T + \Delta x_R^{\parallel}$ and $\Delta x_{\perp} = \sum_{\text{bodies}} \Delta x_R^{\perp}$. Thus a first order correction to the position of the constrained site is

$$\begin{aligned} \Delta \underline{x} &= \Delta \underline{x}_T + \Delta \underline{x}_R^{\parallel} \\ &= (\Delta t)^2 \underline{F}_{\text{const}} \left\{ \frac{1}{M} + \left[\left(\underline{\mathbf{I}}^{-1} (\underline{d}_a \times \underline{e}) \right) \times \underline{d}_a \right] \cdot \underline{e} \right\} \end{aligned} \quad (2.223)$$

where $\underline{e}$ is the unit vector $\underline{F}_{\text{const}} / \|\underline{F}_{\text{const}}\|$. The term in the $\{\dots\}$ defines an effective mass, $\mathcal{M}$: where $1/\mathcal{M} = 1/M + \left[\left(\underline{\mathbf{I}}^{-1} (\underline{d}_a \times \underline{e}) \right) \times \underline{d}_a \right] \cdot \underline{e}$.

The effective reduced mass for the two linked bodies, a and b , is $\mu = \mathcal{M}_a \mathcal{M}_b / (\mathcal{M}_a + \mathcal{M}_b)$. As in the standard SHAKE algorithm, the reduced mass can then be used to predict the constraint force

$$\underline{F}_{\text{const}} = \frac{\mu}{\Delta t^2} \frac{(r^2 - r'^2)}{2r^2} \quad (2.224)$$

Note that each prediction of the constraint force requires the rigid body equations of motion to be reintegrated accurately. If two bodies are linked by one constraint bond only two or three iterations are necessary for convergence. However convergence may be slower if an extended network of linked bodies is involved. Note also that the reduced mass needs to be re-evaluated every time-step because it depends on the relative orientation of the two bodies. Finally, we note the algorithm reduces to the standard SHAKE algorithm if the rigid bodies are replaced by point atoms. In such a case, even though d_a is zero, care must be taken to avoid the singularity arising from $\underline{\mathbf{I}}^{-1}$.

This “linked rigid body” algorithm is implemented in NVEQ_2 with the QSHAKE corrections to the forces applied in QSHAKE. Again it is straightforward to couple these systems to a Hoover type or Berendsen thermostat and/or barostat. The Hoover and Berendsen thermostated versions are found in NVTQ_H2 and NVTQ_B2 respectively. The isotropic constant pressure implementations are found in NPTQ_H2 and NPTQ_B2, while the anisotropic constant pressure routines are found in NPTQ_H4 and NPTQ_B4. An outline of the parallel version of QSHAKE is given in section 2.6.8.

2.5.7 The DL_POLY_2 Multiple Timestep Algorithm

For simulations employing a large spherical cutoff r_{cut} in the calculation of the interactions DL_POLY_2 offers the possibility of using a multiple timestep algorithm to improve the efficiency. The method is based on that described by Streett *et al* [39, 40] with extension to Coulombic systems by Forester *et al* [41].

In the multiple timestep algorithm there are two cutoffs for the pair interactions: a relatively large cutoff (r_{cut}) which is used to define the standard Verlet neighbour list; and a smaller cutoff r_{prim} which is used to define a *primary list* within the larger cutoff sphere (see figure). Forces derived from atoms in the primary list are generally much larger than those derived from remaining (so-called *secondary*) atoms in the neighbour list. Good energy conservation is therefore possible if the forces derived from the primary atoms are calculated *every* timestep, while those from the secondary atoms are calculated much less frequently, and are merely extrapolated over the interval. DL_POLY_2 handles this procedure as follows.

DL_POLY_2 updates the Verlet neighbour list at irregular intervals, determined by the movement of atoms in the neighbour list (see section 2.1). The interval between updates is usually of the order of ~ 20 timesteps. Partitioning the Verlet list into primary and secondary atoms always occurs when the Verlet list is updated, and thereafter at intervals of `multt` timesteps (i.e. the multi-step interval specified by the user - see section 4.1.1). Immediately after the partitioning, the force contributions from both the primary and secondary atoms are calculated. The forces are again calculated *in total* in the subsequent timestep. Thereafter, for `multt-2` timesteps, the forces derived from the primary atoms are calculated explicitly, while those derived from the secondary atoms are calculated by linear extrapolation of the exact forces obtained in the first two timesteps of the multi-step interval. It is readily apparent how this scheme can lead to a significant saving in execution time.

Extension of this basic idea to simulations using the Ewald sum requires the following:

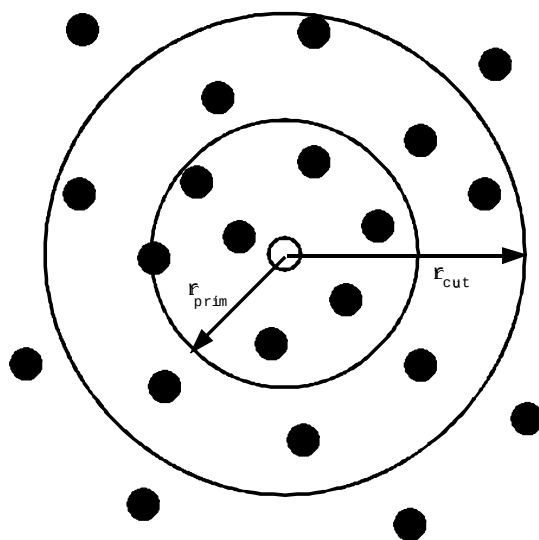
1. the reciprocal space terms are calculated only for the first two timesteps of the multi-step,
2. the contribution to the reciprocal space terms arising from primary interactions are immediately subtracted, leaving only the long-range components. (This is done in real space, by subtracting *erf* terms.);
3. the real space Coulombic forces arising from the secondary atoms are calculated in

the first two timesteps of the multi-step using the normal Ewald expressions (i.e. the *erfc* terms).

4. the Coulombic forces arising from primary atoms are calculated at every timestep in real space assuming the *full Coulombic force*;

In this way the Coulombic forces can be handled by the same multiple timestep scheme as the van der Waals forces. The algorithm is described in detail in [41].

Note that the accuracy of the algorithm is a function of the multi-step interval `multt`, and decreases as `multt` increases. Also, the algorithm is *non-symplectic* (i.e. it is not time reversible) and is therefore susceptible to energy drift. Its use with a thermostat is therefore advised.



The multiple timestep algorithm

The atoms surrounding the central atom (open circle) are classified as primary if they occur within a radius r_{prim} and secondary if outside this radius but within r_{cut} . Interactions arising from primary atoms are evaluated every timestep. Interactions from secondary atoms are calculated exactly for the first two steps of a multi-step and by extrapolation afterwards.

2.5.8 The DL_POLY_2 RESPA Multiple Timestep Implementation

The DL_POLY_2 RESPA Program is a variant of DL_POLY_2 that handles the time reversible, multiple timestep RESPA algorithm due to Tuckerman and Berne [17, 18] as implemented by Procacci and Marchi for atomic and rigid ion systems[19]. However, it is not applicable to systems which have rigid body molecules or constraints, and for this reason the RESPA code is kept separate from the DL_POLY_2 main source in the sub-directory called *respa*.

The DL_POLY_2 implementation allows the user to define three concentric regions around a given atom. Forces deriving from atoms in the innermost region are updated more frequently (i.e. every timestep) than those in the immediate outer region, which in turn are updated more frequently than those deriving from the outermost region. This (depending on the relative sizes of each region and the frequency of the forces update) can represent a great computational saving. This is coupled with the advantage that the algorithm is time reversible and therefore possesses long term stability.

The method is fully documented in [19]. A makefile (Makefile_respa) is available in the *build sub-directory*, which will build the RESPA program (DLRESPA.X). A test case may be found in the *data sub-directory*.

2.6 DL_POLY Parallelisation

DL_POLY_2 is a distributed parallel molecular dynamics package based on the Replicated Data parallelisation strategy [42, 43]. In this section we briefly outline the basic methodology. Users wishing to add new features DL_POLY_2 will need to be familiar with the underlying techniques as they are described (in greater detail) in references [30, 43]).

2.6.1 The Replicated Data Strategy

The Replicated Data (RD) strategy [42] is one of several ways to achieve parallelisation in MD. Its name derives from the replication of the configuration data on each node of a parallel computer (i.e. the arrays defining the atomic coordinates $\underline{r}_i$, velocities $\underline{v}_i$ and forces $\underline{f}_i$, for all N atoms $\{i : i = 1, \dots, N\}$ in the simulated system, are reproduced on every processing node). In this strategy most of the forces computation and integration of the equations of motion can be shared easily and equally between nodes and to a large extent be processed independently on each node. The method is relatively simple to program and is reasonably efficient. Moreover, it can be “collapsed” to run on a single processor very easily. However the strategy can be expensive in memory and have high communication overheads, but overall it has proven to be successful over a wide range of applications. These issues are explored in more detail in [42, 43].

Systems containing complex molecules present several difficulties. They often contain ionic species, which usually require Ewald summation methods [10, 44], and *intra*-molecular interactions in addition to *inter*-molecular forces. These are handled easily in the RD strategy, though the SHAKE algorithm [11] requires significant modification [30].

The RD strategy is applied to complex molecular systems as follows:

1. Using the known atomic coordinates $\underline{r}_i$, each node calculates a subset of the forces acting between the atoms. These are usually comprised of:
 - (a) atom-atom pair forces (e.g. Lennard Jones, Coulombic etc.);
 - (b) non-rigid atom-atom bonds;
 - (c) valence angle forces;
 - (d) dihedral angle forces;

- (e) improper dihedral angle forces.
- 2. The computed forces are accumulated in (incomplete) atomic force arrays $\underline{f}_i$ independently on each node;
- 3. The atomic force arrays are summed globally over all nodes;
- 4. The complete force arrays are used to update the atomic velocities and positions.

It is important to note that load balancing (i.e. equal and concurrent use of all processors) is an essential requirement of the overall algorithm. In DL_POLY_2 this is accomplished for the pair forces with an adaptation of the Brode-Ahlrichs scheme [22].

2.6.2 Distributing the Intramolecular Bonded Terms

DL_POLY_2 handles the intramolecular in which the atoms involved in any given bond term are explicitly listed. Distribution of the forces calculations is accomplished by the following scheme:

1. Every atom in the simulated system is assigned a unique index number from 1 to N ;
2. Every intramolecular bonded term U_{type} in the system has a unique index number i_{type} : from 1 to N_{type} where $type$ represents a bond, angle or dihedral.
3. A pointer array $key_{type}(n_{type}, i_{type})$ carries the indices of the specific atoms involved in the potential term labelled i_{type} . The dimension n_{type} will be 2, 3 or 4, if the term represents a bond, angle or dihedral.
4. The array $key_{type}(n_{type}, i_{type})$ is used to identify the atoms in a bonded term and the appropriate form of interaction and thus to calculate the energy and forces. Each processor is assigned the independent task of evaluating a block of $(Int(N_{total}/N_{nodes}))$ interactions.

The same scheme works for all types of bonded interactions. The global summation of the force arrays does not occur until all the force contributions, including nonbonded forces has been completed.

2.6.3 Distributing the Nonbonded Terms

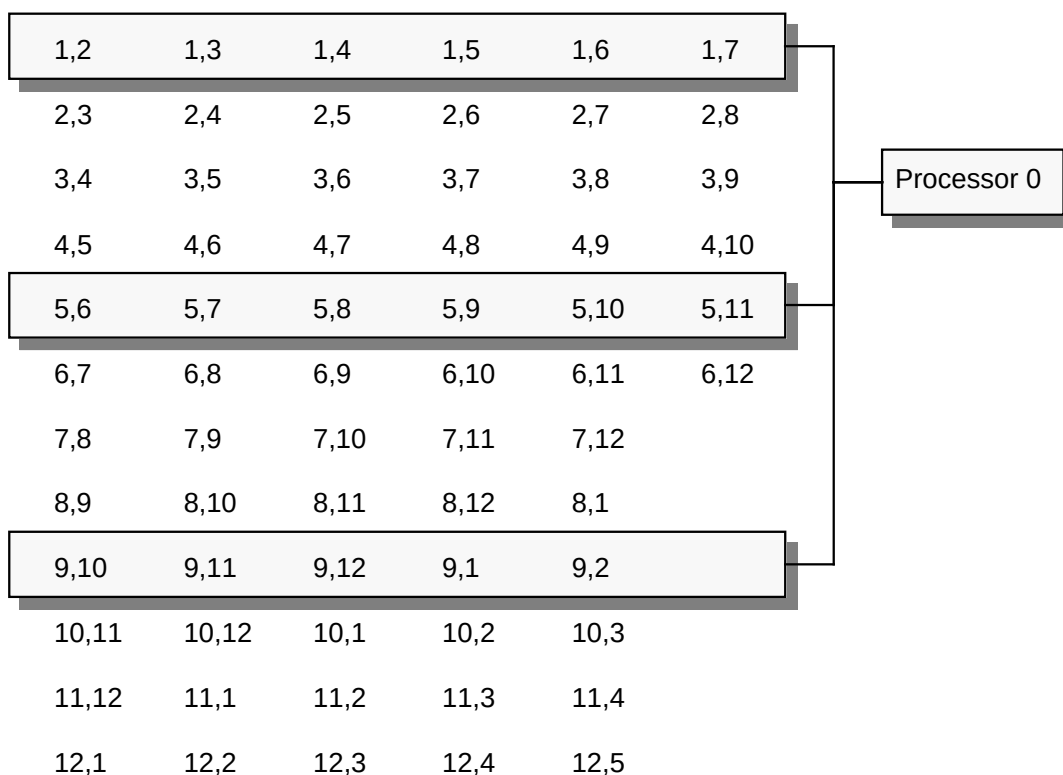
In DL_POLY_2 the nonbonded interactions are handled with a Verlet neighbour list [10] which is reconstructed at intervals during the simulation. This list records the indices of all ‘secondary’ atoms within a certain radius of each ‘primary’ atom; the radius being the cut-off radius (r_{cut}) normally applied to the nonbonded potential function, plus an additional increment (Δr_{cut}). The larger radius ($r_{cut} + \Delta r_{cut}$) permits the same list to be used for several timesteps without requiring an update. The frequency at which the list must be updated clearly depends on the thickness of the region Δr_{cut} . In RD, the neighbour list is constructed *simultaneously* on each node and in such a way as to share the total burden of the work equally between nodes. Each node is responsible for a unique set of nonbonded

interactions and the neighbour list is therefore different on each node. DL_POLY_2 uses a method based on the Brode-Ahlich scheme [22] (see figure below) to construct the neighbour list.

Additional modifications are necessary to handle the excluded atoms [43]. A distributed *excluded atoms list* is constructed by DL_POLY_2 at the start of the simulation. The list is constructed so that the excluded atoms are referenced in the same order as they would appear in the Verlet neighbour list if the bonded interactions were ignored, allowing for the distributed structure of the neighbour list.

Brode Ahlrichs Algorithm

12 Atoms, 4 processors



Schematic diagram of the parallel implementation of the Brode-Ahlich algorithm.

The diagram illustrates the reordering of the upper triangular matrix of $n(n-1)/2$ pair interactions so that the rows of the matrix are of approximately equal length. Each entry in the table consists of a primary atom index (constant within a row) and a “neighbouring” atom index. Rows are assigned sequentially to nodes. In the diagram node 0 deals with rows 1, 5 and 9, node 1 to rows 2, 6, and 10 etc.

When a charge group scheme (as opposed to an atomistic scheme) is used for the

non-bonded terms, the group-group interactions are distributed using the Brode-Ahlrichs approach. This makes the Verlet list considerably smaller, thus saving memory, but also results in a more “coarse grain” parallelism. The consequence of which is that performance with a large number of processors will degrade more quickly than with the atomistic scheme.

Once the neighbour list has been constructed, each node of the parallel computer may proceed independently to calculate the pair force contributions to the atomic forces.

2.6.4 Modifications for the Ewald Sum

For systems with periodic boundary conditions DL_POLY_2 employs the Ewald Sum to calculate the Coulombic interactions (see section 2.4.5).

Calculation of the real space component in DL_POLY_2 employs the algorithm for the calculation of the nonbonded interactions outlined above. The reciprocal space component is calculated using the schemes described in [44], in which the calculation can be parallelised by distribution of either $\underline{k}$ vectors or atomic sites. Distribution over atomic sites requires the use of a global summation of the $q_i \exp(-i\underline{k} \cdot \underline{r}_j)$ terms, but is more efficient in memory usage. Both strategies are computationally straightforward. Subroutine EWALD1 distributes over atomic sites and is often the more efficient of the two approaches. Subroutine EWALD1A distributes over the $\underline{k}$ vectors and may be more efficient on machines with large communication latencies.

Other routines required to calculate the ewald sum include EWALD2, EWALD3 and EWALD4. The first of these calculates the real space contribution, the second the self interaction corrections, and the third is required for the multiple timestep option.

2.6.5 Three Body Forces

DL_POLY_2 can calculate three body interactions of the valence angle type [45]. These are not dealt with in the same way as the normal nonbonded interactions. They are generally very short ranged and are most effectively calculated using a link-cell scheme [23]. No reference is made to the Verlet neighbour list nor the excluded atoms list. It follows that atoms involved in the same three-body term can interact via nonbonded (pair) forces and ionic forces also. The calculation of the three-body terms is distributed over processors on the basis of the identity of the central atom in the bond. A global summation is required to specify the atomic forces fully.

2.6.6 Metal Potentials

The simulation of metals by DL_POLY_2 makes use of density dependent potentials of the Sutton-Chen type [6]. The dependence on the atomic density presents no difficulty however, as this class of potentials can be resolved into pair contributions. This permits the use of the distributed Verlet neighbour list outlined above.

2.6.7 Summing the Atomic Forces

The final stage in the RD strategy, is the global summation of the atomic force arrays. This must be done After all the contributions to the atomic forces have been calculated. To do this DL_POLY_2 employs a global summation algorithm [42], which is generally a system specific utility.

Similarly, the total configuration energy and virial must be obtained as a global sum of the contributing terms calculated on all nodes.

2.6.8 The RD-SHAKE and Parallel QSHAKE Algorithms

The RD-SHAKE algorithm is a parallel adaptation of SHAKE couched in the Replicated Data strategy. The essentials of the RD-SHAKE method are as follows.

1. The bond constraints acting in the simulated system are shared equally between the processing nodes.
2. Each node makes a list recording which atoms are bonded by constraints it is to process. Entries are zero if the atom is not bonded.
3. A copy of the array is passed to each other node in turn. The receiving node compares the incoming list with its own and keeps a record of the shared atoms and the nodes which share them.
4. In the first stage of the SHAKE algorithm, the atoms are updated through the usual Verlet algorithm, without regard to the bond constraints.
5. In the second (iterative) stage of SHAKE, each node calculates the incremental correction vectors for the bonded atoms in its own list of bond constraints. It then sends specific correction vectors to all neighbours that share the same atoms, using the information compiled in step 3.
6. When all necessary correction vectors have been received and added the positions of the constrained atoms are corrected.
7. Steps 5 and 6 are repeated until the bond constraints are converged.
8. After convergence the coordinate arrays on each node are passed to all the other nodes. The coordinates of atoms that are *not* in the constraint list of a given node are taken from the incoming arrays (an operation we term *splicing*).
9. Finally, the change in the atom positions is used to calculate the atomic velocities.

This scheme contains a number of non-trivial operations, which are described in detail in [30]. However some general comments are worth making.

The compilation of the list of constrained atoms on each node, and the circulation of the list (items 1 - 3 above) need only be done once in any given simulation. It also transpires that in sharing bond constraints between nodes, there is an advantage to keeping

as many of the constraints pertaining to a particular molecule together on one node as is possible within the requirement for load balancing. This reduces the data that need to be transferred between nodes during the iteration cycle. It is also advantageous, if the molecules are small, to adjust the load balancing between processors to prevent shared atoms. The loss of balance is compensated by the elimination of communications during the SHAKE cycle. These techniques are exploited by DL_POLY_2 .

The QSHAKE algorithm is an extension of the SHAKE algorithm for constraint bonds between rigid bodies. The parallel strategy is very similar to that of RD-SHAKE. The only significant difference is that increments to the atomic forces, not the atomic positions, are passed between processors at the end of each iteration.

Chapter 3

DL_POLY_2 Construction and Execution

Scope of Chapter

This chapter describes how to compile a working version of DL_POLY_2 and how to run it.

3.1 Constructing DL_POLY_2 : an Overview

3.1.1 Constructing the Standard Version

DL_POLY_2 was designed as a package of useful subroutines rather than a single program, which means that users were to be able to construct a working simulation program of their own design from the subroutines available, which is capable of performing a specific simulation. However we recognise that many, perhaps most, users will be content with creating a standard version that covers all of the possible applications and for this reason we have provided the necessary tools to assemble such a version. The method of creating the standard version is described in detail in this chapter, however a brief step-by-step description follows.

1. DL_POLY_2 is supplied as a UNIX compressed tar file. This must be uncompressed and un-tared to create the DL_POLY_2 directory (section 1.4).
2. In the *build* subdirectory you will find the required DL_POLY_2 makefile (see section 3.2.1 and Appendix A, where the main Makefile is listed). This must be copied into the subdirectory containing the relevant source code. In most cases this will be the *source* subdirectory.
3. The makefile is executed with the appropriate keywords (section 3.2.1) which select for specific computers and if a parallel machine is used, the appropriate communication software.
4. The makefile produces the executable version of the code, which as a default will be named DLPOLY.X and located in the *execute* subdirectory.
5. Versions of DL_POLY_2 from 2.13 also have a Java GUI. The files for this are stored in the subdirectory *java*. Compilation of this is simple and requires running the javac compiler and the jar utility. Details for these procedures are provided later in this document.
6. To run the executable for the first time you require the files CONTROL, FIELD and CONFIG (and possibly TABLE if you have tabulated potentials). These must be present in the directory from which the program is executed. (See section 4.1 for the description of the input files.)
7. Executing the program will produce the files OUTPUT, REVCON and REVIVE (and optionally STATIS, HISTORY, RDFDAT and ZDNDAT) in the executing directory. (See section 4.2 for the description of the output files.)

This simple procedure is enough to create a standard version to run the supplied test cases. (For versions preceding 2.11, it will not run the larger benchmark cases however.) Creating a larger executable, or one which is tailored to an application different from the test cases is more complicated. The main difficulty centres on assigning appropriate array dimensions to meet the application's needs. In versions of DL_POLY_2 after 2.11 these

dimensions are calculated by the subroutine `PARSET.F` and its support routines `CFGSCAN.F`, `CONSCAN.F` and `FLDSCAN.F`. These support routines scan the `CONFIG`, `CONTROL` and `FIELD` files at the start of a run and estimate the required array sizes. (A description of the individual arrays found in `DL_POLY_2` is provided in Appendix C of the `DL_POLY_2` Reference Manual.)

In versions preceeding 2.11, the size of the `DL_POLY_2` executable is determined by the the `DL_PARAMS.INC` (include) file (see section 8.1.1), which specifies all the FORTRAN parameters of the code and is included in each subroutine at compile time. This file is found in the *source* directory and is internally documented (it is also documented in chapter 8). Reconstructing the `DL_PARAMS.INC` for a new simulation is the most difficult aspect of preparing a `DL_POLY_2` executable. If the parameters are not appropriate, `DL_POLY_2` will complain if any parameter is too small and abort in execution. Each such error will entail a recompilation of the code, and `DL_POLY_2` generally detects only one error at a time! Fortunately, the process of creating a new `DL_PARAMS.INC` file is greatly assisted by the utility program `PARSET` (see section 7.3.10). Provided the `CONTROL`, `CONFIG` and `FIELD` files are available, `PARSET` will create a suitable parameters file for you. (See section 3.2.2).

3.1.2 Constructing Nonstandard Versions

In constructing a nonstandard `DL_POLY_2` simulation program, the first requirement is for the user to write a program to function as the root segment. The *source* directory contains an example of such a root program: `DLPOLY`. This root program calls the major routines required to perform the simulation and also controls the normal “molecular dynamics cycle” which consists of forces calculation followed by integration of the equation of motion. `DLPOLY` also monitors the *cpu* usage and brings about a controlled termination of the program if the usage approaches the allotted job time within a pre-set closure time. Lastly, `DLPOLY` is the routine that first opens the `OUTPUT` file (section 4.2.2), which provides the summary of the job. Users are recommended to study the `DLPOLY` root as a model for other implementations of the package they may wish to construct. The dependencies and calling hierarchies of all the `DL_POLY_2` subroutines can be found in the Appendix of the `DL_POLY_2` Reference Manual.

If additional functionality is added to `DL_POLY_2` by the user, the `PARSET.F` routine (and its support routines) will need modifying to allow specification of the dimensions of any new arrays. (A description of the existing arrays found in `DL_POLY_2` is provided in the Appendix of the `DL_POLY_2` Reference Manual.)

Any molecular dynamics simulation performs five different kinds of operation: initialisation; forces calculation; integration of the equations of motion; calculation of system properties; and job termination. It is worth considering these operations in turn and to indicate which `DL_POLY_2` routines are available to perform them. We do not give a detailed description, but provide only a guide. Readers are recommended to examine the different routines described in chapter 8 of the `DL_POLY_2` Reference Manual for further details (particularly regarding further dependencies i.e. additional routines that must be called.) The following outline assumes a system containing flexible molecules held together by rigid

bonds, but without rigid bodies.

Initialisation requires firstly that the program determine what kind of parallel machine it is running on. The routine `MACHINE` determines how many processing nodes are being used and also returns the node identity to each process. Next the job control information is required; this is obtained by the routine `SIMDEF`, which reads the `CONTROL` file (section 4.1.1). The description of the system to be simulated: the types of atoms and molecules present and the intermolecular forces; are obtained by the `SYSDEF` routine, which reads the `FIELD` file (section 4.1.3). Lastly, the atomic positions and velocities must be provided. These are obtained by the `SYSGEN` routine, which reads the `CONFIG` file (section 4.1.2) and also generates the initial velocities if required to do so. If the system contains constraint bonds, the routine `PASSCON` is required to process molecular connectivity data and establish the communication procedure between nodes, and the `QUENCH` routine is required to set the starting velocities correctly. Also needed in the initialisation, is the routine `FORGEN`, which constructs the interpolation arrays for the short-range forces calculations, and the routine `EXCLUDE` which identifies atoms that are explicitly chemically bonded through bonds, constraints or valence angles. The resulting list is known as the *excluded atoms* list.

The calculation of the pair forces represents the bulk of any simulation. A Verlet neighbour list is used by `DL_POLY_2` in calculating the atomic forces. The routine that constructs this is called `PARLST`. This routine builds the neighbour list taking into account the occurrence of atoms in the *excluded atoms* list. The routine `SRFRCE` calculates the short-range (van der Waals) forces, making use of the `IMAGES` routine to handle any periodic boundary conditions. Coulombic forces are handled by a variety of routines: `COUL0`, `COUL1` and `COUL2` handle Coulombic forces *without* periodic boundaries; `EWALD1`, `EWALD2` and `EWALD3` are used for systems *with* periodic boundaries (an additional routine: `EWALD4` is necessary for the multiple timestep algorithm). Intramolecular forces require the routines `ANGFRC`, `BNDFRC` and `DIHFRC`. If the multiple timestep algorithm is required, the routine `MULTIPLE` must be used to call the various forces routines. It also calls the `PRIMLST` routine to split the interaction list into primary and secondary neighbours. The decision to update the neighbour list is handled by the routine `VERTEST`. The routine `EXTNFLD` is required if the simulated system has an external force field (e.g. electrostatic field) operating. To help with equilibration simulations, the routine `FCAP` is sometimes required to reduce the magnitude of badly equilibrated forces. Since `DL_POLY_2` is based on the replicated data strategy, a global sum routine (`GDSUM`) is required to sum the atomic forces on all nodes.

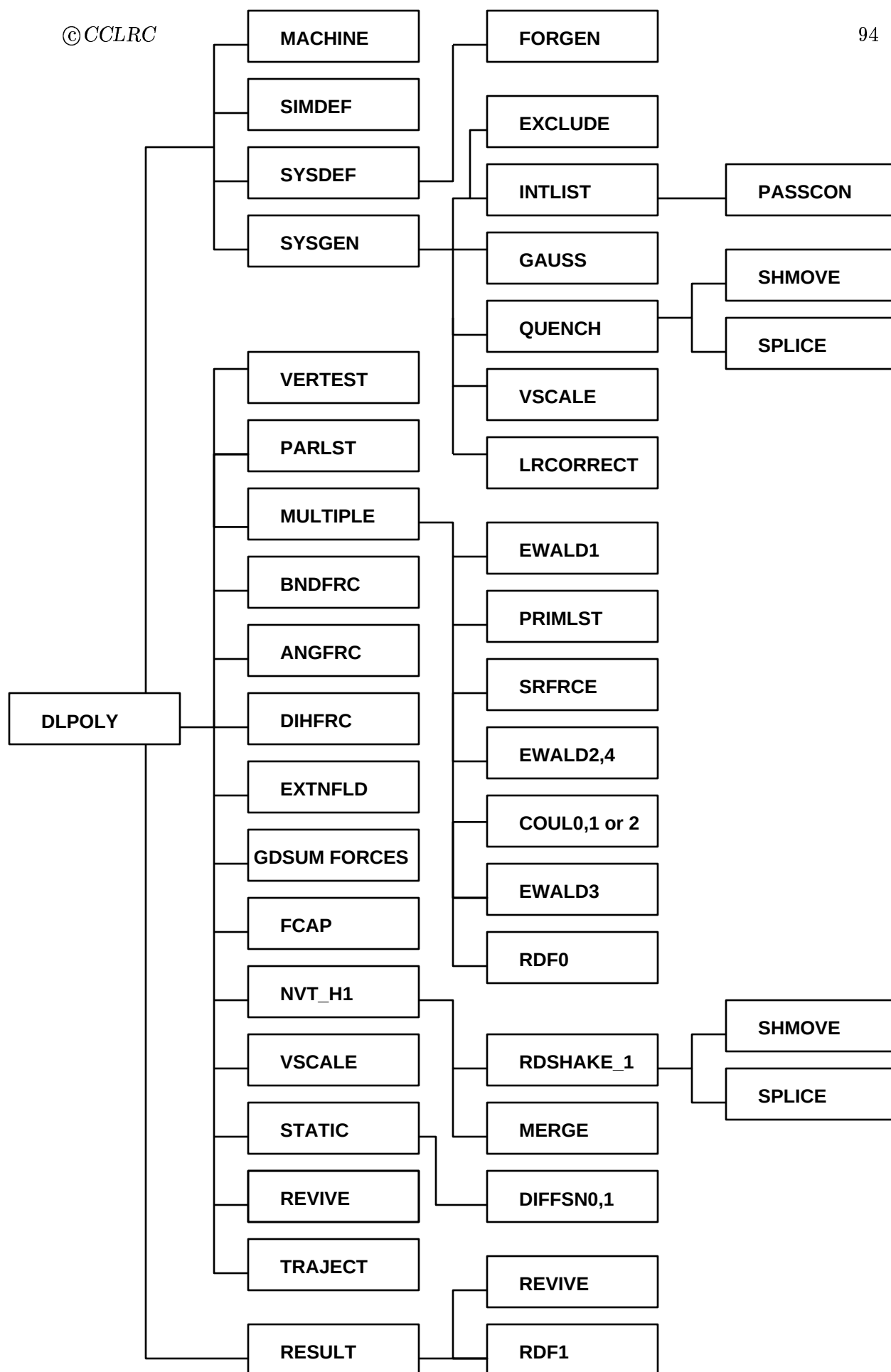
Integration of the equations of motion is handled by one of the routines listed and described in section 2.5. For example routines `NVE_0`, `NVT_E0`, `NVT_H0`, `NVT_B0` etc. are used if no constraint forces are present. These routines treat the NVE, Evans-NVT, Hoover-Nosé-NVT and NVT-Berendsen ensembles respectively. The corresponding versions of these routines which handle constraint forces are `NVE_1`, `NVT_E1`, `NVT_H1` or `NVT_B1`. These versions call the routine `RDSHAKE_1` to handle the constraints. `RDSHAKE_1` itself calls a number of additional routines: `MERGE`, `SHMOVE` and `SPLICE`. For *ad hoc* temperature scaling, the routine `VSCALEG` is required.

As mentioned elsewhere, `DL_POLY_2` does not contain many routines for computing system properties during a simulation. Radial distributions may be calculated however, using the routines `RDF0` and `RDF1`. Similarly `DIFFSN0` and `DIFFSN1` calculate approximate

mean square displacements. Ordinary thermodynamic quantities are calculated by the routine `STATIC`, which also writes the `STATIS` file (section 4.2.7). Routine `TRAJECT` writes the `HISTORY` (section 4.2.1) file for later analysis.

Job termination is handled by the routine `RESULT` which writes the final summaries in the `OUTPUT` file and dumps the restart files `REVIVE` and `REVCN` (sections 4.2.4 and 4.2.3 respectively).

An idea of the construction of a `DLPOLY_2` program can be obtained from the following flowchart. The example represents a `DLPOLY_2` program which uses the multiple timestep algorithm, with bond constraints and the Nosé-Hoover thermostat.



3.2 Compiling and Running DL_POLY_2

3.2.1 Compiling the Source Code

When you have obtained DL_POLY_2 from Daresbury Laboratory and unpacked it, your next task will be to compile it. To aid compilation a general makefile (called “Makefile”) has been provided in the sub-directory *build* (see Appendix A to this document). The general DL_POLY_2 makefile will build an executable with a wide range of functionality - sufficient for the test cases and for most users’ requirements. Other makefiles are found in the *build* sub-directory for variants of DL_POLY_2, such as the *respa* version. Users will need to modify the makefile if they are to add additional functionality to the code, or if it requires adaptation for a non specified computer. Modifications may also be necessary for the Smooth Particle Mesh Ewald method if a system specific 3D FFT routine is desired (see below: “Modifying the makefile”).

Note the following system requirements for a successful build of DL_POLY_2 .

1. a FORTRAN 90 compiler;
2. a C compiler;
3. the Java SDK from Sun Microsystems (if the GUI is required).
4. a UNIX operating system (or Windows if a PC version is required).

Copy the makefile from the *build* sub-directory to the *source* sub-directory and run it there - it will create the executable in the *execute* sub-directory. The compilation of the program is initiated by typing the command:

make target

where *target* is the specification of the required machine (e.g. *sp2-mpi*). For many computer systems this is all that is required to compile a working version of DL_POLY_2 . (To determine which targets are already defined in the makefile, typing the command *make* without a nominated target will produce a list of known targets.)

The full specification of the *make* command is as follows

make <TARGET= ... > <STRESS=... > <TYPE=... > <EX=... > <BINROOT=... >

where some (or all) of the keywords may be omitted. The keywords and their uses are described below. Note that keywords may also be set in the unix environment (e.g. with the “setenv” command in a C-shell).

The makefile first checks for the existence of the DL_POLY_2 *C preprocessor* DPP. If the executable version is not present it will be compiled. Afterwards each FORTRAN routine is taken in turn and scanned by DPP. The purpose of this is to activate the required features in the code (and extract non-selected features from the original source code). For example if MPI is specified, all other non-MPI message passing calls are removed. If NOSTRESS

is specified, the statements for calculating the stress tensor are removed. The result of the preprocessing is a *temporary* FORTRAN file which is then compiled to produce the object code. If all is well, the makefile will combine the object code with the system libraries and produce the executable.

For PCs running Windows, the makefile assumes the user has installed the Cygwin Unix API available from <http://sources.redhat.com/cygwin>. The recommended FORTRAN 90 compiler is Compaq Visual Fortran (see: <http://www.compaq.com/fortran/visual>). Both of these are copyrighted products.

3.2.1.1 Keywords for the Makefile

1. TARGET

The TARGET keyword indicates which kind of computer the code is to be compiled for. This **must** be specified - there is no default value. Valid targets can be listed by the makefile if the command *make* is typed, without arguments. The list frequently changes as more targets are added and redundant ones removed. Users are encouraged to extend the Makefile for themselves, using existing targets as examples.

2. STRESS

The STRESS keyword activates the code that calculates the stress tensor in the DL_POLY_2 code. The arguments are:

- NOSTRESS - if the stress tensor is not required;
- STRESS - if the stress tensor is required (default).

Calculation of the stress tensor is essential if “constant stress” (Rahman-Parrinello) MD is required.

3. TYPE

The TYPE keyword creates variants of the DL_POLY_2 code which determine which scheme is to be used for the interpolation of the short-range potential and force arrays. The arguments are:

- *3pt* - compile with 3-point interpolation (default);
- *4pt* - compile with 4-point interpolation;
- *rsq* - compile with r^2 interpolation.

A discussion of the merits of the different interpolation methods is given below (section 3.2.1.3).

4. EX

The EX keyword specifies the executable name. The default name for the executable is “DLPOLY.X”.

5. BINROOT

The BINROOT keyword specifies the directory in which the executable is to be stored. The default setting is “./execute”.

3.2.1.2 Modifying the Makefile

1. Changing the TARGET

If you do not intend to run DL_POLY_2 on one of the specified machines, you must add appropriate lines to the makefile to suit your circumstances. The safest way to do this is to modify an existing TARGET option for your purposes. The makefile supplied with DL_POLY_2 contains examples for serial and MPI environments as well as for Intel and Cray (T3E) parallel machines so you should find one close to your requirements. You must of course be familiar with the appropriate invocation of the FORTRAN compiler for your local machine and also any alternatives to MPI your local machine may be running. After DL_POLY_2 Version 2.12, the required C preprocessor is supplied with the source code, and is called dpp.c (written by J. Genornowicz, Southampton University, 1999). If you wish to compile for MPI systems remember to ensure the appropriate library directories are accessible to you. If you intend the program for a single processor machine use the flag -DSERIAL (see the examples in the makefile).

2. Enabling the Smooth Particle Mesh Ewald

Users of DL_POLY_2 Version 2.12 (and above) should note that modifications to the makefile will be needed if a system specific Fast Fourier Transform (FFT) routine is required for the Smooth Particle Mesh Ewald (SPME), otherwise a non-optimal default is used. . A good replacement is the public domain FFTW library, for which examples exist in the makefile. The user who wishes to use FFTW should check the relevant part of the makefile to see how this is enabled (this is generally made clear in the makefile comments). The FFTW option is enabled by setting the parameter FFTW_LIBRARY to specify the location of the FFTW software, and by inserting the flag -DFFTW in the CPFLAGS list (see above). Note that for most parallel systems represented in the makefile the appropriate FFT is already enabled (which is usually not FFTW). This is particularly the case for Cray T3E, IBM SP/2 and parallel SGI machines.

3. MPI implementations

The implementation of MPI may differ between sites. On some systems the Fortran callable subroutine names are expected to end with an underscore (“_”). If this is the case the flag -DMPIU must be included as part of the C-preprocessing flags and the file “mpif.h” copied from the MPI library directory into the *source* directory. Alternatively you can set the path to the MPI library either the “-I” option on the

C-preprocessing flags. The DL_POLY_2 makefile assumes you have copied the file over (see the entry *hp-mpi:* in the makefile). This appends an underscore to all MPI subroutine names and to the name of the MPI common block. If the underscores are not required the flag must be omitted (see the entry *sp2-mpi:*).

When using MPI you need a copy of the MPI include file “mpif.h” in the source directory. On many machines this is stored in the /usr/include directory, in which case the make procedure should find it automatically (as it also does for IBM SP/2 and Cray T3E machines, where it is stored elsewhere). However, if the make reports a failure to find the “mpif.h” file, you must amend the makefile to ensure that it copied from the true location before C-preprocessing is attempted. There are several examples in the makefile of how this is done.

4. Problems with optimization ?

Some subroutines do not compile correctly when using optimization on some compilers. This is not the fault of the DL_POLY_2 code, but of the compiler concerned. This is circumvented by compiling the offending subroutines unoptimised. See the entries for various machines in the makefile to see how this is done if you experience problems with other subroutines.

5. Changing the Timer

The only other routine likely to cause problems is TIMCHK, which requires a machine specific timer. The makefile should select the appropriate timer for you, but if the timer routine you require is not included you will need to add the appropriate lines of code. TIMCHK returns the elapsed time in seconds. Note that the C routine ETIME may be used on many unix systems. It is used as the default timer by DL_POLY_2 on serial systems.

6. Adding new functionality

To include a new subroutine in the code simply add *subroutine.o* to the list of object names in the makefile. The simplest way is to add names to the “OBJ_ALL” list.

3.2.1.3 Note on Interpolation Schemes

In DL_POLY_2 the short-range (Van der Waals) contributions to energy and force are evaluated by interpolation of tables constructed at the beginning of execution. DL_POLY_2 caters for three different interpolation schemes: 3-point and 4-point in r -space and linear interpolation in r^2 -space. Tabulation in r^2 avoids the use of the square root function in evaluation of the non-bonded interactions, and thus typically decreases execution time by 10-15 %. Note that tabulation in r^2 usually requires more grid points (and hence more memory) than tabulation in r . This is to ensure sufficient accuracy is retained at small r .

A guide to the *minimum* number of grid points (**mxgrid**) required for interpolation in r to give good energy conservation in a simulation is:

$$\text{mxgrid} \geq 100(\text{rcut}/\text{rmin})$$

where **rmin** is the *smallest* position minimum of the non-bonded potentials in the system. The parameter **mxgrid** is defined in the DL_PARAMS.INC file, and must be set before compilation.

A guide to the *minimum* number of grid points required for interpolation in r^2 is:

$$\text{mxgrid} \geq 100(\text{rcut}/\text{rmin})^2$$

where **rmin** is again the smallest position minimum of the non-bonded potentials in the system.

For users in doubt as whether to use r or r^2 -space interpolation we recommend the former. This is because tabulation in r is less demanding on memory requirements and less prone to inaccuracy should too small a value of **mxgrid**, or too large a value of **rcut**, be used. Tabulation in r is therefore the default option for DL_POLY, r^2 interpolation can be specified at compile time by ‘making’ the executable with the directive **TYPE=rsq**.

The other issue of concern to users is the choice of 3 or 4 point schemes in r -space interpolation. The relative merits are as follows: 4 point interpolation may permit a smaller number of grid points to be used in the interpolation tables thus saving on memory requirements. 3 point interpolation is quicker than 4 point interpolation and normally sufficiently accurate. The choice involves decisions about speed, accuracy and memory requirements. 3-point interpolation is the default option.

A utility program TABCHK is provided in the DL_POLY *utility* sub-directory to help users choose a sufficiently accurate interpolation scheme (including array sizes) for their needs.

3.2.2 Compiling Older Versions with the Utility Program PARSET

A particular difficulty in creating a working version of DL_POLY_2 **prior** to version 2.11 is creating the DL_PARAMS.INC file (section 8.1.1), which specifies (among other things) the array sizes for the compiled code. The copy of DL_PARAMS.INC supplied with DL_POLY_2 contains settings appropriate for the test cases and some of the benchmarks but is not guaranteed to be appropriate for any user’s specific requirements. Users must determine their own requirements and so produce their own version of DL_PARAMS.INC before compiling a working version. While DL_POLY_2 contains many error checks to prevent the arrays being exceeded during a run, it is tedious to have to recompile the code each time an error is detected. Furthermore the large number of parameters required to specify the entire code makes it very unlikely that the code will run successfully first time, or even after several attempts.

To assist with this difficulty, the utility program PARSET has been created (section 7.3.10). It can be used to create a DL_PARAMS.INC file that is suitable for the simulation being attempted. It works by scanning the DL_POLY_2 input files (CONTROL, FIELD

and CONFIG) and then writing out a sample DL_PARAMS.INC file, which has the name NEW_PARAMS.INC.

PARSET is not foolproof however. The user must prepare the correct input files beforehand, which can be difficult without the error checking the DL_POLY_2 code affords. Also, because of the complexity of the requirements, PARSET can only supply a reasonable estimate of some of the array dimensions, and these may not be quite right for every case. (It will be reasonably close however). Notwithstanding the limitations of PARSET its use as a labour saving device is recommended.

PARSET is described in detail in section 7.3.10. The program is found in the *utility* subdirectory.

3.2.3 Running DL_POLY_2

To run the DL_POLY_2 executable (DLPOLY.X) you will initially require three, possibly four, input data files, which you must create in the *execute* sub-directory, (or whichever sub-directory you keep the executable program.) The first of these is the CONTROL file (section 4.1.1), which indicates to DL_POLY_2 what kind of simulation you want to run, how much data you want to gather and for how long you want the job to run. The second file you need is the CONFIG file (section 4.1.2). This contains the atom positions and, depending on how the file was created (e.g. whether this is a configuration created from 'scratch' or the end point of another run), the velocities also. The third file required is the FIELD file (section 4.1.3), which specifies the nature of the intermolecular interactions, the molecular topology and the atomic properties, such as charge and mass. Sometimes you will require a fourth file: TABLE (section 4.1.5), which contains the potential and force arrays for functional forms not available within DL_POLY_2 (usually because they are too complex e.g. spline potentials).

Examples of input files are found in the *data* sub-directory, which can be copied into the *execute* subdirectory using the *select* macro found in the *execute* sub-directory.

A successful run of DL_POLY_2 will generate several data files, which appear in the *execute* sub-directory. The most obvious one is the file OUTPUT (section 4.2.2), which provides an effective summary of the job run: the input information; starting configuration; instantaneous and rolling-averaged thermodynamic data; final configurations; radial distribution functions (RDFs); and job timing data. The OUTPUT file is human readable. Also present will be the restart files REVIVE (section 4.2.4) and REVCON (section 4.2.3). REVIVE contains the accumulated data for a number of thermodynamic quantities and RDFs, and is intended to be used as the input file for a following run. It is *not* human readable. The REVCON file contains the *restart configuration* i.e. the final positions, velocities and forces of the atoms when the run ended and is human readable. The STATIS file (section 4.2.7) contains a catalogue of instantaneous values of thermodynamic and other variables, in a form suitable for temporal or statistical analysis. Finally, the HISTORY file (section 4.2.1) provides a time ordered sequence of configurations to facilitate further analysis of the atomic motions. Depending on which version of the TRAJECT subroutine you compiled in the code, this file may be either formatted (human readable) or unformatted. You may move these output files back into the *data* sub-directory using the *store* macro

found in the *execute* sub-directory.

Note that versions of DL_POLY_2 after 2.10 may also create the files RDFDAT and ZDNDAT, containing the RDF and Z-density data respectively. They are both human readable files.

3.2.4 Restarting DL_POLY_2

The best approach to running DL_POLY_2 is to define from the outset precisely the simulation you wish to perform and create the input files specific to this requirement. The program will then perform the requested simulation, but may terminate prematurely through error, inadequate time allocation or computer failure. Errors in input data are your responsibility, but DL_POLY_2 will usually give diagnostic messages to help you sort out the trouble. Running out of job time is common and provided you have correctly specified the job time variables (using the **close time** and **job time** directives - see section 4.1.1) in the CONTROL file, DL_POLY_2 will stop in a controlled manner, allowing you to restart the job as if it had not been interrupted.

To restart a simulation after normal termination you will again require the CONTROL file, the FIELD (and TABLE) file, and a CONFIG file, which is the exact copy of the REVCON file created by the previous job. You will also require a new file: REVOLD (section 4.1.4), which is an exact copy of the previous REVIVE file. If you attempt to restart DL_POLY_2 without this additional file available, the job will fail. Note that DL_POLY_2 will append new data to the existing STATIS and HISTORY files if the run is restarted, other output files will be **overwritten**.

In the event of machine failure, you should be able to restart the job in the same way from the surviving REVCON and REVIVE files, which are dumped at intervals to meet just such an emergency. In this case check carefully that the input files are intact and use the HISTORY and STATIS files with caution - there may be duplicated or missing records. The reprieve processing capabilities of DL_POLY_2 are not foolproof - the job may crash while these files are being written for example, but they can help a great deal. You are advised to keep backup copies of these files, noting the times they were written, to help you avoid going right back to the start of a simulation.

You can also extend a simulation beyond its initial allocation of timesteps, provided you still have the REVCON and REVIVE files. These should be copied to the CONFIG and REVOLD files respectively and the directive **timesteps** adjusted in the CONTROL file to the new total number of steps required for the simulation. For example if you wish to extend a 10000 step simulation by a further 5000 steps use the directive **timesteps 15000** in the CONTROL file and include the **restart** directive. You can use the **restart scale** directive if you want to reset the temperature at the restart, but note that this also resets all internal accumulators (timestep included) to zero.

3.3 A Guide to Preparing Input Files

The CONFIG file and the FIELD file can be quite large and unwieldy particularly if a polymer or biological molecule is involved in the simulation. This section outlines the paths

to follow when trying to construct files for such systems. The description of the DL_POLY_2 force field in chapter 2 is essential reading. The various utility routines mentioned in this section are described in greater detail in chapter 7. Many of these have been incorporated into the DL_POLY_2 Graphical User Interface [20] and may be conveniently used from there.

3.3.1 Inorganic Materials

The utility GENLAT can be used to construct the CONFIG file for relatively simple lattice structures. Input is interactive. The FIELD file for such systems are normally small and can be constructed by hand. The utility GENLAT.TO constructs the CONFIG file for truncated-octahedral boundary conditions. Otherwise the input of force field data for crystalline systems is particularly simple, if no angular forces are required (notable exceptions to this are zeolites and silicate glasses - see below). Such systems require only the specification of the atomic types and the necessary pair forces. The reader is referred to the description of the DL_POLY_2 FIELD file for further details (section 4.1.3).

DL_POLY_2 allows the simulation of zeolites and silicate (or other) glasses. Both these materials require the use of angular forces to describe the local structure correctly. In both cases the angular terms are included as *three body terms*, the forms of which are described in chapter 2. These terms are entered into the FIELD file with the pair potentials. Note that you cannot use truncated octahedral or rhombic dodecahedral boundary conditions in conjunction with three body forces, due to the use of the link-cell algorithm for evaluating the forces.

An alternative way of handling zeolites is to treat the zeolite framework as a kind of macromolecule (see below). This was the suggested method for users of DL_POLY_1.1, and experience showed that this is quite adequate, provided it is remembered that the periodicity of the system requires that “bonds” be described that cross the boundaries of the MD cell and link with images of atoms from the opposite side. Specifying all this is tedious and is best done computationally: what is required is to determine the nearest image neighbours of all atoms and assign appropriate bond and valence angle potentials. (This may require the definition of new bond forces in subroutine BNDFRC, but this is easy.) What must be avoided at all costs is specifying the angle potentials *without* specifying bond potentials. In this case DL_POLY_2 will automatically cancel the non-bonded forces between atoms linked via valence angles and the system will collapse. The advantage of this method is that the calculation is likely to be faster using three-body forces. This method is not recommended for amorphous systems.

3.3.2 Macromolecules

Simulations of proteins are best tackled using the package DLPROTEIN [21] which is an adaptation of DL_POLY specific to protein modelling. However you may simulate proteins and other macromolecules with DL_POLY_2 if you wish. This is described below.

If you select a *protein* structure from a SEQNET file (e.g. from the Brookhaven database), use the utility PROSEQ to generate the file CONFIG. This will then function as input for DL_POLY_2. Some caution is required here however, as the protein structure

may not be fully determined and atoms may be missing from the CONFIG file.

If you have the “edit.out” file produced by AMBER for your molecule use this as the CONNECT_DAT input file for the utility AMBFORCE. AMBFORCE will produce the DL_POLY_2 FIELD and CONFIG files for your molecule.

If you do not have the “edit.out” file things are a little more tricky, particularly in coming up with appropriate partial charges for atomic sites. However there are a series of utilities that will at least produce the CONNECT_DAT file for use with AMBFORCE. We now outline these utilities and the order in which they should be used.

If you have a structure from the Cambridge Structural database (CSDB) then use the utility FRACCON to take fractional coordinate data and produce a CONNECT_DAT and “ambforce.dat” file for use with AMBFORCE. Note that you will need to modify FRACCON to get the AMBER names correct for sites in your molecule. The version of FRACCON supplied with DL_POLY_2 is specific to the valinomycin molecule.

If you require an all atom force field and the database file does not contain hydrogen positions then use the utility FRACFILL in place of FRACCON. FRACCON produces an output file HFILL which should then be used as input for the utility HFILL. The HFILL utility fills out the structure with the missing hydrogens. (Note that you may need to know what the atomic charges are in some systems, for example the AMBER charges from the literature.)

Note: with minor modifications the utilities FRACFILL and FRACCON can be used on structures from databases other than the Cambridge structural database.

3.3.3 Adding Solvent to a Structure

The utility WATERADD adds water from an equilibrated configuration of 256 SPC water molecules at 300 K to fill out the MD cell. The utility SOLVADD fills out the MD box with single-site solvent molecules from a f.c.c lattice. The FIELD files will then need to be edited to account for the solvent molecules added to the file.

Hint: to save yourself some work in entering the non-bonded interactions variables involving solvent sites to the FIELD file put two bogus atoms of each solvent type at the end of the CONNECT_DAT file (for AMBER force-fields) the utility AMBFORCE will then evaluate all the non-bonded variables required by DL_POLY_2 . Remember to delete the bogus entries from the CONFIG file before running DL_POLY_2 .

3.3.4 Analysing Results

DL_POLY_2 is not designed to calculate every conceivable property you might wish from a simulation. Apart from some obvious thermodynamic quantities and radial distribution functions, it does not calculate anything beyond the atomic trajectories on-line. You must therefore be prepared to post-process the HISTORY file if you want other information. There are some utilities in the DL_POLY_2 package to help with this, but the list is far from exhaustive. In time, we hope to have many more. Our users are invited to submit code to the DL_POLY_2 *public* library to help with this.

The utilities available are described in the DL_POLY_2 Reference Manual, Chapter 7. Users should also be aware that many of these utilities are incorporated into the DL_POLY_2

Graphical User Interface [20] and the Java GUI (see chapter 5.1.

3.3.5 Choosing Ewald Sum Variables

3.3.5.1 Ewald sum and SPME

This section outlines how to optimise the accuracy of the Ewald sum parameters for a given simulation. In what follows the directive **spme** may be used anywhere in place of the directive **ewald** if the user wishes to use the Smooth Particle Mesh Ewald method.

As a guide to beginners DL_POLY_2 will calculate reasonable parameters if the **ewald precision** directive is used in the CONTROL file (see section 4.1.1). A relative error (see below) of 10^{-6} is normally sufficient so the directive

ewald precision 1d-6

will cause DL_POLY_2 to evaluate its best guess at the Ewald parameters α , **kmax1**, **kmax2** and **kmax3**. (The user should note that this represents an *estimate*, and there are sometimes circumstances where the estimate can be improved upon. This is especially the case when the system contains a strong directional anisotropy, such as a surface.) These four parameters may also be set explicitly by the **ewald sum** directive in the CONTROL file. For example the directive

ewald sum 0.35 6 6 8

would set $\alpha = 0.35 \text{ \AA}^{-1}$, **kmax1** = 6, **kmax2** = 6 and **kmax3** = 8. The quickest check on the accuracy of the Ewald sum is to compare the Coulombic energy (U) and the coulombic virial (W) in a short simulation. Adherence to the relationship $U = -W$ shows the extent to which the Ewald sum is correctly converged. These variables can be found under the columns headed **eng_cou** and **vir_cou** in the OUTPUT file (see section 4.2.2).

The remainder of this section explains the meanings of these parameters and how they can be chosen. The Ewald sum can only be used in a three dimensional periodic system. There are three variables that control the accuracy: α , the Ewald convergence parameter; r_{cut} the real space forces cutoff; and the **kmax1,2,3** integers¹ that effectively define the range of the reciprocal space sum (one integer for each of the three axis directions). These variables are not independent, and it is usual to regard one of them as pre-determined and adjust the other two accordingly. In this treatment we assume that r_{cut} (defined by the **cutoff** directive in the CONTROL file) is fixed for the given system.

The Ewald sum splits the (electrostatic) sum for the infinite, periodic, system into a damped real space sum and a reciprocal space sum. The rate of convergence of both sums is governed by α . Evaluation of the real space sum is truncated at $r = r_{\text{cut}}$ so it is important that α be chosen so that contributions to the real space sum are negligible for terms with $r > r_{\text{cut}}$. The relative error (ϵ) in the real space sum truncated at r_{cut} is given

¹**Important note:** For the SPME method the values of **kmax1,2,3** should be double those obtained in this prescription, since they specify the sides of a cube, not a radius of convergence.

approximately by

$$\epsilon \approx \text{erfc}(\alpha r_{\text{cut}})/r_{\text{cut}} \approx \exp[-(\alpha r_{\text{cut}})^2]/r_{\text{cut}} \quad (3.1)$$

The recommended value for α is $3.2/r_{\text{cut}}$ or greater (too large a value will make the reciprocal space sum very slowly convergent). This gives a relative error in the energy of no greater than $\epsilon = 4 \times 10^{-5}$ in the real space sum. When using the directive **ewald precision** DL_POLY_2 makes use of a more sophisticated approximation:

$$\text{erfc}(x) \approx 0.56 \exp(-x^2)/x \quad (3.2)$$

to solve recursively for α , using equation 3.1 to give the first guess.

The relative error in the reciprocal space term is approximately

$$\epsilon \approx \exp(-k_{\text{max}}^2/4\alpha^2)/k_{\text{max}}^2 \quad (3.3)$$

where

$$k_{\text{max}} = \frac{2\pi}{L} \mathbf{kmax} \quad (3.4)$$

is the largest k -vector considered in reciprocal space, L is the width of the cell in the specified direction and $\mathbf{kmax}$ is an integer.

For a relative error of 4×10^{-5} this means using $k_{\text{max}} \approx 6.2\alpha$. $\mathbf{kmax}$ is then

$$\mathbf{kmax} > 3.2 L/r_{\text{cut}} \quad (3.5)$$

In a cubic system, $r_{\text{cut}} = L/2$ implies $\mathbf{kmax} = 7$. In practice the above equation slightly over estimates the value of $\mathbf{kmax}$ required, so optimal values need to be found experimentally. In the above example $\mathbf{kmax} = 5$ or 6 would be adequate.

If your simulation cell is a truncated octahedron or a rhombic dodecahedron then the estimates for the $\mathbf{kmax}$ need to be multiplied by $2^{1/3}$. This arises because twice the normal number of k -vectors are required (half of which are redundant by symmetry) for these boundary contributions [30].

If you wish to set the Ewald parameters manually (via the **ewald sum** or *spme sum* directives) the recommended approach is as follows. Preselect the value of r_{cut} , choose a working a value of α of about $3.2/r_{\text{cut}}$ and a large value for the $\mathbf{kmax}$ (say 10 10 10 or more). Then do a series of ten or so *single* step simulations with your initial configuration and with α ranging over the value you have chosen plus and minus 20%. Plot the Coulombic energy (and $-\mathcal{W}$) versus α . If the Ewald sum is correctly converged you will see a plateau in the plot. Divergence from the plateau at small α is due to non-convergence in the real space sum. Divergence from the plateau at large α is due to non-convergence of the reciprocal space sum. Redo the series of calculations using smaller $\mathbf{kmax}$ values. The optimum values for $\mathbf{kmax}$ are the smallest values that reproduce the correct Coulombic energy (the plateau value) and virial at the value of α to be used in the simulation.

Note that one needs to specify the three integers ($\mathbf{kmax1}$, $\mathbf{kmax2}$, $\mathbf{kmax3}$) referring to the three spatial directions, to ensure the reciprocal space sum is equally accurate in all directions. The values of $\mathbf{kmax1}$, $\mathbf{kmax2}$ and $\mathbf{kmax3}$ must be commensurate with the cell geometry to ensure the same minimum wavelength is used in all directions. For a cubic cell

set **kmax1** = **kmax2** = **kmax3**. However, for example, in a cell with dimensions $2A = 2B = C$ (ie. a tetragonal cell, longer in the c direction than the a and b directions) use **2kmax1** = **2kmax2** = (**kmax3**).

If the values for the **kmax** used are too small, the Ewald sum will produce spurious results. If values that are too large are used, the results will be correct but the calculation will consume unnecessary amounts of *cpu* time. The amount of *cpu* time increases with **kmax1** × **kmax2** × **kmax3**.

3.3.5.2 Hautman Klein Ewald Optimisation

Setting the HKE parameters can also be achieved rather simply, by the use of a **hke precision** directive in the CONTROL file e.g.

hke precision 1d-6 1 1

which specifies the required accuracy of the HKE convergence functions, plus two additional integers; the first specifying the order of the HKE expansion (**nhko**) and the second the maximum lattice parameter (**nlatt**). DLPOLY_2 will permit values of **nhko** from 1-3, meaning the HKE Taylor series expansion may range from zeroth to third order. Also **nlatt** may range from 1-2, meaning that (1) the nearest neighbour, and (2) and next nearest neighbour, cells are explicitly treated in the real space part of the Ewald sum. Increasing either of these parameters will increase the accuracy, but also substantially increase the *cpu* time of a simulation. The recommended value for both these parameters is 1 and if *both* these integers are left out, the default values will be adopted.

As with the standard Ewald and SPME methods, the user may set alternative control parameters with the CONTROL file **hke sum** directive e.g.

hke sum 0.05 6 6 1 1

which would set $\alpha = 0.05 \text{ \AA}^{-1}$, **kmax1** = 6, **kmax2** = 6. Once again one may check the accuracy by comparing the Coulombic energy with the virial, as described above. The last two integers specify, once again, the values of **nhko** and **nlatt** respectively. (Note it is possible to set either of these to zero in this case.)

Estimating the parameters required for a given simulation follows a similar procedure as for the standard Ewald method (above), but is complicated by the occurrence of higher orders of the convergence functions. Firstly a suitable value for α may be obtained when **nlatt**=0 from the rule: $\alpha = \beta/r_{cut}$, where r_{cut} is the largest real space cutoff compatible with a single MD cell and $\beta=(3.46,4.37,5.01,5.55)$ when **nhko**=(0,1,2,3) respectively. Thus in the usual case where **nhko**=1, $\beta=4.37$. When **nlatt**≠0, this β value is multiplied by a factor $1/(2 * nlatt + 1)$.

The estimation of **kmax1,2** is the same as that for the standard Ewald method above. Note that if any of these parameters prove to be insufficiently accurate, DLPOLY_2 will issue an error in the OUTPUT file, and indicate whether it is the real or reciprocal space sums that is questionable.

3.4 DL_POLY_2 Error Processing

3.4.1 The DL_POLY_2 Internal Error Facility

DL_POLY_2 contains a number of in-built error checks scattered throughout the package which detect a wide range of possible errors. In all cases, when an error is detected the subroutine `ERROR` is called, resulting in an appropriate message and termination of the program execution (either immediately, or after additional processing.).

Users intending to insert new error checks should ensure that all error checks are performed *concurrently* on *all* nodes, and that in circumstances where a different result may obtain on different nodes, a call to the global status routine `GSTATE` is made to set the appropriate global error flag on all nodes. Only after this is done, a call to subroutine `ERROR` may be made. An example of such a procedure might be:

```
logical safe
safe=(test_condition)
call gstate(safe)
if(.not.safe) call error(node_id,message_number)
```

In this example it is assumed that the logical operation *test_condition* will result in the answer *.true.* if it is safe for the program to proceed, and *.false.* otherwise. The call to `ERROR` requires the user to state the identity of the calling node (`node_id`), so that only the nominated node in `ERROR` (i.e. node 0) will print the error message. The variable `message_number` is an integer used to identify the appropriate message to be printed.

In all cases, if `ERROR` is called with a *non-negative* message number, the program run terminates. If the message number is *negative*, execution continues, but even in this case DL_POLY_2 will terminate the job at a more appropriate place. This feature is used in processing the `CONTROL` and `FIELD` file directives. A possible modification users may consider is to dump additional data before the call to `ERROR` is made.

A description of the `ERROR` subroutine is found in chapter 8 the DL_POLY_2 Reference Manual.

A full list of the DL_POLY_2 error messages and the appropriate user action can be found in Appendix C of this document.

Chapter 4

DL_POLY_2 Data Files

Scope of Chapter

This chapter describes all the input and output files for DL_POLY_2 , examples of which are to be found in the *data* sub-directory.

4.1 The INPUT files

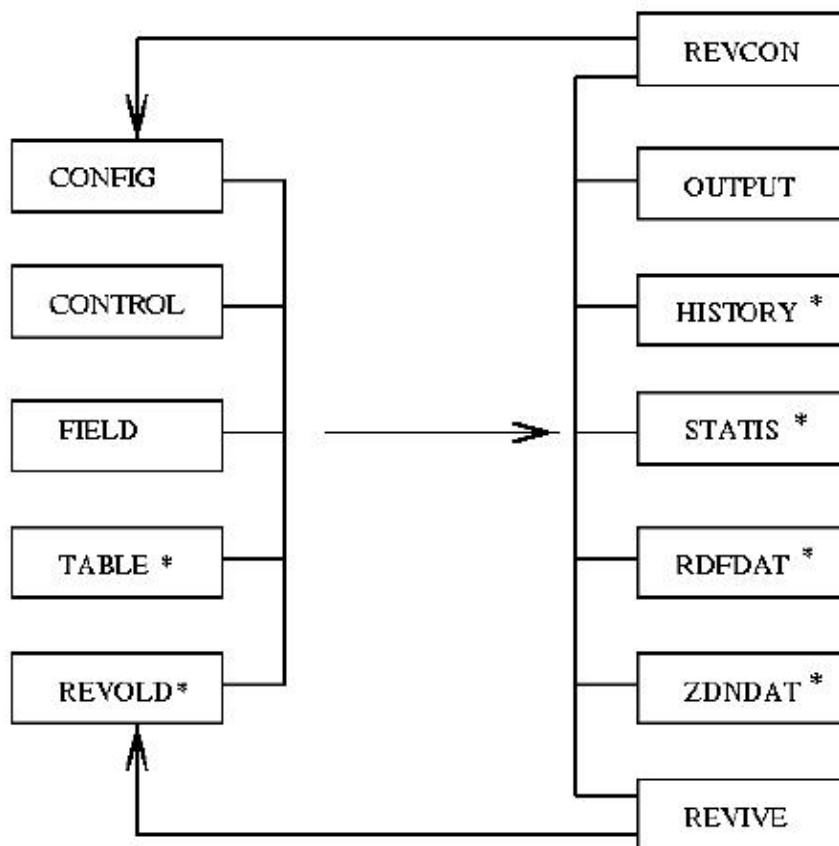


Figure 4.1: DL_POLY_2 input (left) and output (right) files. Note: files marked with an asterisk are non-mandatory.

DL_POLY_2 requires five input files named CONTROL, CONFIG, FIELD, TABLE and REVOLD. The first three files are mandatory, while TABLE is used only to input certain kinds of pair potential, and is not always required. REVOLD is required only if the job represents a continuation of a previous job. In the following sections we describe the form and content of these files.

4.1.1 The CONTROL File

The CONTROL file is read by the subroutine SIMDEF and defines the control variables for running a DL_POLY_2 job. It makes extensive use of **directives** and **keywords**. Directives are character strings that appear as the first entry on a data record (or line) and which

invoke a particular operation or provide numerical parameters. Also associated with each directive may be one or more keywords, which may qualify a particular directive by, for example, adding extra options. Directives can appear in any order in the CONTROL file, except for the **finish** directive which marks the end of the file. Some of the directives are mandatory (for example the **timestep** directive that defines the timestep), others are optional.

This way of constructing the file is very convenient, but it has inherent dangers. It is, for example, quite easy to specify the same directive more than once, or specify contradictory directives, or invoke algorithms that do not work together. By and large DL_POLY_2 tries to sort out these difficulties and print helpful error messages, but it does not claim to be foolproof. Fortunately in most cases the CONTROL file will be small and easy to check visually. It is important to think carefully about a simulation beforehand and ensure that DL_POLY_2 is being asked to do something that is physically reasonable. It should also be remembered that the present capabilities the package may not allow the simulation required and it may be necessary for you yourself to add new features.

An example CONTROL file appears below. The directives and keywords appearing are described in the following section.

Title Record: Example CONTROL file for DL_POLY

```
# define the state point
temperature          300.0

# simulation length and equilibration
steps                2000
equilibration steps  1000
scale every          5 steps
timestep             0.0005 ps
multiple timestep    1 steps

# specify cutoffs
cutoff               7.6 angstrom
delr                 0.5 angstrom

# print controller
print every          100 steps

# rdf options
rdf sampling every   10 steps
print rdf

# job time and permitted wind-up time
job time             21000 seconds
close time           200 seconds
```



```

# forces options
ewald sum          0.48 6 6 6
cap forces in equilibration mode 2000 kT/Å

# ensemble options
ensemble    nve (default option)

# statistics controls
stats every          2000 steps
stack           100 deep

# trajectory dumping controls
trajectory nstraj 1 istraj 50 keytrj 0

finish

```

4.1.1.1 The CONTROL file format

The file is free-formatted, integers, reals and additional keywords are entered following the directive on each record. Real and integer numbers must be separated by a non-numeric character (preferably a space or comma) to be correctly interpreted. No logical variables appear in the control file. Comment records (beginning with a #) and blank lines may be added to aid legibility (see example above). Additional annotation may be added onto the directive line, provided it does not contain numerical characters, or appear where a directive or keyword is expected. The CONTROL file is not case sensitive.

- The first record in the CONTROL file is a header 80 characters long, to aid identification of the file.
- The last record is a **finish** directive, which marks the end of the input data.

Between the header and the **finish** directive, a wide choice of control directives may be inserted. These are described below.

4.1.1.2 The CONTROL File Directives

The directives available are as follows.

directive:	meaning:
all pairs	use all pairs for electrostatic calculations
cap f	cap forces during equilibration period f is maximum cap in units of kT/Å (default $f=1000$)

close time f set job closure time to f seconds
collect include equilibration data in overall statistics
coul calculate coulombic forces
cut f set required forces cutoff to f (Å)
distan calculate coulombic forces using distance dependent dielectric
delr f set Verlet neighbour list shell width to f (Å)
ensemble nve select NVE ensemble (default)
ensemble nvt ber f select NVT ensemble with Berendsen thermostat
with relaxation constant f (ps)
ensemble nvt evans select NVT ensemble with Evans thermostat
ensemble nvt hoover f select NVT ensemble with Hoover-Nose thermostat
with relaxation constant f (ps)
ensemble npt ber $f_1 f_2$ select Berendsen NPT ensemble with f_1, f_2
as the thermostat and barostat relaxation times (ps)
ensemble npt hoover $f_1 f_2$ select Hoover NPT ensemble with f_1, f_2
as the thermostat and barostat relaxation times (ps)
ensemble nst ber $f_1 f_2$ select Berendsen $N\sigma$ T ensemble, with f_1, f_2
as the thermostat and barostat relaxation times (ps)
ensemble nst hoover $f_1 f_2$ select Hoover $N\sigma$ T ensemble with f_1, f_2
as the thermostat and barostat relaxation times (ps)
ensemble pmf select (NVE) potential of mean force ensemble
eps f set relative dielectric constant to f (default 1.0)
equil n equilibrate simulation for first n timesteps
ewald precision f select Ewald sum for electrostatics, with
automatic parameter optimisation ($0 < f < .5$)
ewald sum $\alpha k1 k2 k3$ select Ewald sum for electrostatics, with:
 α = Ewald convergence parameter (Å⁻¹)
 $k1$ = maximum k-vector index in x-direction
 $k2$ = maximum k-vector index in y-direction
 $k3$ = maximum k-vector index in z-direction
finish close the CONTROL file (last data record)
hke precision $f i j$ select HK-Ewald sum for electrostatics, with
automatic parameter optimisation ($0 < f < .5$)
 i = required order of HKE expansion (recommend 1)
 j = required lattice sum order (recommend 1)

hke sum α $k1$ $k2$ i j select HK-Ewald sum for electrostatics, with:
 α = Ewald convergence parameter ($\text{\AA}^{-1}$)
 $k1$ = maximum g-vector index in x-direction
 $k2$ = maximum g-vector index in y-direction
 $nhko$ = required order of HKE expansion (recommend 1)
 $nlatt$ = required lattice sum order (recommend 1)
job time f set job time to f seconds
mult n set multiple timestep (multi-step) interval (activated when $n > 2$)
no elec ignore coulombic interactions
no vdw ignore short range (non-bonded) interactions
pres f set required system pressure to f k.bars
(target pressure for constant pressure ensembles)
prim f set primary cutoff to f ($\text{\AA}$)
(for multiple timestep algorithm only)
print n print system data every n timesteps
print rdf print radial distribution functions
quaternion f set quaternion tolerance to f (default 10^{-8})
rdf f calculate radial distribution functions at intervals
of f timesteps
reaction select reaction field electrostatics
restart restart job from end point of previous run
(i.e. continue current simulation)
restart scale restart job from previous run with temperature scaling
(i.e. begin a new simulation from older run)
scale n rescale atomic velocities every n steps (during equilibration)
shake f set shake tolerance to f (default 10^{-8})
shift calculate electrostatic forces using shifted coulombic potential
spme precision f select Ewald sum for electrostatics, with
automatic parameter optimisation ($0 < f < .5$)
spme sum α $k1$ $k2$ $k3$ select Ewald sum for electrostatics, with:
 α = Ewald convergence parameter ($\text{\AA}^{-1}$)
 $k1$ = maximum k-vector index in x-direction
 $k2$ = maximum k-vector index in y-direction
 $k3$ = maximum k-vector index in z-direction
stack n set rolling average stack to n timesteps
stats n accumulate statistics data every n timesteps
steps n run simulation for n timesteps
temp f set required simulation temperature to f K
traj i j k write HISTORY file with controls:
 i = start timestep for dumping configurations
 j = timestep interval between configurations

	k = data level (i.e. variable keytrj see table 4.3)
timestep f	set timestep to f ps
zden	calculate the z-density profile
zero	perform zero temperature MD run

4.1.1.3 Further Comments on the CONTROL File

1. A number of the directives (or their **mutually exclusive** alternatives) are **mandatory**:
 - (a) **timestep**: specifying the simulation timestep;
 - (b) **temp** or **zero** : specifying the system temperature (not mutually exclusive);
 - (c) **ewald sum** or **ewald precision** or **spme sum** or **spme precision** or **hke sum** or **hke precision** or **coul** or **shift** or **distan** or **reaction** or **no elec**: specifying the required coulombic forces option;
 - (d) **cut** and **delr**: specifying the short range forces cutoff and Verlet strip;
 - (e) **prim**: specifying primary forces cutoff (**if mult>2 only**).
2. The **job time** and **close time** directives are required to ensure a controlled close down procedure when a job runs out of time. The time specified by the **job time** directive indicates the total time allowed for the job. (This must obviously be set equal to the time specified to the operating system when the job is submitted.) The **close time** directive represents the time DL_POLY_2 will require to write and close all the data files at the end of processing. This means the *effective* processing time limit is equal to the job time minus the close time. Thus when DL_POLY_2 reaches the effective job time limit it begins the close down procedure with enough time in hand to ensure the files are correctly written. In this way you may be sure the restart files etc. are complete when the job terminates. Note that setting the close time too small will mean the job will crash before the files have been finished. If it is set too large DL_POLY_2 will begin closing down too early. How large the close time needs to be to ensure safe close down is system dependent and a matter of experience. It generally increases with the job size.
3. The starting options for a simulation are governed by the keyword **restart**. If this is **not** specified in the CONTROL file, the simulation will start as new. If specified, it will either continue a previous simulation (**restart**) or start a new simulation with initial temperature scaling of the previous configuration (**restart scale**). Internally these options are handled by the integer variable **keyres**, which is explained in table 4.1.
4. The various **ensemble** options (i.e. **nve**, **nvt ber**, **nvt evans**, **nvt hoover**, **npt ber**, **npt hoover**, **nst ber**, **nst hoover**) are mutually exclusive, though none is mandatory (the default is the NVE ensemble). These options are handled internally by the integer variable **keyens**. The meaning of this variable is explained in table 4.2.

5. The force selection directives **ewald sum**, **ewald precision**, **reaction**, **coul**, **shift**, **dist**, **no elec** and **no vdw** are handled internally by the integer variable **keyfce**. See table 4.4 for an explanation of this variable. Note that these options are mutually exclusive.
6. The choice of reaction field electrostatics (directive **reaction**) requires also the specification of the relative dielectric constant external to the cavity. This is specified in the **eps** directive.
7. DL_POLY_2 uses as many as three different potential cutoffs. These are as follows:
 - (a) **rcut** - this is the universal cutoff. It applies to the real space part of the electrostatics calculations and to the van der Waals potentials if no other cutoff is applied;
 - (b) **rvdw** - this is the user-specified cutoff for the van der Waals potentials. If not specified its value defaults to **rcut**;
 - (c) **rprim** - this is used in the multiple timestep algorithm to specify the primary atom region (see section 2.5.7). It has no meaning if the multiple timestep option is not used.
8. Some directives are optional. If not specified DL_POLY_2 will take default values if necessary. The defaults appear in the above table.
9. The **zero** directive, enables a zero temperature simulation. This is intended as a crude energy minimizer to help relax a system before a simulation begins. It should not be thought of as a true energy minimization method.
10. The DL_POLY_2 multiple timestep option is invoked if the number appearing with the **mult** directive is greater than 2. This number (stored in the variable **multt**) specifies the number of timesteps (the multi-step) that elapse between partitions of the full Verlet neighbour list into primary and secondary atoms.
11. If a multiple time-step is used, (i.e. **multt**>2), then statistics for radial distribution functions are collected only at updates of the secondary neighbour list. The number specified on the **rdf** directive (the variable **nstbgr**) means that RDF data are accumulated at intervals of **nstbgr**×**multt** timesteps.
12. As a default, DL_POLY_2 does not store statistical data during the equilibration period. If the directive **collect** is used, equilibration data will be incorporated into the overall statistics.
13. The directive **delr** specifies the width of the border to be used in the Verlet neighbour list construction. The width is stored in the variable **delr**. The list is updated whenever two or more atoms have moved a distance of more than **delr**/2 from their positions at the last update of the Verlet list.

Users are advised to study the example CONTROL files appearing in the *data* sub-directory to see how different files are constructed.

Table 4.1: Internal Restart Key

keyres	meaning
0	start new simulation from CONFIG file, and assign velocities from Gaussian distribution.
1	continue current simulation
2	start new simulation from CONFIG file, and rescale velocities to desired temperature

Table 4.2: Internal Ensemble Key

keyens	meaning
0	Microcanonical ensemble (NVE)
1	Evans NVT ensemble
2	Berendsen NVT ensemble
3	Nosé-Hoover NVT ensemble
4	Berendsen NPT ensemble
5	Nosé-Hoover NPT ensemble
6	Berendsen $N\bar{\sigma}T$ ensemble
7	Nosé-Hoover $N\bar{\sigma}T$ ensemble
8	Potential of mean force (NVE) ensemble

Table 4.3: Internal Trajectory File Key

keytrj	meaning
0	coordinates only in file
1	coordinates and velocities in file
2	coordinates, velocities and forces in file

Table 4.4: Non-bonded force key

keyfce	meaning
odd	evaluate short-range potentials and electrostatics
even	evaluate Electrostatic potential only
Electrostatics are evaluated as follows:	
0 [†] , 1 [‡]	Ignore Electrostatic interactions
2, 3	Ewald summation
4, 5	distance dependent dielectric constant
6, 7	standard truncated Coulombic potential
8, 9	truncated and shifted Coulombic potential
10,11	Reaction Field electrostatics
12,13	SPME electrostatics
14,15	Hautman-Klein Ewald electrostatics

[†] **keyfce** = 0 means no non-bonded terms are evaluated.

[‡] **keyfce** = 1 means only short-range potentials are evaluated.

4.1.2 The CONFIG File

The CONFIG file contains the dimensions of the unit cell, the key for periodic boundary conditions and the atomic labels, coordinates, velocities and forces. This file is read by the subroutine SYSGEN. (It is also read by the subroutine SIMDEF if the **ewald precision** directive is used.) The first few records of a typical CONFIG file are shown below:

```
IceI structure 6x6x6 unit cells with proton disorder
      2      3
26.988000000000000  0.000000000000000  0.000000000000000
-13.494000000000000 23.372293600000000  0.000000000000000
 0.000000000000000  0.000000000000000 44.028000000000000
  OW      1
-2.505228382      -1.484234330      -7.274585343
 0.5446573999     -1.872177437     -0.7702718106
3515.939287      13070.74357      4432.030587
  HW      2
-1.622622646     -1.972916834     -7.340573742
 1.507099154     -1.577400769      4.328786484
7455.527553     -4806.880540     -1255.814536
  HW      3
-3.258494716     -2.125627191     -7.491549620
 2.413871957     -4.336956694      2.951142896
-7896.278327     -8318.045939     -2379.766752
  OW      4
0.9720599243E-01 -2.503798635     -3.732081894
 1.787340483     -1.021777575     0.5473436377
9226.455153      9445.662860      5365.202509
```

etc.

4.1.2.1 Format

The file is fixed-formatted: integers as “i10”, reals as “f20.0”. The header record is formatted as 80 alphanumeric characters.

4.1.2.2 Definitions of Variables

record 1		
header	a80	title line
record 2		
levcfg	integer	CONFIG file key. See table 4.5 for permitted values
imcon	integer	Periodic boundary key. See table 4.6 for permitted values
record 3	omitted if imcon = 0	
cell(1)	real	x component of <i>a</i> cell vector


```

    cell(2)    real          y component of a cell vector
    cell(3)    real          z component of a cell vector
record 4      omitted if imcon = 0
    cell(4)    real          x component of b cell vector
    cell(5)    real          y component of b cell vector
    cell(6)    real          z component of b cell vector
record 5      omitted if imcon = 0
    cell(7)    real          x component of c cell vector
    cell(8)    real          y component of c cell vector
    cell(9)    real          z component of c cell vector

```

Subsequent records consists of blocks of between 2 and 4 records depending on the value of the `levcfg` variable. Each block refers to one atom. The atoms must be listed sequentially in order of increasing index. Within each block the data are as follows:

```

record i
    atmnam      a8          atom name.
    index       integer     atom index
    atmnum      integer     atomic number
record ii
    xxx         real        x coordinate
    yyy         real        y coordinate
    zzz         real        z coordinate
record iii    included only if levcfg > 0
    vxx         real        x component of velocity
    vyy         real        y component of velocity
    vzz         real        x component of velocity
record iv     included only if levcfg > 1
    fxx         real        x component of force
    fyy         real        y component of force
    fzz         real        z component of force

```

Note that on **record i** only the atom name is mandatory, any other items are not read by DL_POLY_2 but may be added to aid alternative uses of the file, for example the DL_POLY_2 Graphical User Interface[20].

4.1.2.3 Further Comments

The CONFIG file has the same format as the output file REVCON (section 4.2.3). When restarting from a previous run of DL_POLY_2 (i.e. using the **restart** or **restart scale** directives in the CONTROL file - above), the CONFIG file must be replaced by the REVCON file, which is renamed as the CONFIG file. The `copy` macro in the `execute` sub-directory of DL_POLY_2 does this for you.

Table 4.5: CONFIG file key (record 2)

levcfg	meaning
0	Coordinates included in file
1	Coordinates and velocities included in file
2	Coordinates, velocities and forces included in file

Table 4.6: Periodic boundary key (record 2)

imcon	meaning
0	no periodic boundaries
1	cubic boundary conditions
2	orthorhombic boundary conditions
3	parallelepiped boundary conditions
4	truncated octahedral boundary conditions
5	rhombic dodecahedral boundary conditions
6	x-y parallelogram boundary conditions with no periodicity in the z direction
7	hexagonal prism boundary conditions

4.1.3 The FIELD File

The FIELD file contains the force field information defining the nature of the molecular forces. It is read by the subroutine SYSDEF. Excerpts from a force field file are shown below. The example is the antibiotic Valinomycin in a cluster of 146 water molecules.

Valinomycin Molecule with 146 SPC Waters

UNITS kcal

```

MOLECULES      2
Valinomycin
NUMMOLS 1
ATOMS 168
      O      16.0000      -0.4160      1
      OS     16.0000      -0.4550      1
      "      "          "          "
      "      "          "          "
      HC      1.0080      0.0580      1
      C      12.0100      0.4770      1
BONDS 78
harm  31  19 674.000      1.44900
harm  33  31 620.000      1.52600
      "  "  "  "          "
      "  "  "  "          "
harm 168  19 980.000      1.33500
harm 168 162 634.000      1.52200
CONSTRAINTS 90
      20  19      1.000017
      22  21      1.000032
      "  "          "
      "  "          "
      166 164      1.000087
      167 164      0.999968
ANGLES 312
harm  43   2   44 200.00      116.40
harm  69   5   70 200.00      116.40
      "  "  "  "  "          "
      "  "  "  "  "          "
harm  18 168 162 160.00      120.40
harm  19 168 162 140.00      116.60
DIHEDRALS 371
harm   1  43   2   44 2.3000      180.00
harm  31  43   2   44 2.3000      180.00
      "  "  "  "  "          "
      "  "  "  "  "          "

```

```

cos   149   17  161   16  10.500      180.00
cos   162   19  168   18  10.500      180.00
FINISH
SPC Water
NUMMOLS 146
ATOMS   3
      OW      16.0000      -0.8200
      HW      1.0080      0.4100
      HW      1.0080      0.4100
CONSTRAINTS 3
      1      2      1.0000
      1      3      1.0000
      2      3      1.63299
FINISH
VDW    45
C      C      1j    0.12000      3.2963
C      CT     1j    0.08485      3.2518
"      "      "      "          "
"      "      "      "          "
"      "      "      "          "
OW     OS     1j    0.15100      3.0451
OS     OS     1j    0.15000      2.9400
CLOSE

```

4.1.3.1 Format

The FIELD file is fixed-formatted. Integers are formatted as “i5”, reals as “f12.0” and characters as “a4”, “a8”, “a40” or “a80”, depending on context. The contents of the file are variable and are defined by the use of **directives**. Additional information is associated with the directives, and is free formatted as in the CONTROL file above. The file is not case sensitive.

4.1.3.2 Definitions of Variables

The file divides into three sections: general information, molecular descriptions, and non-bonded interaction descriptions, appearing in that order in the file.

4.1.3.2.1 General information

The first record in the FIELD file is the title. It must be followed by the **units** directive. Both of these are mandatory. These records may optionally be followed by the **neut** directive.

record 1

```

header          a80      field file header

```

record 2

units a40 Unit of energy used for input and output

record 3 (optional)

neut a40 activate the neutral/charge groups option for
the electrostatic calculations

The energy units on the **units** directive are described by additional keywords:

a eV, for electron-volts

b kcal, for k-calories mol⁻¹

c kJ, for k-Joules mol⁻¹

d internal, for DL_POLY_2 internal units (10 J mol⁻¹).

If no units keyword is entered, DL_POLY_2 units are assumed for both input and output. The units keyword may appear anywhere on the data record provided it does not exceed column 40. The **units** directive only affects the input and output interfaces, all internal calculations are handled using DL_POLY_2 units.

4.1.3.2.2 Molecular details

It is important for the user to understand that there is an organisational correspondence between the FIELD file and the CONFIG file described above. It is required that the order of specification of molecular types and their atomic constituents in the FIELD file follows the order in which they appear in the CONFIG file. Failure to adhere to this common sequence will be detected by DL_POLY_2 and result in premature termination of the job. It is therefore essential to work from the CONFIG file when constructing the FIELD file. It is not as difficult as it sounds!

The entry of the molecular details begins with the mandatory directive:

molecules *n*

where *n* is an integer specifying the number of different *types* of molecule appearing in the FIELD file. Once this directive has been encountered, DL_POLY_2 enters the *molecular description* environment in which only molecular description keywords and data are valid.

Immediately following the **molecules** directive, are the records defining individual molecules:

1. *name-of-molecule*

which can be any character string up to 80 characters in length. (Note: this is not a directive, just a simple character string.)

2. **nummols** *n*

where *n* is the number of times a molecule of this type appears in the simulated system. The molecular data then follow in subsequent records:

3. **atoms** *n*

where *n* indicates the number of atoms in this type of molecule. A number of records follow, each giving details of the atoms in the molecule i.e. site names, masses and charges. Each record carries the entries:

sitnam	a8	atomic site name
weight	real	atomic site mass
chge	real	atomic site charge
nrept	integer	repeat counter
ifrz	integer	'frozen' atom (if ifrz > 0)
igrp	integer	neutral/charge group number

The integer **nrept** need not be specified (in which case a value of 1 is assumed.) A number greater than 1 specified here indicates that the next (**nrept** - 1) entries in the CONFIG file are ascribed the atomic characteristics given in the current record. The sum of the repeat numbers for all atoms in a molecule should equal the number specified by the **atoms** directive.

4. **shell** *n*

where *n* is the number of core-shell units. Each of the subsequent *n* records contains:

index 1	integer	site index of core
index 2	integer	site index of shell
spring	real	force constant of core-shell spring

The spring force constant is entered in units of **engunit** Å⁻², where **engunit** is the energy unit specified in the **units** directive.

Note that the atomic site indices referred to in this table are indices arising from numbering each atom in the molecule from 1 to the number specified in the **atoms** directive for this molecule. This same numbering scheme should be used for all descriptions of this molecule, including the **bonds**, **constraints**, **angles**, and **dihedrals** entries described below. DL_POLY_2 will itself construct the global indices for all atoms in the systems.

This directive (and associated data records) need not be specified if the molecule contains no core-shell units.

5. **bonds** *n*

where *n* is the number of flexible chemical bonds in the molecule. Each of the subsequent *n* records contains:

bond key	a4	see table 4.7
index 1	integer	first atomic site in bond
index 2	integer	second atomic site in bond
variable 1	real	potential parameter see table 4.7
variable 2	real	potential parameter see table 4.7
variable 3	real	potential parameter see table 4.7
variable 4	real	potential parameter see table 4.7

The meaning of these variables is given in table 4.7. This directive (and associated data records) need not be specified if the molecule contains no flexible chemical bonds. See the note on the atomic indices appearing under the **shell** directive above.

Table 4.7: Chemical bond potentials

key	potential type	Variables (1-4)				functional form
harm -hrm	Harmonic	k	r_0			$U(r) = \frac{1}{2}k(r - r_0)^2$
mors -mrs	Morse	E_0	r_0	k		$U(r) = E_0[\{1 - \exp(-k(r - r_0))\}^2 - 1]$
12-6 -126	12-6	A	B			$U(r) = \left(\frac{A}{r^{12}}\right) - \left(\frac{B}{r^6}\right)$
rhrm -rhm	Restraint	k	r_0	r_c		$U(r) = \frac{1}{2}k(r - r_0)^2 \quad r - r_0 \leq r_c$ $U(r) = \frac{1}{2}kr_c^2 + kr_c(r - r_0 - r_c) \quad r - r_0 > r_c$
quar -qur	Quartic	k	r_0	k'	k''	$U(r) = \frac{k}{2}(r - r_0)^2 + \frac{k'}{3}(r - r_0)^3 + \frac{k''}{4}(r - r_0)^4$

Note: bond potentials with a dash (-) as the first character of the keyword, do not contribute to the *excluded atoms list* (see section 2.1). In this case DL_POLY_2 will also calculate the nonbonded pair potentials between the described atoms, unless these are deactivated by another potential specification.

6. constraints n

where n is the number of constraint bonds in the molecule. Each of the following n records contains:

index 1	integer	first atomic index
index 2	integer	second atomic index
bondlength	real	constraint bond length

This directive (and associated data records) need not be specified if the molecule contains no constraint bonds. See the note on the atomic indices appearing under the **shell** directive above.

7. **pmf b**

where b is the potential of mean force bondlength (Å). There follows the definitions of two PMF units:

(a) **pmf unit** n_1

where n_1 is the number of sites in the first unit. The subsequent n_1 records provide the site indices and weighting. Each record contains:

index	integer	atomic site index
weight	real	site weighting

(b) **pmf unit** n_2

where n_2 is the number of sites in the second unit. The subsequent n_2 records provide the site indices and weighting. Each record contains:

index	integer	atomic site index
weight	real	site weighting

This directive (and associated data records) need not be specified if no PMF constraints are present. See the note on the atomic indices appearing under the **shell** directive above. The pmf bondlength applies to the distance between the centres of the two pmf units. The centre, $\underline{R}$, of each unit is given by

$$\underline{R} = \frac{\sum_{\alpha} w_{\alpha} \underline{r}_{\alpha}}{\sum_{\alpha} w_{\alpha}}$$

where $\underline{r}_{\alpha}$ is a site position and w_{α} the site weighting. Note that the pmf constraint is intramolecular. To define a constraint between two molecules, the molecules must be described as part of the same DL_POLY “molecule”. This is illustrated in test case 6, where a pmf constraint is imposed between a potassium ion and the centre of mass of a water molecule. DL_POLY_2 allows only one type of pmf constraint per system. The value of **nummols** for this molecule determines the number of pmf constraint in the system.

Note that the directive **ensemble pmf** must be specified in the CONTROL file for this option to be implemented correctly.

8. **angles n**

where n is the number of valence angle bonds in the molecule. Each of the n records following contains:

angle key	a4	potential key. See table 4.8
index 1	integer	first atomic index
index 2	integer	second atomic index (central site)
index 3	integer	third atomic index
variable 1	real	potential parameter see table 4.8
variable 2	real	potential parameter see table 4.8

The meaning of these variables is given in table 4.8. See the note on the atomic indices appearing under the **shell** directive above. This directive (and associated data records) need not be specified if the molecule contains no angular terms.

Table 4.8: Valence Angle potentials

key	potential type	Variables (1-4)				functional form†
harm -hrm	Harmonic	k	θ_0			$U(\theta) = \frac{k}{2}(\theta - \theta_0)^2$
quar -qur	Quartic	k	θ_0	k'	k''	$U(\theta) = \frac{k}{2}(\theta - \theta_0)^2 + \frac{k'}{3}(\theta - \theta_0)^3 + \frac{k''}{4}(\theta - \theta_0)^4$
thrm -thm	Truncated harmonic	k	θ_0	ρ		$U(\theta) = \frac{k}{2}(\theta - \theta_0)^2 \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]$
shrm -shm	Screened harmonic	k	θ_0	ρ_1	ρ_2	$U(\theta) = \frac{k}{2}(\theta - \theta_0)^2 \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]$
bvs1 -bv1	Screened Vessal[24]	k	θ_0	ρ_1	ρ_2	$U(\theta) = \frac{k}{8(\theta - \theta_0)^2} \left\{ [(\theta_0 - \pi)^2 - (\theta - \pi)^2]^2 \right\} \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]$
bvs2 -bv2	Truncated Vessal[25]	k	θ_0	a	ρ	$U(\theta) = k[\theta^a(\theta - \theta_0)^2(\theta + \theta_0 - 2\pi)^2 - \frac{a}{2}\pi^{a-1}(\theta - \theta_0)^2(\pi - \theta_0)^3] \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]$
hcos -hcs	Harmonic Cosine	k	θ_0			$U(\theta) = \frac{k}{2}(\cos(\theta) - \cos(\theta_0))^2$
cos -cos	Cosine	A	δ	m		$U(\theta) = A[1 + \cos(m\theta - \delta)]$

† θ is the a-b-c angle.

Note: valence angle potentials with a dash (-) as the first character of the keyword, do not contribute to the *excluded atoms list* (see section 2.1). In this case DLPOLY_2 will calculate the nonbonded pair potentials between the described atoms.

9. **dihedrals** *n*

where *n* is the number of dihedral interactions present in the molecule. Each of the following *n* records contains:

dihedral key	a4	potential key. See table 4.9
index 1	integer	first atomic index
index 2	integer	second atomic index
index 3	integer	third atomic index
index 4	integer	fourth atomic index
variable 1	real	potential parameter see table 4.9
variable 2	real	potential parameter see table 4.9
variable 3	real	potential parameter see table 4.9
variable 4	real	1-4 electrostatic interaction scale factor.
variable 5	real	1-4 Van der Waals interaction scale factor.

The meaning of the variables 1-3 is given in table 4.9. The variables 4 and 5 specify the scaling factor for the 1-4 electrostatic and Van der Waals nonbonded interactions respectively. This directive (and associated data records) need not be specified if the molecule contains no dihedral angle terms. See the note on the atomic indices appearing under the **shell** directive above.

Table 4.9: Dihedral Angle Potentials

key	potential type	Variables (1-3)			functional form [‡]
cos	Cosine	<i>A</i>	<i>δ</i>	<i>m</i>	$U(\phi) = A[1 + \cos(m\phi - \delta)]$
harm	Harmonic	<i>k</i>	<i>φ</i> ₀		$U(\phi) = \frac{1}{2}k(\phi - \phi_0)^2$
hcos	Harmonic cosine	<i>k</i>	<i>φ</i> ₀		$U(\phi) = \frac{k}{2}(\cos(\phi) - \cos(\phi_0))^2$
cos3	Triple cosine	<i>A</i> ₁	<i>A</i> ₂	<i>A</i> ₃	$U(\phi) = \frac{1}{2}A_1(1 + \cos(\phi)) + \frac{1}{2}A_2(1 - \cos(2\phi)) + \frac{1}{2}A_3(1 + \cos(3\phi))$

[‡]*φ* is the a-b-c-d dihedral angle.

10. **inversions** *n*

where *n* is the number of inversion interactions present in the molecule. Each of the following *n* records contains:

inversion key	a4	potential key. See table 4.10
index 1	integer	first atomic index

index 2	integer	second atomic index
index 3	integer	third atomic index
index 4	integer	fourth atomic index
variable 1	real	potential parameter see table 4.10
variable 2	real	potential parameter see table 4.10

The meaning of the variables 1-2 is given in table 4.10. This directive (and associated data records) need not be specified if the molecule contains no inversion angle terms. See the note on the atomic indices appearing under the **shell** directive above.

Table 4.10: Inversion Angle Potentials

key	potential type	Variables (1-2)		functional form [‡]
harm	Harmonic	k	ϕ_0	$U(\phi) = \frac{1}{2}k(\phi - \phi_0)^2$
hcos	Harmonic cosine	k	ϕ_0	$U(\phi) = \frac{k}{2}(\cos(\phi) - \cos(\phi_0))^2$
plan	Planar	A		$U(\phi) = A[1 - \cos(\phi)]$

[‡] ϕ is the inversion angle.

11. **rigid** n

where n is the number of rigid units in the molecule. It is followed by at least n records, each specifying the sites in a rigid unit:

m	integer	number of sites in rigid unit
site 1	integer	first site atomic index
site 2	integer	second site atomic index
site 3	integer	third site atomic index
..	..	<i>etc.</i>
site m	integer	m'th site atomic index

Up to 15 sites can be specified on the first record. Additional records are used if necessary. Up to 16 sites are specified per record thereafter.

This directive (and associated data records) need not be specified if the molecule contains no rigid units. See the note on the atomic indices appearing under the **shell** directive above.

12. **teth** n

where n is the number of tethered atoms in the molecule. It is followed by n records specifying the tethered sites in the molecule:

tether key	a4	tethering potential key	see table 4.11
index	integer	atomic index	
variable 1	real	potential parameter	see table 4.11
variable 2	real	potential parameter	see table 4.11
variable 3	real	potential parameter	see table 4.11
variable 4	real	potential parameter	see table 4.11

This directive (and associated data records) need not be specified if the molecule contains no tethered atoms. See the note on the atomic indices appearing under the **shell** directive above.

Table 4.11: Tethering potentials

key	potential type	Variables (1-3)			functional form
harm	Harmonic	k			$U(r) = \frac{1}{2}kr^2$
rhrm	Restraint	k	r_c		$U(r) = \begin{cases} \frac{1}{2}kr^2 & r \leq r_c \\ \frac{1}{2}kr_c^2 + kr_c(r - r_c) & r > r_c \end{cases}$
quar	Quartic	k	k'	k''	$U(r) = \frac{k}{2}r^2 + \frac{k'}{3}r^3 + \frac{k''}{4}r^4$

13. **finish**

This directive is entered to signal to DL_POLY_2 that the entry of the details of a molecule has been completed.

The entries for a second molecule may now be entered, beginning with the *name-of-molecule* record and ending with the **finish** directive.

The cycle is repeated until all the types of molecules indicated by the **molecules** directive have been entered.

The user is recommended to look at the example FIELD files in the *data* directory to see how typical FIELD files are constructed.

4.1.3.3 Non-bonded Interactions

Non-bonded interactions are identified by atom types as opposed to specific atomic indices. The first type of non-bonded potentials are the pair potentials. The input of pair potential data is signalled by the directive:

vdw *n*

where n is the number of pair potentials to be entered. There follows n records, each specifying a particular pair potential in the following manner:

atmnam 1	a8	first atom type
atmnam 2	a8	second atom type
key	a4,1X	potential key. See table 4.12
variable 1	real	potential parameter see table 4.12
variable 2	real	potential parameter see table 4.12
variable 3	real	potential parameter see table 4.12
variable 4	real	potential parameter see table 4.12
variable 5	real	potential parameter see table 4.12

The variables pertaining to each potential are described in table 4.12. **Note** that there is an empty column after the potential key. This means that the potential parameters are entered in columns 22 to 81. Note that any pair potential not specified in the FIELD file, will be assumed to be zero. Note also that the Sutton-Chen potential for metals is classified as a pair potential for input purposes.

The specification of three body potentials is initiated by the directive:

tbp n

where n is the number of three-body potentials to be entered. There follows n records, each specifying a particular three body potential in the following manner:

atmnam 1	a8	first atom type
atmnam 2	a8	second atom type (central site)
atmnam 3	a8	third atom type
key	a4	potential key. See table 4.13

Table 4.12: Definition of pair potential functions and variables

key	potential type	Variables (1-5)					functional form
12-6	12-6	A	B				$U(r) = \left(\frac{A}{r^{12}}\right) - \left(\frac{B}{r^6}\right)$
lj	Lennard-Jones	ϵ	σ				$U(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^6 \right]$
nm	n-m	E_o	n	m	r_0		$U(r) = \frac{E_o}{(n-m)} \left[m \left(\frac{r_o}{r}\right)^n - n \left(\frac{r_o}{r}\right)^m \right]$
buck	Buckingham	A	ρ	C			$U(r) = A \exp\left(-\frac{r}{\rho}\right) - \frac{C}{r^6}$
bhm	Born-Huggins -Meyer	A	B	σ	C	D	$U(r) = A \exp[B(\sigma - r)] - \frac{C}{r^6} - \frac{D}{r^8}$
hbnd	12-10 H-bond	A	B				$U(r) = \left(\frac{A}{r^{12}}\right) - \left(\frac{B}{r^{10}}\right)$
snm	Shifted force [†] n-m [27]	E_o	n	m	r_0	$r_c^{\ddagger}$	$U(r) = \frac{\alpha E_o}{(n-m)} \times$ $\left[m\beta^n \left\{ \left(\frac{r_o}{r}\right)^n - \left(\frac{1}{\gamma}\right)^n \right\} - n\beta^m \left\{ \left(\frac{r_o}{r}\right)^m - \left(\frac{1}{\gamma}\right)^m \right\} \right]$ $+ \frac{nm\alpha E_o}{(n-m)} \left(\frac{r-\gamma r_o}{\gamma r_o} \right) \left\{ \left(\frac{\beta}{\gamma}\right)^n - \left(\frac{\beta}{\gamma}\right)^m \right\}$
stch	Sutton-Chen	ϵ	a	n	m	C	$U_i(r) = \epsilon \left[\left(\frac{a}{r}\right)^n - \frac{C}{2} \sqrt{\rho_i} \right]$ $\rho_i = \sum_j \left(\frac{a}{r_{ij}}\right)^m$
tab	Tabulation						tabulated potential

[†] Note: in this formula the terms α , β and γ are compound expressions involving the variables E_o , n , m , r_0 and r_c . See section 2.3.1 for further details.

[‡] Note: r_c defaults to the general van der Waals cutoff (**rvdw** or **rcut**) if it is set to zero or not specified or not specified in the FIELD file.

Table 4.13: Three-body potentials

key	potential type	Variables (1-4)				functional form†
thrm	Truncated harmonic	k	θ_0	ρ		$U(\theta) = \frac{k}{2}(\theta - \theta_0)^2 \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]$
shrm	Screened harmonic	k	θ_0	ρ_1	ρ_2	$U(\theta) = \frac{k}{2}(\theta - \theta_0)^2 \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]$
bvs1	Screened Vessal[24]	k	θ_0	ρ_1	ρ_2	$U(\theta) = \frac{k}{8(\theta - \theta_0)^2} \left\{ [(\theta_0 - \pi)^2 - (\theta - \pi)^2]^2 \right\} \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]$
bvs2	Truncated Vessal[25]	k	θ_0	a	ρ	$U(\theta) = k[\theta^a(\theta - \theta_0)^2(\theta + \theta_0 - 2\pi)^2 - \frac{a}{2}\pi^{a-1}(\theta - \theta_0)^2(\pi - \theta_0)^3] \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]$
hbnd	H-bond [7]	D_{hb}	R_{hb}			$U(\theta) = D_{hb} \cos^4(\theta) [5(R_{hb}/r_{jk})^{12} - 6(R_{hb}/r_{jk})^{10}]$

† θ is the a-b-c angle.

variable 1	real	potential parameter see table 4.13
variable 2	real	potential parameter see table 4.13
variable 3	real	potential parameter see table 4.13
variable 4	real	potential parameter see table 4.13
variable 5	real	cutoff range for this potential (Å)

The variables pertaining to each potential are described in table 4.13. Note that the fifth variable is the range at which the three body potential is truncated. The distance is in Å, measured from the central atom.

The specification of four body potentials is initiated by the directive:

fbp n

where n is the number of four-body potentials to be entered. There follows n records, each specifying a particular four-body potential in the following manner:

atmnam 1	a8	first atom type (central site)
atmnam 2	a8	second atom type
atmnam 3	a8	third atom type
atmnam 4	a8	fourth atom type
key	a4	potential key. See table 4.14
variable 1	real	potential parameter see table 4.14
variable 2	real	potential parameter see table 4.14
variable 3	real	cutoff range for this potential (Å)

The variables pertaining to each potential are described in table 4.14. Note that the third variable is the range at which the four-body potential is truncated. The distance is in Å, measured from the central atom.

Table 4.14: Four-body Potentials

key	potential type	Variables (1-2)		functional form [‡]
harm	Harmonic	k	ϕ_0	$U(\phi) = \frac{1}{2}k(\phi - \phi_0)^2$
hcos	Harmonic cosine	k	ϕ_0	$U(\phi) = \frac{k}{2}(\cos(\phi) - \cos(\phi_0))^2$
plan	Planar	A		$U(\phi) = A[1 - \cos(\phi)]$

[‡] ϕ is the inversion angle.

4.1.3.4 External Field

The presence of an external field is flagged by the **extern** directive. The next line in the FIELD file should have another directive indicating what type of field is to be applied. On the following lines comes the **mxfld** parameters, five per line, that describe the field. In the include files supplied with DL_POLY_2 **mxfld** is set to 10.

The variables pertaining to each potential are described in table 4.15.

Table 4.15: External fields

key	potential type	Variables (1-4)				functional form [†]
elec	Electric field	E_x	E_y	E_z		$\underline{F} = q.\underline{E}$
oshm	Oscillating Shear	A	n			$\underline{F}_x = A\cos(2n\pi.z/L_z)$
shrx	Continuous Shear	A	z_0			$ z > z_0: \underline{v}_x = (1/2)A(z /z)$
grav	Gravitational Field	G_x	G_y	G_z		$\underline{F} = m.\underline{G}$
magn	Magnetic Field	H_x	H_y	H_z		$\underline{F} = q(\underline{v} \times \underline{H})$
sphr	Containing Sphere	A	R_0	n	R_{cut}	$r > R_{\text{cut}}: \underline{F} = A(R_0 - r)^{-n}$

4.1.3.5 Closing the FIELD File

The FIELD file must be closed with the directive:

close

which signals the end of the force field data. Without this directive DL_POLY_2 will abort.

4.1.4 The REVOLD File

This file contains statistics arrays from a previous job. It is not required if the current job is not a continuation of a previous run (ie. if the **restart** directive is not present in the CONTROL file - see above). The file is unformatted and therefore not readable by normal people. DL_POLY_2 normally produces the file REVIVE (see section 4.2.4) at the end of a job which contains the statistics data. REVIVE should be copied to REVOLD before a continuation run commences. This may be done by the *copy* macro supplied in the *execute* sub-directory of DL_POLY_2 .

4.1.4.1 Format

The REVOLD file is unformatted. All variables appearing are written in native real*8 representation. Nominally integer quantities (e.g. the timestep number **nstep**) are represented by the nearest real number. The contents are as follows (the dimensions of array variables are given in brackets, in terms of parameters from the DL_PARAMS.INC file - see section 8.1.1).

record 1:

nstep	timestep of final configuration
numacc	number of configurations used in averages
numrdf	number of configurations used in rdf averages
chit	relaxation time of thermostat
chip	relaxation time of barostat
conint	conserved quantity for selected ensemble
nzden	number of configurations used in z density

record 2:

eta	scaling factors for simulation cell matrix elements (9)

record 3:

stpval	instantaneous values of thermodynamic variables (mxnstk)

record 4:

sumval	average values of thermodynamic variables (mxnstk)

record 5:

ssqval	fluctuation (squared) of thermodynamic variables (mxnstk)

record 6:

zumval	running totals of thermodynamic variables (mxnstk)

```

record 7:
  ravval      rolling averages of thermodynamic variables (mxnstk)
record 8:
  stkval      stacked values of thermodynamic variables (mxstak×mxnstk)
record 9:
  xx0         atomic x-coordinates at MSD time origin (mxatms)
  yy0         atomic y-coordinates at MSD time origin (mxatms)
  zz0         atomic z-coordinates at MSD time origin (mxatms)
record 10:
  xxs         instantaneous atomic x-coordinates for MSD (mxatms)
  yys         instantaneous atomic y-coordinates for MSD (mxatms)
  zzs         instantaneous atomic z-coordinates for MSD (mxatms)
record 11:
  rdf         (Optional) RDF array (mxrdf×mxvdw)
record 12:
  zdens       (Optional) z-density array (mxrdf×mxsvdw)

```

4.1.4.2 Further Comments

Note that recompiling DL_POLY_2 with a different DL_PARAMS.INC file, may render any existing REVOLD file unreadable by the code.

4.1.5 The TABLE File

The TABLE file provides an alternative way of reading in the short range potentials - in tabular form. This is particularly useful if an analytical form of the potential does not exist or is too complicated to specify in the FORGEN subroutine. The table file is read by the subroutine FORTAB (see chapter 8).

The option of using tabulated potentials is specified in the FIELD file (see above). The specific potentials that are to be tabulated are indicated by the use of the **tab** keyword on the record defining the short range potential (see table 4.12). The directive **vdwtable** may be used in place of **vdw** to indicate that one or more of the short ranged potentials is specified in the form of a table.

4.1.5.1 Format

The file is fixed-formatted with integers as “i10”, reals as “e15.8”. Character variables are read as “a8”. The header record is formatted as 80 alphanumeric characters.

4.1.5.2 Definitions of Variables

```

record 1
  header      a80          file header
record 2

```

delpot	real	mesh resolution in Å
cutpot	real	cutoff used to define tables Å
ngrid	integer	number of grid points in tables

The subsequent records define each tabulated potential in turn, in the order indicated by the specification in the FIELD file. Each potential is defined by a header record and a set of data records with the potential and force tables.

header record:

atom 1	a8	first atom type
atom 2	a8	second atom type

potential data records: (*number of data records = Int((ngrid+3)/4)*)

data 1	real	data item 1
data 2	real	data item 2
data 3	real	data item 3
data 4	real	data item 4

force data records: (*number of data records = Int((ngrid+3)/4)*)

data 1	real	data item 1
data 2	real	data item 2
data 3	real	data item 3
data 4	real	data item 4

4.1.5.3 Further Comments

It should be noted that the number of grid points in the TABLE file should not be less than the number of grid points DL_POLY_2 is expecting. (This number is given by the parameter `mxgrid` in the DL_PARAMS.INC file - see section 8.1.1.) DL_POLY_2 will re-interpolate the tables if `ngrid` ≥ `mxgrid`, but will abort if `ngrid` < `mxgrid`.

The potential and force tables are used to fill the internal arrays `vvv` and `ggg` respectively (see section 2.3.1). The contents of force arrays are derived from the potential via the formula:

$$G(r) = -r \frac{\partial}{\partial r} U(r).$$

Note this is *not* the same as the true force.

4.2 The OUTPUT Files

DL_POLY_2 produces up to seven output files: HISTORY, OUTPUT, REVCON, REVIVE, RFDAT, ZDNDAT and STATIS. These respectively contain: a dump file of atomic coordinates, velocities and forces; a summary of the simulation; the restart configuration; statistics accumulators; radial distribution data, Z-density data and a statistical history.

4.2.1 The HISTORY File

The HISTORY file is the dump file of atomic coordinates, velocities and forces. Its principal use is for off-line analysis. The file is written by the subroutines TRAJECT or TRAJECT_U. The control variables for this file are `ltraj`, `nstraj`, `istraj` and `keytrj` which are created internally, based on information read from the `traj` directive in the CONTROL file (see above). The HISTORY file will be created only if the directive `traj` appears in the CONTROL file. Note that the HISTORY file can be written in either a formatted or unformatted version. We describe each of these separately below.

The HISTORY file can become *very* large, especially if it is formatted. For serious simulation work it is recommended that the file be written to a scratch disk capable of accommodating a large data file. Alternatively the file may be written as unformatted (below), which has the additional advantage of speed. However, writing an unformatted file has the disadvantage that the file may not be readily readable except by the machine on which it was created. This is particularly important if graphical processing of the data is required.

4.2.1.1 The Formatted HISTORY File

The formatted HISTORY file is written by the subroutine TRAJECT and has the following structure.

record 1 (a80)		
header	a80	file header
record 2 (2i10)		
keytrj	integer	trajectory key (see table 4.3)
imcon	integer	periodic boundary key (see table 4.6)
natms	integer	number of atoms in simulation cell

For timesteps greater than `nstraj` the HISTORY file is appended at intervals specified by the `traj` directive in the CONTROL file, with the following information for each configuration:

record i (a8,4i10,f12.6)		
timestep	a8	the character string "timestep"
nstep	integer	the current time-step
natms	integer	number of atoms in configuration
keytrj	integer	trajectory key (again)

imcon	integer	periodic boundary key (again)
tstep	real	integration timestep
record ii (3g12.4) for imcon > 0		
cell(1)	real	x component of <i>a</i> cell vector
cell(2)	real	y component of <i>a</i> cell vector
cell(3)	real	z component of <i>a</i> cell vector
record iii (3g12.4) for imcon > 0		
cell(4)	real	x component of <i>b</i> cell vector
cell(5)	real	y component of <i>b</i> cell vector
cell(6)	real	z component of <i>b</i> cell vector
record iv (3g12.4) for imcon > 0		
cell(7)	real	x component of <i>c</i> cell vector
cell(8)	real	y component of <i>c</i> cell vector
cell(9)	real	z component of <i>c</i> cell vector

This is followed by the configuration for the current timestep. i.e. for each atom in the system the following data are included:

record a (a8,i10,2f12.6)		
atmnam	a8	atomic label
iatm	i10	atom index
weight	f12.6	atomic mass (a.m.u.)
charge	f12.6	atomic charge (e)
record b (3e12.4)		
xxx	real	x coordinate
yyy	real	y coordinate
zzz	real	z coordinate
record c (3e12.4) only for keytrj > 0		
vxx	real	x component of velocity
vyy	real	y component of velocity
vzz	real	z component of velocity
record d (3e12.4) only for keytrj > 1		
fx	real	x component of force
fy	real	y component of force
fz	real	z component of force

Thus the data for each atom is a minimum of two records and a maximum of 4.

4.2.1.2 The Unformatted HISTORY File

The unformatted HISTORY file is written by the subroutine `TRAJECT_U` and has the following structure:

```

record 1
  header                configuration name (character*80)
record 2
  natms                 number of atoms in the configuration (real*8)
record 3
  atname(1,...,natms)   atom names or symbols (character*8)
record 4
  weight(1,...,natms)   atomic masses (real*8)
record 5
  charge(1,...,natms)   atomic charges (real*8)

```

For time-steps greater than **nstraj**, the HISTORY file is appended, at intervals specified by the **traj** directive in the CONTROL file, with the following information:

```

record i
  nstep                the current time-step (real*8)
  natms                number of atoms in configuration (real*8)
  keytrj               trajectory key (real*8)
  imcon                image convention key (real*8)
  timestep              integration timestep (real*8)
record ii for imcon > 0
  cell(1,...,9)        a, b and c cell vectors (real*8)
record iii
  xxx(1,...,natms)     atomic x-coordinates (real*8)
record iv
  yyy(1,...,natms)     atomic y-coordinates (real*8)
record v
  zzz(1,...,natms)     atomic z-coordinates (real*8)
record vi only for keytrj > 0
  vxx(1,...,natms)     atomic velocities x-component (real*8)
record vii only for keytrj > 0
  vyy(1,...,natms)     atomic velocities y-component (real*8)
record viii only for keytrj > 0
  vzz(1,...,natms)     atomic velocities z-component (real*8)
record ix only for keytrj > 1
  fxx(1,...,natms)     atomic forces x-component (real*8)
record x only for keytrj > 1
  fyy(1,...,natms)     atomic forces y-component (real*8)
record xi only for keytrj > 1
  fzz(1,...,natms)     atomic forces z-component (real*8)

```

Note the implied conversion of integer variables to real on record i.

4.2.2 The OUTPUT File

The job output consists of 7 sections: Header; Simulation control specifications; Force field specification; Summary of the initial configuration; Simulation progress; Summary of statistical data; Sample of the final configuration; and Radial distribution functions. These sections are written by different subroutines at various stages of a job. Creation of the OUTPUT file *always* results from running DL_POLY_2 . It is meant to be a human readable file, destined for hardcopy output.

4.2.2.1 Header

Gives the DL_POLY_2 version number, the number of processors used and a title for the job as given in the header line of the input file CONTROL. This part of the file is written from the subroutines DLPOLY and SIMDEF

4.2.2.2 Simulation Control Specifications

Echoes the input from the CONTROL file. Some variables may be reset if illegal values were specified in the CONTROL file. This part of the file is written from the subroutine SIMDEF.

4.2.2.3 Force Field Specification

Echoes the FIELD file. A warning line will be printed if the system is not electrically neutral. This warning will appear immediately before the non-bonded short-range potential specifications. This part of the file is written from the subroutine SYSDEF.

4.2.2.4 Summary of the Initial Configuration

This part of the file is written from the subroutine SYSGEN. It states the periodic boundary specification, the cell vectors and volume (if appropriate) and the initial configuration of (a maximum of) 20 atoms in the system. The configuration information given is based on the value of `levcfg` in the CONFIG file. If `levcfg` is 0 (or 1) positions (and velocities) of the 20 atoms are listed. If `levcfg` is 2 forces are also written out.

For periodic systems this is followed by the long range corrections to the energy and pressure.

4.2.2.5 Simulation Progress

This part of the file is written by the DL_POLY_2 root segment DLPOLY. The header line is printed at the top of each page as:

```
-----
      step  eng_tot  temp_tot  eng_cfg  eng_vdw  eng_cou  eng_bnd  eng_ang  eng_dih  eng_tet
time(ps)  eng_pv   temp_rot  vir_cfg  vir_vdw  vir_cou  vir_bnd  vir_ang  vir_con  vir_tet
cpu  (s)   volume  temp_shl  eng_shl  vir_shl   alpha    beta    gamma  vir_pmf  press
```

The labels refer to :

line 1

<code>step</code>	MD step number
<code>eng_tot</code>	total internal energy of the system
<code>temp_tot</code>	system temperature
<code>eng_cfg</code>	configurational energy of the system
<code>eng_vdw</code>	configurational energy due to short-range potentials
<code>eng_cou</code>	configurational energy due to electrostatic potential
<code>eng_bnd</code>	configurational energy due to chemical bond potentials
<code>eng_ang</code>	configurational energy due to valence angle and three-body potentials
<code>eng_dih</code>	configurational energy due to dihedral inversion and four-body potentials
<code>eng_tet</code>	configurational energy due to tethering potentials

line 2

<code>time(ps)</code>	elapsed simulation time (ps) since the beginning of the job
<code>eng_pv</code>	enthalpy of system
<code>temp_rot</code>	rotational temperature
<code>vir_cfg</code>	total configurational contribution to the virial
<code>vir_vdw</code>	short range potential contribution to the virial
<code>vir_cou</code>	electrostatic potential contribution to the virial
<code>vir_bnd</code>	chemical bond contribution to the virial
<code>vir_ang</code>	angular and three body potentials contribution to the virial
<code>vir_con</code>	constraint bond contribution to the virial
<code>vir_tet</code>	tethering potential contribution to the virial

line 3

<code>cpu (s)</code>	elapsed cpu time since the beginning of the job
<code>volume</code>	system volume
<code>temp_shl</code>	core-shell temperature
<code>eng_shl</code>	configurational energy due to core-shell potentials
<code>vir_shl</code>	core-shell potential contribution to the virial
<code>alpha</code>	angle between <i>b</i> and <i>c</i> cell vectors
<code>beta</code>	angle between <i>c</i> and <i>a</i> cell vectors
<code>gamma</code>	angle between <i>a</i> and <i>b</i> cell vectors
<code>vir_pmf</code>	Potential of mean force constraint contribution to the virial
<code>press</code>	pressure

Note: The total internal energy of the system (variable `tot_energy`) includes all contributions to the energy (including system extensions due to thermostats etc.) It is nominally the *conserved variable* of the system, and is not to be confused with conventional system energy, which is a sum of the kinetic and configuration energies.

The interval for printing out these data is determined by the directive **print** in the CONTROL file. At each time-step that printout is requested the instantaneous values of

the above statistical variables are given in the appropriate columns. Immediately below these three lines of output the rolling averages of the same variables are also given. The maximum number of time-steps used to calculate the rolling averages is determined by the parameter **mxstak** in the DL_PARAMS.INC file (see section 8.1.1). The working number of time-steps for rolling averages is controlled by the directive **stack** in file CONTROL (see above). The default value is **mxstak**.

Energy Units: The energy unit for the data appearing in the OUTPUT is defined by the **units** directive appearing in the CONTROL file.

Pressure units: The unit of pressure is *k bar*, irrespective of what energy unit is chosen.

4.2.2.6 Summary of Statistical Data

This portion of the OUTPUT file is written from the subroutine RESULT. The number of time-steps used in the collection of statistics is given. Then the averages over the production portion of the run are given for the variables described in the previous section. The root mean square variation in these variables follow on the next two lines. The energy and pressure units are as for the preceeding section.

Also provided in this section is an estimate of the diffusion coefficient for the different species in the simulation, which is determined from a *single time origin* and is therefore very approximate. Accurate determinations of the diffusion coefficients can be obtained using the MSD utility program, which processes the HISTORY file (see chapter 7).

If an NPT or N σ T simulation is performed the OUTPUT file also provides the mean stress (pressure) tensor and mean simulation cell vectors.

4.2.2.7 Sample of Final Configuration

The positions, velocities and forces of the 20 atoms used for the sample of the initial configuration (see above) are given. This is written by the subroutine RESULT.

4.2.2.8 Radial Distribution Functions

If both calculation and printing of radial distribution functions have been requested (by selecting directives **rdf** and **print rdf** in the CONTROL file) radial distribution functions are printed out. This is written from the subroutine RDF1. First the number of time-steps used for the collection of the histograms is stated. Then each function is given in turn. For each function a header line states the atom types ('a' and 'b') represented by the function. Then r , $g(r)$ and $n(r)$ are given in tabular form. Output is given from 2 entries before the first non-zero entry in the $g(r)$ histogram. $n(r)$ is the average number of atoms of type 'b' within a sphere of radius r around an atom of type 'a'. Note that a readable version of these data is provided by the RDFDAT file (below).

4.2.2.9 Z Density Profile

If both calculation and printing of Z density profiles has been requested (by selecting directives **zden** and **print rdf** in the CONTROL file Z density profiles are printed out as the last part of the OUTPUT file. This is written by the subroutine ZDEN1. First the number of time-steps used for the collection of the histograms is stated. Then each function is given in turn. For each function a header line states the atom type represented by the function. Then z , $\rho(z)$ and $n(z)$ are given in tabular form. Output is given from $Z = [-L/2, L/2]$ where L is the length of the MD cell in the Z direction and $\rho(z)$ is the mean number density. $n(z)$ is the running integral from $-L/2$ to z of $(xy \text{ cell area})\rho(s)ds$. Note that a readable version of these data is provided by the ZDNDAT file (below).

4.2.3 The REVCON File

This file is formatted and written by the subroutine REVIVE. REVCON is the restart configuration file. The file is written every **ndump** time steps in case of a system crash during execution and at the termination of the job. A successful run of DL_POLY_2 will always produce a REVCON file, but a failed job may not produce the file if an insufficient number of timesteps have elapsed. **ndump** is a parameter defined in the DL_PARAMS.INC file found in the *source* directory of DL_POLY_2 (section 8.1.1). Changing **ndump** necessitates recompiling DL_POLY_2. REVCON is identical in format to the CONFIG input file (see section 4.1.2). REVCON should be renamed CONFIG to continue a simulation from one job to the next. This is done for you by the *copy* macro supplied in the *execute* directory of DL_POLY_2.

4.2.4 The REVIVE File

This file is unformatted and written by the subroutine REVIVE. It contains the accumulated statistical data. It is updated whenever the file REVCON is updated (see previous section). REVIVE should be renamed REVOLD to continue a simulation from one job to the next. This is done by the *copy* macro supplied in the *execute* directory of DL_POLY_2. In addition, to continue a simulation from a previous job the **restart** keyword must be included in the CONTROL file.

The format of the REVIVE file is identical to the REVOLD file described in section 4.1.4.

4.2.5 The RDFDAT File

This is a formatted file containing em Radial Distribution Function (RDF) data. Its contents are as follows:

record 1		
cfgname	character (A80)	configuration name
record 2		
ntpdw	integer (i10)	number of RDFs in file

mxrdf	integer (i10)	number of data points in each RDF

There follow the data for each individual RDF i.e. *ntpvduw* times. The data supplied are as follows:

first record

atname 1	character (A8)	first atom name
atname 2	character (A8)	second atom name

following records (*mxrdf* records)

radius	real (e14)	interatomic distance (A)
g(r)	real (e14)	RDF at given radius.

Note the RDFDAT file is optional and appears when the **print rdf** option is specified in the CONTROL file.

4.2.6 The ZDNDAT File

This is a formatted file containing the Z-density data. Its contents are as follows:

record 1

cfgname	character (A80)	configuration name

record 2

mxrdf	integer (i10)	number of data points in the Z-density function

following records (*mxrdf* records)

z	real (e14)	distance in z direction (A)
$\rho(z)$	real (e14)	Z-density at given height z

Note the ZDNDAT file is optional and appears when the **print rdf** option is specified in the CONTROL file.

4.2.7 The STATIS File

The file is formatted, with integers as “i10” and reals as “e14.6”. It is written by the subroutine *STATIC*. It consists of two header records followed by many data records of statistical data.

record 1

cfgname	character	configuration name

record 2

string	character	energy units

Data records

Subsequent lines contain the instantaneous values of statistical variables dumped from the array **stpval**. A specified number of entries of **stpval** are written in the format “(1p,5e14.6)”. The number of array elements required (determined by the parameter **mxnstk** in the DL_PARAMS.INC file) is

$$\begin{aligned} \text{mxnstk} \geq & 27 + \text{nptatm}(\text{number of unique atomic sites}) \\ & + 9(\text{if stress tensor calculated}) \\ & + 9(\text{if constant pressure simulation requested}) \end{aligned}$$

The STATIS file is appended at intervals determined by the **stats** directive in the CONTROL file. The energy unit is as specified in the CONTROL file with the the **units** directive, and are compatible with the data appearing in the OUTPUT file. The contents of the appended information is:

record i		
nstep	integer	current MD time-step
time	real	elapsed simulation time = nstep × Δt
nument	integer	number of array elements to follow
record ii stpval(1) – stpval(5)		
engcns	real	total extended system energy (i.e. the conserved quantity)
temp	real	system temperature
engcfg	real	configurational energy
engsrp	real	short range potential energy
engcpe	real	electrostatic energy
record iii stpval(6) – stpval(10)		
engbnd	real	chemical bond energy
engang	real	valence angle and 3-body potential energy
engdih	real	dihedral interaction energy
engtet	real	tethering energy
enthal	real	enthalpy (total energy + PV)
record iv stpval(11) – stpval(15)		
tmprot	real	rotational temperature
vir	real	total virial
virsrp	real	short-range virial
vircpe	real	electrostatic virial
virbnd	real	bond virial
record v stpval(16) – stpval(20)		
virang	real	valence angle and 3-body virial
vircon	real	constraint virial
virtet	real	tethering virial
volume	real	volume
tmpshl	real	core-shell temperature

```

record vi stpval(21) -stpval(25)
  engshl      real      core-shell potential energy
  virshl      real      core-shell virial
  alpha       real      MD cell angle  $\alpha$ 
  beta        real      MD cell angle  $\beta$ 
  gamma       real      MD cell angle  $\gamma$ 
record vii stpval(26) -stpval(27)
  virpmf      real      Potential of Mean Force virial
  press       real      pressure
the next ntpatm entries
  amsd(1)     real      mean squared displacement of first atom types
  amsd(2)     real      mean squared displacement of second atom types
  ...
  amsd(ntpatm) real      mean squared displacement of last atom types
the next 9 entries - if the stress tensor is calculated
  stress(1)   real      xx component of stress tensor
  stress(2)   real      xy component of stress tensor
  stress(3)   real      xz component of stress tensor
  stress(4)   real      yx component of stress tensor
  ...
  stress(9)   real      zz component of stress tensor
the next 9 entries - if a NPT simulation is undertaken
  cell(1)     real      x component of a cell vector
  cell(2)     real      y component of a cell vector
  cell(3)     real      z component of a cell vector
  cell(4)     real      x component of b cell vector
  ...
  cell(9)     real      z component of c cell vector

```

Note. The stress tensor is calculated only if the code is compiled with the “-STRESS” option (see section 3.2.1). Cell shape varying constant pressure simulations (keyword **ensemble nst ...** in the CONTROL file) are only possible if the stress tensor is calculated. If isotropic constant pressure simulations are required, where the cell volume but not the shape may vary, (keyword **ensemble npt ...**) the stress tensor need not be calculated.

Chapter 5

The DL_POLY_2 Java Graphical User Interface

Scope of Chapter

This chapter describes the Java GUI in detail, its functionality, compilation and operation.

5.1 The DL_POLY_2 Java Graphical User Interface

The DL_POLY_2 Java GUI was designed as an alternative to the original GUI based on the CERIU2 Visualiser and SDK software from MSI. The advantages of this second GUI lie in the free availability of Java, a product of Sun Microsystems, its suitability for constructing GUIs of any description, and its portability to all computers running Java. The result is a portable GUI that can be released freely to all DL_POLY users and which can be developed as desired by users with special needs, provided they are of course willing to learn the Java language.

The first release of the Java GUI came with version 2.13 of DL_POLY_2 . It incorporated most of the features of the original GUI, at (it must be said) a price of somewhat inferior graphics. However the result isn't so bad and the future promises improvement in all areas.

before proceeding, the reader is invited to pay particular attention to the points below.

1. **The DL_POLY_2 Java GUI is not an applet. It is a full Java application program and cannot be run from within an internet browser.**
2. **The appearance of the GUI, its windows and browsers etc may vary from system to system. The broad appearance will remain the same as will the functionality. This is merely an aspect of Java portability.**
3. **For the purpose of rendering different atom types in a configuration, the GUI requires the adoption of a naming convention based on the Periodic Table. Up to eight characters may specify an atom name, but the first two must be taken from the proper chemical symbol. If the chemical symbol is a single letter, the underscore (_) must be used. Thus hydrogen and oxygen have the names O_ and H_, while copper and sodium have Cu and Na respectively. Failure to follow this convention results in the 'grey ball' syndrome. The only exceptions to these rules are the use of symbols OW and HW for oxygen and hydrogen in water molecules. Other atom naming conventions can be fitted into this scheme by extension beyond two characters. The GUI exploits this in the Dreiding [7] and Ceramics [46, 47] force fields. The user may see what the atom names are in these force fields by inspecting the MINDREI and CERAMICS files in the java subdirectory, or by listing them from the Information menu of the GUI (see section 5.11).**

5.2 Compiling the Java GUI

The first requirement for compiling the Java GUI is the availability of the Java software. For national supercomputers and the like, Java should already be available. If it isn't the system managers should be reminded of their responsibilities to the user community. To install Java on local machines, the Java home page at the Sun Microsystems website

[http : //java.sun.com](http://java.sun.com)

provides the Java Development Kit (JDK 1.3 or above) for any particular computer. Installation instructions are available from the same site, and are generally straightforward. Once the JDK is installed, the DL_POLY_2 Java GUI may be compiled.

The source code for the Java GUI is found in the DL_POLY_2 *java* subdirectory. On entering this type the command:

```
javac *.java
```

which will compile the java source code and construct the Java classes (i.e. the Java executable ‘objects’).

Next it is necessary to make a Java ‘jar’ file from the classes. This neatly encapsulates, in a single file, all of the GUI Java classes. (The jar file effectively becomes the GUI executable, which in fact is transportable between systems.) This is done with the command:

```
jar -cfm GUI.jar manifest.mf *.class
```

the *jar* command thus works somewhat like the UNIX *tar* command. Note however the requirement to incorporate the file manifest.mf, which is one of the files in the *java* subdirectory. The contents of this file inform the java program, which of the incorporated classes represent the entry point at execution. (It follows that this file should never be deleted!) The result of this command is the GUI.jar file, which becomes the working GUI.

In addition to compiling the Java source, it is necessary to compile some FORTRAN programs that the Java calls upon to perform some of the calculations. Inside the *java* subdirectory are the lower directories RDF, GDF and SKW. These contain programs for calculating radial distribution functions, van Hove distinct correlation functions, and dynamic structure factors respectively. Enter each of these directories and type *make* to build the FORTRAN programs. This will complete the build of the GUI. Note it may be necessary to change a few of the parameters in the makefiles for these, such as the specification of the FORTRAN77 compiler or the C compiler.

5.3 Starting the Java GUI

To run the GUI successfully, it must be started from within the DL_POLY_2 *execute* subdirectory. Only from there will the GUI be able to locate the support files it needs from within the DL_POLY_2 directory structure. (Users who experience strange problems running the GUI, should consider this as the most likely cause.) To start the GUI enter the command (MS Windows users are advised to open a command window for this purpose):

```
java -jar ../java/GUI.jar
```

Users may alternatively use the *gui* script in the *execute* subdirectory, which incorporates the syntax of this command. After a minute or two, the GUI **Graphics** window and the **Monitor** window (figure 1) will open on-screen.

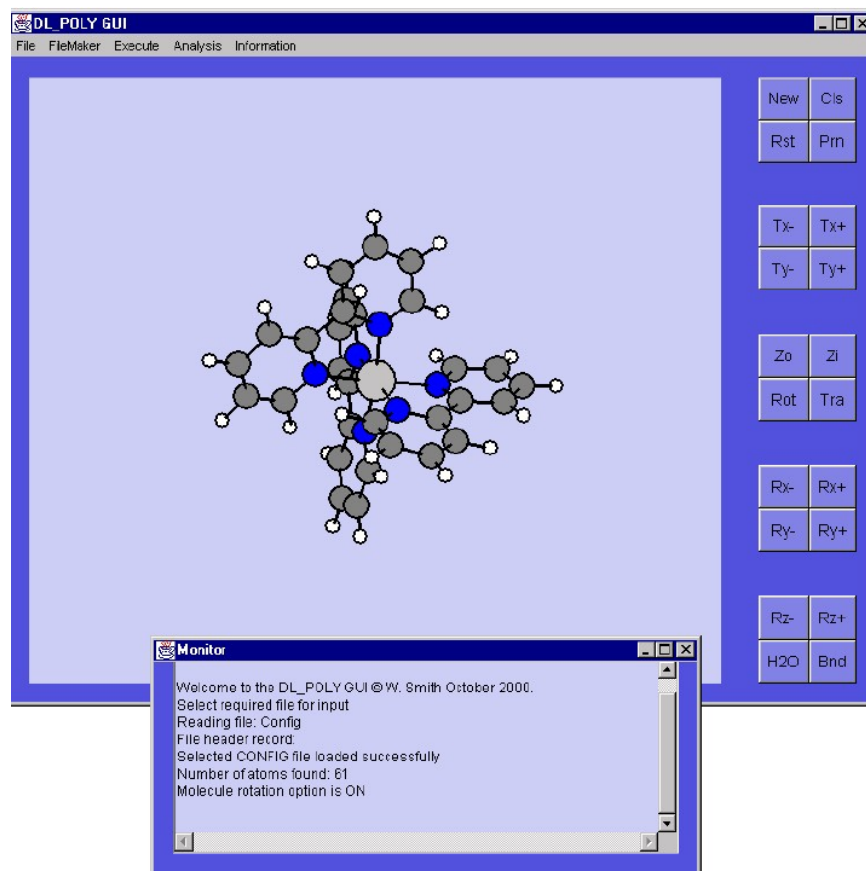


Figure 1: The Java GUI and Monitor window

Both of these windows may be handled in the usual X manner, e.g. moved with the mouse, by clicking and dragging the header panel. The size of each may be changed by clicking and dragging from a corner or edge. The usual window widgets for hiding, enlarging or closing are present. However **note that the close widget has been disabled for all Java GUI windows except the Graphics Window.**

If the colour scheme of the GUI offends, try the command:

```
java -jar ../java/GUI.jar scheme
```

where *scheme* is one of the following: **photo**, **monet**, **picasso**, **vangoch**, **cezanne** or **mondrian**, and see that things could be worse. The default colour scheme is **picasso**.

5.4 The Graphics Window

The Graphics window is the area in which the GUI draws pictures of any molecular configuration selected by the user, or produced by the GUI. The view represents the x,y plane, with the x-axis horizontal, y-axis vertical and the z-axis projecting out of the screen. On the right of this window is a collection of buttons, which manipulate the image on display. These are activated by a single click of the mouse button. Their functions are as follows:

- **New** - opens a file browser (see figure 2) for the user to select a new configuration file for viewing. This button assumes the configuration file is a DL_POLY_2 CONFIG or REVCON file.¹
- **Cls** - clears the image from the screen and deletes the configuration data from the GUI internal memory.
- **Rst** - resets the image to the original picture when the configuration was first loaded. i.e. undoes all user manipulation with the GUI.
- **Prn** - throws up a print panel so the user may either print the image or save the image as a file.
- **Tx-** - moves (translates) the image to the left.
- **Tx+** - moves the image to the right.
- **Ty-** - moves the image down.
- **Ty+** - moves the image up.
- **Zo** - zooms out from the image.
- **Zi** - zooms in on the image.
- **Rot** - rotates the image to follow the dragged cursor.
- **Tra** - moves the image to follow the dragged cursor.
- **Rx-** - rotates the image clockwise about the x axis.
- **Rx+** - rotates the image anticlockwise about the x axis.
- **Ry-** - rotates the image clockwise about the y axis.
- **Ry+** - rotates the image anticlockwise about the y axis.
- **Rz-** - rotates the image clockwise about the z axis.

¹The GUI normally performs some preselection of the files appearing in the browser to reduce the risk of improper selection. However, the Windows file browser currently ignores the preselection attempted by the GUI, so inappropriate files may still appear. Beware.

- **Rz+** - rotates the image anticlockwise about the z axis.
- **H2O** - toggles the visibility of water molecules.
- **Bnd** - toggles the visibility of stick bonds.

Note that toggling the visibility of water molecules assumes that the oxygen atom is labelled as OW and the hydrogen atoms as HW in the CONFIG file.

5.5 The Monitor Window

The Monitor window is the medium through which the GUI informs the user of actions taken, files created or deleted, errors made by the user etc. It is also the area where text files are displayed if the user requests it. Ideally this window should be kept visible at all times. It is expandable if required and can be reduced (hidden) but not deleted.

Note that if there are catastrophic failures during the running of the GUI, the error messages will, of necessity, not appear in this window, but in the X or command window in which the GUI was first invoked.

5.6 The GUI Application Menus

Also appearing in the Graphics window (top left) is a menu bar with a series of drop down menus (made visible by clicking on the menu name). These menus select the various applications buried in the GUI. The applications are discussed in detail below. The current menus are:

- **File** - handles file operations, such as viewing and deleting, resets the GUI and various defaults, and also quits (shuts down) the GUI.
- **FileMaker** - enables construction of various input files for DL_POLY_2 .
- **Execute** - controls the execution of DL_POLY_2 and the selection and storage of I/O files.
- **Analysis** - runs the DL_POLY_2 analysis programs to analyse the simulation results.
- **Information** - provides copyright, licencing and other information.

The contents of each menu are made visible by clicking on the menu header.

5.7 File Menu

The File menu contains the following items

1. **Quit**
2. **View File**

3. Delete File

4. Defaults

5. Reset

To select any item from the menu, the mouse cursor must be dragged down the list and released on the item of choice. Selecting any of these items will initiate a particular action by the GUI. These actions are now described.

5.7.1 Quit

Selection of this item closes down the GUI, provided it is not in a busy state (i.e. performing some other action). No data saving results from this action. Amendments to the GUI defaults will be lost. The GUI may also be shut down by clicking on the standard close widget on the GUI window. Note this is the only instance where a default close widget will have any effect.

5.7.2 View File

Any text file in the *execute* subdirectory may be viewed by selecting this item. On selection the GUI opens a file browser (figure 2) allowing the user to select the file of interest. Selection of the file will result in its display in the Monitor window (which may need to be enlarged for a convenient viewing).



Figure 2: A typical file browser

Note that only text files should be viewed, for obvious reasons. Large files may take quite a while to list.

5.7.3 Delete File

Any file in the *execute* subdirectory may be deleted by selecting this item. On selection a file browser appears to permit the user to choose the required file. Selection of the file

will cause its deletion. As a fail safe, the GUI will first throw up a dialog window to give opportunity to cancel the action.

5.7.4 Defaults

The GUI contains a number of defaults that may be altered after start up. These include array dimension specifications and GUI operational parameters, such as default rotation angles and translation distances. Selection of this item opens the Change Defaults panel (figure 3).

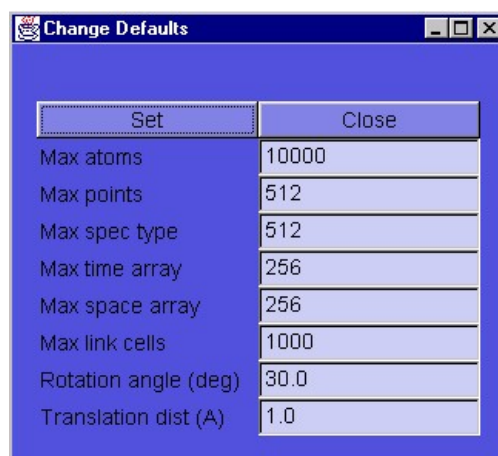


Figure 3: The Change Defaults Panel

The text boxes on the panel show the current values for the defaults. Clicking on the text box beside the parameter description will allow the user to change the value. Clicking the **Set** button will reset the GUI default. Sometimes this also resets the entire GUI, if necessary. Clicking the **Close** button will delete the panel.

The parameters that may be changed are:

- Max atoms - maximum number of atoms in a configuration.
- Max points - maximum number of data points in a graph plot.
- Max spec type - maximum number of different types of atom in configuration.
- Max time array - max number of data points in time correlation functions.
- Max space array - max number of data points in spatial correlation functions.
- max link cells - maximum number of link cells in solvent adding methods.
- Rotation angle (deg) - angle of rotation for Graphics window rotation buttons.
- Translation dist (A) - distance of translation for Graphics window translation buttons.

5.7.5 Reset

This action resets the GUI variables back to their starting values (not current defaults) and clears all loaded data from the GUI.

5.8 FileMaker Menu

The FileMaker menu has the following items, some of which are themselves submenus.

1. **CONTROL**
2. **CONFIG**
3. **FIELD**
4. **Display**
5. **Tools**

The functions of these are as follows.

5.8.1 CONTROL

Selecting this menu item will open the first of three panels dedicated to constructing a DL_POLY_2 CONTROL file. A second panel is opened by clicking the **More** button at the bottom of this, and a third panel is opened by clicking the **More** button at the bottom of the second panel. The three panels (figure 4) allow specification of all the options available in DL_POLY_2, as specified in the DL_POLY_2 manual, section 4.1.1. The details of each item do not require elaboration here.

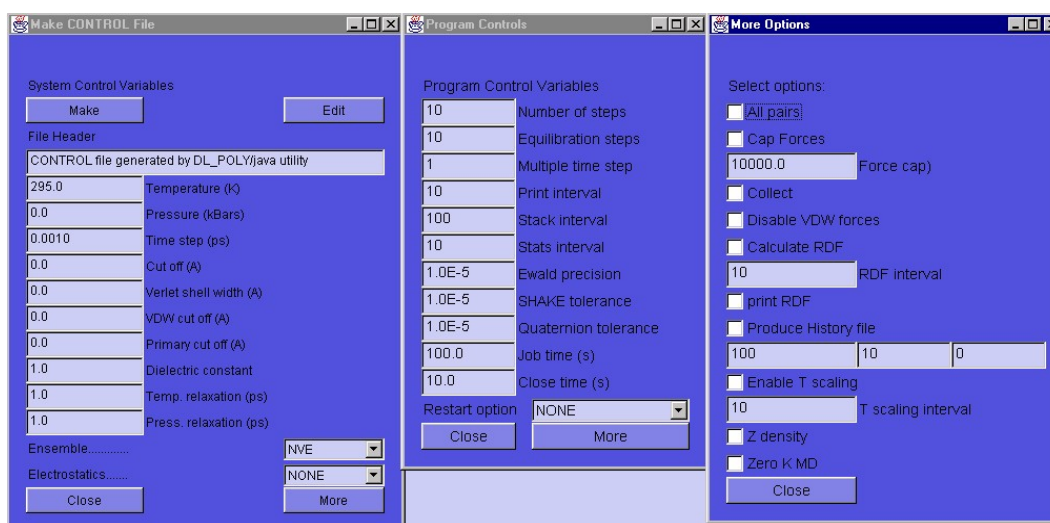


Figure 4: The CONTROL file panels

Specifying most values required by the CONTROL file is matter of amending the contents of the associated text box. Some variables however, require activation of logical options using ‘check boxes’ (particularly panel 3). Yet others require a choice from a menu of options. The choice of ensemble or electrostatic method on panel 1 or the choice of restart option on panel 2 are examples of this. The user will find the panels intuitively straightforward.

When the variables have been defined, the user should click the **Make** button on panel 1, to create the required CONTROL file. The GUI will name this file CNTRL.n, where n is some integer. Note that the GUI has some internal consistency checks and may refuse to make a CONTROL file that is improperly specified. Watch the Monitor window for details.

Another way of making a CONTROL file is to use the **Edit** button on panel 1 to load an existing CONFIG file. This may then be altered in some way through the panels and saved using the **Make** button. A new CONTROL file will be made alongside the old.

When all is finished, the CONTROL file panels may be deleted using the close button on panel 1. Note you cannot open more than one set of CONTROL panels. This is a restriction that applies to most other panels of the GUI, otherwise things could get confusing.

5.8.2 CONFIG

This menu item is intended for building DL_POLY_2 CONFIG files. It presents a submenu with the following items

1. Lattice

This item is used for building CONFIG files that may be constructed from a small unit cell, such as a crystal structure. A panel is opened (figure 5) by this option which enables full specification of the CONFIG file contents.

On the panel are several text boxes grouped in threes. The first of these specifies the unit cell A vector, the second the B vector and the third, the C vector. The distances are expressed in Å. Beneath these is a set of three boxes to specify the integer multiplication factor for the unit cell in each of the principal directions ($n_a \times n_b \times n_c$). The atomic basis for the unit cell is defined at the bottom of the panel. For each atom in the unit cell an atom name may be entered in the text box, followed by the three fractional coordinates of this atom in the unit cell, The **Enter** button will register the specified atom, with an accumulation number displayed on the panel, bottom right. (Note: It is not essential to use the GUI naming conventions here, but it is wise to do so, especially if one of the recognised GUI force fields is to be used later.)

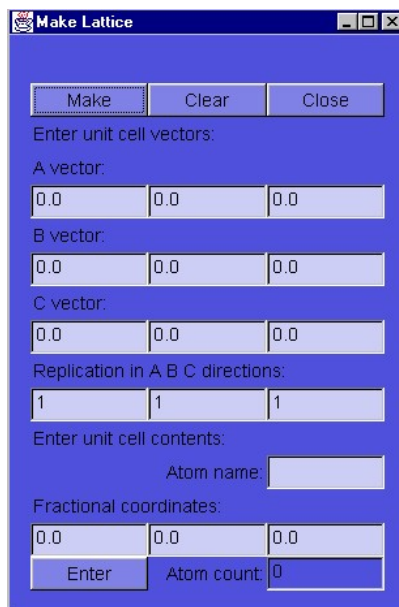


Figure 5: The Make Lattice Panel

When all the required basis atoms have been entered, the user should click the **Make** button to create the desired CONFIG file, which the GUI will name CFGLAT.n, with n some integer. Simultaneous with this, the lattice is displayed in the Graphics window. The panel can be reset by using the **Clear** button.

The **Close** button deletes the 'Make Lattice' panel.

2. Chain

This panel invoked by this option (figure 6) enables construction of a small range of chain molecules, particularly surfactants, which may then form the basis for a layered system.

The panel supports several labelled text boxes, with which the user may specify the number of carbon atoms in the chain and the XY area per chain (in $\text{\AA}^2$), as required for a layer definition. (A hexagonal arrangement of the chains is assumed.) The chain may be assigned a head group from the menu box on the panel. The current choices are:

- (a) none;
- (b) soap i.e. $-\text{CO.ONa}$;
- (c) carboxy i.e. $-\text{CO.OH}$;
- (d) phenol i.e. $-\text{C}_6\text{H}_4\text{OH}$;
- (e) TAB, trimethylammino bromide i.e. $(-\text{N}(\text{CH}_3)_3^+.\text{Br}^-)$;
- (f) $(\text{EO})_n$, polyethylether i.e. $(-\text{C}_2\text{H}_4\text{O}-)_n$.

The number n in the $(EO)_n$ case is specified in a labelled text box on the panel. There are also two check boxes. One activates the 'flip' option, which turns the chain through 180 degrees, the other duplicates the chain and stacks the two chains one above the other in the z-direction, as in a double layer. The separation between the two is determined by the value in the 'Z-gap' text box. When the user has selected the required details, clicking the **Make** button will create the CFGCHN.i file, where 'i' is some integer, and simultaneously display the molecule in the Graphics window. Note that the atom naming convention adopted by this facility is compatible with the DL_POLY_2 and Dreiding conventions.



Figure 6: The Make Chain Panel

Clicking the **Close** button will delete the 'Make Chain' panel.

Note that in order to build a full layered system, the user should use the N_fold option that appears under the FileMaker/Tools menu.

3. Polymer

The polymer panel opened by this option (figure 7) provides a means to construct an amorphous polymer chain by a temperature dependent self avoiding random walk.

Make Polymer

Make Close

1 Number of C atoms

System Volume (A³) and Temperature (K):

1000.0 100.0

C-C and C-H Bondlengths (A):

1.54 1.09

C-C, C-H and H-H LJ Epsilon (kJ/mol):

0.2977 0.1573 0.0831

C-C, C-H and H-H LJ Sigma (A):

3.4 3.1 2.8

348.0 C-C Bond Energy (kJ/mol)

Dihedral Energy Constants (kJ/mol):

8.832 18.087 4.88 -31.8

0.5 Dihedral LJ Scaling Factor

☒ PBC

Figure 7: The Make Polymer Panel

This is a rather complicated panel, with many adjustable parameters, but the user need not know too much detail to use it. The first requirement is to specify the desired length of the chain in the first text box. The user needs then to specify the required system volume and temperature in the appropriate labelled text boxes. Clicking the **Make** button will start the Monte Carlo process that generates the chain. If this is successful a CONFIG file named CFGPOL.n, where n is some integer, will be written. The atom naming convention adopted by this facility is compatible with the DLPOLY_2 and Dreiding conventions.

Other labelled text boxes on the panel allow the user to change the particulars of the model. The bondlengths, Lennard-Jones parameters (ϵ, σ), the C-C bond energy, and the (third order cosine polynomial) dihedral potential parameters are all modifiable, as is the strength of the dihedral 1-4 Lennard-Jones term, which may be scaled by a specified factor.

The chain is always grown with a cubic periodic boundary condition applied. The volume of the box is that specified by the user. However, selecting the PBC check box on the panel, will result in the chain coordinates being written into the CFGPOL file in a 'folded' manner, as opposed to a contiguous, unbroken chain.

The **Close** button deletes the 'Make Polymer' panel.

The chain build may not be successful, depending on the temperature and volume specified by the user. If unsuccessful the user is advised to increase the volume until

success is achieved. The required density may always be obtained in a DL_POLY_2 simulation using one of the NPT ensemble options.

4. Bucky

The panel for this option (figure 8) enables the building of a buckminsterfullerene (C60) molecule or a fullerene tube.

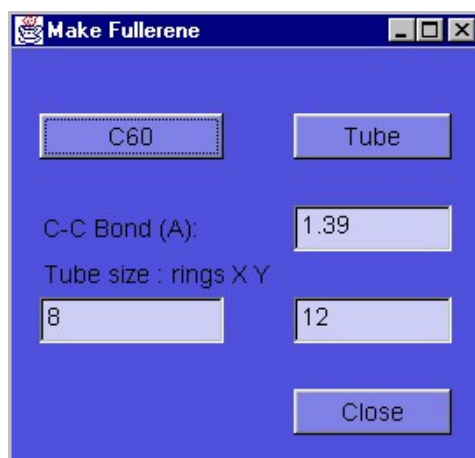


Figure 8: The Make Fullerene Panel

To make a C60 molecule the user need only click the **C60** button on the panel. This will create a CONFIG file named CFGBUK.n, with n some integer, and simultaneously display the molecule in the Graphics window. The C-C bondlength may be set to a new value using the labelled text box. To build a fullerene tube, the circumference and length of the required tube must first be specified. These are given in terms of the number of C₆ rings in each direction. Two text boxes are available for this, the labelled X box refers to the circumference, and the Y box to the length. (Note that an odd number of rings specified in the Y box, will produce a continuous tube in the system z direction, by virtue of the periodic boundary condition. An even number will be incommensurate with the PBC.) Clicking the **Tube** button will create the CONFIG file CFGBUK.n and display the structure in the Graphics window.

The **Close** button will delete the 'Make Fullerene' panel.

Note that the atom naming convention adopted by this facility is compatible with the DL_POLY_2 and Dreiding conventions.

5.8.3 FIELD

This menu item is for building DL_POLY_2 FIELD files. It opens a submenu with the following items;

1. Blank

The Blank FIELD panel (figure 9) is useful for analysing the molecular topology for a CONFIG file, since it will identify the atoms, bonds, angles, dihedrals and inversions in the system and list them in a form compatible with the DL_POLY_2 FIELD specification. However it does not assign force field parameters.

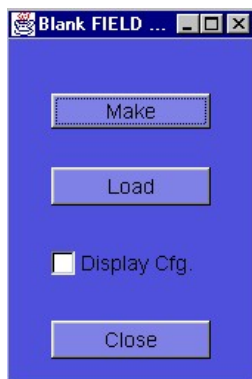


Figure 9: The Blank Field Panel

Operation of this panel is simple. The user needs only to click the **Make** button (if the required molecule is already loaded into the GUI), or the **Load** button (to throw up a file browser to select the required CONFIG file), to produce the blank FIELD file, which will have the name FLDuvw.n, where uvw is a three character string derived from the input CONFIG file name and n is an integer. If the **Load** button is used, the user can display the loaded file by activating the 'Display cfg' option.

The **Close** button deletes the 'Blank Field' panel.

Note that for the Blank FIELD facility to work properly, atomic names in the CONFIG file must obey the GUI atomic naming convention (see 5.1).

2. Dreiding

For CONFIG files that are specified with the Dreiding naming convention [7] (see also the MINIDREI file details under the Information menu described below), the panel for this option (figure 10) may be used to build a compatible FIELD file.

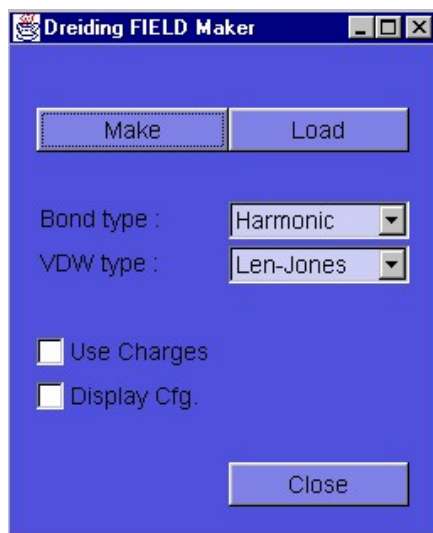


Figure 10: The Dreiding FIELD Maker Panel

This is also a simple panel to operate. A FIELD file may be built from a pre-loaded CONFIG file by clicking the **Make** button, or by clicking the **Load** button to load a file from a browser. The file created will have the name FLDuvw.n , where uvw is a three character string taken from the loaded CONFIG file name and n is an integer. If the **Load** option is taken, the user may display the loaded structure by setting the 'Display cfg.' check box. The nature of the bonding forces may be altered by selecting either *Harmonic* or *Morse* bond potentials, and by choosing either *Len-Jones* or *Buckingham* nonbonded potentials from the box menus. The 'Use charges' check box enables the GUI to pick up prescribed charges for some atoms (such as the chloride ion). In general however this is not very useful, as the charges for the rest of the molecule (if any) have to be obtained from elsewhere.

The **Close** button deletes the 'Dreiding FIELD Maker' panel.

3. Ceramics

For CONFIG files that are specified with the naming conventions for the Ceramics force fields (see the CERAMICS file details under the Information menu described below), the panel for this option (figure 11) will build an appropriate FIELD file.



Figure 11: The Ceramics FIELD Maker Panel

With this panel a ceramic FIELD file may be built from a pre-loaded CONFIG file by clicking the **Make** button, or by clicking the **Load** button to load a file from a browser. The FIELD file will have the name FLDuvw.n, where uvw is a three character string extracted from the loaded CONFIG file name and n is an integer. The user may display the loaded structure, if not already visible, by selecting the 'Display cfg.' check box. Five ceramic force fields are available:

- (a) LC_a, divalent and tetravalent rigid ions [46];
- (b) LC_b, trivalent rigid ions [46];
- (c) LC_c1, shell model ions [46];
- (d) LC_c2, shell model ions [46];
- (e) GULP, shell model ions [47];

The user may choose any of these from the menu on the panel. The LC_c1 force field distinguishes between ions in tetrahedral and other environments, for which different parameters are available. The labelled check box enables the user to specify the tetrahedral option. The **Close** button deletes the 'Ceramics Field Maker' panel.

4. Table

DLPOLY_2 allows the user to specify nonbonded pair potentials in tabular form. The 'Make TABLE File' panel (figure 12) offers some facilities for making such files.



Figure 12: The Make TABLE File Panel

The prime purpose of this panel is to fit a set of data points describing a potential function, and from the fit construct a full TABLE file for DL_POLY_2 input. Two kinds of fit are available: fitting by splines, or by gaussian functions. The selection is made from the 'Fitting Option' menu on the panel. The user must also specify, in the text boxes provided, the names of the atoms for which the pair potential is being defined. Also required are the range of the potential cut off and the number of points for the tabulation (not to be confused with the number of data points). The user may then enter the data to be fitted either by hand or by loading a data file. The **Load** button enables loading from an XY file, which will then be fitted and a TABLE file created with the name TABSPL.n (spline) or TABGSS.n (gaussian), with n some integer. Specifying the data by hand requires the user to enter the r and V(r) coordinates, one pair at a time, into the text boxes provided. After each of which the **Enter** button is pressed. This can be tedious and thus error prone, and a **Clear** button is available to restart the process. When all data coordinates have been entered, clicking the **Make** button will produce the TABLE file.

The panel also has some built-in potentials under the 'Special options' menu. One is for silica (SiO₂) [45] and the other for silver iodide [48]. (These have proved useful for student exercises.) The TABLE files produced are named TABSiO2.n and TABAgI.n respectively.

The **Close** button deletes the 'Make Table' panel.

5.8.4 Display

This menu item enables the user to load configuration files with a format different from the standard DL_POLY_2 CONFIG file. The options on its submenu are as follow.

- **CFG** - displays DL_POLY_2 CONFIG files. This option is the same as the **New** button on the GUI button panel.
- **XYZ** - displays a standard XYZ file.
- **SEQ** - displays a protein sequence PDB file.
- **MSI** - displays a CERIOUS 2 msi file.

Each of these options (except **CFG**) will also produce a CONFIG file suitable for input to DL_POLY_2. These appear with names CFGXYZ.n, CFGSEQ.n or CFGMSI.n respectively, with n some integer.

5.8.5 Tools

The GUI provides some utilities that are useful when building DL_POLY_2 input files. The following are available.

1. N_fold

This panel (figure 13) is useful for scaling up systems to multiples of the original, for example taking a single polymer chain and generating a layer.



Figure 13: The N_fold Panel

The panel has three text boxes to enter the integer scaling factors in the x, y and z direction. (For a layer, expand only in x and y.) The button **Make** can be used for a CONFIG file that is loaded already. The **Load** button will load a different CONFIG file from a browser. Both options produce a file called CFGBIG.n, where n is the

suffix of the loaded CONFIG file. A check box enables the display of the expanded structure. The 'Z-bilayer' check box is used to create bilayers, by doubling the system in the z-direction. The text box 'Z gap' specifies the separation between the layers. (Note that a bilayer can also be created using the bilayer options in the 'Make Chain' application above and expanding the system using N_fold, but ignoring the bilayer option here.) The **Close** button deletes the 'N_fold' panel.

2. Bondlengths

The CONFIG file builders for the GUI, such as the polymer and chain options assume certain bond lengths for particular types of bonds. This panel (figure 14) gives the user the opportunity to alter the default values.

A generic bond length is reset by typing in the new value into the appropriate text box and clicking the **Set** button. All subsequent CONFIG files will use the new bond length. Bondlengths are specified in Å.

The **Close** button deletes the 'Set Bondlengths' panel.

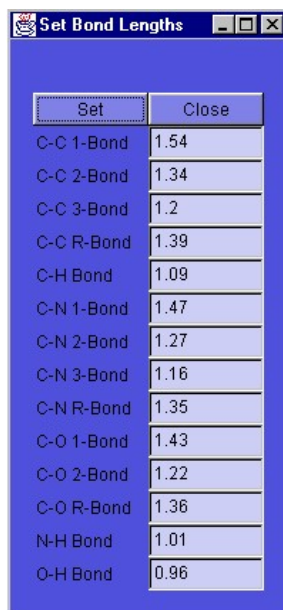


Figure 14: The Set Bondlengths Panel

3. Add water

The 'Add Water' panel (figure 15) inserts water into a CONFIG file to make. The water configuration is taken from a file of water molecules equilibrated at 300 K and inserted into the CONFIG file, with replication and truncation if necessary. Overlap of the water with the solute is prevented by removal of the offending water molecules.

The **Make** button will add water to an already loaded CONFIG file and the **Load** button to a file selected from a browser. The file created has the name CFGH2O.n, where n is the integer suffix of the input file. The user may specify the closest approach of water molecules to the solute and between the water molecules at the periodic boundary (necessary if the water source file has been truncated) using the labelled text boxes.

The 'Add Water' panel also has a check box to enable insertion of water in a 'slab', for example in a bilayer gap. To do this the user must specify a direction vector in the text boxes provided, and the upper and lower bounds of slab perpendicular to this vector. It is assumed the vector is drawn from the centre of the simulation cell. For example a direction vector (0,0,1) and upper and lower bounds of 3.0 and -3.0 respectively will create a water layer in the x, y direction, 6 Å wide, centred on the middle of the simulation cell.

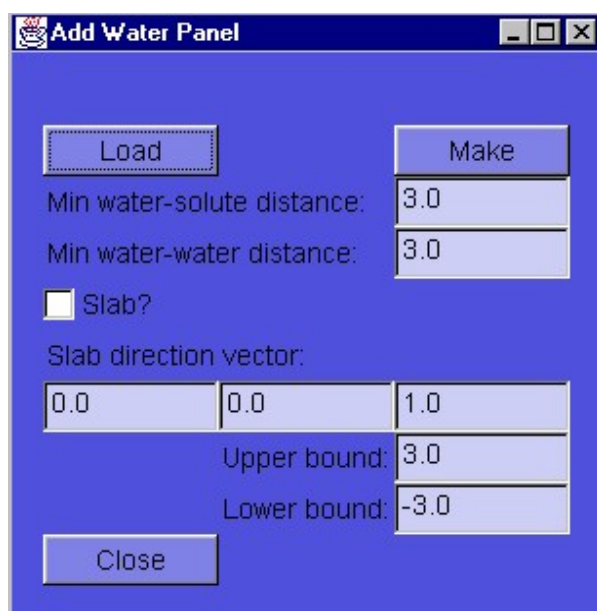


Figure 15: The Add Water Panel

The **Close** button deletes the 'Add Water' panel.

Note that in order to display the full CFGH2O file, the Graphical window button **H2O** must be used to toggle the visibility of the water molecules.

5.9 Execute Menu

The Execute Menu offers facilities for running the DL_POLY_2 executable and for managing the input and output files. The menu has two items:

1. **Run DL_POLY**
2. **Store/Fetch Data**

The operation of each of these is given below.

5.9.1 Run DL_POLY

This menu item opens the ‘Run DL_POLY’ panel (figure 16), which hosts several buttons to control a DL_POLY_2 execution.

The upper middle of the panel is dominated by a group of four buttons: **CONTROL**, **CONFIG**, **FIELD** and **TABLE**. Each of these buttons opens a file browser, for selection of the corresponding input file for a DL_POLY_2 run. It is assumed that all of these files are present in the *execute* subdirectory and are the products of the GUI. Thus all potential CONTROL files will begin with the letters CNTROL, all potential FIELD files with FLD, all potential CONFIG files with CFG and all potential TABLE files (if any) with TAB. On unix systems, the file browser will list files on this basis, but Windows file browsers (to date) are not discriminating. It is the responsibility of the user to select a consistent set of input files. The selected file is copied into one of CONTROL, CONFIG, FIELD or TABLE, which are the files DL_POLY_2 expects to be in the *execute* subdirectory at run time.

When the files have been selected, and provided the *execute* subdirectory properly contains the DLPOLY.X executable, clicking the **Run** button will start the DL_POLY_2 program running in the background. Note that this represents a spawned process, and the program should keep running even if the GUI is closed down. There are advantages to keeping the GUI running however.



Figure 16: The Run DL_POLY Panel

The status of the job can be obtained by clicking the panel **Status** button. This will result either in a display of the job elapsed time, if still running, or a statement that the job has finished. This message will appear in the Monitor window.

The GUI will not allow two jobs to be run at the same time, since there is danger of file corruption. So all jobs must be formally killed, using the **Kill** button, after completion to permit a following run. A job may also be terminated prematurely by clicking the **Kill** button.

Neither the **Status** nor the **Kill** buttons will operate if the GUI has been closed down meanwhile.

Input and output files for DL_POLY_2 left lying around in the *execute* subdirectory after a run may be cleared away using the **Clear** button. Files required for a subsequent run may be set up using the **Update** button, which will result in backup copies of the CONFIG and REVOLD files being taken (and given the suffix .BAK), and the files REVCON and REVIVE being renamed CONFIG and REVOLD respectively. Note that it is up to the user to amend the CONTROL file if necessary, using the CONTROL file editor under the **FileMaker** menu.

Note that because both **Clear** and **Update** may result in data loss if inappropriately used, so a dialog box will appear when either is clicked, to provide opportunity to abort the operation.

The **Close** button deletes the 'Run DL_POLY' panel.

5.9.2 Store/Fetch Data

Selection of this menu item invokes the 'Data Archive' panel (figure 17), which provides facilities for storing and retrieving DL_POLY_2 I/O files.

The first option available on the 'Data Archive' panel is the selection of the standard test cases of DL_POLY_2. The user must first choose the required test case from the menu box on the panel, then clicking the **Select** button will result in the data files for this test case being copied into the *execute* subdirectory. These files may be used immediately for a DL_POLY_2 run, since they already have the correct file names and do not need to be selected again.

The user may define a storage directory under the DL_POLY_2 *data* subdirectory and use the Data Archive panel to store DL_POLY_2 I/O files. The user should enter a *new* directory name in the text box beside the **Store** button and then click **Store**. (This action also deletes the original files from the *execute* subdirectory.) If the nominated directory already exists, an error message appears in the Monitor window and no action is taken.



Figure 17: The Data Archive Panel

In a similar fashion the **Fetch** button will copy DL_POLY_2 I/O files from the directory nominated in the adjacent text box into the *execute* subdirectory. The nominated directory must be in the DL_POLY_2 *data* subdirectory.

Note that both the **Select** and **Fetch** buttons will first produce a dialog box, to alert the user to a possible data overwrite.

The **Close** button deletes the 'Data Archive' panel.

5.10 Analysis Menu

The Analysis menu provides access to a range of tools for analysing DL_POLY_2 output. The menu items that appear are as follows:

1. **Structure** - spatial correlation functions;
2. **Dynamics** - time correlation functions;
3. **van Hove** - space-time correlation and dynamic structure;
4. **Display** - visualisation;
5. **Tools** - utilities.

Each of these items presents a submenu of applications, which are described below.

5.10.1 Structure

This menu item invokes a submenu of facilities for analysing the static structure of a system. The following facilities are provided.

1. **RDF_Plot**

The 'RDF Plotter' panel (figure 18) enables the user to plot a radial distribution function produced by DL_POLY_2 .



Figure 18: The RDF Plot Panel

The RDF data are stored in the file RDFDAT, but if the user has renamed this file, the name in the text box available for this purpose must be changed. To produce a RDF plot all the user need do is nominate the two required atom names in the text boxes provided and click the **Plot** button. (The names are those used to label atoms in the simulation CONFIG or FIELD files.) The GUI will produce a screen plot of

the RDF and also create an associated plot file named RDFn.XY, where n is some integer. The on-screen plot is produced by the GUI Graph Plotter (see section 5.12).

Note the names of the atoms present in any CONFIG file may easily be obtained with the 'What Atoms?' facility under the Analysis/Tools menu (see section 5.10.5).

The **Close** button deletes the 'RDF Plot' panel.

2. RDF_Calc

The data stored in the RDFDAT file does not necessarily provide a complete account of pair correlations in a system. For example bonded pairs are not described, nor are pairs for which the interaction potential is defined as zero. The 'RDF_Calc' panel (Figure 19) provides the means to calculate these missing correlations using a DLPOLY_2 HISTORY file. The panel can also calculate a total RDF for the system, combining all atom types.



Figure 19: The RDF Calculator Panel

The data required by this panel are as follows. Firstly the name of the HISTORY file is required. (This must be a file in the *execute* subdirectory.) The default file name is HISTORY. If the file is formatted, the check box on the panel must be set. Next the user must supply the atom names for the pair correlation, as for the RDF plotter. Note that the name ALL may be used if a total RDF is required. The user must specify the required number of configurations in the HISTORY file using

the associated text box. (This may exceed the actual number without harm.) Also required are the length of the RDF array (how many data points in the plot), the HISTORY file sampling interval (e.g. setting 1 will sample all configurations, 2 will sample every other configuration, and so on) and the cut off radius (in Å) for the RDF. Text boxes are available for all of these.

Clicking the **Run** button will start the RDF program running *in the background*. It may be monitored or terminated using the **Status** and **Kill** buttons respectively, in a similar manner to a DL_POLY_2 run.

When finished, the program produces a file named RDFDAT.n, where n is an integer. This file may be plotted using the RDF plotter described above (section 1).

The **Close** button deletes the 'RDF Calculator' panel.

3. **S(k)**

The S(k) plotter panel (figure 20) is used to plot a structure factor, based on the RDF data in the RDFDAT file.



Figure 20: The S(k) Plotter Panel

This panel works in exactly the same way as the RDF plotter above (section 1). The only difference is that the RDF data are Fourier transformed immediately to give the structure factor. An on-screen plot of this appears, drawn by the GUI Graph Plotter (section 5.12, and a plot file SOKn.XY, with n some integer, is produced.

4. **Z_Density**

The 'Z-Density Plotter' panel (figure 21) plots the particle density of a system along the z-direction, taking data from a DL_POLY_2 ZDNDAT file. This has particular application to layered systems, where, by convention, the layers lie in the x, y plane.

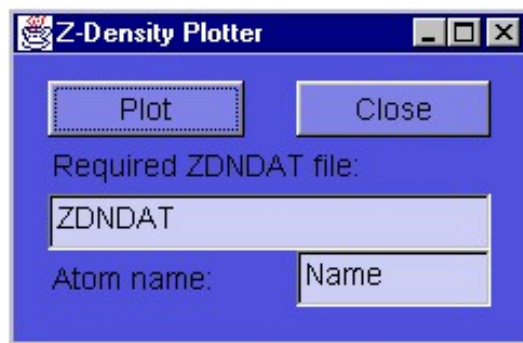


Figure 21: The Z-Density Plotter Panel

To operate this panel the user must nominate the ZDNDAT file in the appropriate text box and the name of the atom of interest. The **Plot** button produces an on-screen plot using the GUI Graph Plotter facility (section 5.12) and also writes the plot file $Zden\alpha.XY$, where α is an integer.

Note the names of the atoms present in any CONFIG file may easily be obtained with the ‘What Atoms’ facility under the Analysis/Tools menu (see section 5.10.5).

The **Close** button deletes the panel.

5. Slice

The ‘Slice’ panel (figure 22) enables a user to cut a slice, or slab of atoms from REVCON file or a loaded configuration. This can be useful to isolate areas of interest for closer inspection.

To use the slice option the user must define a slice direction vector, (which is perpendicular to the slice faces,) and the upper and lower bounds of the slice measured along the chosen direction. It is assumed the direction vector starts at the centre of the REVCON file. Labelled text boxed are provided for these inputs. To perform the slice operation on a loaded configuration the **Make** button must be clicked. Alternatively the **Load** button may be used, to enable selection of a REVCON file from a browser. Both buttons produce a file CFGSLC.n, with n an integer (absent in some cases) taken from the configuration file suffix. The sliced configuration appears in the Graphics window.

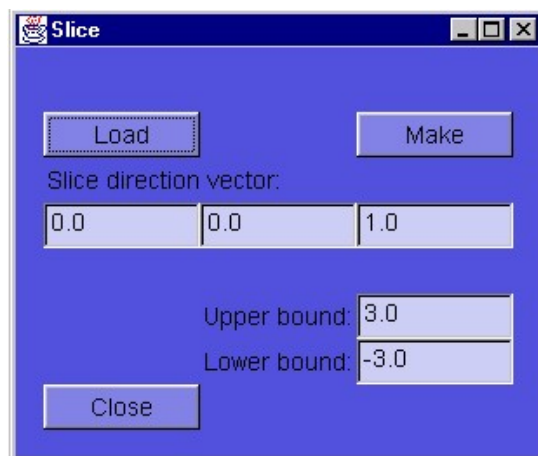


Figure 22: The Slice Panel

Note: Being an analysis tool, this facility is targetted at REVCON files, though it will work equally well on CONFIG files also.

The **Close** button deletes the 'Slice' panel.

5.10.2 Dynamics

The Dynamics submenu offers some standard time correlation functions, namely the mean square displacement (MSD), velocity autocorrelation (VAF) and force autocorrelation (FAF). These are calculated from the data in a DL_POLY_2 HISTORY file.

1. MSD

The MSD Panel (figure 23) enables a multiple origin MSD calculation to be performed and the result plotted on-screen.



Figure 23: The MSD Panel

The user must specify the HISTORY file name in the labelled text box. The file must be in the `DL_POLY_2 execute` subdirectory. The default name is HISTORY. A check box is used to indicate that the file is formatted or otherwise. A text box is available for the name of the atom of interest, which may be specified as ALL if a non-discriminating MSD is required. Text boxes are also provided for the required number of configurations in the HISTORY file, the MSD array length, sampling interval and origin interval. The sampling interval defines the interval between selected configurations in the HISTORY file, for example an assignment of 1 means every configuration is used, while 2 would use every second configuration and 3 every third and so on. The origin interval specifies which of the sampled configurations is to be used as an origin for a MSD array. Thus a specification 1 means every sampled configuration is used as an origin, 2 means every second sampled configuration is used etc. By a quirk of bookkeeping the MSD array length must be divisible by the origin interval. The GUI will enforce this if necessary.

Clicking the **Run** button starts the MSD calculation. On completion the MSD is plotted by the GUI Graph Plotter (see section 5.12) and a plot file MSDn.XY created, with n an integer.

The **Close** button deletes the MSD panel.

2. VAF

The VAF panel is identical to the MSD panel in appearance and operation. Its outputs are a plot of the Velocity Autocorrelation Function data and a plot file: VAFn.XY, with n an integer. Please consult the above section describing the MSD

panel. Note that the HISTORY file must contain velocity data, and DL_POLY_2 should be directed to produce this when the simulation is performed.

3. FAF

The FAF panel is identical to the MSD panel in appearance and operation. Its outputs are a plot of the Force Autocorrelation Function data and a plot file: FAFn.XY, with n an integer. Please consult the above section describing the MSD panel. The HISTORY file must, of course, contain force data.

5.10.3 van Hove

This menu item provides a selection of density correlation tools of the kind pioneered by van Hove. (i.e. correlations in both space and time). The facilities available are:

1. $G_s(r,t)$

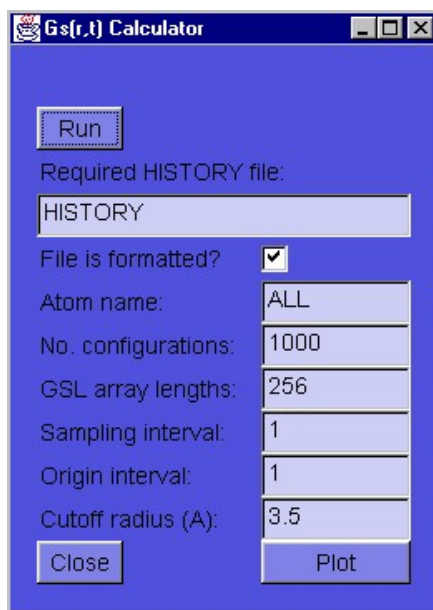


Figure 24: The $G_s(r,t)$ Calculator Panel

The panel for the van Hove self correlation function (see figure 24) resembles that for the MSD above, but requires an additional control parameter: the cutoff radius, which is the distance over which the spatial correlations are to be evaluated. The calculations are commenced when the RUN button on the file browser is clicked and may take several minutes to complete. The results are stored in a file: HOVGSL.n,

where n is an integer. Unlike the MSD, VAF and FAF panels this panel does not produce a graph window automatically, since, potentially, a large number of $G_s(r,t)$ functions are produced. (The actual number is announced in the Monitor Window when the calculation finishes.) It is necessary to use the **Plot** button to invoke the van Hove Plotter and select which of the many functions is required for plotting by entering the sequence number in the text box (see figure 25). The selected plot appears on-screen in the Graph Plotter and each plot produces a plot file: HOVm.XY, where m is the sequence number of the van Hove function in the parent HOVGSL. n file.

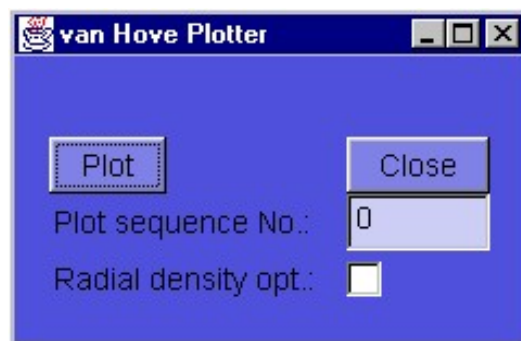


Figure 25: The van Hove Plotter Panel

2. $G_d(r,t)$

The van Hove distinct correlation function panel (figure 26) resembles that for the self correlation function, except that an additional atom name is required, since this is a pair correlation function. The name ALL may be used for indiscriminate correlation functions. The calculations are performed *in a background job*, which may be monitored using the **Status** button or terminated with the **Kill** button on the panel. When the job is finished, plotting the distinct correlation functions, (which are stored in the file HOVGDF. n .) is identical to the self correlation function case.

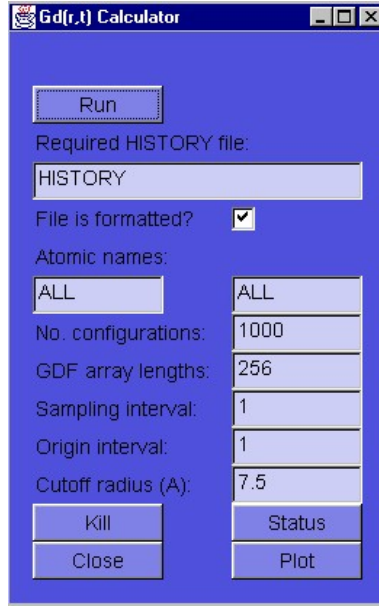


Figure 26: The Gd(r,t) Calculator Panel

3. $\mathbf{S}(\mathbf{k}, \mathbf{w})$

The dynamic structure factor panel (figure 27) also operates in a manner resembling the self correlation function panel, however there are important differences. Firstly, the calculation (*which runs in the background*) will accept only *formatted* HISTORY files. It does not require the names of the atoms, but does require the user to specify whether or not the atomic charges are to be used (to calculate charge density as opposed to particle density). A check box is available for this purpose. The maximum $\mathbf{k}$ vector is specified by an integer index, which determines the maximum in all three principal directions (and, incidentally, has a huge impact in the job time!) as in:

$$\underline{k} = 2\pi/L(\ell, m, n)^{\dagger},$$

where ℓ, m, n are the integers concerned and L is the cell width. The calculation proceeds via the particle density $\rho(k, t)$, through the intermediate scattering function $F(k, t)$ to the dynamic structure factor $S(k, \omega)$. These stages respectively produce files called SPCDEN, DENFKT and DENSKW. The latter two may be displayed by clicking the **Plot** button, which invokes the dynamic structure factor plotting panel. The panel supports **Status** and **Kill** buttons to help manage the background job.

Figure 27: The $S(k,w)$ Calculator Panel

The plotting panel (figure 28) resembles the van Hove plotter, but the choice of function to be plotted is specified by the three k-vector indices, for which individual text boxes are provided.

Figure 28: The $S(k,w)$ Plotter Panel

5.10.4 Display

The Display menu item offers a facility for displaying REVCON files and (re-)plotting various graphs.

1. **REVCON**

This option displays the contents of the REVCON file produced at the end of a

DL_POLY_2 simulation. Its function is entirely analogous to the Display CFG option in the **FileMaker** menu above, and the **New** button on the Graphics Window. The action invokes a file browser for selection of the REVCON file, and can also be used to display CONFIG files.

2. Plot

This option invokes a simple file browser, which allows the user to select a plot file (abc.XY etc.) to display with the Graph Plotter.

5.10.5 Tools

1. What Atoms?

The ‘What Atoms?’ menu item provides a mechanism by which the user may conveniently determine the different types of atom that occur in a CONFIG file. It invokes a file browser with which the required CONFIG file may be selected. A list of atom types within the selected file then appears in the Monitor window. This facility is useful for analysis purposes, as the atom types (names) are frequently required by the GUI analysis tools.

2. Statistics

The ‘Statistics’ Panel invoked by this option (figure 29) allows the user to calculate average values and statistical errors of particular system variables, using information stored in the DL_POLY_2 STATIS file.

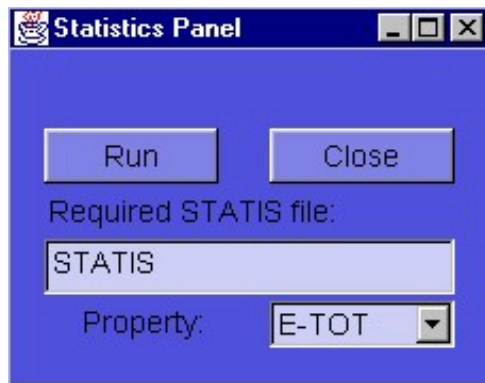


Figure 29: The Statistics Panel

The panel supports a text box for the name of the STATIS file and a menu box for the variables that may be analysed. These are:

- (a) E_TOT - system total energy;

- (b) TEMP - system temperature;
- (c) E_CFG - configuration energy;
- (d) E_VDW - van der Waals energy;
- (e) E_COUL - Coulombic energy;
- (f) VOLUME - system volume;
- (g) PRESS - system pressure;
- (h) PMF - potential of mean force virial;

The statistical calculations are initiated by the **Run** button. While calculating the average value of the selected variable, the GUI also performs a blocking analysis to obtain the optimal statistical error and the error uncertainty. The results are displayed in the Monitor window. A graph plot of the variable is produced on-screen by the GUI Graph Plotter.

5.11 Information Menu

The Information menu provides some on-line information of a semi-useful nature. The information is displayed in the Monitor window.

1. **About DL_POLY**

Note on authorship and ownership of DL_POLY_2 .

2. **Disclaimer**

Software writer's incantation to ward off litigation.

3. **Licence**

View of the DL_POLY_2 licence on-screen.

4. **Acknowledgements**

Acknowledgements and thanks to various people and organisations.

5. **MINIDREI**

The contents of the MINIDREI Dreiding data file.

6. **CERAMICS**

The contents of the CERAMICS ceramic data file.

7. **Clear Text**

Clear the Monitor window.

5.12 The GUI Graph Plotter Window

The Graph Plotter window (figure 30) is invoked automatically by some of the Analysis tools described above and also by selecting the Analysis/Display/Plot menu item. The

scale of the graph is calculated automatically and the window provides facilities to display or print a graph plot and also perform some editing.

The Graph Plotter presents a large drawing area, with three text boxes at its base and a number of buttons stacked vertically on the right hand side. Most of the functionality of the plotter is controlled by the buttons, which are as follows:

- **Load** - load and plot a new XY file;
- **Spline** - Fit data points with spline functions and plot result;
- **Dots** - Show/hide dots marking data points;
- **Lines** - Set plot line thickness.
- **Print** - Open a print dialog box for printing;
- **Zoom** - Zoom in on selected region of plot (marked with drag box);
- **Clear** - Delete plot and clear arrays;
- **Close** - Delete Graph Plotter window.

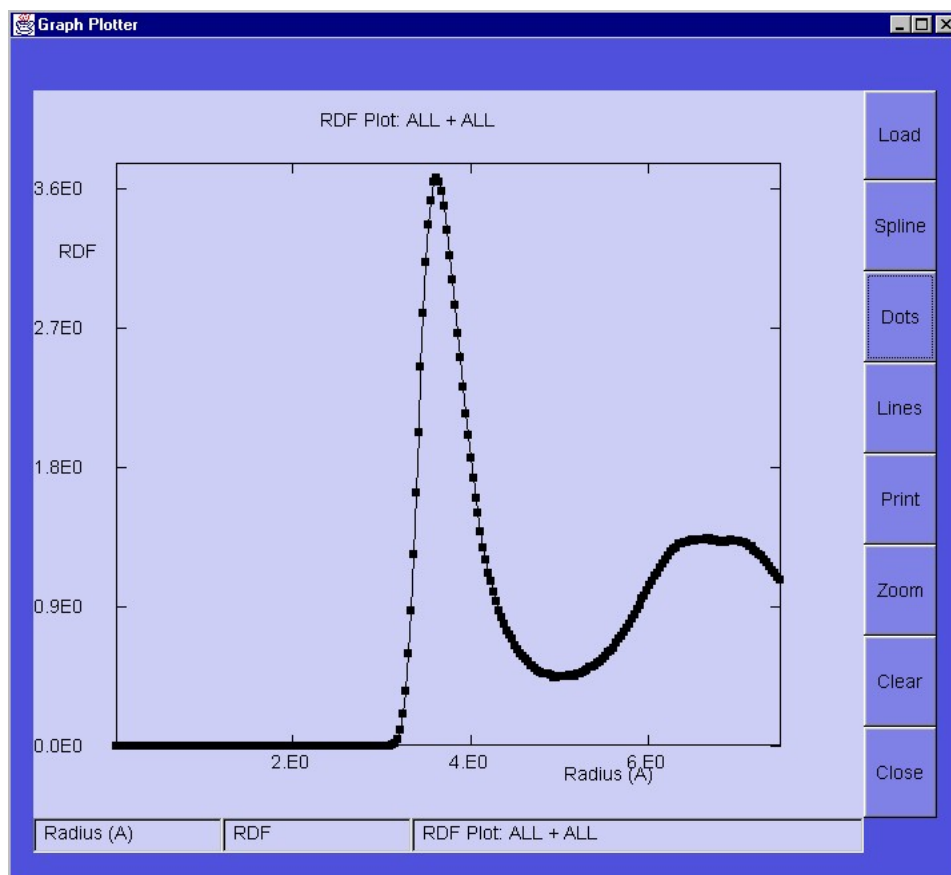


Figure 30: The GUI Graph Plotter

In addition to the button operations, the text boxes at the foot of the drawing area allow the user to change the graph annotation. From the left, the first box defines the x axis notation, the second box the y-axis notation and the last defines the graph title. The text in any of these may be edited. Hitting < *Return* > will insert the changed text into the plot.

The GUI Graph Plotter reads and writes data files with the following format:

1. **Record 1: File header record**

Record starts with the '#' character followed by user text;

2. **Record 2: Plot title record**

Record starts with the '#' character followed by text defining the plot title;

3. **Record 3: X-axis label**

Record starts with the '#' character followed by text defining x-axis label;

4. **Record 4: Y-axis label**

Record starts with the '#' character followed by text defining y-axis label;

5. **Records 5+: Data points**

Records defining the x and y data points of the plot. x and y must be real numbers separated by at least one space. Scientific (E) number format is acceptable.

6. **Last record: terminator**

The data must be terminated with a '&' character.

This format is equivalent to the common XY format used by most data processing packages.

Chapter 6

DL_POLY_2 Examples

Scope of Chapter

This chapter describes the standard test cases for DL_POLY_2 , the input and output files for which are in the *data* sub-directory.

6.1 DL_POLY Examples

6.1.1 Test Cases

The following example data sets (both input and output) are stored in the subdirectory *data*. These are provided so that you may check that your version of DL_POLY is working correctly. All the jobs are short and should require no more than a few minutes execution time, even on a single processor computer. The output files are stored in compressed format. The test cases can be run by typing

```
select n
```

from the *execute* directory, where *n* is the number of the test case. The *select* macro will copy the appropriate CONTROL, CONFIG, and FIELD files to the *execute* directory ready for execution. The output files OUTPUT, REVCON and STATIS may be compared with the files supplied in the *data* directory.

The example output files provided in the *data* directory were obtained on 4 processors of an IBM SP/2 parallel system (120 MHz P2SC ‘thin’ nodes). The program was compiled with the three point interpolation option.

It should be noted that the potentials and the simulation conditions used in the following test cases are chosen to demonstrate functionality only. **They are not necessarily appropriate for serious simulation of the test systems.** Note also that the DL_POLY_2 Graphical User Interface [20] provides a convenient means for running and viewing these test cases.

6.1.1.1 Test Case 1: KNaSi_2O_5

Potassium Sodium disilicate glass (NaKS_2O_5) using two and three body potentials. Some of the two body potentials are read from the TABLE file. Electrostatics are handled by a multiple timestep Ewald sum method. Cubic periodic boundaries are in use.

6.1.1.2 Test Case 2: Metal simulation with Sutton Chen potentials

FCC Aluminium using Sutton-Chen potentials. Temperature is controlled by the method of Gaussian constraints.

6.1.1.3 Test Case 3: An antibiotic in water

Valinomycin in 1223 spc water molecules. The temperature is controlled by a Nosé-Hoover thermostat while electrostatics are handled by a shifted Coulombic potential. The water is defined as a rigid body while bond constraints are applied to all chemical bonds in the valinomycin. Truncated octahedral boundary conditions are used.

6.1.1.4 Test Case 4: Shell model of water

256 molecules of water with a polarizable oxygen atom. Temperature is controlled by the Berendsen thermostat while electrostatics are handled by the reaction field method with

a “charge group” cutoff scheme. “Slab” period boundary conditions are used. The water molecule (apart from the shell) is treated as a rigid body.

6.1.1.5 Test Case 5: Shell model of MgCl_2 at constant pressure

Dynamical Shell model simulation of MgCl_2 . Temperature and pressure are controlled by a Berendsen thermostat and barostat. An Ewald sum is used with cubic periodic boundary conditions.

6.1.1.6 Test Case 6: PMF calculation

Potential of mean force calculation of a potassium ion in SPC water. Electrostatics are handled by the Ewald sum. The water is treated as a constrained triangle.

6.1.1.7 Test Case 7: Linked rigid bodies

8 biphenyl molecules in cubic boundary conditions. The NVE ensemble is used. Each phenyl ring is treated as a rigid body, with a constraint bond to the other ring of the molecule. In the centre of each ring are three massless charge sites which imparts a quadrupole moment to the ring.

6.1.1.8 Test Case 8: An osmosis experiment with a semi permeable membrane

The membrane is a collection of tethered sites interconnected by harmonic springs. There are no electrostatic forces in the system. The simulation is run with the Hoover anisotropic constant pressure algorithm.

6.1.1.9 Test Case 9: A surfactant at the air-water interface

The system is comprised of 32 surfactant molecules (trimethylaminododecane bromide or TAB-C12) arranged either side of a slab of 342 water molecules approximately 30 Å thick. The surfactant chains are treated with rigid bonds and the water molecules are treated as rigid bodies. The TAB headgroup has fractional charges summing to +1 (the bromide ion has charge -1). The Ewald sum handles the electrostatic calculations. The short range forces are taken from the Dreiding force field.

6.1.1.10 Test Case 10: DNA strand in water

This system consists of a strand of DNA 1260 atoms in length in a solution of 706 (SPC) water molecules. The DNA is aligned in the Z-direction and *hexagonal prism* periodic boundary conditions applied. The electrostatic interactions are calculated using the Smooth Particle Mesh Ewald method. Note that the system has a strong overall negative charge which is strongly anisotropic in distribution. The short range forces are taken from the Dreiding force field, and constraints are used for all covalent bonds. For simplicity H-bonds are treated as harmonic bonds with an equilibrium bondlength of 1.724 Å.

6.1.1.11 Test Case 11: Hautman-Klein test case

The system consists of 100 short chain surfactant molecules in a layer simulated under NVE conditions. The total system size is 2300 atoms and the XY periodicity is a square. The Dreiding force field describes the molecular interactions. All bonds are harmonic and all atoms are explicit. The link-cell algorithm is in operation.

6.1.2 Benchmark Cases

These represent rather larger test cases for DL_POLY_2 that are also suitable for benchmarking the code on large scale computers. They have been selected to show fairly the capabilities and limitations of the code.

6.1.2.1 Benchmark 1

Simulation of metallic aluminium at 300K using a Sutton-Chen density dependent potential. The system is comprised of 19652 identical atoms. The simulation runs on 16 to 512 processors only.

6.1.2.2 Benchmark 2

Simulation of a 15-peptide in 1247 water molecules. This was designed as an AMBER comparison. The system consists of 3993 atoms in all and runs on 8-512 processors. It uses neutral group electrostatics and rigid bond constraints and is one of the smallest benchmarks in the set.

6.1.2.3 Benchmark 3

Simulation of the enzyme transferrin in 8102 water molecules. The simulation makes use of neutral group electrostatics and rigid bond constraints. The system is 27539 atoms and runs on 8-512 processors.

6.1.2.4 Benchmark 4

Simulation of a sodium chloride melt with Ewald sum electrostatics and a multiple timestep algorithm to enhance performance. The system is comprised of 27000 atoms and runs on 8-512 processors.

6.1.2.5 Benchmark 5

Simulation of a sodium-potassium disilicate glass. Uses Ewald sum electrostatics, a multiple timestep algorithm and a three-body valence angle potentials to support the silicate structure. It also using tabulated two-body potentials stored in the file TABLE. The system is comprised of 8640 atoms and runs on 16-512 processors.

6.1.2.6 Benchmark 6

Simulation of a potassium-valinomycin complex in 1223 water molecules using an adapted AMBER forcefield and truncated octahedral periodic boundary conditions. The system size is 3838 atoms and runs on 16-512 processors.

6.1.2.7 Benchmark 7

Simulation of gramicidin A molecule in 4012 water molecules using neutral group electrostatics. The system is comprised of 12390 atoms and runs on 8-512 processors. This example was provided by Lewis Whitehead at the University of Southampton.

6.1.2.8 Benchmark 8

Simulation of an isolated magnesium oxide microcrystal comprised of 5416 atoms originally in the shape of a truncated octahedron. Uses full coulombic potential. Runs on 16-512 processors.

6.1.2.9 Benchmark 9

Simulation of a model membrane with 196 41-unit membrane chains, 8 valinomycin molecules and 3144 water molecules using an adapted AMBER potential, multiple timestep algorithm and Ewald sum electrostatics. The system is comprised of 18866 atoms and runs on 8-512 processors.

Chapter 7

DL_POLY_2 Utilities

Scope of Chapter

This chapter describes the utility programs and subroutines of DL_POLY_2 , found in the sub-directories *utility* and *utility/macromol*.

Three kinds of utility are distinguished: Initialisation, Analysis and Miscellaneous.

7.1 Initialisation Utilities

7.1.1 BUCKY

7.1.1.1 Header records

```

      program bucky
c
c*****
c
c      construction of buckminster fullerene molecule
c
c      copyright daresbury laboratory  november 1994
c      author w. smith      november 1994
c
c*****

```

7.1.1.2 Function

BUCKY creates a data file with the XYZ coordinates of the 60 atomic positions of a buckminster fullerene molecule. Input and output are in Å.

7.1.1.3 Dependencies

- NONE

7.1.1.4 Parameters

- NONE

7.1.1.5 Input

Data Files:

- standard input

The input required is as follows:

record 1:

bnd	real	length of C-C bondlength in Å.

7.1.1.6 Output

Data Files:

- filename: FULLERENE

This is a formatted file. The output consists of the following:

1. three columns of data (format: 1p,3e16.8) giving the X, Y and Z coordinates of each atom in turn. 60 records in all. Units are Å.

7.1.1.7 Comments

BUCKY generates the coordinates for one molecule only. Adaptation for many molecules in the same MD cell is trivial.

7.1.2 CSD2EDT

7.1.2.1 Header records

program csd2edt

```

C*****
C
C    DL_POLY utility to convert fractional coordinates into real space
C    coordinates in form suitable for EDTOUT (see ffggen.f). The
C    FF.dat file is also created.
C    Input data assumed to be in form compatible with Cambridge
C    Structural Database
C
C    input a,b,c (cell lengths)
C    input alp,bet,gam (cell angles)
C    input xi,yi,zi - fractional coordinates
C
C    copyright darsebury laboratory 1993
C    author - t. forester      june 1993
C
C*****

```

7.1.2.2 Function

The program converts fractional coordinates to real space coordinates. The input data is assumed to be compatible with files from the Cambridge structural database. The output file, "EDTOUT", is designed as input for the utility FFGEN. Atomic charges are not assigned.

7.1.2.3 Dependencies

- none

7.1.2.4 Parameters

integers:

mxn	10000	maximum number of atoms in the structure

7.1.2.5 Input

Interactive:

record 1:

title	a40	title line

a0	real	<i>a</i> cell vector length (f8.0)
b0	real	<i>b</i> cell vector length (f8.0)
c0	real	<i>c</i> cell vector length (f8.0)

record 2:

dummy	22x	dummy
alp	real	crystal angle α
bet	real	crystal angle β
gam	real	crystal angle γ

record 3:

natms	integer	number of atoms in unit cell

The next natms records are of the form

next record:

dummy	5x	
name	a5	atom name
xb	real	x fractional coordinate (f10.5)
yb	real	y fractional coordinate (f10.5)
zb	real	z fractional coordinate (f10.5)
dummy	1x	
icn(1)	integer	index of first atom bonded to this site (i4)
icn(2)	integer	index of second atom bonded to this site (i4.)
icn(3)	integer	index of third atom bonded to this site (i4)
icn(4)	integer	index of fourth atom bonded to this site (i4)

7.1.2.6 Comments

The CSDB files do not have residue information in them, simply the atomic coordinates. This means the assignment of atom types in a molecule is non-trivial, and the logic applied in the code (lines 123 - 164) is unique for the AMBER force-field applied to the Valinomycin molecule. You will need to edit this portion to suit your own needs.

7.1.3 EIGHTFOLD

7.1.3.1 Header records

```

        program fold8
c
c*****
c
c      dl_poly utility to expand a simulation cell by a factor of
c      eight in size using a 2x2x2 multiplication
c
c      copyright daresbury laboratory 1994
c
c      author w. smith september 1994
c
c      Note: under no circumstances should the velocity arrays be
c            treated using this program
c
c*****

```

7.1.3.2 Function

EIGHTFOLD reads a DL_POLY_2 CONFIG file and expands it to eight times the original size (i.e. eight times the number of atoms, eight times the volume) using a 2x2x2 expansion in the principal directions.

7.1.3.3 Dependencies

- NONE

7.1.3.4 Parameters

integers:

mega	1000000	Maximum number of records in CONFIG file

7.1.3.5 Input

Data Files:

- standard input

The input records are as follows

record 1:

fname char*40 working name of CONFIG file

- **filename: CONFIG**

This is the configuration file for a normal DL_POLY_2 run. Note that the working name of the trajectory file need not be CONFIG. The actual name is specified in standard input.

7.1.3.6 Output

Data Files:

- **filename: CONFIG_EIGHT**

This is a formatted file, written in standard CONFIG file format, containing eight times as many atoms as the original CONFIG file.

7.1.3.7 Comments

Note that EIGHTFOLD ignores any velocity or force information in the CONFIG file. It is generally not advisable to assign the original velocities to the expanded file, though this is easily done. The catch is that the original periodicity of the smaller system will then be *completely preserved*, rendering the eightfold expansion statistically worthless.

Beware of using EIGHTFOLD to expand systems that contain molecules (as opposed to unconnected atoms or ions). Bonds extending half the cell width are the inevitable result!

EIGHTFOLD can be adapted to do other expansions of the system relatively easily.

7.1.4 FFGEN

7.1.4.1 Header records

program ffggen

```

C*****
C
C    program to extract force field for a macromolecule
C    and interface it to DL_POLY.
C    input file: EDTOUT (AMBER 3, 4 edit.out style)
C                : FF.dat  (title plus additional info)
C    output    : CONFIG (coordinates etc)
C                : FIELD  (force-field)
C
C    copyright -  daresbury laboratory 1993
C
C    author    -  t forester    April 1993
C    modified  -  t.forester    feb 1996
C
C*****

```

7.1.4.2 Function

Constructs the DL_POLY CONFIG file and FIELD file from the “AMBER” like edtout file. The actual files created are “CONFIG” and “FIELD”. The program requires input files “EDTOUT” and “FF.dat”.

Interactive input is also required. The user will be prompted for the force-field file. DL_POLY_2 comes supplied with three such files: “amb89.ff”, “amb94.ff” and “gro37C.ff”. They correspond to the Amber forcefields as of 1989 and 1994 and the GROMOS 37C forcefield.

7.1.4.3 Dependencies

- none

7.1.4.4 Parameters

integers:

ntyp	100	number of forcefield atom types
mga	500	number of forcefield angle potentials
mgb	500	number of forcefield bond potentials
mgd	500	number of forcefield torsion potentials
mgj	200	number of forcefield improper torsion potentials
mhb	80	number of forcefield hydrogen bond potentials
mlj	100	number of forcefield Lennard-Jones types

mxang	50000	maximum number of valence angles in system
mxatm	30000	number of atoms in FIELD specifications
mxbnd	30000	maximum number of bonds in system

7.1.4.5 Input

Data Files:

filename: EDTOUT

This file should be in the same format as the “edit.out” file produced by AMBER ver 3.x and 4.x. Excerpts from the example file “EDTOUT.example” supplied in the “utility/macromol” directory are given below:

Header line: Example EDTOUT file :

```

MOLECULE

RESIDUE  1 =  VAL

BOND ARRAY BEGINS WITH      1

1  -99  HN    H           64.4190  46.8190  24.7180  0.312000
2  -99   N    N           63.8740  46.5770  25.5330 -0.263000
3   2   HN    H           63.8900  47.3790  26.1480  0.312000
4   2   HN    H           62.9190  46.4580  25.2270  0.312000
5   2   CA    CH           64.3940  45.3590  26.2030  0.256000
6   5   CB    CH           65.8590  45.5760  26.6120  0.033000
7   6   CG1   C3           66.4480  44.4950  27.5490  0.006000
8   6   CG2   C3           66.7810  45.7110  25.4410  0.006000
9   5    C    C           63.5200  45.0210  27.4620  0.526000
10  9    O    O           63.4750  45.8700  28.3270 -0.500000

RESIDUE  2 =  PRO

BOND ARRAY BEGINS WITH      11

11 -99   N    N           62.8790  43.8270  27.6380 -0.257000
12 11   CD    C2           62.4980  42.8170  26.6360  0.084000
13 12   CG    C2           61.3280  41.9640  27.1470  0.036000
14 13   CB    C2           61.6170  42.0510  28.5940 -0.001000
15 11   CA    CH           62.2170  43.4140  28.8420  0.112000
16 15    C    C           63.0990  43.3140  30.0780  0.526000
17 16    O    O           62.6210  43.7130  31.1660 -0.500000

```

the file consists of blocks of the following form:

header lines

MOLECULE
 RESIDUE = ...
 BOND ARRAY
blank line
bonding data - one atom per line

The “MOLECULE” line indicates the start of a new molecular species. The word MOLECULE must be in upper case and begin in column 7. There must be at least one such record in the file.

The RESIDUE line indicates the start of a new residue (Amino acid or nucleotide) within the molecule. The actual residue name and numbers are not required by FFGEN, and no checks are made on the consistency of the atomic data with the residue name.

The “BOND ARRAY” line signals that bonding data follows the “RESIDUE” line. The word BOND must be in uppercase and begin at column 6 in the file. It must be followed by a blank line. Thereafter, until the next “MOLECULE” or “RESIDUE” record, the input is as follows:

next record:

ia	i5	atomic index
ib	i5	neighbour index
dummy	3x	
at	a6	atom name
an	a6	force field atom type
dummy	4x	
x	f10.4	x coordinate
y	f10.4	y coordinate
z	f10.4	z coordinate
qq	f8.4	partial charge

7.1.4.6 Comments

The atom numbering must start from 1 and increase sequentially throughout the file. The neighbour index need only be given if it is less than the atomic index. If there are no such neighbours then enter “-99” for the neighbour index. If the atom is bonded to more than one neighbour with indices less than the atomic index then all these may be given, (one per line). For example if atom 168 is bonded to atoms 18, 19 and 162, this may be entered as:

168	18	C	C	2.4980	6.0633	10.1126	0.616
168	19	C	C	2.4980	6.0633	10.1126	0.616
168	162	C	C	2.4980	6.0633	10.1126	0.616

However in general it is NOT necessary to do this. The program will allocate the expected number of bonds to an atom from the atom type (this is done in the subroutine

NCONNECT for the AMBER and GROMOS forcefields) and then seeks to build bonds to other unsaturated sites that are within a reasonable bonding distance. The distance criteria are set in the subroutine CONCHECK by the variables `r11` and `r12`, which give the upper and lower bounds for a bond distance. Only when an unusually long bond is made is information echoed to the screen (as, for example, when a disulphide connection is made). If more bonds than expected are found an error message is written to the screen. This means that it is not strictly necessary to indicate all (or indeed any) of the bonded neighbours for a site. However, the locator algorithm is simplistic and the more information you include the better. It should also be said that the AMBER `edtout` file includes at most one neighbour bond for each site in the file. FFGEN was successful in locating all ‘missing’ interactions in all test cases.

FFGEN generates the list of angle and dihedral interactions from the completed bonding data. With the GROMOS forcefield 1.4 Lennard Jones are also defined.

7.1.4.7 FF.dat

FF.dat is a formatted file: integers and reals are in free format. Co-ordinates and partial charges given in the FF.dat file are used preferentially over those supplied in the EDTOUT file. If coordinates are not supplied in the FF.dat file then data is taken from EDTOUT. In addition the FF.dat file contains instructions on whether or not to use constraint bonds, chemical bonds or a combination of both in the force field, and on the energy units to be used. The example file “FF.dat.example” supplied in the “utility/macromol” directory of DL_POLY is shown below:

```
title line
CONSTRAIN hydrogen
UNITS kcal
```

The FF.dat file is of the form:

record 1		
header	a80	title line
record 2	optional	
constraint	a80	constraint information
record 3	optional	
units	a80	units of energy

record 2 is of the form

```
CONSTRAIN string
```

where `string` is one of “all”, “hydrogen” or “none”. This will result in : constraints apply to all bonds ; constraints apply only to bonds involving hydrogen atoms, other bonds are extensible; all bonds are extensible. `string` is not case sensitive and may appear anywhere

after the keyword **CONSTRAIN** provided it does not exceed column 80. The default value of **string** is “none” and this will be used if record 2 is omitted or if the record is incorrectly entered.

Record 3 is also optional. It determines the units of energy used to generate the force-field: It takes the form:

UNITS units

where **units** is one of “eV”, “kcal”, “kJ”, or “INTERNAL” referring, respectively to units of: electron volts, kcal mol⁻¹, kJ mol⁻¹ and the DL-POLY internal energy units (10 J mol⁻¹). If this line is omitted the default value will that for the AMBER force field *viz* kcal mol⁻¹.

The remainder of the file consists blocks of 2 lines each that indicate atomic coordinates and partial charges that are either additional to, or a modification of, those found in the EDTOUT file.

record i

atmnam	a8	atom type
index	i10	index

record ii

x	real	x coordinate
y	real	x coordinate
z	real	z coordinate
qq	real	partial charge

7.1.4.8 forcefield file .FF

Contains parameters characterizing the terms in the forcefield description. Integers are as i5, reals as f12 and atom name as a4.

The file contains the following information

record 1

ffname	a8	forcefiled name

Followed by blocks of data each block is of the form

record 1

header	a80	BLOCK = ...

record 2

nrec	integer	number of records

record 3 (for some)

fac	real	prefactor in energy expression

record 4 (for some)

potyp	a4	potential type

followed by **nrec** records giving data for that particular block. The blocks are in the order

block 1	name and masses	
name	a4	atom name
mass	real	atom mass (amu)
block 2	bond potentials	
bnda	a4	first atom type
bndb	a4	second atom type
cb	real	force constant
bo	real	bond length
block 3	angular potentials	
anga	a4	first atom type
angb	a4	second (central) atom type
angc	a4	third atom type
ct	real	force constant
to	real	bond angle (degrees)
block 4	dihedral potential	
dia	a4	first atom type
dib	a4	second atom type
dic	a4	third atom type
did	a4	fourth atom type
np	integer	repetition factor
cq	real	force constant
qd	real	angle (degrees)
td	real	periodicity
block 5	specific dihedral potentials	
as per block 4		
block 6	improper dihedral potentials	
as per block 4		
block 7	specific improper dihedral potentials	
as per block 4		
block 8	Hydrogen bond (12-10) potentials	
hba	a4	first atom type
hbb	a4	second atom type
h12	real	frict coefficient
h10	real	second coefficient
block 9	Van der Waals equivalents	
name	a4	primary atom type
name 1	a4	equivalent VdW type
...	a4	...
name 18	a4	equivalent VdW type
block 10	Van der Waal parameters	
nlj	a4	atom type

c12	real	first coefficient
c6	real	second coefficient
block 11	Van der Waal cross terms	
nlja	a4	first atom type
nljb	a4	second atom type
c12a	real	first coefficient
c6a	real	second coefficient
block 11	1..4 Van der Waal terms	
nljh	a4	atom type
c12b	real	first coefficient
c6b	real	second coefficient

7.1.4.9 Comments

Other forcefields can be accommodated by the construction of a suitable .FF file. Note that for dihedral terms the “* ” string acts as a wild card. Specific interactions override any general interaction parameters.

The utility PDB2EDT will convert a brookhaven pdb file into the EDTOUT form. The utility FILLEDT adds missing hydrogens (and Sulphur lone pairs) to the crystallographic data. The utility POMEDT reformats phosphate groups in nucleotide units into a suitable form for input into FFGEN and FILLEDT. Together these four utilities provide a direct route from a crystallographic structure to the DL-POLY input files.

7.1.5 FILLEDT

7.1.5.1 Header records

program filledt

```

C*****
C
C      DL_POLY utility to add missing hydrogens and Sulphur lone pairs
C      to a structure.
C      files are in dl_poly edit.out style
C
C      author t. forester  feb 1996
C      copyright daresbury laboratory 1996
C
C*****

```

7.1.5.2 Function

Adds all missing hydrogens and Sulphur lone pairs to a structure. Input (and output) are in a form compatible with the input for the utility FFGEN. It is intended that this utility be used before FFGEN to ensure the structure is complete.

7.1.5.3 Dependencies

- HADD4_0
- HADD3_1
- HADD2_1
- HADD1_1
- HADD2_2
- HADD1_2
- HADD1_3

7.1.5.4 Parameters

integers:

mxatm	50000	maximum number of atoms in final structure
mxtms	10	maximum number of molecular types
mxlist	10	maximum number of bonds to an atom
mxres	50	maximum number of different residues in database
mxsit	30	maximum number of unique atom types in a residue
mxunt	20000	maximum number of residues in a structure

`mxbond` 50000 maximum number of bonds in a structure

7.1.5.5 Input

Data Files:

filename 1: from interactive input

This file contains the residue maps to which the structural data in input file 2 is compared. The three residue map files supplied with DL_POLY_2 are “resmap.amb” (for the all atom version of the AMBER forcefield) “resmap_u.amb” (the united atom version of the AMBER forcefield) and “resmap_u.gro” (for the GROMOS forcefield). The format of these files is the same as the “respdb” files described in PDB2EDT.

filename 2: from interactive input

The file is in the format of the EDTOUT file produced by the utility PDB2EDT and required as input by FFGEN. See the utility FFGEN for a description of the format of this file.

7.1.5.6 Comments

The makefile “makefile.filledt” is supplied in *utility/macromol* to aid compiling a working version of this utility. It can be invoked by typing

```
make -f makefile.filledt
```

from the within the *utility/macromol* subdirectory. The executable it produces is “filledt.x”.

It is worth running “filledt” twice in order to scan for missing heavy ($Z \geq 1$) atoms in a structure. This utility will not attempt to add heavy atoms to the structure but will let you know which ones are missing.

The EDTOUT style input file can be generated by the utility PDB2EDT (followed by POM2EDT for files with nucleotide units. The “edtout” files generated by AMBER 3.0 - 4.1 can also be used provided they contain the atomic coordinates. An example can be found in the file “EDTOUT.example” in the *utility/macromol* directory of DL_POLY”.

7.1.6 GENLAT

7.1.6.1 Header records

```

      program genlat
c
c*****
c
c      program to generate a perfect lattice with general unit cell
c      author - w.smith oct. 1992
c
c*****
c

```

7.1.6.2 Function

GENLAT is used to generate the position coordinates of a simulation cell by replication of a unit cell. i.e. it generates a crystal lattice within the simulation cell. It is not confined to simple cubic lattices. The execution is interactive; the user must supply in turn:

1. a suitable text header;
2. the number of basis atoms;
3. the three components of the unit cell $\underline{a}$ vector;
4. the three components of the unit cell $\underline{b}$ vector;
5. the three components of the unit cell $\underline{c}$ vector;
6. the integer multiples of the $\underline{a}$, $\underline{b}$, $\underline{c}$, in the simulation cell; and
7. the name and fractional coordinates of the basis atoms.

The generated atomic lattice is output in a file called LATTICE, which is directly readable by DL_POLY. (See description of CONFIG file.)

7.1.6.3 Dependencies

- DCELL

7.1.6.4 Parameters

integers:

mxatom	1000	maximum number of basis atoms

7.1.6.5 Input**Interactive:**

record 1:		
title	a80	title for output file
record 2:		
natms	integer	number of basis atoms (max 1000)
record 3:		
<u>a</u>	3×real	unit cell vector <u>a</u>
record 4:		
<u>b</u>	3×real	unit cell vector <u>b</u>
record 5:		
<u>c</u>	3×real	unit cell vector <u>c</u>
record 6:		
nx,ny,nz	3×integer	multipliers of <u>a</u> , <u>b</u> , <u>c</u>
records 7 to natms+6:		
name,x,y,z	a8,3×real	names, and coords of basis atoms

7.1.6.6 Comments

If a large number of atoms are present in the basis, it is recommended that the input data be written into an input file and this fed to the program via standard input.

7.1.7 GENLAT.TO

7.1.7.1 Header records

```

      program genlat2
c
c*****
c
c      program to generate a truncated octahedral cell from a perfect
c      lattice
c      author - w.smith oct. 1992
c
c*****
c

```

7.1.7.2 Function

GENLAT is used to generate the position coordinates of a **truncated octahedral** simulation cell by replication of a unit cell. i.e. it generates a crystal lattice within the TO simulation cell. A cubic lattice is assumed. The execution is interactive; the user must supply in turn:

1. a suitable text header;
2. the number of basis atoms;
3. the cubic unit cell width;
4. the required spherical cutoff; and
5. the name and fractional coordinates of the basis atoms.

The generated TO simulation cell is output in a file called LATTICE, which is directly readable by DL_POLY. (See description of CONFIG file.) The cell produced will be the smallest TO compatible with the spherical cutoff provided.

7.1.7.3 Dependencies

- none

7.1.7.4 Parameters

integers:

mxatom	10	maximum number of basis atoms

7.1.7.5 Input**Interactive:****record 1:**

title	a80	title for output file

record 2:

natms	integer	number of basis atoms (max 10)

record 3:

width	real	unit cell width

record 4:

radius	real	spherical cutoff

records 5 to natms+4:

name,x,y,z	a8,3×real	names, and coords of basis atoms

7.1.7.6 Comments

If a large number of atoms are present in the basis, it is recommended that the input data be written into an input file and this fed to the program via standard input. (The maximum number of atoms in the basis must also be redefined in the source code.)

7.1.8 HADD1_1

7.1.8.1 Header records

```
subroutine hadd1_1(x,y,z,hx,hy,hz,bl,angd)
```

```
C*****
C
C      subroutine to add 1 terminal hydrogen
C              b-a...h
C
C      copyright daresbury laboratory 1996
C      author -      t forester   feb 1996
C
C*****
```

7.1.8.2 Function

Adds one bond of length `bl`, and bond angle `angd` to a site with one existing bond.

7.1.8.3 Dependencies

- none

7.1.8.4 Arguments

reals:

<code>x</code>	input	x coordinate of existing atoms (2)
<code>y</code>	input	y coordinate of existing atoms (2)
<code>z</code>	input	z coordinate of existing atoms (2)
<code>hx</code>	output	x coordinate of added atom
<code>hy</code>	output	y coordinate of added atom
<code>hz</code>	output	z coordinate of added atom
<code>bl</code>	input	bond length
<code>angd</code>	input	bond angle (in degrees)

7.1.8.5 Comments

The bond is added to the atom at $[x(1), y(1), z(1)]$.

7.1.9 HADD1_2**7.1.9.1 Header records**

```
subroutine hadd1_2(x,y,z,hx,hy,hz,bl,angd)
```

```

C*****
C
C      subroutine to add 1 hydrogen onto a 2 coordinate group
C
C              b - a ... H
C              |
C              c
C
C      copyright daresbury laboratory 1996
C      author -      t forester      feb 1996
C
C*****

```

7.1.9.2 Function

Adds one bond of length `bl`, and bond angle `angd` to a site with two existing bonds.

7.1.9.3 Dependencies

- none

7.1.9.4 Arguments

reals:

<code>x</code>	input	x coordinate of existing atoms (3)
<code>y</code>	input	y coordinate of existing atoms (3)
<code>z</code>	input	z coordinate of existing atoms (3)
<code>hx</code>	output	x coordinate of added atom
<code>hy</code>	output	y coordinate of added atom
<code>hz</code>	output	z coordinate of added atom
<code>bl</code>	input	bond length
<code>angd</code>	input	bond angle (in degrees)

7.1.9.5 Comments

The bond is added to the atom at $[x(1), y(1), z(1)]$ along the bisector of the other two bonds.

7.1.10 HADD1_3**7.1.10.1 Header records**

```

      subroutine hadd1_3(x,y,z,hx,hy,hz,bl)
c
c*****
c
c      dlpoly utility to add 1 hydrogen to a tertiary site
c
c      copyright daresbury laboratory 1993
c      author -      t forester      feb 1993
c      modified      t forester      feb 1996
c
c*****

```

7.1.10.2 Function

Adds one bond of length **bl**, and bond angle **angd** to a site with three existing bonds.

7.1.10.3 Dependencies

- none

7.1.10.4 Arguments

reals:

x	input	x coordinate of existing atoms (4)
y	input	y coordinate of existing atoms (4)
z	input	z coordinate of existing atoms (4)
hx	output	x coordinate of added atom
hy	output	y coordinate of added atom
hz	output	z coordinate of added atom
bl	input	bond length

7.1.10.5 Comments

The bond is added to the atom at $[x(1), y(1), z(1)]$ in the direction of the plane normal, where the plane is defined by the positions of the other three substituents.

7.1.11 HADD2_1

7.1.11.1 Header records

```
subroutine hadd2_1(x,y,z,hx,hy,hz,bl,angd,angt)
```

```
C*****
C
C      subroutine to add 2 bonds to a 1 coordinate site
C
C      copyright daresbury laboratory 1996
C      author -      t forester      feb 1996
C
C      $Author: trf $
C      $Date: 1996/03/04 14:20:59 $
C      $Revision: 1.1 $
C      $State: Exp $
C
C*****
```

7.1.11.2 Function

Adds two bonds of length **bl**, and bond angle **angd** to a site with two existing bonds. **angt** is the angle between the projection of the two added bonds onto the plane perpendicular to the existing bond.

7.1.11.3 Dependencies

- none

7.1.11.4 Arguments

reals:

x	input	x coordinate of existing atoms (2)
y	input	y coordinate of existing atoms (2)
z	input	z coordinate of existing atoms (2)
hx	output	x coordinate of added atom (2)
hy	output	y coordinate of added atom (2)
hz	output	z coordinate of added atom (2)
bl	input	bond length
angd	input	bond angle (in degrees)
angt	input	projection angle (in degrees)

7.1.11.5 Comments

The bond is added to the atom at $[x(1), y(1), z(1)]$. **angt** should be 120.0 for a tetrahedral site. In the case of adding lone pairs to a sulphur, say, **angt** should be 180.0.

7.1.12 HADD2_2**7.1.12.1 Header records**

```

      subroutine hadd2_2(x,y,z,hx,hy,hz,b1,angd)
c*****
c
c   program to add 2 bonds onto a 2 coordinate site
c
c           h2
c           :
c       b-a...h1
c           |
c           c
c
c   author t forester      feb 1996
c   copyright daresbury laboratory 1996
c
c*****

```

7.1.12.2 Function

Adds two bonds of length **b1**, and bond angle **angd** to a site with two existing bonds.

7.1.12.3 Dependencies

- none

7.1.12.4 Arguments

reals:

x	input	x coordinate of existing atoms (3)
y	input	y coordinate of existing atoms (3)
z	input	z coordinate of existing atoms (3)
hx	output	x coordinate of added atom (2)
hy	output	y coordinate of added atom (2)
hz	output	z coordinate of added atom (2)
b1	input	bond length
angd	input	bond angle (in degrees)

7.1.12.5 Comments

The bonds are added to the atom at $[x(1), y(1), z(1)]$ in the plane that bisects the other two bonds.

7.1.13 HADD3_1**7.1.13.1 Header records**

```

      subroutine hadd3_1(x,y,z,hx,hy,hz,bl,angd)
c
c*****
c
c      dl_poly utility to add 3 bonds to a 1 coordinate site
c
c          h1
c          :
c          b-a...h2
c          :
c          h3
c
c      copyright daresbury laboratory 1993
c      author      t forester   feb 1993
c
c*****

```

7.1.13.2 Function

Adds three bonds of length **bl**, and bond angle **angd** to a site with one existing bond. **angt** is the angle between the existing bond and the new bonds.

7.1.13.3 Dependencies

- none

7.1.13.4 Arguments**reals:**

x	input	x coordinate of existing atoms (2)
y	input	y coordinate of existing atoms (2)
z	input	z coordinate of existing atoms (2)
hx	output	x coordinate of added atoms (3)
hy	output	y coordinate of added atoms (3)
hz	output	z coordinate of added atoms (3)
bl	input	bond length
angd	input	bond angle (in degrees)

7.1.13.5 Comments

The bond is added to the atom at $[\mathbf{x}(1), \mathbf{y}(1), \mathbf{z}(1)]$. The existing bond is assumed to be an axis of three fold symmetry.

7.1.14 HADD4_0**7.1.14.1 Header records**

```

c
c*****
c
c      dl_poly utility to add 4 bonds to an uncoordinate site
c
c          h1
c          :
c      h4...a...h2
c          :
c          h3
c
c      copyright daresbury laboratory 1996
c      author      t forester   feb 1996
c
c*****

```

7.1.14.2 Function

Adds four bonds of length **bl** to an uncoordinated site. Tetrahedral symmetry is assumed.

7.1.14.3 Dependencies

- HADD3_1

7.1.14.4 Arguments

reals:

x	input	x coordinate of existing atom(2)
y	input	y coordinate of existing atom(2)
z	input	z coordinate of existing atom(2)
hx	output	x coordinate of added atoms (4)
hy	output	y coordinate of added atoms (4)
hz	output	z coordinate of added atoms (4)
bl	input	bond length
angd	input	bond angle (in degrees)

7.1.14.5 Comments

The first bond is added along the positive x direction.

7.1.15 NCONNECT**7.1.15.1 Header records**

```
subroutine nconnect(icn,name)
```

```
C*****
C
C      DL_POLY utility to assign number of neighbours for an atom type.
C      entries are for the AMBER and GROMOS  force fields
C
C      author t.forester feb 1996
C      copyright daresbury laboratory 1996
C
C*****
```

7.1.15.2 Function

Assigns the number of bonds expected to an atom type.

7.1.15.3 Dependencies

- NONE

7.1.15.4 Arguments

integers:

icn	integer	output	number of bonds

character:

name	a4	input	atom type

7.1.15.5 Comments

It may be necessary to differentiate between atom types that have the same names in different force fields. For example, we distinguish the AMBER sulphur “S” from the GROMOS sulphur “S” by relabelling the latter as “Sg”. “S” has 4 bonds (including 2 lone pairs) while “Sg” has only 2.

7.1.16 NFOLD

7.1.16.1 Header records

```

      program nfold
c*****
c
c      dl_poly utility to expand a simulation cell by a
c      nx*ny*nz multiplication
c
c      copyright daresbury laboratory 1997
c
c      author w. smith june 1997
c
c      Note: under no circumstances should the velocity arrays be
c            treated using this program
c
c*****

```

7.1.16.2 Function

NFOLD reads a DL_POLY_2 CONFIG file and expands it to n ($=n_x*n_y*n_z$) times the original size (i.e. n times the number of atoms, n times the volume) using a $n_x.n_y.n_z$ expansion in the principal directions.

7.1.16.3 Dependencies

- DCELL

7.1.16.4 Parameters

integers:

mega	1000000	Maximum number of records in CONFIG file

7.1.16.5 Input

Data Files:

- standard input

The input records are as follows

record 1:

fname	char*40	working name of CONFIG file

record 2:

nx,ny,nz integer expansion indices for x,y,z directions

- **filename: CONFIG**

This is the configuration file for a normal DL_POLY_2 run. Note that the working name of the trajectory file need not be CONFIG. The actual name is specified in standard input.

7.1.16.6 Output

Data Files:

- **filename: CONFIG_N**

This is a formatted file, written in standard CONFIG file format, containing n times as many atoms as the original CONFIG file.

7.1.16.7 Comments

Note that NFOLD ignores any velocity or force information in the CONFIG file. It is generally not advisable to assign the original velocities to the expanded file, though this is easily done. The catch is that the original periodicity of the smaller system will then be *completely preserved*, rendering the nfold expansion statistically worthless.

Beware of using NFOLD to expand systems that contain molecules (as opposed to unconnected atoms or ions). Bonds extending over half the cell width are the inevitable result!

7.1.17 PDB2EDT**7.1.17.1 Header records**

program pbd2edt

```

C*****
C
C      dl_poly routine to read a brookhaven pdb file
C      (or the AMBER processed equivalent) and to extract ...
C      1)  atom  names and types
C      2)  residue name and numbers
C      3)  x,y,z coordinates
C      4)  site charges
C      and then print in an edit.out compatible form
C
C      author t.forester feb 1996
C      copyright daresbury laboratory 1996
C
C*****

```

7.1.17.2 Function

The program processes a brookhaven database file into an “edit.out” format suitable for further processing with the DL_POLY_2 utilities POMEDT, FILLEDIT and **ffgen**.

7.1.17.3 Dependencies

- none

7.1.17.4 Parameters

integers:

mxres	50	maximum number different residue types
mxsit	30	maximum number unique atom types per residue

7.1.17.5 Input

residue map: filename from Interactive input:

record 1:

header	a	dummy

record 2:

nres	integer	number of residues in file

The next records are then in blocks pertaining to individual residues.

next record:

dummy	10x	
resnam	a3	residue name

next record:

dummy	8x	
nsite	integer	number of unique atom types in residue

next nsite records:

dummy	4x	site type number
join	i4	site atom is joined to
sitnam	a4	pdb site name
ffnam	a4	forcefield atom type
qsit	f8.0	partial charge
rept	f8.0	number of sites in residue

7.1.17.6 Comments

Only the following lines are processed from the pdb files

- CRYST1
- ATOM
- HETATM

All other lines are ignored.

Some hand editing of the pdb files may be required before running this utility:

- all residues are expected to be labelled by a three letter code, which must correspond to an entry in the residue map file. In the residue map files provided (“respdb.amb”, “respdb_u.amb” and “respdb_u.gro”) the codes for the 5 standard nucleotide units are “ADE”, “CYT”, “GUA”, “THY”, and “URA”, rather than the single letter symbols employed by the brookhaven database.
- In accordance with the AMBER forcefield cystine residues (“CYS”) involved in disulphide bonds should be relabelled as “CYX”.
- If the AMBER files are used Histidine residues (“HIS”) should be relabelled “HID”, “HIE” or “HIP” depending on the level of protonation required. By default they will be set to be “HID”.
- If the GROMOS files is used Histidine residues (“HIS”) should be relabelled “HIA”, “HIB” or “HIH” depending on the level of protonation required. By default they will be set to be “HIA”.
- The utility ignores any symmetry information in the pdb files. It thus does not attempt to generate any atoms in crystallographic equivalent positions. If they are required they will need to be added explicitly to the pdb file.

Three residue map files are provided with DL_POLY_2 : “respdb.amb”, “respdb_u.amb” (which correspond to the All-atom and the united atom versions of the AMBER forcefield) and “respdb_u.gro” (for the GROMOS forcefield based on the 37C4 parameterization). The files are very similar to “resmap.amb”, “resmap_u.amb” and “resmap_u.gro”. They differ only in that in the AMBER “respdb” files the nucleotide backbone sites are labelled with a “”, while in the AMBER “resmap” files they are labelled with a “/”. This labelling problem does not occur with the GROMOS forcefield.

For nucleotide units two points are of note. First, the residues used to compare with the information in the pdb file are constructed by combining a backbone residue (“dna” or “rna”) with the rest of the nucleotide at runtime. So additional nucleotide units in the database should not include backbone atoms. Moreover they **must** appear in the residue map file after the entries for the backbones. New Amino acid residue entries **must** appear before the nucleotide backbone entries. Second, the residues in the “resmap” files do not include the phosphate groups, which in accordance with the AMBER parameterization are listed as a residue in their own right. The utility POMEDT processes an edit.out style to separate the atoms of the phosphate groups into “POM” residues, distinct from the atoms of the nucleotide base.

Output: Some hand editing of the output file may also be required before filtering it through further utilities:

- Terminating groups will probably not have the correct atom charges or atom types assigned.
- The boundary between one molecule and another may not have been detected. It may be necessary to insert extra “MOLECULE” records into the file to delineate between one molecule and another.

7.1.18 POLYGROW

7.1.18.1 Header records

```

c      program polygrow
c
c*****
c
c      dl_poly utility for generating an amorphous polymer chain
c
c      copyright daresbury laboratory march 1995
c
c      author      w.smith march 1995
c
c*****
c

```

7.1.18.2 Function

POLYGROW generates the coordinates of a linear hydrocarbon chain using a weighted random walk. Each CH₂ unit is added with a random orientation about the dihedral and is checked for overlap with the existing units within a periodic boundary. A Monte Carlo procedure is used to accept or reject the addition. Dihedral and Lennard Jones interactions, temperature and volume are taken into account. The result is NOT a true equilibrated amorphous polymer chain, but it offers a configuration from which to start a simulation that is free of excessive overlap.

7.1.18.3 Dependencies

- DUNI - random number generator
- EULER - constructs rotation matrix
- OUTCON - writes out configuration data
- ROTATE - performs rotation operation
- SELECT - selects or rejects added monomer

7.1.18.4 Parameters

integers:

mxatms	864	Maximum number of atoms permitted
nsearch	100	Maximum number of attempts to build chain
mxtry	25	Maximum number of tries to add one unit
mxfail	1000	Maximum number attempts to restart chain growth

7.1.18.5 Input

Data Files:

• standard input

The input records are as follows:

record 1:		
nlinks	integer	number of monomer units in chain
record 2:		
Volm,Temp	2*real	volume (Å ³) and temperature (K)
record 3:		
ccbond,chbond	2*real	C-C and C-H bondlengths (Å)
record 4:		
ebond	real	C-C bond energy (kJ/mol)
record 5:		
eps(1),eps(2),eps(3)	3*real	C-C, C-H and H-H LJ epsilons (kJ/mol)
record 6:		
sig(1),sig(2),sig(3)	3*real	C-C, C-H and H-H LJ sigmas (Å)
record 7:		
dihsc1	real	dihedral force 1-4 scaling factor
record 5:		
c0,c1,c2,c3	4*real	dihedral angle potential parameters (kJ/mol)

7.1.18.6 Output

Data Files:

• standard output

This displays on-screen the input data plus any error messages and comments during the execution of the program.

• filename: POLYCON

This is a formatted file, containing the cartesian coordinates and chemical names of the polymer atoms.

7.1.18.7 Comments

The dihedral potential energy function has the form

$$V(r) = C_0 + C_1 * \cos(\theta) + C_2 * \cos^2(\theta) + C_3 * \cos^3(\theta) \quad (7.1)$$

Note that POLYGROW may fail to create a polymer chain within the time allowed, if the density of monomers is too high. The program has **no** automatic termination (it was designed for interactive use). If it hasn't converged in a reasonable time, it should be killed. Consider using an expanded system followed by an NPT simulation using DL_POLY_2 to produce the required density.

In effect the program tries to estimate the energy cost of adding a monomer unit to the end of an existing chain. Thus the molecule gains a C-C bond and the dihedral term, plus the new LJ interactions from the extra carbon and hydrogens, but loses the LJ terms from one terminal hydrogen atom (which is replaced by the carbon). There is no cost associated with the C-H bonds. Adjusting the C-C bond energy and the C_0 parameter of the dihedral term has an effect on the success ratio of the trial moves and can thus be adjusted to affect the rate of growth and chain steric energy.

7.1.18.8 Sample input file:

100				No. Units
8000.0	600.00			Volume, temperature ($\text{\AA}^3, \text{K}$)
1.5400	1.0900			C-C, CH bondlengths ($\text{\AA}$)
348.00				C-C Bond energy (kJ/mol)
0.2977	0.1573	0.0831		C-C, C-H, H-H Epsilons (kJ/mol)
3.4000	3.1000	2.8000		C-C, C-H, H-H Sigmas ($\text{\AA}$)
0.5000				Dihedral scale factor 1-4
8.8320	18.0870	4.8800	-31.8000	Dihedral coefficients (kJ/mol)

7.1.19 POMEDT

7.1.19.1 Header records

program pomedt

```
C*****
C
C      dl_poly utility to reprocesses files produced by the dl_poly
C      utility pdb2edt, for files with nucleotide residues.
C      In particular residues entries 'POM' (phosphate groups) are
C      seperated from the rest of the residue.
C
C      author t. forester feb 1996
C      copyright daresbury laboratory 1996
C
C*****
```

7.1.19.2 Function

As per headrer records.

7.1.19.3 Dependencies

- NONE

7.1.19.4 Comments

Input files will be prompted for by interactive input. The file to be processed, and the Output file are in the “edtout” format required by **ffgen**.

7.1.20 PROSEQ**7.1.20.1 Header records**

```

        program proseq
c
c*****
c
c      dl_poly utility program for reading SEQNET data files and
c      extracting the protein structure in a format compatible
c      with the dl_poly CONFIG file
c
c      copyright - daresbury laboratory 1993
c      author    - w. smith jan 1993
c
c*****
c

```

7.1.20.2 Function

Extracts a protein structure from a SEQNET file and formats in a way compatible with the CONFIG file required by DL_POLY.

7.1.20.3 Dependencies

- none

7.1.20.4 Parameters

integers:

mxatm	100000	maximum number of atoms in database file
mxseq	5000	maximum number of amino acids in sequence

7.1.20.5 Input

The input file must be in SEQNET form.

7.1.20.6 Comments

Information on the Daresbury Laboratory Protein Database service (SEQNET) is available from seqnet@daresbury.ac.uk.

7.1.21 RESIZE

7.1.21.1 Header records

```

        program resize
c
c*****
c
c      dl_poly utility program to resize a config file by linearly
c      scaling all the dimensions
c
c      author - w.smith december 1994
c      copyright daresbury laboratory 1994
c
c*****
c

```

7.1.21.2 Function

RESIZE reads a DL_POLY_2 CONFIG file and expands it to a new volume, while retaining the number of atoms. i.e. a linear scale factor α is applied to all independent directions in the system, resulting in an *isotropic* expansion. The new volume will be α^3 times the old volume.

7.1.21.3 Dependencies

- NONE

7.1.21.4 Parameters

integers:

mega	1000000	Maximum number of records in CONFIG file

7.1.21.5 Input

Data Files:

- : standard input

The input records are as follows:

record 1:

fname	char*40	name of CONFIG file

record 2:

scale	real	linear scale factor α for expansion

- **filename: CONFIG**

This is the configuration file for a normal DL_POLY_2 run. Note that the working name of the trajectory file need not be CONFIG. The actual name is specified in standard input, record 1.

7.1.21.6 Output

Data Files:

- **filename: NEW_CONFIG**

This is a formatted file, written in standard CONFIG file format, containing the same number of atoms as the original CONFIG file, but with a new volume.

7.1.21.7 Comments

Note that RESIZE will read any velocity or force information in the CONFIG file and write them unchanged in the NEW_CONFIG file.

Beware of using RESIZE to expand systems that contain molecules (as opposed to unconnected atoms or ions). This will be incorrect if the bonds are defined by constraints, and may lead to unrealistic bondlengths (and subsequently large forces) if the bonds are defined as extensible.

RESIZE can be used to do reductions in volume as well as expansions.

7.1.22 SOLVADD

7.1.22.1 Header records

program solvadd

```

C*****
C
C      DL_POLY utility
C
C      Program to add single site solvent molecules to a structure to
C      fill out the MD cell.
C      Assumes atomic postions are in a form compatible
C      with the CONFIG file used in DL_POLY with periodic boundary
C      conditions.
C
C      solvent is added from a f.c.c. lattice
C
C      input file = CONFIG
C      output file = CONFIG.solv
C
C      copyright Daresbury laboratory 1993
C      author -      t. forester  feb 1993
C
C*****

```

7.1.22.2 Function

The program superimposes a f.c.c. lattice of solvent atoms on top of the current configuration. Solvent molecules which are too close to the solute are removed, as are solvent molecules that are too close to other solvents. The latter may occur because the size of the f.c.c. lattice will, in general, not be commensurate with the MD cell. Input is read from the file CONFIG. Output is placed in the file CONFIG.solv.

7.1.22.3 Dependencies

- none

7.1.22.4 Parameters

integers:

mxatm	100000	maximum number of atoms in MD cell
mxcell	1000000	maximum number of link cells

7.1.22.5 Input

Interactive:

record 1:		
axcub	real	lattice size of f.c.c. solvent cell
record 2:		
ow	a8	name for solvent atom
record 3:		
tolnce	real	minimum permitted solvent-solute site distance
record 4:		
tolow	real	minimum permitted solvent - solvent distance

Data Files:

filename: CONFIG

See the file CONFIG in the section 4.1.2 for a description of this file.

7.1.22.6 Comments

The program uses a link-cell algorithm for searching out interaction pairs of interest. For very large systems the maximum number of link cells, `mxcell`, may need to be increased.

The cell size, periodic boundary conditions, etc, are set in the CONFIG file. The CONFIG file *must* have a unit cell and a image key, `imcon`, recognised by the DLPOLY subroutine IMAGES or equal to 10 (see below). For non-periodic systems mock cell vectors and a non-zero `imcon` must be specified in order for the program to know how much water to add.

The program will ask for interactive data specifying the closest solute-solvent distance and the closest permitted solute-solute distance. The latter is primarily to check for close interactions across periodic boundaries.

If `imcon` is set to 10 in the CONFIG file an ellipsoid of solvent will be added to the cell based on a cutoff scheme in reduced coordinates (so the cell vectors need not be orthogonal). The intention is that this option will be used only in preparation for a simulation in a non-periodic system. Consequently the output configuration will have `imcon` set to 0 and no cell vectors present.

The utility WATERADD provides similar functionality for the solvating of solute with thermalised spc water.

7.1.23 WATERADD

7.1.23.1 Header records

program wateradd

```

C*****
C
C      DL_POLY utility
C
C      Program to add SPC water molecules to a structure to fill
C      out the MD cell.
C      Assumes atomic postions are in a form compatible
C      with the CONFIG file used in DL_POLY with periodic boundary
C      conditions.
C      Water is added from the 'water.300K' file
C
C      input file = CONFIG
C      output file = CONFIG.plusH2O
C
C      copyright Daresbury laboratory 1993
C      author -      t. forester  feb 1993
C
C*****

```

7.1.23.2 Function

Fills out an MD cell with SPC water molecules. The waters are added from a configuration of SPC waters at 300 K. Input is read from the file CONFIG in the *utility*sub-directory. Output is placed in the file CONFIG.plusH2O in the *utility* directory.

7.1.23.3 Dependencies

- none

7.1.23.4 Parameters

integers:

mxatm	10000	maximum number of atoms in MD cell

7.1.23.5 Input

Data Files:

filename: CONFIG

See the file CONFIG in section 4.1.2 for a description of this file.

7.1.23.6 Comments

Cell size, periodic boundary conditions, etc, are set in the CONFIG file.

7.2 Analysis Utilities

7.2.1 ACF3D

7.2.1.1 Header records

```
program acf3d
```

```

c*****
c
c      DLPOLY routine for auto correlation function of three
c      dimensional vector -eg. velocities, forces.
c
c      Data is assumed to be formatted and compatible with the
c      HISTORY file written by DL_POLY subroutine traject.f
c
c      parallel version.
c      This program contains dummy routines for machine,gisum,and gdsum
c      to allow running on a single processor.
c
c      copyright daresbury laboratory 1993.
c
c      author t. forester  April 1993.
c
c*****

```

7.2.1.2 Function

Calculates the auto-correlation function and its integral for either velocities or forces for one selected atom type. Users will be prompted for:

1. the name of the trajectory file.
2. the number of atoms in the MD cell.
3. the atom label of interest.
4. the time interval (in ps) at which configurations were written.
5. length of the correlation function
6. output file name
7. vector key (1 for velocities, 2 for forces)

7.2.1.3 Dependencies

- MACHINE - SEE DL_POLY/SOURCE
- GDSUM - SEE DL_POLY/SOURCE
- GISUM - SEE DL_POLY/SOURCE

7.2.1.4 Parameters

integers:

<code>mxcor</code>	300	maximum length of correlation function
<code>mxatms</code>	1000	number of atoms in the MD cell
<code>msatms</code>	500	maximum number of atoms of interest for current processor

7.2.1.5 Input

Interactive:

record 1:

<code>dumpfile</code>	a80	name of trajectory file

record 2:

<code>natms</code>	integer	number of atoms in MD cell

record 3:

<code>acfme</code>	a8	atom label of interest

record 4:

<code>tstep</code>	real	time interval (ps) between configurations

record 5:

<code>ncor</code>	integer	number of time-steps to correlate over

record 6:

<code>outfile</code>	a80	output file name

record 7:

<code>lvf</code>	integer	function key (1 = velocities, 2 = forces)

data files:

filename: dumpfile (see above)

This file is assumed to be in the formatted form of the DL_POLY output file "HISTORY".

7.2.1.6 Comments

All calculations are performed in double precision. This may be changed to single precision if memory requirements pose a problem.

7.2.2 CORREL1

7.2.2.1 Header records

```

      program correl1
c
c*****
c
c      dl_poly utility program for calculating correlation functions
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith may 1992.
c
c*****
c

```

7.2.2.2 Function

CORREL1 calculates normalised correlation functions by the standard pedestrian method. The data are supplied as columns of numbers in the input file 'input.data', the first column being the time array. Execution is interactive, the user is first required to enter the number of correlation functions desired (max 10) and the number of data columns in the input file (max 15). Next the user must enter the pairs of numbers identifying the columns to be correlated. A column pair being (say) 4 6, meaning the 4'th and 6'th columns are to be correlated (the index does not count the time column). The columns may be up to 500 elements in length.

Each correlation function is output in its own distinct file, with an appropriate name (e.g. x0406 for the above example). Both cross- and auto-correlation functions may be calculated.

7.2.2.3 Dependencies

- none

7.2.2.4 Parameters

integers:

ndiv	500	maximum length of data arrays
mxcols	15	maximum number of data columns in input file (not including time column)
mxcorr	10	maximum number of correlation functions permitted

7.2.2.5 Input

Interactive:

record 1:
 npairs integer number of correlation functions (max 10)
record 2:
 ncols integer number of data columns in input file (max 15)
records 3 to npairs + 2:
 nc1 nc2 integers id numbers of columns to be correlated

Data Files:

input.data:

File comprised of N records (N<500) with contents:

time, col₁, col₂, . . . , col_{ncols}

where col_i is an entry for the *i*'th data column. (The data are in free format.)

7.2.2.6 Comments

This method is **not** recommended for large arrays or large numbers of correlation functions. The related program CORREL2 should be used for such cases.

7.2.3 CORREL2

7.2.3.1 Header records

```

        program correl2
c
c*****
c
c      dl_poly utility program for calculating correlation functions
c      using the fast fourier transform strategy
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith may 1992.
c
c      note - this program includes a naff fft routine purely for
c      portability and verification reasons. users are strongly
c      recommended to substitute a machine specific fft, which
c      should vastly improve performance.
c
c      note - you will probably need to change some of the array
c      dimensions using the following parameters.
c
c      note - ndiv must be a power of 2. ndiv3 will change according to
c      the fft routine you use. leave ndiv2 as it is.
c
c*****
c

```

7.2.3.2 Function

CORREL2 calculates normalised correlation functions by the Fast Fourier Transform method. The data are supplied as columns of numbers in the input file 'input.data', the first column being the time array. Execution is interactive, the user is first required to enter the number of correlation functions desired (max 10) and the number of data columns in the input file (max 15). Next the user must enter the pairs of numbers identifying the columns to be correlated. A column pair being (say) 4 6, meaning the 4'th and 6'th columns are to be correlated (the index does not count the time column). The columns may be up to 8192 entries in length.

Each correlation function is output in its own distinct file, with an appropriate name (e.g. x0406 for the above example). Both cross- and auto-correlation functions may be calculated.

7.2.3.3 Dependencies

- FFT (use machine specific version where available).

7.2.3.4 Parameters

integers:

<code>ndiv</code>	8192	maximum length of data arrays
<code>ndiv2</code>	16384	maximum length of FFT arrays
<code>ndiv3</code>	16834	maximum length of FFT workspace arrays
<code>mxcols</code>	15	maximum number of data columns in input file (not including time column)
<code>mxcorr</code>	10	maximum number of correlation functions permitted

7.2.3.5 Input

Interactive:

record 1:

<code>npairs</code>	integer	number of correlation functions (max 10)

record 2:

<code>ncols</code>	integer	number of data columns in input file (max 15)

records 3 to `npairs + 2`:

<code>nc1 nc2</code>	integers	id numbers of columns to be correlated

Data Files:

input.data:

File comprised of N records ($N < 8192$) with contents:

time, col_1 , col_2 , ..., col_{ncols}

where col_i is an entry for the i 'th data column. (The data are in free format.)

7.2.3.6 Comments

This method is recommended for large arrays or large numbers of correlation functions. The related program CORREL1 is more suitable for smaller cases.

7.2.4 CURDEN

7.2.4.1 Header records

```

      program curden
c*****
c
c      daresbury laboratory dl_poly program for the calculation
c      of the fourier transform of the particle current density
c      in space and time
c
c      original curden program written by w.smith march 1982
c      adapted for dl_poly by w.smith august 1994
c      parallel version
c
c      author w smith 1994
c      copyright daresbury laboratory 1994
c
c*****

```

7.2.4.2 Function

CURDEN calculates the spatial Fourier Transform (FT) of the particle (or charge) current density (longitudinal and transverse) for a range of selected $\underline{k}$ vectors, and the correlation function of these. It also calculates the temporal FT of the longitudinal and transverse correlations.

7.2.4.3 Dependencies

- CORREL - calculates time correlation functions
- CRDC - calculates dot product of a real and a complex array
- CURFFT - calculates temporal FT of particle current correlation
- FFT
- FOURDC - calculates spatial FT of particle current density
- GDSUM
- INVERT
- MACHINE
- NBITS - calculates number of set bits in an integer
- SDOT - calculates dot product of two real arrays

- START - reads standard input data and checks numbers
- TABULATE - tabulates output
- TIMCHK

The subroutines CORREL, CURFFT, FOURDC, NBITS, SDOT, CRDC, START and TABULATE are specific to CURDEN. The remainder are standard DL_POLY_2 utilities (see within).

7.2.4.4 Parameters

integers:

mcore	116000	size of re-useable working array SPACE
mzp	1080	Maximum number of atoms of required type in MD cell
iread	5	channel number of input file curden_input
irite	6	channel number of standard output
iconf	7	channel number of HISTORY file
iwork	8	channel number of CUR_SPC file
isave1	19	channel number of CUR_COR file
isave2	20	channel number of CUR_FFT file

reals:

pi	π	numerical value of π
tpi	2π	numerical value of 2π

7.2.4.5 Input

Data Files:

- filename: curden_input

This is a free formatted file. The records are as follows:

record 1:

title	char*80	title of job (header record)

record 2:

natm	integer	number of atoms in MD cell

record 3:

ncon	integer	number of configurations in HISTORY file

record 4:

kmax	integer	maximum $\underline{k}$ vector index required

record 5:

lden	logical	TRUE for spatial current density

record 6:

lcor	logical	TRUE for correlations of current density

record 7:		
ltim	logical	TRUE for Fourier transforms of correlations
record 8:		
lchg	logical	TRUE = charge current. FALSE = particle current
record 9:		
ntime	integer	No. of data points in time correlation arrays.
record 10:		
ngap	integer	interval between time origins
record 11:		
tstep	real	time interval between configurations (ps)

• **filename: HISTORY**

This is the trajectory file produced by a DL_POLY_2 run. Note that CURDEN assumes that the name of the trajectory file is HISTORY. The current version of CURDEN assumes this file is *formatted*.

7.2.4.6 Output

Data Files:

• **filename: curden_output**

This is a formatted file. The output consists of any error message and/or the following:

- a summary of the input data specified in the curden_input file, plus the header record of the HISTORY file, followed by:
- a block of data summarising the FT of the particle (or charge) current density: the l,m,n values of the $\underline{k}$ vectors and the squared modulus of the FT of the longitudinal and transverse current densities (i.e. $|j_L(\underline{k})|^2$ and $|j_T(\underline{k})|^2$).
- a block of data summarising the correlation of the FT of the particle current densities (i.e. $\langle j_L(\underline{k}, t) j_L(-\underline{k}, 0) \rangle$ and $\langle j_T(\underline{k}, t) \cdot j_T(-\underline{k}, 0) \rangle$), for each l,m,n index, written as a function of time.
- a block of data summarising the FT of the current density correlation functions for each l,m,n index, written as a function of angular frequency (ω).

• **filename: CUR_SPC**

This is an unformatted file storing all the terms from the FT of the particle (or charge) current. Each unformatted record contains all the terms from a single configuration (i.e. $j_L(\underline{k})$ and $j_T(\underline{k})$ for all $\underline{k}$ vectors required). The file is also used as an *input file* for the correlation function calculations. It is generated by the subroutine FOURDC.

- **filename: CUR_COR**

This is an unformatted file storing the current density correlation functions. Each record consists of a block of several correlation functions, two for each $\underline{k}$ vector ((i.e. $\langle j_L(\underline{k}, t) j_L(-\underline{k}, 0) \rangle$ and $\langle j_T(\underline{k}, t) \cdot j_T(-\underline{k}, 0) \rangle$) written as a function of time. The file is generated by subroutine CORREL.

- **filename: CUR_FFT**

This is an unformatted file storing the FT of the current correlation functions. Each record consists of several FTs, two for each $\underline{k}$ vector (as with the CUR_COR file). The file is generated by the subroutine CURFFT.

7.2.4.7 Comments

1. CURDEN is a parallel code designed for the Intel hypercube. Adaptation to single processor machines is easily accomplished by setting the variables **idnode** and **mxnode** to 0 and 1 respectively in the subroutine MACHINE and commenting out calls to the communication routines CSEND, CRECV etc. Conversion to MPI (for use on other parallel machines can be accomplished by substitution of the Intel specific routines for the MPI equivalents.
2. CURDEN does all its internal calculations using a managed array area called **space**. The size of this area is defined by the parameter **mcure**. The area is used to store several different kinds of data, and is continually re-used, thus making the most economic use of computer memory.
3. CURDEN can calculate the particle current **or** charge current correlations, by selecting appropriate information from the HISTORY file. This is determined by the variable **lchg**, which if TRUE will calculate the charge correlation functions and if FALSE will calculate the particle correlation functions. Other atomic weightings besides charge are easily possible, but these must be coded by the user.
4. CURDEN can calculate the FT of the current density, the longitudinal and transverse correlation functions and their Fourier transforms - in that order. It is not necessary to calculate all these, and some control is offered by the variables **lden**, **lcor** and **ltim**. However it is important to note that the correlation functions cannot be calculated without the CUR_SPC file first being present, and the Fourier transforms require the presence of the CUR_COR file, which are produced at earlier stages.
5. The indices used to define the $\underline{k}$ vectors (l,m,n), simply specify the multiples of the reciprocal space vectors for the MD cell. For example, for a cubic cell of width L , the $\underline{k}$ vectors are described by:

$$\underline{k} = \frac{2\pi}{L}(l, m, n)^{\dagger}.$$

Choosing a specific value for **kmax** in the `curden_input` file, means that the following $\underline{k}$ vectors will be processed:

- when $n = 0, m = 0$ then $l = 1 \rightarrow kmax$, otherwise:
- when $n = 0$ then $m = 0 \rightarrow kmax$ and $l = -kmax \rightarrow kmax$;
- when $n \neq 0$ then $m = -kmax \rightarrow kmax$ and $l = -kmax \rightarrow kmax$.

Thus allowance is made for inversion symmetry in the reciprocal cell.

6. The order in which CURDEN generates $\underline{k}$ vectors is given by the simple FORTRAN algorithm:

```

k=0
mmin=0
lmin=1
do n=0,kmax
  do m=mmin,kmax
    do l=lmin,kmax
      k=k+1
    enddo
    lmin=-kmax
  enddo
  mmin=-kmax
enddo

```

7. The origin interval **ngap** is usually set to 1 in the `curden_input` file, which means that every configuration in the HISTORY file will be used as a time origin to calculate the density time correlations. If it is set to a larger number, the intervening configurations are not used as origins, but are used to calculate the correlation functions. (Note that the time interval **tstep** specified in the `curden_input` file is the time between HISTORY file configurations, and is unaffected by the value of **ngap**.)
8. The imaginary parts of the current correlation functions and their Fourier transforms have no physical significance and should be ignored.

7.2.4.8 Example input file

The following example `curden_input` file is taken from a sodium chloride simulation.

```

Sodium Chloride test for CURDEN
216          natms
1500         ncon
2           kmax
.true.      lden
.true.      lcor

```

.true.	ltim
.true.	lchg
128	ntime
1	ngap
0.004	tstep

7.2.5 DENCOR

7.2.5.1 Header records

```

      program dencor
c*****
c
c      daresbury laboratory dl_poly program for the calculation
c      of the fourier transform of the density in space and time
c
c      original dencor program written by w.smith march 82
c      adapted for dl_poly by w.smith november 1994
c      parallel version
c
c      author w smith 1994
c      copyright daresbury laboratory 1994
c
c*****

```

7.2.5.2 Function

DENCOR calculates the spatial Fourier Transform (FT) of the particle (or charge) density for a range of selected $\underline{k}$ vectors, and the correlation function of this (known as the intermediate scattering function). It also calculates the temporal FT of the intermediate scattering function, which is known as the dynamic structure factor.

7.2.5.3 Dependencies

- CORREL - calculates time correlation functions
- DENFFT - calculates temporal FT of particle density
- FFT
- FORDEN - calculates spatial FT of particle density
- GDSUM
- INVERT
- MACHINE
- NBITS - calculates number of nonzero bits in an integer
- START - reads standard input data and checks numbers
- TABULATE - tabulates output
- TIMCHK

The subroutines CORREL, DENFFT, FORDEN, NBITS, START and TABULATE are specific to DENCOR. The remainder are standard DL_POLY.2 utilities (see within).

7.2.5.4 Parameters

integers:

mcore	116000	size of re-useable working array SPACE
msp	1080	Maximum number of atoms of required type in MD cell
iread	5	channel number of input file dencor_input
irite	6	channel number of standard output
iconf	7	channel number of HISTORY file
iwork	8	channel number of DEN_SPC file
isave1	9	channel number of DEN_COR file
isave2	10	channel number of DEN_FFT file

reals:

pi	π	numerical value of π
tpi	2π	numerical value of 2π

7.2.5.5 Input

Data Files:

• filename: dencor_input

This is a free formatted file. The input records are as follows.

record 1:

title	char*80	title of job (header record)

record 2:

natm	integer	number of atoms in MD cell

record 3:

ncon	integer	number of configurations in HISTORY file

record 4:

kmax	integer	maximum $\underline{k}$ vector index required

record 5:

ldcn	logical	TRUE for spatial density calculation

record 6:

lcor	logical	TRUE for intermediate scattering function

record 7:

ltim	logical	TRUE for dynamic structure factor

record 8:

lchg	logical	TRUE = charge density. FALSE = Particle density

record 9:

ntime	integer	No. of data points in time correlation arrays.

record 10:

ngap	integer	interval between time origins

record 11:

tstep	real	time interval between configurations (ps)

- **filename: HISTORY**

This is the trajectory file produced by a DL_POLY_2 run. Note that DENCOR assumes that the name of the trajectory file is HISTORY. The current version of DENCOR assumes this file is *formatted*.

7.2.5.6 Output

Data Files:

- **filename: dencor_output**

This is a formatted file. The output consists of any error message and/or the following:

- a summary of the input data specified in the dencor_input file, plus the header record of the HISTORY file, followed by:
- a block of data summarising the FT of the particle (or charge) density: the l,m,n values of the $\underline{k}$ vectors, the squared modulus of the FT of the particle density (i.e. $|\rho(\underline{k})|^2$) and the (complex) FT of the particle density (i.e. $\rho(\underline{k})$).
- a block of data summarising the intermediate scattering functions (i.e. the correlation of the FT of the particle density $\langle \rho(\underline{k}, t)\rho(-\underline{k}, 0) \rangle$), for each l,m,n index, written as a function of time.
- a block of data summarising the dynamic structure factors, for each l,m,n index, written as a function of angular frequency (ω).

- **filename: DEN_SPC**

This is an unformatted file storing all the terms from the FT of the particle (or charge) density. Each unformatted record contains all the terms from a single configuration (i.e. $\rho(\underline{k})$ for all $\underline{k}$ vectors processed). The file is also used as an *input file* for the intermediate scattering function and dynamic structure factor calculations. It is generated by the subroutine FORGEN.

- **filename: DEN_COR**

This is an unformatted file storing the intermediate scattering functions. Each record consists of a block of several intermediate scattering functions; one for each $\underline{k}$ vector, written as a function of time. The file is generated by subroutine CORREL.

- **filename:** `DEN_FFT`

This is an unformatted file storing the dynamic structure factors. Each record consists of several dynamic structure factors; one for each $\underline{k}$ vector. The file is generated by the subroutine `DENFFT`.

7.2.5.7 Comments

1. `DENCOR` is a parallel code designed for the Intel hypercube. Adaptation to single processor machines is easily accomplished by setting the internal variable `idnode` and `mxnode` to 0 and 1 respectively in the subroutine `MACHINE` and commenting out calls to the communication routines `CSEND`, `CRECV` etc. Conversion to MPI (for use on other parallel machines) can be accomplished by substitution of the Intel specific routines for the MPI equivalents.
2. `DENCOR` does all its internal calculations using a managed array area called `space`. The size of this area is define by the parameter `mcore`. The area is used to store several different kinds of data, and is continually re-used, this making the most economic use of computer memory.
3. `DENCOR` can calculate the particle density properties **or** the corresponding particle charge properties, by selecting either the particle or charge information from the `HISTORY` file. This is determined by the variable `lchg`, which if `TRUE` will calculate the charge density functions and if `FALSE` will calculate the particle density. Other atomic weightings besides charge are easily possible, but these must be coded by the user.
4. `DENCOR` can calculate the FT of the particle density, the intermediate scattering function and the dynamic structure factor - in that order. It is not necessary to calculate all these, and some control is offered by the variables `lden`, `lcor` and `ltim`. However it is important to note that the intermediate scattering function cannot be calculated without the `DEN_SPC` file first being present, and the dynamic structure factor requires the presence of the `DEN_COR` file, which are produced at earlier stages.
5. The indices used to define the $\underline{k}$ vectors (l,m,n), simply specify the multiples of the reciprocal space vectors for the MD cell. For example, for a cubic cell of width L , the $\underline{k}$ vectors are described by:

$$\underline{k} = \frac{2\pi}{L}(l, m, n)^{\dagger}.$$

Choosing a specific value of $kmax$ in the `dencor_input` file means that the following $\underline{k}$ vectors will be processed:

- when $n = 0, m = 0$ then $l = 1 \rightarrow kmax$, otherwise:
- when $n = 0$ then $m = 0 \rightarrow kmax$ and $l = -kmax \rightarrow kmax$;

- when $n \neq 0$ then $m = -kmax \rightarrow kmax$ and $l = -kmax \rightarrow kmax$.

Thus allowance is made for inversion symmetry in the reciprocal cell.

6. The order in which DENCOR generates $\underline{k}$ vectors is given by the simple FORTRAN algorithm:

```

k=0
mmin=0
lmin=1
do n=0,kmax
  do m=mmin,kmax
    do l=lmin,kmax
      k=k+1
    enddo
    lmin=-kmax
  enddo
  mmin=-kmax
enddo

```

7. The origin interval `ngap` is usually set to 1 in the `dencor_input` file, which means that every configuration in the HISTORY file will be used as a time origin to calculate the density time correlations. If it is set to a larger number, the intervening configurations are not used as origins, but are used to calculate the correlation functions. (Note that the time interval `tstep` specified in the `dencor_input` file is the time between HISTORY file configurations, and is unaffected by the value of `ngap`.)
8. The imaginary parts of the intermediate scattering function and the dynamic structure factor have no physical significance and should be ignored.

7.2.5.8 Example input file

The following example of the `dencor_input` file is taken from a simulation of sodium chloride.:

```

Sodium Chloride test for DENCOR
216          natms
1500         ncon
2           kmax
.true.       lden
.true.       lcor
.true.       ltim
.true.       lchg
128         ntime
1           ngap
0.004       tstep

```

7.2.6 DIPOLE

7.2.6.1 Header records

program dipole

```

C*****
C
C      dl_poly program to calculate cluster dipoles
C      for selected atom clusters from dl_poly HISTORY file
C
C      copyright daresbury laboratory 1996
C      author  w.smith oct 1996
C
C      wl
C      1996/02/15 14:33:26
C      1.1.1.1
C      Exp
C
C*****

```

7.2.6.2 Function

DIPOLE calculates the dipole moment of a selected cluster of atoms using data stored in the HISTORY file. It can read both formatted and unformatted versions of the HISTORY file.

7.2.6.3 Dependencies

- HREAD
- UREAD

7.2.6.4 Parameters

integers:

mxatms	1080	Maximum number of atoms in MD cell
mxcls	2	Maximum number of atomic clusters

7.2.6.5 Input

Data Files:

- filename: dip_input

This is a free formatted file. The input records are as follows.

record 1:		
ncls	integer	number of clusters required
record 2 to ncls+1:		
ibc(i),iec(i)	2*integer	start and end indices of each cluster
record ncls+2:		
fname	char*40	working name of HISTORY file
record ncls+3:		
fmt	char*1	specified format of HISTORY file (U or F)
record ncls+4:		
natms	integer	number of atoms in MD cell
record ncls+5:		
nconf	integer	number of configurations in HISTORY file
record ncls+6:		
tstep	real	time interval between configurations (ps)

- **filename: HISTORY**

This is the trajectory file produced by a DL_POLY_2 run. Note that the working name of the trajectory file need not be HISTORY. The actual name is specified in the dip_input file.

7.2.6.6 Output

Data Files:

- **filename: dip_output**

This is a formatted file. The output consists of the following:

- the time and the calculated components of the dipole clusters; one record per time configuration. Format: (1p,10e12.4).

7.2.6.7 Comments

DIPOLE can handle either formatted or unformatted HISTORY files. The choice is made by specifying either 'U' or 'F' as the **fmt** variable in the dip_input file.

Note that, as written, DIPOLE assumes the configurations in the HISTORY file have **no** periodic boundary condition.

7.2.7 GDIF

7.2.7.1 Header records

```
program gdiff
```

```

C*****
C
C      dl_poly program to calculate cross term of Van Hove density
C      correlation function for selected atoms
C
C      copyright daresbury laboratory 1995
C      author  w.smith march 1995
C
C*****

```

7.2.7.2 Function

GDIF calculates the van Hove distinct correlation function for a nominated pair of atom types, using data stored in the HISTORY file. GDIF regards the first nominated atom type as being the ‘central atom’ about which atoms of the second nominated atom type congregate (the ‘peripheral atoms’). The results are in the form of spherically symmetric functions which describe how the arrangement of the peripheral atoms, which are originally arranged around the site of the central atom at time zero, changes with time.

GDIF can read both formatted and unformatted versions of the HISTORY file.

7.2.7.3 Dependencies

- HREAD (modified)
- UREAD (modified)
- INVERT

7.2.7.4 Parameters

integers:

mxatms	1080	Maximum number of atoms in MD cell
nstr	240	Maximum number of atoms of selected type
mxcrs	64	Number of bins in radial correlation
mxtim	64	Number of points in time correlation

7.2.7.5 Input

Data Files:

• filename: **gdiff_input**

This is a free formatted file.

record 1:		
atnam1	char*8	name of central atoms in correlation
record 2:		
atnam1	char*8	name of peripheral atoms in correlation
record 3:		
fname	char*40	working name of HISTORY file
record 4:		
fmt	char*1	specified format of HISTORY file (U or F)
record 5:		
natms	integer	number of atoms in MD cell
record 6:		
tstep	real	time interval between configurations (ps)
record 7:		
nconf	integer	number of configurations in HISTORY file
record 8:		
isampl	integer	sampling interval for selecting configurations
record 9:		
iogcrs	integer	origin interval between selected configurations
record 10:		
rcut	real	cutoff radius for spatial correlation (Å)
record 11-19:		
cell(1,...,9)	real	MD cell vectors in order A,B,C (Å)

• filename: **HISTORY**

This is the trajectory file produced by a DL_POLY_2 run. Note that the working name of the trajectory file need not be HISTORY. The actual name is specified in record 3 of the gdiff_input file.

7.2.7.6 Output

Data Files:

• filename: **gdiff_output**

This is a formatted file. The output consists of the following:

- a summary of the input data specified in the gdiff_input file followed by:
- the distinct correlation function tabulated vertically by distance (in Å) and horizontally by time (in ps). It is split into several blocks of eight columns each. The first

column in each block is a tabulation of the radius, and subsequent columns present the van Hove distinct correlation as a function of radius for the time given at the head of the column.

7.2.7.7 Comments

GDIFF can be used to correlate atoms of the same type, but the result will **not** include correlations of an atom with itself i.e. the self correlation function is discarded.

GDIFF can handle either formatted or unformatted HISTORY files. The choice is made by specifying either 'U' or 'F' as the `fmt` variable on record 4 of the `gdiff_input` file.

The number of "bins" in the radial coordinate of the correlation function is given by the parameter `mxcrs` and the range of the correlation is set by the variable `rcut`. It follows that the width of a bin is `rcut/mxcrs`.

A sampling interval `isampl` greater than 1, means that GDIFF processes the configurations in the HISTORY at intervals of `isampl` configurations, missing out the intervening `isampl-1` configurations. However at the end of the HISTORY file, the program returns to the start and begins again, starting from the second configuration in the file and striding through with the same sampling interval. This procedure is repeated `isampl` times, increasing the starting configuration number by one each time, until all configurations in the file have been processed. In effect, the program calculates the correlation function `isampl` times, starting from a slightly different time point on each pass. In this way, maximum use of the HISTORY file data is made. If users find this rather slow, it is permissible to alter the statement on line 137 from

```
do ipass=0,isampl-1
```

to

```
do ipass=0,0
```

which speeds up the correlation function calculations by a factor of `isampl`, at the expense of a noisier function however.

The number of points in the time coordinate of the correlation function is given by the parameter `mxtim`. The time interval between each point is determined jointly by the sampling interval `isampl` and the variable `tstep` (i.e. by the product `tstep × isampl`). It follows that the maximum temporal length of the correlation function is given by `tstep × isampl × mxtim`. Note that `tstep` is the time between successive configurations in the HISTORY file, and is an integer multiple of the molecular dynamics timestep.

The origin interval `iogcrs` is usually set to 1, which means that every *sampled* configuration is used as an origin in each pass of the distinct correlation calculation. If a larger number is used, then the intervening sampled configurations are not used as origins, though they will be used to calculate the distinct correlation. The time interval between origins is thus given by `iogcrs × isampl × tstep`. This facility speeds up the calculation, at the price of poorer statistics. The recommended value for `iogcrs` is 1.

7.2.8 GSELF

7.2.8.1 Header records

program gself

```

C*****
C
C      dl_poly program to calculate van Hove self correlation
C      function for selected atoms from dl_poly HISTORY file
C      includes gaussian analysis as described by rahman
C      in phys rev 136 (1964) a405
C
C      copyright daresbury laboratory 1996
C      author  w.smith jan 1996
C
C*****

```

7.2.8.2 Function

GSELF calculates the van Hove self correlation function for a nominated atom type, using data stored in the HISTORY file. The result is in the form of a spherically symmetric function that changes with time. It thus describes how the probability density of finding an atom, at a given radius from its position at time zero, changes with time.

GSELF can read both formatted and unformatted versions of the HISTORY file.

7.2.8.3 Dependencies

- HREAD (MODIFIED)
- UREAD (MODIFIED)
- INVERT

7.2.8.4 Parameters

integers:

mxatms	1080	Maximum number of atoms in MD cell
nstr	240	Maximum number of atoms of selected type
mxslf	64	Number of bins in radial correlation
mxtim	64	Number of points in time correlation

7.2.8.5 Input

Data Files:

• filename: **gself_input**

This is a free formatted file.

record 1:		
atname	char*8	name of atom of interest
record 2:		
fname	char*40	working name of HISTORY file
record 3:		
fmt	char*1	specified format of HISTORY file (U or F)
record 4:		
natms	integer	number of atoms in MD cell
record 5:		
tstep	real	time interval between configurations (ps)
record 6:		
nconf	integer	number of configurations in HISTORY file
record 7:		
isampl	integer	sampling interval for selecting configurations
record 8:		
iogself	integer	origin interval between selected configurations
record 9:		
rcut	real	cutoff radius for spatial correlation (Å)
record 10-18:		
cell(1,...,9)	real	MD cell vectors in order A,B,C (Å)

• filename: **HISTORY**

This is the trajectory file produced by a DL_POLY_2 run. Note that the working name of the trajectory file need not be HISTORY. The actual name is specified in record 2 of the gself_input file.

7.2.8.6 Output

Data Files:

• filename: **gself_output**

This is a formatted file. The output consists of the following:

- a summary of the input data specified in the gself_input file, followed by:
- the self correlation function tabulated vertically by distance (in Å) and horizontally by time (in ps). It is split into several blocks of eight columns each. The first column in each block is a tabulation of the radius, and subsequent columns present the van

Hove self correlation as a function of radius for the time given at the head of the column.

- the results of an analysis of the correlation function in terms of its Gaussian moments i.e. the functions $\langle r^{2n} \rangle$ and α_n described by Rahman [49].

7.2.8.7 Comments

GSELF can handle either formatted or unformatted HISTORY files. The choice is made by specifying either 'U' or 'F' as the `fmt` variable on record 3 of the `gself_input` file.

The number of "bins" in the radial coordinate of the correlation function is given by the parameter `mxself` and the range of the correlation is set by the variable `rcut`. It follows that the width of a bin is `rcut/mxself`.

A sampling interval `isamp1` greater than 1, means that GSELF processes the configurations in the HISTORY at intervals of `isamp1` configurations, missing out the intervening `isamp1-1` configurations. However at the end of the HISTORY file, the program returns to the start and begins again, starting from the second configuration in the file and striding through with the same sampling interval. This procedure is repeated `isamp1` times, increasing the starting configuration number by one each time, until all configurations in the file have been processed. In effect, the program calculates the correlation function `isamp1` times, starting from a slightly different time point on each pass. In this way, maximum use of the HISTORY file data is made.

The number of points in the time coordinate of the correlation function is given by the parameter `mxtim`. The time interval between each point is determined jointly by the sampling interval `isamp1` and the variable `tstep` (i.e. by the product `tstep × isamp1`). It follows that the maximum temporal length of the correlation function is given by `tstep × isamp1 × mxtim`. Note that `tstep` is the time between successive configurations in the HISTORY file, and is an integer multiple of the molecular dynamics timestep.

The origin interval `iogself` is usually set to 1, which means that every *sampled* configuration is used as an origin in each pass of the self correlation calculation. If a larger number is used, then the intervening sampled configurations are not used as origins, though they will be used to calculate the self correlation. The time interval between origins is thus given by `iogself × isamp1 × tstep`. This facility speeds up the calculation, at the price of poorer statistics. The recommended value for `iogself` is 1.

7.2.9 MSDBLK

7.2.9.1 Header records

```

        program msdblk
c*****
c
c    dl_poly program to calculate mean square displacement
c    for selected atoms from dl_poly HISTORY file
c    this version incorporates a blocking average method
c    to calculate the statistical error
c
c    copyright daresbury laboratory 1996
c    author w.smith jan 1996
c
c*****

```

7.2.9.2 Function

MSDBLK calculates the Mean Square Displacement (MSD) for a nominated atom type, using data stored in the HISTORY file and then proceeds to estimate the statistical error (i.e. error bars) in the MSD obtained using the blocking method of Flyvbjerg and Petersen [50] for each point on the MSD.

MSDBLK can read both formatted and unformatted versions of the HISTORY file.

7.2.9.3 Dependencies

- HREAD
- UREAD
- INVERT

7.2.9.4 Parameters

integers:

mxatms	1080	maximum number of atoms in MD cell
npts	600	number of data points in MSD array
nstr	1080	maximum of atoms of type selected
ndeg	10	maximum degree of data blocking
mxconf	2500	maximum number of configurations to be sampled

7.2.9.5 Input

Data Files:

• **filename: blk_input**

This is a free formatted file.

record 1:		
atname	char*8	name of atom of interest
record 2:		
fname	char*40	working name of HISTORY file
record 3:		
fmt	char*1	specified format of HISTORY file (U or F)
record 4:		
natms	integer	number of atoms in MD cell
record 5:		
tstep	real	time interval between configurations (ps)
record 6:		
nconf	integer	number of configurations in HISTORY file
record 7:		
nmsd	integer	required number of data points in MSD
record 8:		
isampl	integer	sampling interval for selecting configurations
record 9:		
iomsd	integer	origin interval between selected configurations
record 10-18:		
cell(1,...,9)	real	MD cell vectors in order A,B,C (Å)

• **filename: HISTORY**

This is the trajectory file produced by a DL_POLY_2 run. Note that the working name of the trajectory file need not be HISTORY. The actual name is specified in record 2 of the blk_input file.

7.2.9.6 Output

Data Files:

• **filename: blk_output**

This is a formatted file. The output consists of the following:

- a summary of the input data specified in the blk_input file, followed by:
- several columns of data (format: 1p,12e11.4) first column is time, second column is the MSD and subsequent columns are the estimates of sigma at each degree of data blocking (up to order **ndeg**). The time is in the same units as the **tstep** variable in the blk_input file, the MSD is in Å².

7.2.9.7 Comments

MSDBLK can handle either formatted or unformatted HISTORY files. The choice is made by specifying either 'U' or 'F' as the `fmt` variable on record 3 of the `blk_input` file.

In principle, the error estimate should converge with increasing order of blocking (see [50]), but in practice statistical noise may obscure this, and an estimate of the converged value may need to be taken for the true statistical error. It is also possible for the convergence to require more than the `ndeg` orders of blocking to occur. In which case (and assuming a larger value of `ndeg` is impractical), the final column will give a *lower bound* on the error. Note that using a larger value of `ndeg` may not be possible, either because there are insufficient configurations available, or the required memory exceeds the computer - MSDBLK can be a very memory hungry program.

The sampling interval `isampl` is usually set to 1, which means that every configuration in the HISTORY file will be used to calculate the MSD. If it is set to a larger number, the intervening configurations will be ignored. (Note that the time interval `tstep` specified in the `blk_input` file is the time between HISTORY file configurations, and is unaffected by the value of `isampl`.) The true time interval between the sampled configurations is thus `isampl × tstep`.

The origin interval `iomsd` is usually set to 1, which means that every *sampled* configuration is used as an origin in the MSD calculation. If a larger number is used, then the intervening sampled configurations are not used as origins, though they will be used to calculate the MSD. The time interval between origins is thus given by `iomsd × isampl × tstep`.

7.2.10 MSDPRG

7.2.10.1 Header records

program msdprg

```

C*****
C
C      dl_poly program to calculate mean square displacement
C      for selected atoms from dl_poly HISTORY file
C
C      copyright daresbury laboratory 1996
C      author  w.smith jan 1996
C
C*****

```

7.2.10.2 Function

MSDPRG calculates the Mean Square Displacement (MSD) for a nominated atom type, using data stored in the HISTORY file. It can read both formatted and unformatted versions of the HISTORY file.

7.2.10.3 Dependencies

- HREAD
- UREAD
- INVERT

7.2.10.4 Parameters

integers:

mxatms	1080	Maximum number of atoms in MD cell
npts	600	number of data points in MSD array
nstr	1080	Maximum of atoms of type selected

7.2.10.5 Input

Data Files:

- filename: msd_input

This is a free formatted file.

record 1:		
atname	char*8	name of atom of interest
record 2:		
fname	char*40	working name of HISTORY file
record 3:		
fmt	char*1	specified format of HISTORY file (U or F)
record 4:		
natms	integer	number of atoms in MD cell
record 5:		
tstep	real	time interval between configurations (ps)
record 6:		
nconf	integer	number of configurations in HISTORY file
record 7:		
nmsd	integer	required number of data points in MSD
record 8:		
isampl	integer	sampling interval for selecting configurations
record 9:		
iomsd	integer	origin interval between selected configurations
record 10-18:		
cell(1,...,9)	real	MD cell vectors in order A,B,C (Å)

• **filename: HISTORY**

This is the trajectory file produced by a DL_POLY_2 run. Note that the working name of the trajectory file need not be HISTORY. The actual name is specified in record 2 of the msd_input file.

7.2.10.6 Output

Data Files:

• **filename: msd_output**

This is a formatted file. The output consists of the following:

- a summary of the input data specified in the msd_input file, followed by:
- two columns of data (format: 1p,2e15.7) first column is time, second column is the MSD. The former is in the same units as the **tstep** variable in the msd_input file, the second is in Å².

7.2.10.7 Comments

MSDPRG can handle either formatted or unformatted HISTORY files. The choice is made by specifying either 'U' or 'F' as the **fmt** variable on record 3 of the msd_input file.

The sampling interval `isampl` is usually set to 1, which means that every configuration in the HISTORY file will be used to calculate the MSD. If it is set to a larger number, the intervening configurations will be ignored. (Note that the time interval `tstep` specified in the `msd_input` file is the time between HISTORY file configurations, and is unaffected by the value of `isampl`.) The true time interval between the sampled configurations is thus `isampl × tstep`.

The origin interval `iomsd` is usually set to 1, which means that every *sampled* configuration is used as an origin in the MSD calculation. If a larger number is used, then the intervening *sampled* configurations are not used as origins, though they will be used to calculate the MSD. The time interval between origins is thus given by `iomsd × isampl × tstep`.

7.2.11 SLOWMO

7.2.11.1 Header records

```

        program slowmo
c
c*****
c
c    program to process dl_poly trajectories, stripping out
c    selected frequencies
c
c    copyright daresbury laboratory 1993
c    author    w.smith    october    1993
c
c*****
c

```

7.2.11.2 Function

SLOWMO reads the DL_POLY_2 HISTORY file and performs a temporal Fourier transform (FT) on the atomic positions. This allows the user to select a particular frequency bandwidth in the frequency domain and set all other frequencies amplitudes to zero. On completing the inverse FT, the resulting trajectory when viewed graphically (as in a movie for example) will exhibit the motion of the molecules at the chosen frequency. It can be very effective at isolating particular configurational oscillations.

7.2.11.3 Dependencies

- NONE

7.2.11.4 Parameters

integers:

mxatms	200	Maximum number of atoms in HISTORY file
mxconf	200	Maximum number of configurations in HISTORY file

7.2.11.5 Input

Data Files:

- standard input

The input records are as follows:

record 1:

source	char*40	working name of HISTORY file
record 2:		
natms	integer	number of atoms in a configuration
record 3:		
nzapp	integer	'zapp' number - see comments below

- **filename: HISTORY**

This is the configuration file for a normal DL_POLY_2 run. Note that the working name of the trajectory file need not be HISTORY. The actual name is specified in standard input, record 1.

7.2.11.6 Output

Data Files:

- **filename: pivot.01**

This is an unformatted file in which the coordinates of all the atoms in a configuration are written out as a single record. The file is used to 'pivot' the configuration data for the Fourier transforms.

- **filename: pivot.02**

This is an unformatted file in which the processed coordinates of individual atoms are written. One Cartesian component for the full time history of one atom is written on each record.

- **filename: NEW_HISTORY**

This is a formatted file, written in the form of a DL_POLY_2 HISTORY file, containing the coordinates after the deletion of the unwanted frequencies. (Note the suffix 'HISTORY' will be replaced by the working name for the HISTORY file specified in standard input.)

- **filename: DIS_HISTORY**

This is a formatted display version for the processed HISTORY file, which may be written in a form the user desires for graphical processing. The format produced by SLOWMO is standard XYZ form. (Note the suffix 'HISTORY' may change, as above.)

7.2.11.7 Comments

The so-called 'zapp number' described in standard input (variable **nzapp**) simply specifies which parts of the Fourier transform are to be retained in the processing. To be precise:

if the time interval between configurations is defined as (say) **tstep**, then the Fourier transform time window T is **mxconf**×**tstep**. This means that the data interval in the frequency domain is $1/T$ and the frequency window is $-\text{mxconf}/(2T)$ to $\text{mxconf}/(2T)$. Thus a zapp number of **nzapp**, will retain all frequencies from $-\text{nzapp}/(2T)$ to $\text{nzapp}/(2T)$ and all others will be set to zero amplitude. Note that retaining the lower frequencies is most useful for graphical processing purposes, but it is an easy matter to amend this to select other frequencies instead.

Note that the dimension of the Fourier transform is given by the parameter **mxconf**, which must therefore be represented by a number of the form 2^n . Also, if the number of configurations in the HISTORY file is less than **mxconf**, the FFT arrays are padded with zeros. (This is standard practice). Do not be confused by the appearance of negative frequencies in the FFT, this is perfectly normal for a complex to complex FFT. Be careful not to set **nzapp** greater than **mxconf**/2.

Users are warned that SLOWMO is a huge consumer of disc space and computer memory. It may be an idea to produce one of the files NEW_HISTORY or DIS_HISTORY, rather than both. Users are also recommended to get hold of a machine specific FFT routine rather than the one supplied, which could speed things up dramatically. Note that SLOWMO uses a special trick to Fourier transform two real arrays at the same time. Take care not to destroy this advantage.

7.2.12 SMOOTH

7.2.12.1 Header records

program smooth

```

C*****
C
C      dl_poly utility for smoothing data in "xvgr" type file
C      format  x,y data sets separated by &
C      lines beginning "#" = comments
C      lines beginning "@TYPE xy" => x,y data follow on following lines
C
C      smoothing algorithm from Allen and Tildesley, p204
C      3rd degree, five point smooth.
C
C      copyright daresbury laboratory 1996
C      author t.forester      March 1996
C
C*****

```

7.2.12.2 Function

SMOOTH smooths a function by bestfitting a cubic functions to sub groups of 5 points. The points must be evenly spaced. The (x,y) data are supplied as columns of numbers in the input file whose name is requested from interactive input. The first column is x values which are assumed, but not checked, to be evenly spaced. The second column is the value of the function at the corresponding value of x. The data should be preceeded by a line

@TYPE xy

More than one data set can be contained in a file. Each set should be separated by a line containing an ampersand ("&") in the first column.

The program also prompts the user for the number of smoothing iterations required. The minimum value is 1. The smoothed output from one iteration is used as input for the next smoothing iteration.

7.2.12.3 Dependencies

- none

7.2.12.4 Parameters

integers:

mxn	10000	maximum length of data arrays

7.2.12.5 Input

Interactive:

record 1:

name	character	name of input file

record 2:

name1	character	name of output file

records 3:

nsmooth	integers	numbers of smoothing iterations

Data Files:

input:

File comprised of N records ($N < 10000$) with contents as noted above.

7.2.12.6 Comments

Several iterations (ie `nsmooth` > 1) may be required to give optimal results. However excessive smoothing can also remove functional features in addition to statistical noise.

7.2.13 SPECTRUM

7.2.13.1 Header records

```

        program spectrum
c
c*****
c
c      dl_poly utility program for spectral analysis using the fast
c      fourier transform
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith june 1992.
c
c      note - this program includes a naff fft routine purely for
c      portability and verification reasons. users are strongly
c      recommended to substitute a machine specific fft, which
c      should vastly improve performance.
c
c      note - you will probably need to change some of the array
c      dimensions using the following parameters.
c
c      note - ndiv must be a power of 2. ndiv2 will change according to
c      the fft routine you use.
c
c*****
c

```

7.2.13.2 Function

SPECTRUM calculates the discrete Fourier transform of a number of data columns. The data are supplied as columns of numbers in the input file 'input.data'. Execution is interactive, the user is first required to enter the number of transforms desired (max 10), the number of data columns in the input file (max 15) and the stride index (i.e. the number of records between each data point - not all points in the input need be used). Next the user must enter the numbers identifying the columns to be transformed. The columns may be up to 8192 entries in length.

Each Fourier transform is output in its own distinct file, with an appropriate name (e.g. ft06 for the transform of the 6'th column).

7.2.13.3 Dependencies

- FFT (use machine specific version if available).

7.2.13.4 Parameters

integers:

<code>ndiv</code>	8192	maximum length of data arrays
<code>ndiv2</code>	16384	maximum length of FFT arrays
<code>mxcols</code>	15	maximum number of data columns in input file
<code>mxcorr</code>	10	maximum number of transforms permitted

7.2.13.5 Input

Interactive:

record 1:

<code>nspecs</code>	integer	number of Fourier transforms (max 10)

record 2:

<code>ncols</code>	integer	number of data columns in input file (max 15)

record 3:

<code>nstride</code>	integer	integer stride through data columns

records 4 to `nspecs+3`:

<code>nc1 nc2</code>	integers	id numbers of columns to be transformed

Data Files:

input.data:

File comprised of N records ($N < 8192$) with contents:

`col1, col2, ..., colncols`

where `coli` is an entry for the *i*'th data column. (The data are in free format.)

7.2.13.6 Comments

This program uses a Blackman-Harris 4 term window function. If you don't like it, use your own!

7.2.14 STATBLOCK

7.2.14.1 Header records

```

        program statblock
c
c*****
c
c      blocking method for estimating standard error of mean
c
c      (reference - Flyvbjerg and Petersen  JCP 91 (1989) 461)
c
c      author - w. smith nov 1992
c
c*****
c

```

7.2.14.2 Function

statblock calculates the mean values of a set of data columns and performs a blocking analysis to determine the correct standard error. The method is that of Flyvbjerg and Petersen. Up to 20 columns of data can be processed at once. The data are supplied as columns of numbers in an input file. Execution is interactive, the user must first supply the name of the input file holding the data. Next the number of data columns in the input file (max 20) and the number of columns to be processed (max 20) are entered. Finally the user must enter the numbers identifying the columns selected for processing. The columns may be up to 10000 entries in length. Results are streamed to standard output.

7.2.14.3 Dependencies

- None

7.2.14.4 Parameters

integers:

ndeg	13	max degree of blocking to be performed
mxcols	20	maximum number of data columns in input file
mxrows	10000	maximum number of entries per data column

7.2.14.5 Input

Interactive:

record 1:

statfile	a40	name of data file to be processed

record 2:		
mcols	integer	number of data columns in data file
record 3:		
ncols	integer	number of columns to be processed
record 4:		
id ₁ ... id _{ncols}	integers	id numbers of columns to be processed

Data Files:

statfile:

File comprised of N records (N<10000) with contents:

col₁, col₂, ..., col_{ncols}

where col_i is an entry for the *i*'th data column. (The data are in free format.)

7.2.14.6 Comments

It is advisable to plot the calculated standard errors as a function of degree of blocking, to determine the optimal value.

7.2.15 VACF

7.2.15.1 Header records

program vacf

```

C*****
C
C      dl_poly routine to calculate velocity autocorrelation function
C      for selected atoms
C
C      copyright daresbury laboratory 1995
C      author w.smith jan 1995
C
C*****

```

7.2.15.2 Function

VACF calculates the velocity autocorrelation function (VACF) for a selected atom type, using data available in a formatted DL_POLY_2 HISTORY file. VACF also calculates the power spectrum of the VACF.

7.2.15.3 Dependencies

- FFT - fast Fourier transform routine.

7.2.15.4 Parameters

integers:

mxatms	1080	Maximum number of atoms in MD cell
nstr	250	Maximum number of atoms of selected type
npts	128	Number of points in time correlation

7.2.15.5 Input

Data Files:

- standard input

The input consists of the following:

record 1:

natms	integer	number of atoms in MD cell

record 2:

nvacf	integer	working length of vacf array

record 3:		
<i>iovacf</i>	integer	origin interval between selected configurations
record 4:		
<i>tstep</i>	real	time interval between configurations (ps)
record 5:		
<i>atname</i>	char*8	name of atom in correlation
record 6:		
<i>fname</i>	char*40	working name of HISTORY file
record 7:		
<i>nconf</i>	integer	number of configurations in HISTORY file

• filename: HISTORY

This is the formatted trajectory file produced by a DL_POLY_2 run. Note that the working name of the trajectory file need not be HISTORY. The actual name is specified in record 6 of the standard input file.

7.2.15.6 Output

Data Files:

• standard output

The output consists of the following:

- a summary of the input data specified in the standard input file, plus any error or warning messages followed by:
- the VACF for the atom type chosen, plus its accumulated sum (i.e. integral.) as functions of time.
- the Fourier transform of the VACF as a function of frequency.

7.2.15.7 Comments

The VACF produced by VACF is normalised. That is, the value of the VACF is unity at time zero.

The number of “bins” in the time coordinate of the correlation function is given by the input variable *nvacf*. Note that this number must be a multiple of the interval between origins *iovacf*. If this is not so, VACF will reset it so it is.

Note that *tstep* is the time between successive configurations in the HISTORY file, and is an integer multiple of the molecular dynamics timestep.

The origin interval *iovacf* is usually set to 1, which means that every configuration in the HISTORY file is used as an origin in the VACF calculation. If a larger number is used,

then the intervening sampled configurations are not used as origins, though they will be used to calculate the VACF. The recommended value is 1.

The Fourier transform of the VACF is *real*.

7.3 Miscellaneous Utilities

7.3.1 BESTFIT

7.3.1.1 Header records

```
program bestfit
```

```
c*****
c
c      Least squares fitting of arbitrary functions to data
c      uses Gram-Schmidt Orthogonalisation procedure.
c
c      copyright daresbury laboratory 1993
c      author t. forester may 1993.
c
c*****
```

7.3.1.2 Function

The program best-fits a linear combination of M functions to N data points. The program will prompt for the following information from the standard input device:

- number of functions to use
- n data points as : x y

The program will generate a list of the best coefficients for the functions specified and a comparison for each data point of the input data and the best-fit value.

7.3.1.3 Dependencies

- SYMINV

7.3.1.4 Parameters

integers:

m1	20	maximum number of functions
n1	1000	maximum number of data points

7.3.1.5 Input

Interactive:

record 1:

m	integer	number of functions to use (max value is M1)

record 2:

x(1)	real	first x data point
y(1)	real	first y data point
record k+1:		
x(k)	real	<i>k</i> th x data point
y(k)	real	<i>k</i> th y data point
last record:		
-999	real	end of data list
-999	real	end of data list

7.3.1.6 Comments

The functions used are defined starting at line 53 of the source code. Change the definitions to suit and recompile before running.

The number of data points given must equal or exceed the number of functions used.

The program makes use of a version of the matrix inversion utility SYMINV.

7.3.2 CFGCHK

7.3.2.1 Header records

program cfgchk

```

C*****
C
C      dl_poly utility program for checking a CONFIG file to ensure
C      an absence of close or zero atomic contacts
C
C      author w.smith december 1998
C      copyright daresbury laboratory 1998
C
C*****

```

7.3.2.2 Function

CFGCHK scans through a nominated CONFIG file and reports the minimum separation between all atoms. It is intended to highlight unfavourable contacts (or overlap) between atoms in a system.

7.3.2.3 Dependencies

- INVERT

7.3.2.4 Parameters

- mxatms - MAXIMUM NUMBER OF ATOMS IN THE SYSTEM (30000)

7.3.2.5 Input

Data Files:

- standard input

The input required is as follows:

record 1:

filnam	character	name of CONFIG file to be analysed.

- filename: Nonimated CONFIG file (filnam)

7.3.2.6 Output

Data Files:

- **standard output**

The output consists of the following:

1. the coordinates and indices of any atomic pairs found to have zero separation;
2. a report of the minimum separation found.

7.3.2.7 Comments

7.3.3 CFGCMP

7.3.3.1 Header records

```

      program cfgcmp
c*****
c
c      dl_poly utility program for comparing two CONFIG files
c
c      author w.smith december 1998
c      copyright daresbury laboratory 1998
c
c*****

```

7.3.3.2 Function

CFGCCMP scans through two nominated CONFIG files and reports maximum differences found between them. It is intended to allow a low output comparison of the two (unlike the UNIX diff command).

7.3.3.3 Dependencies

- NONE

7.3.3.4 Parameters

- mxatms - MAXIMUM NUMBER OF ATOMS IN THE SYSTEM (13824)

7.3.3.5 Input

Data Files:

- standard input

The input required is as follows:

record 1:

file1	character	name of first CONFIG file to be analysed.
file2	character	name of second CONFIG file to be analysed.

- filename: Nonimated CONFIG file (file1)
- filename: Nonimated CONFIG file (file2)

7.3.3.6 Output

Data Files:

- **standard output**

The output consists of the following:

1. the maximum difference in cell vectors (a,b,c);
2. the maximum difference in coordinates (x,y,z);
3. the maximum difference in velocities (vx,vy,vz);
4. the maximum difference in forces (fx,fy,fz);

7.3.3.7 Comments

Can be used to find the maximum distance any atom has moved during a simulation.

7.3.4 DLPXYZ

7.3.4.1 Header records

```

      program dlpxyz
c*****
c
c      dl_poly program to convert CONFIG and REVCON files to
c      a standard XYZ file for the XMOL package
c
c      copyright daresbury laboratory 1997
c      author  w.smith april 1997
c
c*****

```

7.3.4.2 Function

DLPXYZ converts a nominated DL_POLY_2 CONFIG file into a standard XYZ file, suitable for graphics programs such as XMOL.

7.3.4.3 Dependencies

- STRIP

7.3.4.4 Parameters

- mxatms - MAXIMUM NUMBER OF ATOMS IN THE SYSTEM (2490)

7.3.4.5 Input

Data Files:

- standard input

The input required is as follows:

record 1:

fname	character	name of CONFIG file to be analysed.
natms	integer	number of atoms in the CONFIG file.

record 2:

- filename: Nominated CONFIG file (fname)

7.3.4.6 Output

Data Files:

- **XYZ.xyz**

The XYZ file corresponding to the CONFIG file nominated.

- **standard output**

The output consists of the following:

1. a report of the conversion process, plus any error messages.

7.3.4.7 Comments

The program attempts to rename the atoms to the corresponding proper chemical names. There is only a limited dictionary for this, and users is advised to look at the code and amend it to suit their needs.

7.3.5 FFT

7.3.5.1 Header records

```

      subroutine fft(ind,isw,ndiv,key,aaa,wfft,bbb)
c*****
c
c      fast fourier transform routine
c
c      copyright daresbury laboratory 1994
c
c      author w smith
c
c*****

```

7.3.5.2 Function

FFT performs a complex to complex Fast Fourier Transform (FFT) on N data points where $N = 2^m$ (i.e. a radix 2 FFT).

7.3.5.3 Dependencies

- none

7.3.5.4 Arguments

integers:

ind	input	Set 0 for FFT, 1 for initialisation of complex exponentials
isw	input	Sign of transform, as in $\exp(isw \times i2\pi ft)$
ndiv	input	Number of data points

integer arrays:

key	work	'reverse bit address' array, (FFT unscrambler) (ndiv)

real arrays:

aaa	input	<i>Complex</i> array to be Fourier transformed (ndiv)
wfft	work	Storage array for complex exponentials (ndiv)
bbb	output	<i>Complex</i> Fourier transform of array aaa (ndiv)

7.3.5.5 Comments

FFT must be called once with **ind**=1 to initialise the bit reverse address array and the complex exponentials. The input array *must* have 2^m data points. Arrays **aaa** and **bbb** may be the same array (with the result overwriting the input).

7.3.6 GEARPC

7.3.6.1 Header records

```

      subroutine gearpc
      x      (mode,idnode,mxnode,natms,imcon,engke,weight,cell,
      x      xxx,yyy,zzz,xx1,yy1,zz1,xx2,yy2,zz2,xx3,yy3,zz3,xx4,yy4,
      x      zz4,xx5,yy5,zz5,fx,fy,fz,buffer)

c
c*****
c
c      dl_poly subroutine for integrating newtonian equations of
c      motion in molecular dynamics - gear predictor corrector
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1993
c      author    - w. smith february 1993.
c
c*****
c

```

7.3.6.2 Function

The Gear 5'th order predictor corrector for second order linear differential equations, to integrate the Newtonian equations of motion.

7.3.6.3 Dependencies

- GDSUM
- IMAGES
- MERGE

7.3.6.4 Arguments

integers:

mode	input	predictor/corrector selection mode=1 for predictor step mode=2 for corrector step
idnode	input	identity number of node
mxnode	input	total number of nodes on the cube
natms	input	total number of sites in the simulated system.
imcon	input	periodic boundary condition key. See IMAGES

reals:

engke	output	kinetic energy

real arrays:

weight	input	atomic mass array (mxatms).
cell	input	cell vectors (9). See IMAGES.
xxx	in/out	x coordinate array (mxatms).
yyy	in/out	y coordinate array (mxatms)
zzz	in/out	z coordinate array (mxatms)
xx1	in/out	1st deriv term of xxx (mxatms).
yy1	in/out	1st deriv term of yyy (mxatms)
zz1	in/out	1st deriv term of zzz (mxatms)
xx2	in/out	2nd deriv term of xxx (mxatms).
yy2	in/out	2nd deriv term of yyy (mxatms)
zz2	in/out	2nd deriv term of zzz (mxatms)
xx3	in/out	3rd deriv term of xxx (mxatms).
yy3	in/out	3rd deriv term of yyy (mxatms)
zz3	in/out	3rd deriv term of zzz (mxatms)
xx4	in/out	4th deriv term of xxx (mxatms).
yy4	in/out	4th deriv term of yyy (mxatms)
zz4	in/out	4th deriv term of zzz (mxatms)
xx5	in/out	5th deriv term of xxx (mxatms)
yy5	in/out	5th deriv term of yyy (mxatms)
zz5	in/out	5th deriv term of zzz (mxatms)
fxx	input	x component of force array (mxatms)
fyy	input	y component of force array (mxatms)
fzz	input	z component of force array (mxatms)
buffer	work	buffer array (mxbuff). See MERGE

7.3.6.5 Comments

Calling this routine with **mode**=1 results on the predictor step being performed. Calling with **mode**=2 causes the corrector step to be performed.

7.3.7 HISXYZ

7.3.7.1 Header records

```

      program hisxyz
c*****
c
c      dl_poly program to convert HISTORY files to
c      a standard XYZ file for the XMOL package
c
c      copyright daresbury laboratory 1997
c      author w.smith june 1997
c
c*****

```

7.3.7.2 Function

HISXYZ converts a nominated DL_POLY_2 HISTORY file into a standard XYZ format file, suitable for graphics programs such as XMOL.

7.3.7.3 Dependencies

- STRIP
- WHATNAME

7.3.7.4 Parameters

- **mxatms** - MAXIMUM NUMBER OF ATOMS IN THE SYSTEM (2490)

7.3.7.5 Input

Data Files:

- **standard input**

The input required is as follows:

record 1:

fname	character	name of HISTORY file to be analysed.	record 2:
nconf	integer	number of configurations in the HISTORY file.	record 3:
natms	integer	number of atoms in the HISTORY file.	

- **filename:** Nominated HISTORY file (**fname**)

7.3.7.6 Output

Data Files:

- **HIS.xyz**

The XYZ file corresponding to the HISTORY file nominated.

- **standard output**

The output consists of the following:

1. a report of the conversion process, plus any error messages.

7.3.7.7 Comments

The program attempts to rename the atoms to the corresponding proper chemical names. There is only a limited dictionary for this, and users is advised to look at the subroutine WHATATMS within and amend it to suit their needs.

7.3.8 JACOBI

7.3.8.1 Header records

```

      subroutine jacobi(a,v,n)
c
c*****
c
c      diagonalisation of real symmetric matrices by jacobi method
c
c      input parameters:
c
c      a(n,n) is the matrix to be diagonalised
c      v(n,n) is the eigenvector matrix
c      n   is the dimension of the matrices
c
c      jacobi processes lower triangle only (upper triangle unchanged)
c
c      variable rho sets absolute tolerance on convergence
c      variable tes is a moving tolerance that diminishes
c      on each pass until at true convergence tes<rho
c
c      copyright daresbury laboratory 1993
c      author w.smith 1993
c
c*****
c

```

7.3.8.2 Function

Diagonalises a real symmetric matrix.

7.3.8.3 Dependencies

- none

7.3.8.4 Arguments

integers:

n input dimension of matrix

real arrays:

a input real symmetric matrix

v output eigenvectors of matrix

7.3.8.5 Comments

7.3.9 Useful Macros

7.3.9.1 Macros

Macros are simple executable files containing standard unix commands. A number of the are supplied with DL_POLY and are found in the *execute* sub-directory. The available macros are as follows.

- *cleanup*
- *copy*
- *gopoly*
- *gui*
- *select*
- *store*

The function of each of these is described below. It is worth noting that most of these functions can be performed by the DL_POLY_2 GUI [20] or Java GUI 5.1.

7.3.9.2 *cleanup*

cleanup removes several standard data files from the *execute* sub-directory. It contains the unix commands:

```
rm OUTPUT REVCON REVOLD STATIS REVIVE gopoly.*
```

and removes the files OUTPUT, REVCON, REVOLD, STATIS, REVIVE and gopoly.* (all variants). It is useful for cleaning the sub-directory up after a run. (Useful data should be stored elsewhere however!)

7.3.9.3 *copy*

copy invokes the unix commands:

```
mv CONFIG CONFIG.OLD  
mv REVCON CONFIG  
mv REVIVE REVOLD
```

which collectively prepare the DL_POLY files in the *execute* sub-directory for the continuation of a simulation. It is always a good idea to store these files elsewhere in addition to using this macro.

7.3.9.4 *gopoly*

gopoly is used to submit a DL_POLY job to the Daresbury IBM SP/2, which operates a LOADLEVELLER job queuing system. It invokes the following script.

```
#@ min_processors = 4
#@ max_processors = 4
#@ job_type = parallel
#@ requirements = (Adapter == "hps_ip") && ( Pool == 2)
#@ executable = /usr/bin/poe
#@ cpu_limit = 00:10:00
#@ arguments = ~/dl_poly_2.10/execute/DLPOLY.X -euilib ip
#@ output = gopoly.o
#@ error = gopoly.e
#@ class = dev
#@ queue
```

Using LOADLEVELLER, the job is submitted by the unix command:

llsubmit gopoly

where *llsubmit* is a local command for submission to the SP/2. The number of required nodes and the job time are indicated in the above script.

7.3.9.5 *gui*

gui is a macro that starts up the DL_POLY_2 Java GUI. It invokes the following unix commands:

```
java -jar ../java/GUI.jar
```

In other words the macro invokes the Java Virtual Machine which executes the instructions in the Java archive file GUI.jar, which is stored in the *java* subdirectory of DL_POLY_2. (Note: Java 1.3.0 or a higher version is required to run the GUI.)

7.3.9.6 *select*

select is a macro enabling easy selection of one of the test cases. It invokes the unix commands:

```
cp ../data/TEST$1/CONTROL CONTROL
cp ../data/TEST$1/FIELD FIELD
cp ../data/TEST$1/CONFIG CONFIG
```

select requires one argument (an integer) to be specified:

select n

where n is test case number, which ranges from 1 to 10.

This macro sets up the required input files in the *execute* sub-directory to run the n -th test case.

7.3.9.7 *store*

The *store* macro provides a convenient way of moving data back from the *execute* sub-directory to the *data* sub-directory. It invokes the unix commands:

```
mkdir ../data/TEST$1
mv OUTPUT ../data/TEST$1/OUTPUT
mv REVCON ../data/TEST$1/REVCON
mv STATIS ../data/TEST$1/STATIS
mv REVIVE ../data/TEST$1/REVIVE
mv RDFDAT ../data/TEST$1/RDFDAT
mv ZDNDAT ../data/TEST$1/ZDNDAT
chmod 400 ../data/TEST$1/*
```

which first creates a new DL_POLY *data/TEST..* sub-directory and then moves the standard DL_POLY output data files into it.

store requires one argument:

store n

where n is a unique string or number to label the output data in the *data/TEST n* sub-directory.

Note that *store* sets the file access to read-only. This is to prevent the *store* macro overwriting existing data without your knowledge.

7.3.10 PARSET

7.3.10.1 Header records

```

      program parset

c
c*****
c
c      dl_poly utility program to prepare the dl_params.inc file
c      for specific dl_poly applications
c
c      author - w.smith, t.forester jan 1995
c      copyright daresbury laboratory 1995
c
c*****
c

```

7.3.10.2 Function

PARSET is designed to provide estimates of the FORTRAN parameters (e.g. array dimensions) required by versions of DL_POLY_2 prior to 2.11 to simulate a given system i.e. those required for the `dl_params.inc` file. By scanning the CONTROL, FIELD and CONFIG files of the intended simulation PARSET is able to calculate estimates of minimum parameters needed. The calculated parameters are written into a file: *new_params.inc*, which is FORTRAN compatible. Once this has been renamed `DL_PARAMS.INC`, it can be used directly when DL_POLY_2 is compiled. Use of PARSET is strongly recommended as a means of reducing the effort needed to create a working DL_POLY_2 executable.

Note. PARSET is required only for versions of DL_POLY_2 prior to version 2.11.

7.3.10.3 Dependencies

- None (PARSET is self contained.)

7.3.10.4 Parameters

- None (There are no internal arrays.)

7.3.10.5 Input

Interactive:

record 1:

<code>min_nodes</code>	integer	minimum number of processing nodes

record 2:

<code>max_nodes</code>	integer	maximum number of processing nodes

Data Files:**input.data:**

CONTROL	Standard DL_POLY_2 CONTROL input file
CONFIG	Standard DL_POLY_2 CONFIG input file
FIELD	Standard DL_POLY_2 FIELD input file
TABLE	Standard DL_POLY_2 TABLE input file (optional)

output.data:

<i>new_params.inc</i>	New parameters include file

7.3.10.6 Comments

PARSET does not undertake any error checking of the CONTROL, FIELD and CONFIG files. Any errors in these files will result in an incorrect parameters file. It must also be recognised that for some of the parameters, there is no straightforward formula for calculating suitable values from the input files. In these cases the parameters produced represent a reasonable estimate only. Fortunately, these cases are few in number and as a rule PARSET will significantly reduce the time needed to prepare a working version of DL_POLY_2.

It is useful to prepare the input files assuming the largest simulation you are likely to attempt on a given system. This will generate the largest executable necessary to prevent you having to recompile the code in the course of your study.

7.3.11 ROTATE

7.3.11.1 Header records

```
      subroutine rotate(xo,yo,zo,x1,y1,z1,x2,y2,z2,alp
      x      ,bet,gam)
```

```
C*****
C
C      DL_POLY utility to rotate water molecule through the
C      Euler angles (alp,bet,gam) in lab fixed axes system.
C      The first data point is taken as the origin for the rotation
C
C      copyright daresbury laboratory 1993
C      author - t. forester      july 1993
C
C*****
```

7.3.11.2 Function

Creates a water molecule with the dipole direction along the x axis then rotates it through the Euler angles (α, β, γ) in the laboratory fixed frame.

7.3.11.3 Dependencies

- none

7.3.11.4 Arguments

reals:

xo	input	x coordinate of oxygen
yo	input	y coordinate of oxygen
zo	input	z coordinate of oxygen
x1	output	x coordinate of first hydrogen
y1	output	y coordinate of first hydrogen
z1	output	z coordinate of first hydrogen
x2	output	x coordinate of second hydrogen
y2	output	y coordinate of second hydrogen
z2	output	z coordinate of second hydrogen
alp	input	α (in degrees)
bet	input	β (in degrees)
gam	input	γ (in degrees)

7.3.11.5 Comments

7.3.12 SOLVE

7.3.12.1 Header records

```

      subroutine solve(ndim,lpv,aaa,xxx,bbb)
c
c*****
c
c      gaussian elimination routine with pivoting
c
c      copyright daresbury laboratory 1987
c      w. smith aug. 1987
c
c*****
c

```

7.3.12.2 Function

SOLVE is a no-frills gaussian elimination routine with pivoting, for solving a system of linear equations. i.e. solves the equation:

$$[A][x] = [b] \quad (7.1)$$

7.3.12.3 Dependencies

- none

7.3.12.4 Arguments

integers:

ndim	input	the dimension of the arrays and number of unknowns.

integer arrays:

lpv	work	Pivot unscrambler - record of pivots (ndim)

real arrays:

aaa	input	Matrix of coefficients (ndim, ndim)
xxx	output	Solution array (ndim)
bbb	input	Array of coefficients (ndim)

7.3.12.5 Comments

SOLVE overwrites the contents of the matrix $[A]$ and the vector $[b]$, so keep back up copies if needed.

7.3.13 SORT

7.3.13.1 Header records

```

      subroutine sort(n,index,x)
c
c*****
c
c      DL_POLY utility
c      shell sort of index array
c
c      copyright daresbury laboratory 1993
c      author -      t.forester may 1993
c
c*****
c

```

7.3.13.2 Function

Sorts *n* real indexed numbers into ascending order.

7.3.13.3 Dependencies

- none

7.3.13.4 Arguments

integers:

<i>n</i>	input	number of items to be sorted

integer arrays:

<i>index</i>	output	index of sorted list

real arrays:

<i>x</i>	input	numbers to be sorted

7.3.13.5 Comments

The sort is based on the *index* not the array itself. The array *X* is unchanged on exit from this routine. Thus *X*(INDEX(1)) is the smallest number in the array and *X*(INDEX(*N*)) the largest.

7.3.14 SYMINV

7.3.14.1 Header records

```

      subroutine syminv(nnn,aaa,bbb)
c
c*****
c
c      routine to invert a real symmetric matrix (reference:
c      computing methods v.ii, i.s. berezin and n.p.zhidkov,
c      pergamon press 1965). note that the matrices are
c      packed in the minimum storage mode, (i.e. a(k)=a(i,j),
c      where i>j and k=(i*(i-1))/2+j ).
c      the matrices aaa and bbb may be equivalenced though
c      this will destroy the contents of the original
c      array.
c
c      general version for all real symmetric matrices
c
c      copyright daresbury laboratory 1993
c      author w.smith (added to dl_poly july 1993)
c
c*****
c

```

7.3.14.2 Function

Finds the inverse of a real symmetric matrix.

7.3.14.3 Dependencies

- none

7.3.14.4 Arguments

integers:

nnn input dimension of matrix

real arrays:

aaa input matrix in triangular form (*)

bbb output inverse matrix in triangular form (*)

7.3.14.5 Comments

The program makes use of minimum storage requirements for the matrices. The array AAA has $NNN(NNN+1)/2$ entries defining the lower triangle of the real symmetric matrix. The

inverse matrix $\mathbf{B}^{-1}$ is given in the same form.

7.3.15 TABCHK

7.3.15.1 Header records

```

      program tabchk
c
c*****
c
c      dl_poly utility to check accuracy of tabulation procedures
c
c      copyright daresbury laboratory 1994
c      author -      t.forester october 1994
c
c*****
c

```

7.3.15.2 Function

The program tests the accuracy of the tabulation procedures in DL_POLY prior to a simulation. The tabulation schemes are 3-point and 4-point interpolation in R space and 2 point interpolation in R^2 space.

The program outputs the maximum, the mean and the variance in the deviations between the interpolated values and exact values. This is done for the absolute and relative deviations in both the energy and the forces. Thus a total of 12 numbers are reported for each potential function.

7.3.15.3 Dependencies

- none

7.3.15.4 Parameters

integers:

mega	10000	maximum number of records in FIELD file
mxvdw	1000	maximum number of potential functions
mxpvdw	5	maximum number of variables to describe potential functions

7.3.15.5 Input

Interactive:

record 1:

ngrid	integer	tabulation grid size

record 2:

rcut	real	proposed cutoff

record 3:

rmin	real	minimum distance for function evaluation

record 4:

npts	integer	number of test points to use

Data Files:**FIELD**

FIELD file as per standard input for DL_POLY

7.3.15.6 Comments

The best choice of tabulation procedure is the one that gives you sufficient accuracy within your memory requirements. The key test is probably the mean relative error in the energy and forces. This should be about 10^{-5} or less. In general four point scheme is more accurate than the three point which in turn is more accurate than the R^2 based procedure. However in terms of computational efficiency the order is reversed: the R^2 scheme is the quickest and the four point scheme the slowest.

Increasing the number of grid points will improve the accuracy of all schemes. We recommend that the number of test points to be used should be non-commensurate with the number of grid points. A large prime number (*e.g.* 2311) seems to work well.

The program also requires a minimum distance, **rmin**, to start testing from as at very small separations the pair forces and energies, in general, rise very rapidly and any interpolation scheme becomes suspect. The user should choose **rmin** to be safely below the smallest pair separation expected in a simulation.

Rerun the program several times varying the number of grid points, **ngrid** and the minimum cutoff **rmin**. Once a suitable tabulation scheme and number of grid points has been found, adjust the **mxgrid** parameter in the DL_PARAMS.INC file (in the *source* directory and “make” DL_POLY with the appropriate tabulation directive. See chapter 3 for instructions on compiling the code.

Chapter 8

DL_POLY_2 Subroutines and Functions

Scope of Chapter

This chapter describes all the subroutines of DL_POLY_2 to be found in the *source* and *respa* subdirectories.

8.1 Subroutine and Function Specifications

8.1.1 DL_PARAMS.INC: The DL_POLY Parameters (Include) File

8.1.1.1 Header

```

C*****
C
C    dl_poly insert file specifying fundamental parameters for the
C    entire package
C
C    copyright - daresbury laboratory 1997
C    authors - w. smith
C
C
C    note the following internal units apply everywhere
C
C    unit of time      (to)    =          1 x 10**(-12) seconds
C    unit of length    (lo)    =          1 x 10**(-10) metres
C    unit of mass      (mo)    = 1.6605402 x 10**(-27) kilograms
C    unit of charge    (qo)    = 1.60217733 x 10**(-19) coulombs
C    unit of energy     (eo)    = 1.6605402 x 10**(-23) joules
C    unit of pressure  (po)    = 1.6605402 x 10**( 7) pascals
C                                = 163.842151          atmospheres
C
C*****
C

```

8.1.1.2 Function

The DL_POLY_2 parameters file for version 2.10 (and earlier) contains all the parameters defining the arrays dimensions of DL_POLY_2 plus the fundamental constants and conversion factors. In the include file for version 2.11 (and after) the bulk of these parameters take the form of FORTRAN integer variables, which are stored in the COMMON block /params/. Apart from this difference, the meaning and function of the parameters is the same. The DL_PARAMS.INC file is included in all relevant subroutines at compile time.

8.1.1.3 The Parameters and their Function

In the following table the values of parameters that are dependent on the simulated system are given as *variable*. Some parameter values may be derived from others, and a simple formula is given. An example of a working parameters file can be found in the *source* subdirectory.

parameter value	function

boltz	0.831451115	boltzmann constant in internal units
kmaxa	<i>variable</i>	max reciprocal space vector index (a direction)
kmaxb	<i>variable</i>	max reciprocal space vector index (b direction)
kmaxc	<i>variable</i>	max reciprocal space vector index (c direction)
kmaxd	<i>variable</i>	SPME FFT array dimension (a direction)
kmaxe	<i>variable</i>	SPME FFT array dimension (b direction)
kmaxf	<i>variable</i>	SPME FFT array dimension (c direction)
minnode	<i>variable</i> ≥ 1	min number of nodes for code execution
msatms	$\text{mxatms}/\text{minnode}+1$	max number of atoms in working arrays
msbad	$\max(\text{mxbond}, \text{mxangl}, \text{mxdihd}, \text{mxteth}, \text{mxinv}, \text{mxshl})$	max size of working arrays for bond, angle, dihedral, tether, inversion and shell routines
msggrp	$\text{mxgrp}/\text{minnode}+1$	max number of rigid groups per node
mspmf	<i>variable</i>	max number of potential of mean force constraints per node
msteth	<i>variable</i>	max number of tethered atoms per node
mx2tbp	<i>variable</i> $\leq \text{mxvdw}$	array dimension of 3-body potential parameters
mx3fbp	<i>variable</i>	array dimension of 4-body potential parameters
mxangl	<i>variable</i>	max number of bond angles on a node
mxatms	<i>variable</i>	max number of atoms in system
mxbond	<i>variable</i>	max number of chemical bonds on a node
mxbuff	$\max(6*\text{mxatms}, 8*(\text{mxcons}+1), 8*(\text{mxgrp}+1), \text{mxnstk}*\text{mxstak}, \text{mxebuf}, \text{mxgrid})$	max dimension of atomic transfer buffer
mxcell	<i>variable</i>	max number of link cells in system
mxcons	<i>variable</i>	max number of constraint bonds on a node
mxdihd	<i>variable</i>	max number of torsion angles on a node
mxegrđ	<i>variable</i>	max number of grid points in HKE interpolation arrays
mxewld	msatms or mxatms	max of array elements in EWALD1(A) work arrays
mxexcl	<i>variable</i>	max number of excluded interactions per atom
mxfbd	<i>variable</i>	max number of defined 4-body force potentials
mxfld	10	max number of external field parameters
mxftab	<i>variable</i>	dimension of <code>fftttable</code> array ($2*(\text{kmaxd}+\text{kmaxe}+\text{kmaxf})$)
mxgatm	<i>variable</i> $\leq \text{mxatms}$	max number of sites in rigid units
mxgrid	<i>variable</i>	max number of grid points in potential arrays
mxgrp	<i>variable</i>	max number of rigid body units in system
mxhke	<i>variable</i>	dimension of HK-Ewald work arrays
mxhko	<i>variable</i>	max order of HK-Ewald convergence function
mxinv	<i>variable</i>	max number of inversion potentials per node
mxlist	<i>variable</i>	max number of atoms in verlet list
mxlshp	$2*\text{mxcons}$	max dimension of shape routine transfer array
mxmols	<i>variable</i>	max number of molecules

<code>mxneut</code>	<i>variable</i>	max number of neutral groups + 1
<code>mxngp</code>	<i>variable</i> ≥ 3	max number of sites in a rigid unit
<code>mxnstk</code>	<code>mxsvdw</code> +45	max number of stacked variables
<code>mxpang</code>	4	max number of angle potential parameters
<code>mxpbnd</code>	4	max number of bond potential parameters
<code>mxpdih</code>	5	max number of dihedral potential parameters
<code>mxpfbp</code>	3	max number of 4-body potential parameters
<code>mxpinv</code>	2	max number of inversion potential parameters
<code>mxpmf</code>	<i>variable</i> $\leq$ <code>mxatms</code>	number of atoms in potential of mean force constraints
<code>mxproc</code>	<i>variable</i>	max number of nodes (used in parallel constraint algorithm)
<code>mxptbp</code>	<code>mxpang</code> +1	max number of 3-body potential parameters
<code>mxpvdw</code>	5	max number of van der Waals potential parameters
<code>mxquat</code>	<i>variable</i>	max iterations in quaternion integration
<code>mxrdf</code>	<i>variable</i>	number of tabulation points for radial distribution functions
<code>mxshak</code>	<i>variable</i>	max number of iterations in shake algorithm
<code>mxshl</code>	<i>variable</i>	max number of core-shell units
<code>mxsite</code>	<i>variable</i>	max number of molecular sites
<code>mxspme</code>	<i>variable</i>	max number of charged atoms in SPME
<code>mxspl</code>	<i>variable</i>	max order of B-spline function in SPME
<code>mxspmf</code>	<i>variable</i>	max number sites to define pmf units
<code>mxstak</code>	<i>variable</i>	dimension of stack arrays for rolling averages
<code>mxsvdw</code>	<i>variable</i>	max number of different types of sites for pair potentials
<code>mxtang</code>	<i>variable</i>	max number of different bond angle potentials
<code>mxtbnd</code>	<i>variable</i>	max number of chemical bond potentials
<code>mxtbp</code>	<code>mxvdw</code> * <code>mxsvdw</code>	max number of 3-body potentials
<code>mxtcon</code>	<i>variable</i>	max number of specified bondlength constraints
<code>mxt dih</code>	<i>variable</i>	max number of different dihedral potentials
<code>mxteth</code>	<i>variable</i>	max number of tethered atom potentials
<code>mxtinv</code>	<i>variable</i>	max number of inversion potentials
<code>mxtmls</code>	<i>variable</i>	max number of molecule types
<code>mxtshl</code>	<i>variable</i>	max number of core-shell unit types
<code>mxungp</code>	<i>variable</i>	max number unique rigid body units
<code>mxvdw</code>	$((\text{mxsvdw}+1)*\text{mxsvdw})/2$	max number of different pair potentials ($\times 2$ if Sutton-Chen potentials used)
<code>mxxdf</code>	<code>max(mxlist,msatms, mxcons,mxn1*mxn1*(mxneut+1)/2)</code>	max number of atoms in xdf,ydf and zdf arrays (<code>mxn1</code> is number of sites in largest neutral group)
<code>nconf</code>	8	configuration file input channel
<code>ndump</code>	<i>variable</i>	data dumping interval in event of system crash
<code>nfield</code>	7	force field input channel

<code>nhist</code>	21	trajectory history file channel
<code>nrdfdt</code>	24	output channel for RDF data
<code>nread</code>	5	main input channel
<code>nrest</code>	22	output channel accumulators restart dump file
<code>nrite</code>	6	main output channel
<code>nstats</code>	20	statistical data file output channel
<code>ntable</code>	23	tabulated potentials file input channel
<code>nzdfdt</code>	25	output channel for Z-density data file
<code>pi</code>	3.141592653589793	π
<code>prsunt</code>	1.63842151d-1	pressure conversion factor
<code>r4pie0</code>	138935.4835	electrostatic conversion factor i.e. (unit(charge)**2/(4 pi eps0 unit(length)))/ unit(energy)
<code>sqrpi</code>	1.7724538509055159	square root of π

8.1.1.4 Comments

1. A working version of the parameters file (for DL_POLY_2 versions prior to 2.11), for any given application, can be constructed with the aid of the utility program PARSET see section 7.3.10 of the DL_POLY_2 Reference Manual.
2. Many of the above parameters refer to array dimensions. If the application does not use the associated functionality, these parameters may be set to 1. e.g. `mxtang` and `mxangl` may both be set to 1 if the simulation does not require bond angles.
3. Any changes made to the parameters file requires the entire program to be recompiled. (The *build* directory contains example makefiles which do this automatically.)
4. The parameters to found in the /params/ COMMON block (versions 2.11 and above) may be determined from the following FORTRAN extract, taken from the DL_PARAMS.INC file. The variables appearing have the same meaning as the parameters described above.

c array allocation parameters (set by subroutine parset)

```
common/params/kmaxa,kmaxb,kmaxc,minnode,msatms,msbad,msgrp,
x      mspmf,msteth,mxangl,mxatms,mxbond,mxbuff,mcxcell,
x      mxcons,mxdihd,mxewld,mxexcl,mxfbp,mxfld,mxgadm,mxgrid,
x      mxgrp,mxinv,mxlist,mxlshp,mxneut,mxngp,mxnstk,mxpang,
x      mxpbnd,mxpdih,mxpfbp,mxpinv,mxpmf,mxproc,mxptbp,mxpvdw,
x      mxrdf,mxshl,mxsite,mxspmf,mxstak,mxsvdw,mxtang,mxtbnd,
x      mxtbp,mxtcon,mxtdih,mxteth,mxtinv,mxtmls,mxtshl,mxungp,
x      mxvdw,mxxdf,mx2tbp,mx3fbp,mxebuf,mxquat,mxshak,mxspl,
x      kmaxd,kmaxe,kmaxf,mxspme,mxftab,mxhko,mxegrp,mxhke
```

8.1.2 ANGFRFC

8.1.2.1 Header records

```

      subroutine angfrfc
      x  (idnode,imcon,mxnode,ntangl,engang,virang,keyang,listang,
      x  cell,fx,fyy,fzz,prmang,xxx,yyy,zzz,xdab,ydab,zdab,xdbc,
      x  ydbc,zdbc,stress)
c
c*****
c
c      dl_poly subroutine for calculating bond angle energy and
c      force terms in molecular dynamics.
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith             may 1992
c      modified  - t. forester          feb 1993
c      modified  - t.forester           nov 1994 : block data
c      stress tensor - t.forester      may 1995
c
c*****
c

```

8.1.2.2 Function

Calculates valence angle interactions.

8.1.2.3 Dependencies

- ERROR
- GSTATE
- IMAGES
- GDDSUM
- GSTATE

8.1.2.4 Arguments

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors
ntangl	input	total number of interactions in the system

reals:

engang	output	valence angle interaction energy
virang	output	valence angle interaction virial
integer arrays:		
keyang	input	key for angle potential (mxtang)
listang	input	index for potential variables and atomic coordinates (mxangl,4)
real arrays:		
cell	input	MD cell vectors (9)
fx	output	x component of force(mxatms)
fy	output	y component of force(mxatms)
fz	output	z component of force(mxatms)
prmang	input	potential variables (mxtang,mxpang)
xxx	input	x coordinate(mxatms)
yyy	input	y coordinate(mxatms)
zzz	input	z coordinate(mxatms)
xdab	work	x component of a-b inter atomic vector (msbad)
ydab	work	y component of a-b inter atomic vector (msbad)
zdab	work	z component of a-b inter atomic vector (msbad)
xdb	work	x component of b-c inter atomic vector (msbad)
ydb	work	y component of b-c inter atomic vector (msbad)
zdb	work	z component of b-c inter atomic vector (msbad)
stress	in/out	stress tensor (9)

8.1.2.5 Comments

Entries in the array **keyang** determine the interaction potential to be used and the significance of entries in the **prmang** array. Each interaction has its own interaction type and interaction parameters. Allowed combinations are summarised in the table. In each case k is the interaction strength, θ_0 is the equilibrium angle between atoms abc , ρ a screening length and r_{ab} and r_{bc} the ab and bc bond lengths.

NB: If **keyang** is positive then the ab , bc and ac pairs are added to the exclude list, if **keyang** is negative they are not. (Normally the ab and bc pairs will be excluded anyway because a bond will have been defined involving those pairs). If a pair is in the exclude list then DL_POLY_2 ignores the electrostatic and short range interactions between them, otherwise they are evaluated.

Table 8.1: Angle potential parameters

keyang	potential type	prmang (1-4)				functional form
1	Harmonic	k	θ_0			$U = (k/2)(\theta - \theta_0)^2$
2	Quartic	k	θ_0	k_3	k_4	$U = (k/2)(\theta - \theta_0)^2 + (k_3/3)(\theta - \theta_0)^3 + (k_4/4)(\theta - \theta_0)^4$
3	Truncated Harmonic	k	θ_0	ρ		$U = (k/2)(\theta - \theta_0)^2 \exp[-(r_{ab}^8 + r_{bc}^8)/\rho^8]$
4	Screened Harmonic	k	θ_0	ρ_1	ρ_2	$U = (k/2)(\theta - \theta_0)^2 \exp[-(r_{ab}/\rho_1 + r_{bc}/\rho_2)]$
5	Screened Vessal	k	θ_0	ρ_1	ρ_2	$U = [k/8(\theta - \pi)^2] \left\{ [(\theta_0 - \pi)^2 - (\theta - \pi)^2]^2 \right\} \exp[-(r_{ab}/\rho_1 + r_{bc}/\rho_2)]$
6	Truncated Vessal	k	θ_0	a	ρ	$U = k[\theta^a(\theta - \theta_0)^2(\theta + \theta_0 - 2\pi)^2 - (a/2)\pi^{a-1}(\theta - \theta_0)^2(\pi - \theta_0)^3] \exp[-(r_{ab}^8 + r_{bc}^8)/\rho^8]$

8.1.3 BNDFRC

8.1.3.1 Header records

```

      subroutine bndfrc
      x  (idnode,imcon,mxnode,ntbond,engbnd,virbnd,keybnd,listbnd,cell,
      x  fxx,fyy,fzz,prmbnd,xxx,yyy,zzz,xdab,ydab,zdab,stress,buffer)
c
c*****
c
c      dl_poly subroutine for calculating chemical bond energy and
c      force terms in molecular dynamics.
c
c      replicated data - blocked  data version
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith              july 1992
c
c      'lennard-jones' bonds added for GROMOS 1-4 interactions
c      - t forester      march 1993
c      keybnd > 0 ==> bond (excluded interaction)
c      keybnd < 0 ==> distance restraint (non-excluded interaction)
c      - t. forester     march 1994
c
c      stress tensor added : t.forester may 1994
c      block data version :  t.forester nov 1994
c
c*****

```

8.1.3.2 Function

Calculates chemical bond energy and force terms.

8.1.3.3 Dependencies

- ERROR
- GSTATE
- IMAGES
- GDSUM

8.1.3.4 Arguments

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors
ntbond	input	total number of interactions in the system
reals:		
engbnd	output	chemical bond interaction energy
virbnd	output	chemical bond interaction virial
integer arrays:		
keybnd	input	key for bond potential (mxtbnd)
listbnd	input	index for potential variables and atomic coordinates (mxbond ,3)
real arrays:		
cell	input	MD cell vectors (9)
fx	output	x component of force (mxatms)
fy	output	y component of force (mxatms)
fz	output	z component of force (mxatms)
prmbnd	input	potential variables (mxtbnd , mxbnd)
xxx	input	x coordinate (mxatms)
yyy	input	y coordinate (mxatms)
zzz	input	z coordinate (mxatms)
xdab	work	x component of a-b inter atomic vector (msbad)
ydab	work	y component of a-b inter atomic vector (msbad)
zdab	work	z component of a-b inter atomic vector (msbad)
stress	in/out	stress tensor (9)
buffer	in/out	message buffer (mxbuff)

8.1.3.5 Comments

The sign of **keybnd** indicates whether the designated bond is a *bona fide* bond (**keybnd** > 0), an additional excluded interaction (**keybnd** = 0), or a distance restraint (**keybnd** < 0). Electrostatic and short-range potentials are calculated for the interaction (elsewhere in the code) only for distance restraints. The book-keeping for this takes place in the subroutine EXCLUDE.

The bond potentials / distance restraints supplied with DL_POLY_2 are summarised in the table. In the table r_0 is the equilibrium bond length, α a reciprocal length, and the k , A and B interaction strengths.

Table 8.2: Bond potential parameters

keybnd	potential type	prmbnd(1-4)				functional form
0	null					$U = 0$
1	Harmonic	k	r_0			$U = \frac{1}{2}k(r - r_0)^2$
2	Morse	A	r_0	α		$U = A \{1 - \exp[-\alpha(r - r_0)]\}^2$
3	12 - 6	A	B			$U = A/r^{12} - B/r^6$
4	restrained harmonic	k	r_0	r_a		$ r - r_0 < r_a: U = \frac{1}{2}k(r - r_0)^2$ $ r - r_0 > r_a: U = kr_a r - r_0 - \frac{1}{2}kr_a^2$
5	Quartic	k	r_0	k_3	k_4	$U = \frac{1}{2}k(r - r_0)^2 + \frac{1}{3}k_3(r - r_0)^3 + \frac{1}{4}k_4(r - r_0)^4$

8.1.4 BSPCOE

8.1.4.1 Header records

```

      subroutine bspcoe
      x  (nospl,kmax1,kmax2,kmax3,csp,bscx,bscy,bscz,ww1,ww2,ww3)

c*****
c
c      dl_poly subroutine to calculate B-spline coefficients for
c      Euler exponential splines.
c
c      copyright - daresbury laboratory 1998
c      author    - w. smith july 1998
c
c*****

```

8.1.4.2 Function

Calculates the B-spline coefficients for the Ewald SPME method.

8.1.4.3 Dependencies

- NONE

8.1.4.4 Arguments

integers:

nospl	input	required interpolation order of B-splines
kmax1	input	order of interpolation grid (1st direction)
kmax2	input	order of interpolation grid (2nd direction)
kmax3	input	order of interpolation grid (3d direction)

real arrays:

csp	output	B-spline coefficients at knots

complex arrays:

bscx	output	B-spline coefficients (1st direction) (kmaxd)
bscy	output	B-spline coefficients (2nd direction) (kmaxe)
bscz	output	B-spline coefficients (3d direction) (kmaxf)
ww1	input	complex exponentials (1st direction) (kmaxd)
ww2	input	complex exponentials (2nd direction) (kmaxe)
ww3	input	complex exponentials (3d direction) (kmaxf)

8.1.4.5 Comments

A full description of the algorithms underpinning this subroutine can be found in [31].

8.1.5 BSPGEN

8.1.5.1 Header records

```

      subroutine bspgen
      x  (nospl,natms,xxx,yyy,zzz,bspdx,bspdy,bspdz,bsdxd,bsdxdy,bsdxdz)

c*****
c
c      dl_poly subroutine to calculate B-splines for SPME method
c
c      copyright - daresbury laboratory 1998
c      author    - w. smith july 1998
c
c*****

```

8.1.5.2 Function

Calculates the B-splines (to order `nospl`) used to interpolate the charge distribution in the Ewald SPME method. Also calculated are the first derivatives needed to determine the atomic forces.

8.1.5.3 Dependencies

- NONE

8.1.5.4 Arguments

integers:

<code>nospl</code>	input	required interpolation order of B-splines
<code>natms</code>	input	number of atoms in system

real arrays:

<code>xxx</code>	input	reduced X-coordinate of atoms (<code>mxatms</code>)
<code>yyy</code>	input	reduced Y-coordinate of atoms (<code>mxatms</code>)
<code>zzz</code>	input	reduced Z-coordinate of atoms (<code>mxatms</code>)

complex arrays:

<code>bspdx</code>	output	B-splines (1st direction) (<code>mxspme,mxspl</code>)
<code>bspdy</code>	output	B-splines (2nd direction) (<code>mxspme,mxspl</code>)
<code>bspdz</code>	output	B-splines (3d direction) (<code>mxspme,mxspl</code>)
<code>bsdxd</code>	output	First derivative of B-splines (1st direction) (<code>mxspme,mxspl</code>)
<code>bsdxdy</code>	output	First derivative of B-splines (2nd direction) (<code>mxspme,mxspl</code>)
<code>bsdxdz</code>	output	First derivative of B-splines (3d direction) (<code>mxspme,mxspl</code>)

8.1.5.5 Comments

A full description of the algorithms underpinning this subroutine can be found in [31].

8.1.6 CFGSCAN

8.1.6.1 Header records

```

      subroutine cfgscan
      x  (idnode,mxnode,nconf,imcon,volm,xhi,yhi,zhi,cell,buffer)

c*****
c
c      dl_poly subroutine for scanning the initial configuration
c      file to determine the number of atoms present
c
c      copyright - daresbury laboratory 1997
c      author    - w. smith  june      1997
c
c      note: volm is volume containing all particles, not system volume
c
c*****
c

```

8.1.6.2 Function

Scans the DL_POLY_2 CONFIG file to obtain estimates of required DL_POLY_2 array sizes for dynamic array allocation and passes estimates back to subroutine PARSET.

8.1.6.3 Dependencies

DCELL GDSUM

8.1.6.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors
nconf	input	channel number of CONFIG file

reals:

volm	output	estimated system volume
xhi	output	max atom x-displacement from system centre
yhi	output	max atom y-displacement from system centre
zhi	output	max atom z-displacement from system centre

real arrays:

cell	output	simulation cell vectors (9)
buffer	work	(10)

8.1.6.5 Comments

8.1.7 CONSCAN

8.1.7.1 Header records

```

      subroutine conscan
      x      (idnode,mxnode,nread,imcon,mxstak,kmaxa,kmaxb,kmaxc,
      x      kmaxd,kmaxe,kmaxf,rcut,rvdw,delr,cell)
c
c*****
c
c      dl_poly subroutine for scanning the contents of the control file
c
c      copyright - daresbury laboratory 1997
c      author    - w. smith  june      1997
c
c*****
c

```

8.1.7.2 Function

Scans the DL_POLY_2 CONTROL file to obtain estimates of required DL_POLY_2 array sizes for dynamic array allocation and pass estimates back to subroutine PARSET.

8.1.7.3 Dependencies

DBLSTR	ERROR	GETREC
INTSTR	LOWCASE	STRIP

8.1.7.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors
nread	input	channel number of CONTROL file
imcon	input	system image convention
mxtak	output	stack array dimension
kmaxa	output	Ewald max k-vector x-component
kmaxb	output	Ewald max k-vector y-component
kmaxc	output	Ewald max k-vector z-component
kmaxd	output	SPME max k-vector x-component
kmaxe	output	SPME max k-vector y-component
kmaxf	output	SPME max k-vector z-component

reals:

rcut	input	maximum potential cutoff
rvdw	input	VDW potential cutoff

<code>delr</code>	input	potential cutoff shell width
real arrays:		
<code>cell</code>	input	simulation cell vectors (9)

8.1.7.5 Comments

8.1.8 CORSHL

8.1.8.1 Header records

```

      subroutine corshl
      x (idnode,mxnode,ntshl,shlke,listshl,weight,vxx,vyy,vzz)
c
c*****
c
c      dl_poly subroutine for calculating the internal kinetic
c      energy of core-shell units in the shell polarisation model
c
c      copyright - daresbury laboratory 1994
c      author    - w. smith july 1994
c
c*****
c

```

8.1.8.2 Function

Calculates core-shell interaction energy and forces.

8.1.8.3 Dependencies

- GDSUM

8.1.8.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
ntshl	input	total number of core-shell units in system

reals:

shlke	output	core-shell kinetic energy

integer arrays:

listshl	input	indices for core-shell interactions on current processor (mxshl,3)

real arrays:

weight	input	atomic masses (mxatms)
vxx	input	x component of velocity (mxatms)
vyy	input	y component of velocity (mxatms)
vzz	input	z component of velocity (mxatms)

8.1.8.5 Comments

Assumes a dynamical shell model implementation.

8.1.9 COUL0

8.1.9.1 Header records

```

      subroutine coul0
      x  (iatm,ik,engcpe,vircpe,rcut,epsq,ilist,
      x  chge,rsqdf,xdf,ydf,zdf,fx,fyy,fzz,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic force.
c      1/r potential, no truncation or damping
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1993
c      author    - t. forester february 1993
c      stress tensor - t.forester may 1994
c
c*****
c

```

8.1.9.2 Function

Calculates interaction energy and forces for the Coulombic potential. The potential energy is

$$U = \frac{1}{4\pi\epsilon_0\epsilon_r} \frac{q_i q_j}{r} \quad (8.1)$$

8.1.9.3 Dependencies

- none

8.1.9.4 Arguments

integers:

iatm	input	index of the central site
ik	input	number of neighbours around the central site

reals:

engcpe	output	Coulombic energy
vircpe	output	Coulombic virial
rcut	input	potential cut off radius
epsq	input	relative dielectric constant, ϵ_r

integer arrays:

ilist	input	indices of neighbours of the central site (mxxdf)

real arrays:

<code>chge</code>	input	electrostatic charge (<code>mxatms</code>)
<code>rsqdf</code>	input	square of intersite distance (<code>mxxdf</code>)
<code>xdf</code>	input	x component of intersite vector (<code>mxxdf</code>)
<code>ydf</code>	input	y component of intersite vector (<code>mxxdf</code>)
<code>zdf</code>	input	z component of intersite vector (<code>mxxdf</code>)
<code>fxx</code>	output	x component of force (<code>mxatms</code>)
<code>fyy</code>	output	y component of force (<code>mxatms</code>)
<code>fzz</code>	output	z component of force (<code>mxatms</code>)
<code>stress</code>	in/out	stress tensor (9)

8.1.9.5 Comments

DL_POLY_2 employs an *atomistic* cutoff to determine which interactions should be calculated. Only those less than `rcut` are evaluated. The recommended way of evaluating electrostatic interactions in a *periodic* system is to use the Ewald sum and not this routine. Use of `COUL0` is recommended only for *non-periodic* systems in which *all* pair interactions fall within the cutoff sphere. If these precautions are not heeded severe heating effects and unphysical structural properties may be seen in the simulation. Subroutines `COUL1`, `COUL2` and `COUL3` deal with the cutoff somewhat better than `COUL0` although they offer modified forms of the Coulombic potential. The user may also consider using a group based cutoff scheme as opposed to an atomistic one, as this avoids unphysical “charging” effects in the cutoff sphere - see `COUL0NEU`, `COUL2NEU` and `COUL3NEU`.

8.1.10 COUL0_SW

8.1.10.1 Header records

```

      subroutine coul0_sw
      x  (iatm,ik,iregn,mxregn,engcpe,vircpe,rcut2,rcut1,epsq,
      x  swlen,ilist,chge,rsqdf,xdf,ydf,zdf,fx,fy,fz,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic force.
c      1/r potential, with cubic switching function between
c      rcut1 - swlen < r < rcut2
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1996
c      author    - t. forester june 1996
c
c*****
c

```

8.1.10.2 Function

Calculates interaction energy and forces for the Coulombic potential with a cubic switching function. The potential energy is

$$U = \frac{1}{4\pi\epsilon_0\epsilon_r} \frac{q_i q_j}{r} S(r, R_1, R_2, \lambda). \quad (8.2)$$

Where the Switching function, S, is

$$\begin{aligned}
 r < R_1 - \lambda : \quad S &= 0 \\
 R_1 - \lambda \leq r < R_1 : \quad S &= \Delta_1^2(3 - 2\Delta_1) \\
 R_1 \leq r < R_2 - \lambda : \quad S &= 1 \\
 R_2 - \lambda \leq r < R_2 : \quad S &= 1 - \Delta_2^2(3 - 2\Delta_2) \\
 R_2 \leq r : \quad S &= 0
 \end{aligned}$$

where $\Delta_i = [r - (R_i - \lambda)]/\lambda$. However if $\text{iregn} = \text{mxregn}$ then $\Delta_2 = 0$.

8.1.10.3 Dependencies

- none

8.1.10.4 Arguments

integers:

iatm	input	index of the central site
ik	input	number of neighbours around the central site
iregn	input	Switching region index
mxregn	input	total number of Switching regions

reals:

engcpe	output	Coulombic energy
epsq	input	relative dielectric constant, ϵ_r
rcut1	input	lower cut off radius R_1
rcut2	input	upper cut off radius R_2
swlen	input	switching length λ
vircpe	output	Coulombic virial

integer arrays:

ilist	input	indices of neighbours of the central site (mxxdf)

real arrays:

chge	input	electrostatic charge (mxatms)
rsqdf	input	square of intersite distance (mxxdf)
xdf	input	x component of intersite vector (mxxdf)
ydf	input	y component of intersite vector (mxxdf)
zdf	input	z component of intersite vector (mxxdf)
fxx	output	x component of force (mxatms)
fyy	output	y component of force (mxatms)
fzz	output	z component of force (mxatms)
stress	in/out	stress tensor (9)

8.1.10.5 Comments

See comments on COUL0. The switching function has a non-negligible effect on the forces (as well as the energy) in the switching region.

8.1.11 COULONEU

8.1.11.1 Header records

```

      subroutine coul0neu
      x  (ik,engcpe,vircpe,epsq,ilist,jlist,chge,
      x  rsqdf,xdf,ydf,zdf,fx,fyy,fzz,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic force.
c      1/r potential, no truncation or damping.
c      neutral group implementation
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1994
c      author    - t. forester    march 1994
c
c*****

```

8.1.11.2 Function

Calculates interaction energy and forces for the Coulombic potential with a group based cutoff scheme. The potential energy is of the form

$$U = \frac{1}{4\pi\epsilon_0\epsilon_r} \frac{q_i q_j}{r} \quad (8.3)$$

8.1.11.3 Dependencies

- none

8.1.11.4 Arguments

integers:

ik	input	number of interactions (site-site)

reals:

engcpe	output	Coulombic energy
vircpe	output	Coulombic virial
epsq	input	relative dielectric constant

integer arrays:

ilist	input	indices of first site (mxxdf)
jlist	input	indices of second site (mxxdf)

real arrays:

chge	input	electrostatic charge (mxatms)

rsqdf	input	square of intersite distance (mxxdf)
xdf	input	x component of intersite vector (mxxdf)
ydf	input	y component of intersite vector (mxxdf)
zdf	input	z component of intersite vector (mxxdf)
fx	output	x component of force (mxatms)
fy	output	y component of force (mxatms)
fz	output	z component of force (mxatms)
stress	in/out	stress tensor (9)

8.1.11.5 Comments

This routine assumes a group based cutoff has been used to generate the arrays **ilist** and **jlist** and so the cutoff is not implemented within the routine. The recommended way of evaluating electrostatic interactions in a three dimensional *periodic* system is to use the Ewald sum and not this routine. Use of COUL0NEU is recommended only for *non-periodic* systems and even there the user may wish to consider using a reaction field approach instead (see COUL3NEU). If these precautions are not heeded severe heating effects and unphysical structural properties may be seen in the simulation.

8.1.12 COUL1

8.1.12.1 Header records

```

      subroutine coul1
      x  (iatm,ik,engcpe,vircpe,rcut,epsq,ilist,chge,
      x  rsqdf,xdf,ydf,zdf,fxx,fyy,fzz,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic forces
c      assuming a standard coulomb potential truncated at rcut
c      and shifted to zero at rcut.
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith december 1992.
c
c      stress tensor t.forester may 1994
c
c*****
c

```

8.1.12.2 Function

Calculates interaction energy and forces for a truncated, shifted Coulombic potential. The potential energy is of the form

$$U = \frac{1}{4\pi\epsilon_0\epsilon_r} q_i q_j \left(\frac{1}{r} - \frac{1}{r_{\text{cut}}} \right) \quad (8.4)$$

8.1.12.3 Dependencies

- none

8.1.12.4 Arguments

integers:

iatm	input	index of the central site
ik	input	number of neighbours around the central site

reals:

engcpe	output	Coulombic energy
vircpe	output	Coulombic virial
rcut	input	potential cut off radius
epsq	input	relative dielectric constant, ϵ_r

integer arrays:

ilist	input	indices of neighbours of the central site (mxxdf)

real arrays:

chge	input	electrostatic charge (mxatms)
rsqdf	input	square of intersite distance (mxxdf)
xdf	input	x component of intersite vector (mxxdf)
ydf	input	y component of intersite vector (mxxdf)
zdf	input	z component of intersite vector (mxxdf)
fxx	output	x component of force (mxatms)
fyy	output	y component of force (mxatms)
fzz	output	z component of force (mxatms)
stress	in/out	stress tensor (9)

8.1.12.5 Comments

DL_POLY_2 employs an *atomistic* cutoff to determine which interactions should be calculated. Only those less than **rcut** are evaluated. The recommended way of evaluating electrostatic interactions in a *periodic* system is to use the Ewald sum and not this routine. Use of COUL1 is recommended only for crude structural optimizations. Note that the potential energy, but not the force, is zero at the cutoff.

8.1.13 COUL2

8.1.13.1 Header records

```

      subroutine coul2
      x  (iatm,ik,engcpe,vircpe,rcut,epsq,ilist,chge,
      x  rsqdf,xdf,ydf,zdf,fx,fyy,fzz,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic forces
c      assuming a distance dependant dielectric 'constant'.
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1993
c      author    - t. forester    april 1993
c      stress tensor added - t.forester may 1994
c
c*****
c

```

8.1.13.2 Function

Calculates interaction energy and forces for a Coulombic potential with a distance dependent dielectric constant. The form of the potential is

$$U = \frac{1}{4\pi\epsilon_0\epsilon_r} \frac{q_i q_j}{r^2} \quad (8.5)$$

8.1.13.3 Dependencies

- none

8.1.13.4 Arguments

integers:

iatm	input	index of the central site
ik	input	number of neighbours around the central site

reals

engcpe	output	Coulombic energy
vircpe	output	Coulombic virial
rcut	input	potential cut off radius
epsq	input	relative dielectric constant, ϵ_r , at $r = 1$

integer arrays:

ilist	input	indices of neighbours of the central site (mxxdf)

real arrays:

chge	input	electrostatic charge (mxatms)
rsqdf	input	square of intersite distance (mxxdf)
xdf	input	x component of intersite vector (mxxdf)
ydf	input	y component of intersite vector (mxxdf)
zdf	input	z component of intersite vector (mxxdf)
fx	output	x component of force (mxatms)
fy	output	y component of force (mxatms)
fz	output	z component of force (mxatms)
stress	in/out	stress tensor (9)

8.1.13.5 Comments

DL_POLY_2 employs an *atomistic* cutoff to determine which interactions should be calculated. Only those less than **rcut** are evaluated. The recommended way of evaluating electrostatic interactions in a *periodic* system is to use the Ewald sum and not this routine. The potential form used in COUL2 has traditionally been used to model the effects of implicit solvent in biomolecular simulations. It should only be used with caution, and users should consider using a group based cutoff scheme rather than atomistic cutoff scheme (see COUL2NEU).

8.1.14 COUL2_SW

8.1.14.1 Header records

```

      subroutine coul2_sw
      x  (iatm,ik,iregn,mxregn,engcpe,vircpe,rcut2,rcut1,epsq,
      x  swlen,ilist,chge,rsqdf,xdf,ydf,zdf,fx,ffy,fzz,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic forces
c      assuming a distance dependant dielectric 'constant'.
c      with cubic switching functions
c      rcut1 - swlen < r < rcut2
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1996
c      author    - t. forester      june 1996
c
c*****
c

```

8.1.14.2 Function

Calculates interaction energy and forces for a Coulombic potential with a distance dependent dielectric constant. The form of the potential is

$$U = \frac{1}{4\pi\epsilon_0\epsilon_r} \frac{q_i q_j}{r^2} S(r, R_1, R_2, \lambda) \quad (8.6)$$

Where S is the Switching function:

$$\begin{aligned}
 r < R_1 - \lambda : \quad S &= 0 \\
 R_1 - \lambda \leq r < R_1 : \quad S &= \Delta_1^2(3 - 2\Delta_1) \\
 R_1 \leq r < R_2 - \lambda : \quad S &= 1 \\
 R_2 - \lambda \leq r < R_2 : \quad S &= 1 - \Delta_2^2(3 - 2\Delta_2) \\
 R_2 \leq r : \quad S &= 0
 \end{aligned}$$

where $\Delta_i = [r - (R_i - \lambda)]/\lambda$. However if $\text{iregn} = \text{mxregn}$ then $\Delta_2 = 0$.

8.1.14.3 Dependencies

- none

8.1.14.4 Arguments

integers:

iatm	input	index of the central site
ik	input	number of neighbours around the central site
iregn	input	Switching region index
mxregn	input	total number of Switching regions

reals:

engcpe	output	Coulombic energy
epsq	input	relative dielectric constant, ϵ_r
rcut1	input	lower cut off radius R_1
rcut2	input	upper cut off radius R_2
swlen	input	switching length λ
vircpe	output	Coulombic virial

integer arrays:

ilist	input	indices of neighbours of the central site (mxxdf)

real arrays:

chge	input	electrostatic charge (mxatms)
rsqdf	input	square of intersite distance (mxxdf)
xdf	input	x component of intersite vector (mxxdf)
ydf	input	y component of intersite vector (mxxdf)
zdf	input	z component of intersite vector (mxxdf)
fxx	output	x component of force (mxatms)
fyy	output	y component of force (mxatms)
fzz	output	z component of force (mxatms)
stress	in/out	stress tensor (9)

8.1.14.5 Comments

See comments on COUL2. The switching function has a non-negligible effect on the forces (as well as the energy) in the switching region.

8.1.15 COUL2NEU

8.1.15.1 Header records

```

      subroutine coul2neu
      x  (ik,engcpe,vircpe,epsq,ilist,jlist,
      x  chge,rsqdf,xdf,ydf,zdf,fx,fyy,fzz,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic forces
c      assuming a distance dependant dielectric 'constant'.
c      neutral group implementation
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1994
c      author    - t. forester    march 1994
c
c*****
c

```

8.1.15.2 Function

Calculates interaction energy and forces for a Coulombic potential with a distance dependent dielectric constant. A group based cutoff scheme is assumed. The form of the potential is

$$U = \frac{1}{4\pi\epsilon_0\epsilon_r} \frac{q_i q_j}{r^2} \quad (8.7)$$

8.1.15.3 Dependencies

- none

8.1.15.4 Arguments

integers:

ik	input	number of interactions (site-site)

reals

engcpe	output	Coulombic energy
vircpe	output	Coulombic virial
epsq	input	relative dielectric constant, ϵ_r , at $r = 1$

integer arrays:

ilist	input	indices of first site (mxxdf)
jlist	input	indices of second site (mxxdf)

real arrays:

chge	input	electrostatic charge (mxatms)
rsqdf	input	square of intersite distance (mxxdf)
xdf	input	x component of intersite vector (mxxdf)
ydf	input	y component of intersite vector (mxxdf)
zdf	input	z component of intersite vector (mxxdf)
fx	output	x component of force (mxatms)
fy	output	y component of force (mxatms)
fz	output	z component of force (mxatms)
stress	in/out	stress tensor (9)

8.1.15.5 Comments

The potential form used in this routine has been popular in simulations of bio-molecules in implicit solvent. It should only be used with caution.

8.1.16 COUL3

8.1.16.1 Header records

```

      subroutine coul3
      x  (iatm,ik,engcpe,vircpe,rcut,epsq,ilist,
      x  chge,rsqdf,xdf,ydf,zdf,fx,fyy,fzz,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic force.
c      reaction field  potential
c      Ref: M Neumann, J Chem Phys, 82, 5633, (1985)
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1995
c      author    - t. forester february 1995
c      stress tensor - t.forester   feb 1995
c
c*****
c

```

8.1.16.2 Function

Calculates interaction energy and forces for the Coulombic potential with a reaction field. The potential is shifted to zero at the cutoff. The potential energy is of the form

$$U = \frac{q_i q_j}{4\pi\epsilon_0} \left[\frac{1}{r} + \left(\frac{\epsilon - 1}{2\epsilon + 1} \right) \left(\frac{r^2}{r_{cut}^3} \right) - \left(1 + \frac{\epsilon - 1}{2\epsilon + 1} \right) \left(\frac{1}{r_{cut}} \right) \right] \quad (8.8)$$

8.1.16.3 Dependencies

- none

8.1.16.4 Arguments

integers:

iatm	input	index of the central site
ik	input	number of neighbours around the central site

reals:

engcpe	output	Coulombic energy
vircpe	output	Coulombic virial
rcut	input	potential cut off radius
epsq	input	reaction field dielectric constant, ϵ

integer arrays:

<code>ilist</code>	input	indices of neighbours of the central site (<code>mxxdf</code>)
real arrays:		
<code>chge</code>	input	electrostatic charge (<code>mxatms</code>)
<code>rsqdf</code>	input	square of intersite distance (<code>mxxdf</code>)
<code>xdf</code>	input	x component of intersite vector (<code>mxxdf</code>)
<code>ydf</code>	input	y component of intersite vector (<code>mxxdf</code>)
<code>zdf</code>	input	z component of intersite vector (<code>mxxdf</code>)
<code>fx</code>	output	x component of force (<code>mxatms</code>)
<code>fy</code>	output	y component of force (<code>mxatms</code>)
<code>fz</code>	output	z component of force (<code>mxatms</code>)
<code>stress</code>	in/out	stress tensor (9)

8.1.16.5 Comments

DL_POLY_2 employs an *atomistic* cutoff to determine which interactions should be calculated. Only those less than `rcut` are evaluated. The user may also wish to consider using a group based cut off scheme (see COUL3NEU).

8.1.17 COUL3_SW

8.1.17.1 Header records

```

      subroutine coul3_sw
      x  (iatm,ik,iregn,mxregn,engcpe,vircpe,rcut,rcut2,rcut1,
      x  swlen,epsq,ilist,chge,rsqdf,xdf,ydf,zdf,fx,fy,fz,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic force.
c      reaction field potential
c      Ref: M Neumann, J Chem Phys, 82, 5633, (1985)
c
c      with cubic switching function
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1996
c      author    - t. forester      june 1996
c
c*****
c

```

8.1.17.2 Function

Calculates interaction energy and forces for the Coulombic potential with a reaction field. The potential is shifted to zero at the cutoff and modified by a Switching function. The potential energy is of the form

$$U = \frac{q_i q_j}{4\pi\epsilon_0} \left[\frac{1}{r} + \left(\frac{\epsilon - 1}{2\epsilon + 1} \right) \left(\frac{r^2}{r_{cut}^3} \right) - \left(1 + \frac{\epsilon - 1}{2\epsilon + 1} \right) \left(\frac{1}{r_{cut}} \right) \right] S(r, R_1, R_2, \lambda) \quad (8.9)$$

Where S is the Switching function:

$$\begin{aligned}
 r < R_1 - \lambda : \quad S &= 0 \\
 R_1 - \lambda \leq r < R_1 : \quad S &= \Delta_1^2(3 - 2\Delta_1) \\
 R_1 \leq r < R_2 - \lambda : \quad S &= 1 \\
 R_2 - \lambda \leq r < R_2 : \quad S &= 1 - \Delta_2^2(3 - 2\Delta_2) \\
 R_2 \leq r : \quad S &= 0
 \end{aligned}$$

where $\Delta_i = [r - (R_i - \lambda)]/\lambda$. However if $\text{iregn} = \text{mxregn}$ then $\Delta_2 = 0$.

8.1.17.3 Dependencies

- none

8.1.17.4 Arguments

integers:

iatm	input	index of the central site
ik	input	number of neighbours around the central site
iregn	input	Switching region index
mxregn	input	total number of Switching regions

reals:

engcpe	output	Coulombic energy
epsq	input	reaction field dielectric, ϵ
rcut	input	potential cut off radius r_{cut} ($= R_2$ in the outermost region)
rcut1	input	lower cut off radius R_1
rcut2	input	upper cut off radius R_2
swlen	input	switching length λ
vircpe	output	Coulombic virial

integer arrays:

ilist	input	indices of neighbours of the central site (mxxdf)

real arrays:

chge	input	electrostatic charge (mxatms)
rsqdf	input	square of intersite distance (mxxdf)
xdf	input	x component of intersite vector (mxxdf)
ydf	input	y component of intersite vector (mxxdf)
zdf	input	z component of intersite vector (mxxdf)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
stress	in/out	stress tensor (9)

8.1.17.5 Comments

See comments on COUL3. The switching function has a non-negligible effect on the forces (as well as the energy) in the switching region.

8.1.18 COUL3NEU

8.1.18.1 Header records

```

      subroutine coul3neu
      x  (ik,engcpe,vircpe,epsq,rcut,ilist,jlist,
      x  chge,rsqdf,xdf,ydf,zdf,fxx,fyy,fzz,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic force.
c      reaction field  potential
c      Ref: M Neumann, J Chem Phys, 82, 5633, (1985)
c
c      neutral group implementation
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1995
c      author    - t. forester february 1995
c      stress tensor - t.forester  feb 1995
c
c*****
c

```

8.1.18.2 Function

Calculates interaction energy and forces for the Coulombic potential with a reaction field and a group based cutoff scheme. The potential energy is of the form

$$U = \frac{q_i q_j}{4\pi\epsilon_0} \left[\frac{1}{r} + \left(\frac{\epsilon - 1}{2\epsilon + 1} \right) \left(\frac{r^2}{r_{cut}^3} \right) \right] \quad (8.10)$$

8.1.18.3 Dependencies

- none

8.1.18.4 Arguments

integers:

ik	input	number of interactions (site-site)

reals:

engcpe	output	Coulombic energy
vircpe	output	Coulombic virial
rcut	input	potential cut off radius
epsq	input	reaction field dielectric constant, ϵ

integer arrays:

<code>ilist</code>	input	indices of first site (<code>mxxdf</code>)
<code>jlist</code>	input	indices of second site (<code>mxxdf</code>)

real arrays:

<code>chge</code>	input	electrostatic charge (<code>mxatms</code>)
<code>rsqdf</code>	input	square of intersite distance (<code>mxxdf</code>)
<code>xdf</code>	input	x component of intersite vector (<code>mxxdf</code>)
<code>ydf</code>	input	y component of intersite vector (<code>mxxdf</code>)
<code>zdf</code>	input	z component of intersite vector (<code>mxxdf</code>)
<code>fxx</code>	output	x component of force (<code>mxatms</code>)
<code>fyy</code>	output	y component of force (<code>mxatms</code>)
<code>fzz</code>	output	z component of force (<code>mxatms</code>)
<code>stress</code>	in/out	stress tensor (9)

8.1.18.5 Comments

8.1.19 COUL4

8.1.19.1 Header records

```

      subroutine coul4
      x  (iatm,ik,engcpe,vircpe,rcut,epsq,ilist,chge,
      x  rsqdf,xdf,ydf,zdf,fxx,fyy,fzz,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic forces
c      assuming a force shifted coulomb potential.
c      U proportional to ( 1/r + ar + b )
c      such that dU(rcut)/dr = 0 => a = 1/(rcut)**2
c      and          U(rcut)    = 0 => b = -2/(rcut)
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1995
c      author    - t.forester october 1995
c
c*****
c

```

8.1.19.2 Function

Calculates interaction energy and forces for a shifted force Coulombic potential. The potential energy is of the form

$$U = \frac{1}{4\pi\epsilon_0\epsilon_r} q_i q_j \left(\frac{1}{r} + \frac{r}{r_{cut}^2} - \frac{2}{r_{cut}} \right) \quad (8.11)$$

8.1.19.3 Dependencies

- none

8.1.19.4 Arguments

integers:

iatm	input	index of the central site
ik	input	number of neighbours around the central site

reals:

engcpe	output	Coulombic energy
vircpe	output	Coulombic virial
rcut	input	potential cut off radius
epsq	input	relative dielectric constant, ϵ_r

integer arrays:

ilist	input	indices of neighbours of the central site (mxxdf)

real arrays:

chge	input	electrostatic charge (mxatms)
rsqdf	input	square of intersite distance (mxxdf)
xdf	input	x component of intersite vector (mxxdf)
ydf	input	y component of intersite vector (mxxdf)
zdf	input	z component of intersite vector (mxxdf)
fxx	output	x component of force (mxatms)
fyy	output	y component of force (mxatms)
fzz	output	z component of force (mxatms)
stress	in/out	stress tensor (9)

8.1.19.5 Comments

DL_POLY_2 employs an *atomistic* cutoff to determine which interactions should be calculated. Only those less than **rcut** are evaluated. The recommended way of evaluating electrostatic interactions in a *periodic* system is to use the Ewald sum and not this routine. The use of COUL4 is recommended only for crude structural optimizations. Note that both potential energy and the force are zero at the cutoff.

8.1.20 CPY_RTC**8.1.20.1 Header records**

```
subroutine cpy_rtc(nnn,aaa,bbb)
```

```
c*****
c
c      dl_poly subroutine for copying a real array into a complex array
c      of the same dimension
c
c      copyright daresbury laboratory 1998
c      author w.smith oct 1998
c
c*****
```

8.1.20.2 Function

This routine loads the content of a real array into a complex array, setting the imaginary component to zero.

8.1.20.3 Dependencies

- NONE

8.1.20.4 Arguments

integers:

nnn	input	number of array elements required

real arrays:

aaa	input	real array (*)

complex arrays:

bbb	output	complex array (*)

8.1.20.5 Comments

It is used to avoid the loop nesting needed for multidimensional arrays.

8.1.21 CRECV**8.1.21.1 Header records**

```
subroutine crecv(msgtag,buf,length)
```

```
c*****
c
c      Intel-like  crecv (double precision)
c
c      MPI version - t.forester may 1995
c      CPP version - w.smith may 1995
c
c*****
```

8.1.21.2 Function

MPI mimic of the INTEL message receive function CRECV.

8.1.21.3 Dependencies

- MPI message receive routines
- ERROR

8.1.21.4 Arguments

integers:

msgtag	input	message tag number
length	input	message length in bytes

real arrays:

buf	input	buffer containing message (tt *)

8.1.21.5 Comments

C preprocessing is required to select the appropriate communication harness.

8.1.22 CSEND**8.1.22.1 Header records**

```
subroutine csend(msgtag,buf,length,pe,idum)
```

```
c*****
c
c      Intel-like  csend (double precision)
c
c      MPI version - t.forester may 1995
c      CPP version - w.smith may 1995
c
c*****
```

8.1.22.2 Function

MPI mimic of the INTEL message send function CSEND.

8.1.22.3 Dependencies

- MPI message send routines.
- ERROR

8.1.22.4 Arguments**integers:**

msgtag	input	message tag number
length	input	message length in bytes
pe	input	identity of receiving processor
idum	input	dummy argument

real arrays:

buf	input	buffer containing message (*)

8.1.22.5 Comments

C preprocessing is required to select the appropriate communication harness.

8.1.23 DBLSTR

8.1.23.1 Header records

```

        function dblstr(word,len,lst)
c
c*****
c
c    dl_poly function for extracting double precisions from a
c    character string.
c    modified from dl_poly function intstr
c
c    copyright - daresbury laboratory 1994
c    author    - w. smith may 1994.
c    modified  - t. forester april 1994
c
c    parameters:
c    word      - input character string
c    len       - working length of character string
c    lst       - location of space character at end of
c    double precision string
c
c*****
c

```

8.1.23.2 Function

Extracts the first (real*8) number from a character string.

8.1.23.3 Dependencies

- INTSTR

8.1.23.4 Arguments

character arrays:

word	input	character string (len)

integers:

len	input	length of string
lst	output	location of first character after number in the string

8.1.23.5 Comments

The function returns the double precision *real* value of the first number in the character string. Integer, real, 'd', and 'e' formats are all acceptable.

8.1.24 DCELL**8.1.24.1 Header records**

```

      subroutine dcell(aaa,bbb)
c
c*****
c
c      dl_poly subroutine to calculate the dimensional properies of
c      a simulation cell specified by the input matrix aaa.
c      the results are returned in the array bbb, with :
c
c      bbb(1 to 3) - lengths of cell vectors
c      bbb(4 to 6) - cosines of cell angles
c      bbb(7 to 9) - perpendicular cell widths
c      bbb(10)      - cell volume
c
c      copyright daresbury laboratory 1992
c      author - w. smith          july 1992
c
c*****
c

```

8.1.24.2 Function

Calculates the dimensional properties of the MD cell.

8.1.24.3 Dependencies

- none

8.1.24.4 Arguments

real arrays:

aaa	input	MD cell vectors (9) (stored contiguously)
bbb	output	cell dimensional properties as per header records (10)

8.1.24.5 Comments

8.1.25 DENLOC

8.1.25.1 Header records

```

      subroutine denloc
      x  (iatm,ik,ntpvdw,rcut,dlrpot,ilist,ltype,lstvdw,ltpvdw,rsqdf,
      x  rho,vvv)
c
c*****
c
c      dl_poly subroutine for calculating local atomic density
c      for sutton-chen metal potential
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1995
c      author    - w. smith      june 1995
c
c*****
c

```

8.1.25.2 Function

Calculates the local atomic density for Sutton Chen potentials.

8.1.25.3 Dependencies

- none

8.1.25.4 Arguments

integers:

iatm	input	index of the central site
ik	input	number of entries in the Verlet list
ntpvdw	input	number of van der Waals/metal potentials

reals:

rcut	input	the non-bonded potential cutoff
dlrpot	input	tabulation interval for interpolation tables

integer arrays:

ilist	input	the Verlet neighbour list (mxlist)
ltype	input	atom type (mxatms)
lstvdw	input	index number of non-bonded potential (mxvdw)
ltpvdw	input	key for non-bonded potentials (mxvdw)

real arrays:

rsqdf	input	square of inter site vector length (mxxdf)
rho	output	Sutton Chen particle density (mxatms)

vvv input interpolation table for non-bonded potentials (**mxgrid,mxvdw**)

8.1.25.5 Comments

This subroutine calculates only the density dependent terms of the Sutton Chen potentials. The full calculation requires an additional pair body term (see the subroutine SUTTCHEN).

Both 4-point and r^2 interpolation versions of this routine exist. They are named DENLOC_4PT and DENLOC_RSQ respectively.

8.1.26 DIFFSN0

8.1.26.1 Header records

```

      subroutine diffsn0
      x  (idnode,natms,mxnode,tstep,vxx,vyy,vzz,xx0,yy0,zz0)

c*****
c
c      DL_POLY routine for calculating displacements of sites from
c      t=0
c
c      use diffsn1 for mean squared displacements
c
c      parallel version - replicated data.
c
c      copyright daresbury laboratory 1993
c
c      author - t. forester      june 1993
c
c*****

```

8.1.26.2 Function

Calculates increment in atomic positions due to the current time-step.

8.1.26.3 Dependencies

- none

8.1.26.4 Arguments

integers:

idnode	input	identity of current processor
natms	input	number of atoms in MD cell
mxnode	input	total number of processors

reals:

tstep	input	time step, Δt

real arrays:

vxx	input	x component of velocity (mxatms)
vyy	input	y component of velocity (mxatms)
vzz	input	z component of velocity (mxatms)
xx0	in/out	running sum of x increments (mxatms)
yy0	in/out	running sum of y increments (mxatms)
zz0	in/out	running sum of z increments (mxatms)

8.1.26.5 Comments

This subroutine should be called every time step. Mean squared displacements are calculated in DIFFSN1.

8.1.27 DIFFSN1

8.1.27.1 Header records

```

      subroutine diffsn1
      x  (idnode,natms,ntpatm,mxnode,ltype,numtyp,amsd,
      x  xx0,yy0,zz0,buffer)

c*****
c
c    DL_POLY routine for calculating mean squared displacements
c
c    displacements calculated in diffsn0
c
c    parallel version - replicated data.
c
c    copyright daresbury laboratory 1993
c
c    author - t. forester      june 1993
c
c*****

```

8.1.27.2 Function

Calculates mean squared displacements of atomic types.

8.1.27.3 Dependencies

- GDSUM

8.1.27.4 Arguments

integers:

idnode	input	identity of current processor
natms	input	number of atoms in MD cell
ntpatm	input	number of unique atomic types
mxnode	input	total number of processors

integer arrays:

ltype	input	atomic type index (mxatms)
numtyp	input	the number of atoms of each atomic type (mxsvdw)

real arrays:

amsd	output	mean squared displacements (mxsvdw)
xx0	input	running sum of x increments (mxatms)
yy0	input	running sum of y increments (mxatms)
zz0	input	running sum of z increments (mxatms)
buffer	work	work array for global sum (mxbuff)

8.1.27.5 Comments

Use in conjunction with subroutine DIFFSN0. Called from within subroutine STATIC.

8.1.28 DIHFRC**8.1.28.1 Header records**

```

      subroutine dihfrc
      x  (idnode,imcon,mxnode,ntdihd,keyfce,dlrpot,epsq,engcpe,
      x  engdih,engsrp,rcut,vircpe,virdih,virsrp,keydih,listdih,
      x  ltype,lstvdw,buffer,cell,chge,fx,fy,fz,prmdih,xxx,yyy,
      x  zzz,xdab,ydab,zdab,xdbc,ydbc,zdbc,xdcd,ydcd,zdcd,vvv,ggg,
      x  stress)
c
c*****
c
c      dl_poly subroutine for calculating dihedral energy and force
c      terms in molecular dynamics.
c
c      version 3: scale factors for reduces electrostatic and vdw
c      1-4 interactions.
c
c      NOTE: assumes 1-4 interactions are in the exclude list
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith          march 1992
c
c      version 3  december 1993
c      author    - t. forester.
c
c      stress tensor added t.forester june 1995
c
c*****
c

```

8.1.28.2 Function

Calculates dihedral angle interactions.

8.1.28.3 Dependencies

- IMAGES
- ERROR
- GDSUM
- GSTATE

8.1.28.4 Arguments**integers:**

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors
ntdihd	input	total number of interactions in the system
keyfce	input	key for non-bonded force field

reals:

dlrpot	input	tabulation interval for short-range forces
epsq	input	relative dielectric constant
engcpe	in/out	electrostatic energy
engdih	output	dihedral interaction energy
engsrp	in/out	non-bonded potential energy
rcut	input	cut-off radius
vircep	in/out	electrostatic virial
virdih	output	dihedral interaction virial
virsrp	in/out	non-bonded virial

integer arrays:

keydih	input	key for dihedral potential (mxtdih)
listdih	input	index for potential parameters and atomic coordinates (mxdihd ,5)
ltype	input	atomic type index (mxatms)
lstvdw	input	pointer from potential index to interpolation tables, (mxvdw)

real arrays:

buffer	work	work array (mxbuff)
cell	input	MD cell vectors (9)
chge	input	partial charges (mxatms)
fxx	output	x component of force (mxatms)
fyy	output	y component of force (mxatms)
fzz	output	z component of force (mxatms)
prmdih	input	potential parameters (mxtdih , mxpdih)
xxx	input	x coordinate (mxatms)
yyy	input	y coordinate (mxatms)
zzz	input	z coordinate (mxatms)
xdab	work	x component of a-b inter atomic vector (msbad)
ydab	work	y component of a-b inter atomic vector (msbad)
zdab	work	z component of a-b inter atomic vector (msbad)
xdbc	work	x component of b-c inter atomic vector (msbad)
ydbc	work	y component of b-c inter atomic vector (msbad)
zdbc	work	z component of b-c inter atomic vector (msbad)
xcdcd	work	x component of c-d inter atomic vector (msbad)
ydcdd	work	y component of c-d inter atomic vector (msbad)

zdc	work	z component of c-d inter atomic vector (msbad)
vvv	input	short-range potential energy interpolation table (mxgrid,mxvdw)
ggg	input	short-range potential force interpolation table (mxgrid,mxvdw)
stress	in/out	stress tensor (9)

8.1.28.5 Comments

Available potential forms are given in the table:

Table 8.3: Dihedral potential parameters

keydih	potential type	prmdih(1-5)					functional form
1	Cosine	A	δ	n	W_{elec}	W_{Vdw}	$U = A [1 + \cos(n\phi - \delta)]$
2	Harmonic	k	ϕ_0		W_{elec}	W_{Vdw}	$U = \frac{1}{2}k(\phi - \phi_0)^2$

In each case the 1..4 electrostatic and short-range interactions are evaluated in this routine as the 1..4 interaction is in the ‘exclude’ list (see EXCLUDE). They are weighted by variables W_{elec} and W_{Vdw} respectively. *E.g.* to ignore these contributions the weighting parameters should be set to zero, for ‘normal’ interactions they should be set to 1.0.

Note that parameters for each 1..4 interaction must be specified. In the case where the same 1..4 pair is listed in more than one dihedral interaction the weighting parameters are taken from the *first* appropriate entry in the force field (see SYSDEF). Subsequent entries are set to zero to avoid accidental overcounting of interactions.

The calculation of the 1..4 short range interactions requires the interpolation arrays **vvv** and **ggg**. These arrays are defined in the subroutine FORGEN.

In the default version of DIHFRC a three point interpolation scheme is used for the Van der Waals potentials. Other versions using different interpolation schemes are also available. These include tabulations in r^2 or with 4-point interpolation formula. These are indicated by the suffixes **_RSQ.F** and **_4PT.F** on the source code files. See the notes on FORGEN for a discussion of the relative merits of the different interpolation schemes.

8.1.29 DLPFFT3**8.1.29.1 Header records**

```

      subroutine dlpfft3
      x  (ind,isw,ndiv1,ndiv2,ndiv3,key1,key2,key3,
      x  ww1,ww2,ww3,aaa)

c*****
c
c      dl-poly 3D fast fourier transform routine (in place)
c
c      copyright daresbury laboratory 1998
c
c      author w smith july 1998
c
c*****

```

8.1.29.2 Function

Calculates a radix 2 discrete fast Fourier transform in 3D.

8.1.29.3 Dependencies

- none

8.1.29.4 Arguments**integers:**

ind	input	initialise work arrays if ind>0
isw	input	use -i or +i in transform according to isw=-/+1
ndiv1	input	Number of data points in first direction
ndiv2	input	Number of data points in second direction
ndiv3	input	Number of data points in third direction

integer arrays:

key1	work	bit reversed address array for first direction (ndiv1)
key2	work	bit reversed address array for second direction (ndiv2)
key3	work	bit reversed address array for third direction (ndiv3)

complex arrays:

ww1	work	complex exponentials in first direction (ndiv1)
ww2	work	complex exponentials in second direction (ndiv2)
ww3	work	complex exponentials in third direction (ndiv3)
aaa	in/out	3D array to be Fourier transformed (ndiv1,ndiv2,ndiv3)

8.1.29.5 Comments

This is a radix 2 Fourier transform, so all of `ndiv1`, `ndiv2` and `ndiv3` must be powers of 2. The subroutine must be initialised first by a call with `ind=1`.

8.1.30 DLPOLY

8.1.30.1 Header records

```
c
c*****
c
c    dl_poly is an cclrc/ccp5 program package for the dynamical
c    simulation of molecular systems.
c
c    dl_poly is the property of the cclrc daresbury laboratory,
c    daresbury, warrington wa4 4ad. no part of the package may
c    be redistributed to third parties without the consent of
c    daresbury laboratory.
c
c    dl_poly is available free of charge to academic institutions
c    engaged in non-commercial research only. potential users not
c    in this category must consult the ccp5 program librarian at
c    daresbury to negotiate terms of use.
c
c    neither the cclrc, daresbury laboratory, ccp5 nor the authors
c    of this package claim that it is free from errors and do not
c    accept liability for any loss or damage that may arise from
c    its use. it is the users responsibility to verify that the
c    package dl_poly is fit for the purpose the user intends for
c    it.
c
c    users of this package are recommended to consult the dl_poly
c    user and reference manuals for the full terms and conditions
c    of its use.
c
c    authors: w.smith and t.r.forester 1995
c    copyright daresbury laboratory 1995
c
c    release 2.0
c
c*****
c
```

8.1.30.2 Function

Overall control of the simulation.

8.1.30.3 Dependencies

ANGFRC	BNDFRC	CORSHL	DIHFRC
DCELL	ERROR	EXITCOMMS	EXTNFLD
FCAP	FORCES	FORCESNEU	FREEZE
GDSUM	GSTATE	GSYNC	INITCOMMS
LRCMETAL	MACHINE	MULTIPLE	MULTIPLE_NSQ
MULTIPLENEU	NPT_B1	NPT_B3	NPT_H1
NPT_H3	NPTQ_B1	NPTQ_B2	NPTQ_B3
NPTQ_B4	NPTQ_H1	NPTQ_H2	NPTQ_H3
NPTQ_H4	NVE_1	NVEQ_1	NVEQ_2
NVT_B1	NVT_E1	NVT_H1	NVTQ_B1
NVTQ_B2	NVTQ_H1	NVTQ_H2	PARLINK
PARLST	PARLST_NSQ	PARNEULST	PARSET
PMF_1	QUATQNCH	REVIVE	RESULT
SHLFRC	SHLQNCH	SIMDEF	STATIC
STRUCOPT	SYSDEF	SYSGEN	TETHFRC
THBFRC	TIMCHK	TRAJECT	VERTEST
VSCALEG	XSCALE	ZDEN0	

8.1.30.4 Arguments

none

8.1.30.5 Comments

It is anticipated that users will write their own versions of this routine, calling the modules appropriate for their purposes. The sample program provides a reasonable array of functionality but is by no means exhaustive. Each user constructed variant should have at least the following features:

- machine identification (e.g. MACHINE)
- memory management/array allocation (e.g. PARSET)
- error handling routine (e.g. ERROR)
- define simulation parameters (e.g. SIMDEF)
- define force field (e.g. SYSDEF)
- define atomic configuration (e.g. SYSGEN)
- a simulation cycle consisting of :
 - evaluation of forces (e.g. FORCES, ANGFRC, BNDFRC, DIHFRC, EXTNFLD)

- integration of equations of motion (e.g. NVE_1)
 - collection of statistics (e.g. STATIC)
 - restart facility (e.g. REVIVE)
- summary of simulation (e.g. RESULT)
- exit from parallel harness (e.g. EXITCOMMS)

A sample makefile for a DL-POLY_2 program appears in the *build* sub-directory.

8.1.31 DLPOLY_RESPA

8.1.31.1 Header records

```

      program dlpoly_respa
c
c*****
c
c      dl_poly is an cclrc/ccp5 program package for the dynamical
c      simulation of molecular systems.
c
c      dl_poly is the property of the cclrc daresbury laboratory,
c      daresbury, warrington wa4 4ad. no part of the package may
c      be redistributed to third parties without the consent of
c      daresbury laboratory.
c
c      dl_poly is available free of charge to academic institutions
c      engaged in non-commercial research only. potential users not
c      in this category must consult the ccp5 program librarian at
c      daresbury to negotiate terms of use.
c
c      neither the cclrc, daresbury laboratory, ccp5 nor the authors
c      of this package claim that it is free from errors and do not
c      accept liability for any loss or damage that may arise from
c      its use. it is the users responsibility to verify that the
c      package dl_poly is fit for the purpose the user intends for
c      it.
c
c      users of this package are recommended to consult the dl_poly
c      user and reference manuals for the full terms and conditions
c      of its use.
c
c      This is a RESPA version of the code. The manipulation of the
c      ewald sum is based on the decomposition due to Procacci and
c      Marchi, J Chem Phys 104 (1996) 3003-3012.
c
c      authors: w.smith and t.r.forester 1996
c      copyright daresbury laboratory 1996
c
c*****
c

```

8.1.31.2 Function

Overall control of the simulation.

8.1.31.3 Dependencies

ANGFRC	BNDFRC	CORSHL	DCELL
DIHFRC	ERROR	EXITCOMMS	EXTNFLD
FAST_FORCE	FCAP	FREEZE	GDSUM
GSTATE	GSYNC	HALF_STEP	INITCOMMS
KINENG	MACHINE	PARLST_RESPA	PARNEULST_RESPA
RESPA_R	RESPA_V	RESULT	REVIVE
SHLFRC	SHLQNCH	SIMDEF_RESPA	STATIC
SYSDEF_RESPA	SYSGEN_RESPA	TETHFRC	THBFRC
TIMCHK	TRAJECT	VERTEST	VSCALE
WARNING	ZDEN0		

8.1.31.4 Arguments

none

8.1.31.5 Comments

This is a variant of DLPOLY to implement the RESPA multiple timestep algorithm [17, 18, 19]. It is anticipated that users will write their own versions of this routine, calling the modules appropriate for their purposes. The sample program provides a reasonable array of functionality but is by no means exhaustive. See notes on DLPOLY for general notes on the minimum functionality this routine should provide. Note that in this version velocities are integrated in the routine RESPA_V and coordinates in the routine RESPA_R.

The sample makefile, MAKEFILE.RESPA, for this program appears in the *build* sub-directory.

8.1.32 DUNI**8.1.32.1 Header records**

```

        function duni()
c
c*****
c
c      dl_poly random number generator based on the universal
c      random number generator of marsaglia, zaman and tsang
c      (stats and prob. lett. 8 (1990) 35-39.) it must be
c      called once to initialise parameters u,c,cd,cm
c
c      copyright daresbury laboratory 1992
c      author - w.smith          july 1992
c
c*****
c

```

8.1.32.2 Function

Random number generator. A uniform distribution on the interval $[0,1)$ is produced.

8.1.32.3 Dependencies

- none

8.1.32.4 Comments

DUNI must be called once to initialise internal parameters, thereafter, the function is called once for each random number required. This routine is universally transportable (allegedly).

8.1.33 ERFCGEN

8.1.33.1 Header records

```

      subroutine erfcgen
      x  (keyfce,alpha,drewd,rcut,erc,fer)

c*****
c
c      dlpoly routine for generating interpolation tables for
c      erfc and its derivative - for use with ewald sum.
c
c      copyright daresbury laboratory 1994
c      author t.forester dec 1994
c
c*****

```

8.1.33.2 Function

This subroutine calculates the interpolation arrays for the Ewald sum.

8.1.33.3 Dependencies

- none

8.1.33.4 Arguments

integers:

keyfce	input	force field key

reals:

alpha	input	Ewald convergence parameter, α
drewd	input	Spacing for interpolation arrays
rcut	input	real space cutoff

real arrays:

erc	output	interpolation array of function $\text{erfc}(\alpha r)/r$ (mxgrid)
fer	input	first derivative of interpolation array erc (mxgrid)

8.1.33.5 Comments

The functions are generated from a rational function approximation to the complementary error function [51]

$$\text{erfc}(x) \approx \exp(-x^2)[a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5] \quad (8.1)$$

where $t = 1/(1 + 0.3275911x)$ and the a_i have the values: $a_1 = 0.254829592$, $a_2 = -0.284496736$, $a_3 = 1.421413741$, $a_4 = -1.453152027$ and $a_5 = 1.061405429$.

A version of this routine, which tabulates in r^2 is available (ERFCGEN_RSQ).

8.1.34 ERROR**8.1.34.1 Header records**

```

      subroutine error(idnode,iode)
c
c*****
c
c      dl_poly subroutine for printing error messages and bringing
c      about a controlled termination of the program
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith march 1992.
c
c      warning - this routine terminates the job. user must ensure
c      that all nodes are informed of error condition before this
c      subroutine is called. e.g. using subroutine gstate().
c
c*****
c

```

8.1.34.2 Function

Writes error messages and terminates execution.

8.1.34.3 Dependencies

- none

8.1.34.4 Arguments

integers:

idnode	input	identity of current node
iode	input	key for error type

8.1.34.5 Comments

This routine should be called by all nodes simultaneously. Failure to do this will result in the program 'hanging', otherwise known as 'deadlock'.

8.1.35 EWALD1

8.1.35.1 Header records

```

      subroutine ewald1
      x  (idnode,mxnode,natms,imcon,kmax1,kmax2,kmax3,
      x  engcpe,vircpe,alpha,volm,epsq,cell,chge,xxx,
      x  yyy,zzz,fx,fyy,fzz,elc,emc,enc,els,ems,ens,
      x  ckc,cks,clm,slm,stress,buffer)

c
c*****
c
c      dl_poly subroutine for calculating coulombic forces in a
c      periodic system using ewald's method
c
c      parallel replicated data version (part 1)
c
c      copyright - daresbury laboratory 1992
c      author      - w. smith march 1992.
c
c      version 2
c      author      - t. forester april 1993
c      T3D optimizaion . t.forester july 1994
c
c      part 1 - reciprocal space terms (fourier part)
c
c      note - in loop over all k vectors k=2pi(ll/cl,mm/cl,nn/cl)
c      the values of ll,mm and nn are selected so that the symmetry of
c      reciprocal lattice is taken into account i.e. the following
c      rules apply.
c
c      ll ranges over the values 0 to kmax1 only.
c
c      mm ranges over 0 to kmax2 when ll=0 and over
c      -kmax2 to kmax2 otherwise.
c      nn ranges over 1 to kmax3 when ll=mm=0 and over
c      -kmax3 to kmax3 otherwise.
c
c      hence the result of the summation must be doubled at the end.
c
c      stress tensor added t.forester may 1994
c
c*****

```

c

8.1.35.2 Function

Calculation of reciprocal space contributions to the Coulombic energy and forces for the Ewald sum. Applicable to **periodic systems only**.

8.1.35.3 Dependencies

- INVERT
- DCELL
- GDSUM
- ERROR

8.1.35.4 Arguments**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in the MD cell
imcon	input	periodic boundaries control key
kmax1	input	number of k-vectors in the <i>a</i> reciprocal space direction
kmax2	input	number of k-vectors in the <i>b</i> reciprocal space direction
kmax3	input	number of k-vectors in the <i>c</i> reciprocal space direction

reals:

engcpe	output	Coulombic energy
vircpe	output	Coulombic virial
alpha	input	Ewald sum convergence variable
volm	input	volume of the MD cell
epsq	input	relative dielectric constant

real arrays:

cell	input	cell vectors (9).
chge	input	atomic electrostatic charge(mxatms)
xxx	input	x coordinate array (mxatms).
yyy	input	y coordinate array (mxatms)
zzz	input	z coordinate array (mxatms)
fxx	output	x component of force(mxatms)
fyy	output	y component of force(mxatms)
fzz	output	z component of force(mxatms)
elc	work	x component of the real part of $\exp(i.\mathbf{k}.\mathbf{r})$ (msatms,0:1)
emc	work	y component of the real part of $\exp(i.\mathbf{k}.\mathbf{r})$ (msatms,0:kmaxb)
enc	work	z component of the real part of $\exp(i.\mathbf{k}.\mathbf{r})$ (msatms,0:kmaxc)
els	work	x component of the imag. part of $\exp(i.\mathbf{k}.\mathbf{r})$ (msatms,0:1)

ems	work	y component of the imag. part of $\exp(i.\mathbf{k}.\mathbf{r})$ (<code>msatms,0:kmaxb</code>)
ens	work	z component of the imag. part of $\exp(i.\mathbf{k}.\mathbf{r})$ (<code>msatms,0:kmaxc</code>)
ckc	work	work array for reciprocal space sum (<code>msatms</code>)
cks	work	work array for reciprocal space sum (<code>msatms</code>)
clm	work	work array for reciprocal space sum (<code>msatms</code>)
slm	work	work array for reciprocal space sum (<code>msatms</code>)
stress	in/out	stress tensor (9)
buffer	work	work space (<code>mxbuff</code>)

8.1.35.5 Comments

In order to save memory the second dimension of the work arrays `elc` and `els` is `[0:1]`. The higher order terms ($l > 0$) of $\exp(k_a.r)$ are evaluated recursively from the $l = 1$ and $l = l - 1$ values and stored in the `(...,0)` elements of the arrays.

The amount of cpu time used in this routine is proportional to `kmax1 * kmax2 * kmax3`. The optimal choice of `alpha` and the `kmax` is system dependent - see section 3.3.5 for advice on selection of Ewald sum parameters.

Note: two versions of this routine exist : `EWALD1.F` and `EWALD1A.F`. They differ only in the parallelization strategy employed. `EWALD.F` parallelizes over k vectors while `EWALD1A.F` parallelises over atoms. It is anticipated that the first strategy will be best on parallel machines with fast communications while the second may be better for workstation clusters etc.

8.1.36 EWALD2

8.1.36.1 Header records

```

      subroutine ewald2
      x  (iatm,ik,engcpe,vircpe,drewd,rcut,epsq,
      x  ilist,chge,rsqdf,xdf,ydf,zdf,fx,fy,fz,erc,fer,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic forces in a
c      periodic system using ewald's method
c
c      parallel replicated data version (part 2)
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith march 1992.
c      T3d optimised. t.forester july 1994
c
c      part 2 - real space terms.
c
c      Tabulated potential in r space
c      3pt interpolation
c
c      t. forester March 1993
c      {stress tensor : t.forester june 1994}
c
c*****

```

8.1.36.2 Function

Calculates the Ewald sum real-space contribution to energy, virial and forces. For periodic systems with Coulombic forces.

8.1.36.3 Dependencies

- none

8.1.36.4 Arguments

integers:

iatm	input	index of central atom
ik	input	number of atoms in neighbour list array

reals:

engcpe	output	calculated contribution to electrostatic energy
vircpe	output	calculated contribution to electrostatic virial

drewd	input	tabulation interval for interpolation tables
rcut	input	cutoff for real space calculation
epsq	input	relative dielectric constant
integer arrays:		
ilist	input	list of secondary atom indices (mxlist)
real arrays:		
chge	input	ionic charges of atoms(mxatms)
rsqdf	input	square of interatomic distance (mxxdf)
xdf	input	x component of interatomic separation vector (mxxdf)
ydf	input	y component of interatomic separation vector (mxxdf)
zdf	input	z component of interatomic separation vector (mxxdf)
fx	output	x component of atomic forces(mxatms)
fy	output	y component of atomic forces(mxatms)
fz	output	z component of atomic forces(mxatms)
erc	input	interpolation array of function $\text{erfc}(\alpha r)/r$ (mxgrid)
fer	input	first derivative of interpolation array erc (mxgrid)
stress	in/out	stress tensor (9)

8.1.36.5 Comments

The algorithm requires values of $\text{erfc}(\alpha r)/r$ and its derivative. These are stored in the interpolation tables **erc** and **fer**. A three point interpolation scheme is used to approximate the functions between tabulation points. The tables are generated in the subroutine **ERFCGEN**.

In the default version of **EWALD2** a three point interpolation scheme is used to approximate the functions of r . Other versions using different interpolation schemes are also available. These include tabulations in r^2 or with 4-point interpolation formula. These are indicated by the suffixes **_RSQ.F** and **_4PT.F** on the source code files. See the notes on **FORGEN** for a discussion of the relative merits of the different interpolation schemes.

8.1.37 EWALD2_SW

8.1.37.1 Header records

```

      subroutine ewald2_sw
      x  (iatm,ik,iregn,mxregn,engcpe,vircpe,drewd,rcut1,rcut2,
      x  swlen,epsq,ilist,chge,rsqdf,xdf,ydf,zdf,fx,fy,fz,
      x  erc,fer,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic forces in a
c      periodic system using ewald's method and cubic switching functions
c
c      parallel replicated data version (part 2)
c
c      copyright - daresbury laboratory 1996
c      author (ewald2) - w. smith march 1992.
c      author (sw functions) t. forester june 1996
c
c      part 2 - real space terms.
c
c      Tabulated potential in r space
c      3pt interpolation
c
c*****
c

```

8.1.37.2 Function

Calculates the Ewald sum real-space contribution to energy, virial and forces. For periodic systems with Coulombic forces. A cubic switching function is applied to the potential.

8.1.37.3 Dependencies

- none

8.1.37.4 Arguments

integers:

iatm	input	index of the central site
ik	input	number of neighbours around the central site
iregn	input	Switching region index
mxregn	input	total number of Switching regions

reals:

drewd	input	tabulation interval for interpolation tables

engcpe	output	Coulombic energy
epsq	input	relative dielectric constant, ϵ_r
rcut1	input	lower cut off radius R_1
rcut2	input	upper cut off radius R_2
swlen	input	switching length λ
vircpe	output	Coulombic virial
integer arrays:		
ilist	input	indices of neighbours of the central site (mxxdf)
real arrays:		
chge	input	electrostatic charge (mxatms)
rsqdf	input	square of intersite distance (mxxdf)
xdf	input	x component of intersite vector (mxxdf)
ydf	input	y component of intersite vector (mxxdf)
zdf	input	z component of intersite vector (mxxdf)
fxx	output	x component of force (mxatms)
fyy	output	y component of force (mxatms)
fzz	output	z component of force (mxatms)
erc	input	interpolation array of function $\text{erfc}(\alpha r)/r$ (mxgrid)
fer	input	first derivative of interpolation array erc (mxgrid)
stress	in/out	stress tensor (9)

8.1.37.5 Comments

See comments on EWALD2 (for calculation of real space part of ewald sum) and COUL0_SW (for notes on the switching function used).

8.1.38 EWALD2NEU

8.1.38.1 Header records

```

      subroutine ewald2neu
      x  (ik,engcpe,vircpe,drewd,rcut,epsq,ilist,jlist,
      x  chge,rsqdf,xdf,ydf,zdf,fx,fyy,fzz,erc,fer,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic forces in a
c      periodic system using ewald's method for charge groups.
c
c      parallel replicated data version (part 2)
c
c      copyright - daresbury laboratory 1996
c      ewald2 author   - w. smith march 1992.
c      charge groups version  t.forester june 1996
c
c      part 2 - real space terms.
c      with charge group implementation
c
c      Tabulated potential in r space
c      3pt interpolation
c
c*****
c

```

8.1.38.2 Function

Calculates the Ewald sum real-space contribution to energy, virial and forces. For periodic systems with Coulombic forces.

8.1.38.3 Dependencies

- none

8.1.38.4 Arguments

integers:

ik	input	number of atoms in neighbour list arrays

reals:

engcpe	output	calculated contribution to electrostatic energy
vircpe	output	calculated contribution to electrostatic virial
drewd	input	tabulation interval for interpolation tables
rcut	input	cutoff for real space calculation

epsq	input	relative dielectric constant
integer arrays:		
ilist	input	list of first atom indices (mxxdf)
jlist	input	list of second atom indices (mxxdf)
real arrays:		
chge	input	ionic charges of atoms(mxatms)
rsqdf	input	square of interatomic distance (mxxdf)
xdf	input	x component of interatomic separation vector (mxxdf)
ydf	input	y component of interatomic separation vector (mxxdf)
zdf	input	z component of interatomic separation vector (mxxdf)
fx	output	x component of atomic forces(mxatms)
fy	output	y component of atomic forces(mxatms)
fz	output	z component of atomic forces(mxatms)
erc	input	interpolation array of function $\text{erfc}(\alpha r)/r$ (mxgrid)
fer	input	first derivative of interpolation array erc (mxgrid)
stress	in/out	stress tensor (9)

8.1.38.5 Comments

The algorithm requires values of $\text{erfc}(\alpha r)/r$ and its derivative. These are stored in the interpolation tables **erc** and **fer**. A three point interpolation scheme is used to approximate the functions between tabulation points. The tables are generated in the subroutine **ERFCGEN**.

8.1.39 EWALD3

8.1.39.1 Header records

```

      subroutine ewald3
      x  (iatm,ilst,engcpe,vircpe,alpha,epsq,nexatm,
      x  lexatm,chge,xdf,ydf,zdf,fxx,fyy,fzz,stress)
c
c*****
c
c      dl_poly subroutine for calculating coulombic forces in a
c      periodic system using ewald's method
c
c      parallel replicated data version (part 3)
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith dec 1992.
c
c      stress stensor added t.forester may 1994
c
c*****
c

```

8.1.39.2 Function

Calculates the intramolecular corrections to the Ewald sum as defined by the excluded atoms list. For periodic systems with Coulombic forces.

8.1.39.3 Dependencies

- none

8.1.39.4 Arguments

integers:

iatm	input	index of primary atom
ilst	input	pointer for iatm in excluded atom arrays

reals:

engcpe	output	calculated contribution to electrostatic energy
vircpe	output	calculated contribution to electrostatic virial
alpha	input	Ewald convergence parameter
epsq	input	relative dielectric constant, ϵ_r

integer arrays:

nexatm	input	last entry in excluded atom list for atom iatm (msatms)
lexatm	input	list of excluded atoms for atom iatm (msatms,mxexcl)

real arrays:

chge	input	atomic charges(mxatms)
xdf	input	x component of interatomic separation vector (mxxdf)
ydf	input	y component of interatomic separation vector (mxxdf)
zdf	input	z component of interatomic separation vector (mxxdf)
fxx	output	x component of atomic forces(mxatms)
fyy	output	y component of atomic forces(mxatms)
fzz	output	z component of atomic forces(mxatms)
stress	in/out	stress tensor (9)

8.1.39.5 Comments

The algorithm requires the function $\text{erf}(\alpha r)$ and its derivative. These are calculated from the same rational function approximations used in FORGEN where interpolation tables for $\text{erfc}(\alpha r)/r$ are constructed. Interpolation tables do not always provide sufficient accuracy as the interatomic distances required in this routine tend to be small.

8.1.40 EWALD4

8.1.40.1 Header records

```

      subroutine ewald4
      x  (iatm,ik,engcpe,vircpe,engcpl,vircpl,drewd,
      x  rcut,epsq,ilist,chge,rsqdf,xdf,ydf,zdf,
      x  fxx,fyy,fzz,flx,fly,flz,erc,fer,stress,stresl)
c
c*****
c
c      dl_poly subroutine for calculating coulombic forces in a
c      periodic system using ewald's method
c
c      modified to allow direct calculation of primary (short-range)
c      interactions for multiple-time step corrections
c
c      primary neighbours are taken out of the Ewald sum
c      electrostatics are evaluated directly instead
c
c      parallel replicated data version (part 2)
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith march 1992.
c
c      modified  - t. forester february 1993
c
c      part 2 - real space terms
c
c*****
c

```

8.1.40.2 Function

Calculates real space contributions to Ewald sum and the corresponding direct coulombic potential contributions. This routine is used only when all the following conditions are met:

- the multiple timestep algorithm is in use
- the current timestep corresponds to the first or second step of a multiple timestep iteration.
- the Ewald sum is being used to evaluate the Coulombic potential.
- the system is periodic (a prerequisite for the Ewald sum).

8.1.40.3 Dependencies:

- none

8.1.40.4 Arguments

integers:

iatm	input	index of the central site
ik	input	number of primary neighbours around the central site

reals

engcpe	output	Coulombic energy
virupe	output	Coulombic virial
engcpl	output	long time Coulombic energy
virupl	output	long time Coulombic virial
dreud	input	tabulation interval in interpolation tables
rcut	input	potential cut off radius
epsq	input	relative dielectric constant

integer arrays:

ilist	input	indices of primary neighbours of the central site (mxxdf)

real arrays:

chge	input	atomic charge (mxatms)
rsqdf	input	square of intersite vector (mxxdf)
xdf	input	x component of intersite vector (mxxdf)
ydf	input	y component of intersite vector (mxxdf)
zdf	input	z component of intersite vector (mxxdf)
fxx	output	x component of force (mxatms)
fyy	output	y component of force (mxatms)
fzz	output	z component of force (mxatms)
flx	output	x component of the long time force (mxatms)
fly	output	y component of the long time force (mxatms)
flz	output	z component of the long time force (mxatms)
erc	input	complimentary error function (potential) (mxgrid)
fer	input	complimentary error function (forces) (mxgrid)
stress	in/out	stress tensor (9)
stresl	in/out	“long-range” contribution to stress tensor (9)

8.1.40.5 Comments

Based on Forester’s multiple time-step strategy for the Ewald sum [41]. The algorithm requires values of $\text{erfc}(\alpha r)/r$ and its derivative. These are stored in the interpolation arrays **erc** and **fer**. The arrays are generated in the subroutine FORGEN.

In the default version of EWALD4 a three point interpolation scheme is used to approximate the functions between tabulation points and arrays **erc** and **fer** are tabulated as functions of r . Other versions using different interpolation schemes are available. These

include tabulations in r^2 or with 4-point interpolation formulas. These are indicated in the *source* directory by the additional qualifiers _RSQ and _4PT attached to the file names. See the notes on subroutine FORGEN for a discussion of relative merits of the different interpolation schemes.

8.1.41 EWALD_SPME

8.1.41.1 Header records

```

      subroutine ewald_spme
      x (idnode,mxnode,natms,imcon,kmax1,kmax2,kmax3,nospl,fplan,bplan,
      x  engcpe,vircpe,alpha,volm,epsq,nauxfft,key1,key2,key3,cell,chge,
      x  xxx,yyy,zzz,txx,tyy,tzz,fxx,fyy,fzz,stress,buffer,csp,ww1,ww2,
      x  ww3,bspvx,bspvy,bspvz,bsdxx,bsdyy,bsdzz,qqc,qqq,bscx,bscy,bscz,
      x  ffttable)

C*****
C
C      dl_poly subroutine for calculating coulombic forces in a
C      periodic system using the smooth particle mesh ewald method
C      due to Essmann et al J. Chem. Phys. 103 (1995) 8577.
C
C      parallel replicated data version (part 1)
C
C      copyright - daresbury laboratory 1998
C      author      - w. smith july 1998
C      additional FFT code - j. geronowicz sept 1999
C
C      part 1 - reciprocal space terms (fourier part)
C
C      $Author: wl $
C      $Date: 2001/06/12 12:51:59 $
C      $Revision: 1.5 $
C      $State: Exp $
C
C*****

```

8.1.41.2 Function

Calculation of reciprocal space contributions to the Coulombic energy and forces for the Ewald sum using the Smooth Particle Mesh Ewald (SPME) method [31]. **Applicable to 3D periodic systems only.**

8.1.41.3 Dependencies

- BSPCOE
- BSPGEN
- CCFFT3D (**CRAY** option only)
- CPY_RTC

- DCELL
- DCFT3 (**ESSL** option only)
- DLPFFT3 (default FFT routine)
- ELE_PROD
- ERROR
- FFTWND_F77_ONE (**FFTW** option only)
- FFTWND (**FFTW** option only)
- GDSUM
- INVERT
- SCL_CSUM
- SET_BLOCK
- SPL_EXP
- SPME_FOR
- ZZFFT3D (**SGICRAY** option only)

8.1.41.4 Arguments

integers:

<code>idnode</code>	input	identity of current processor
<code>mxnode</code>	input	total number of processors in use
<code>natms</code>	input	number of atoms in the MD cell
<code>imcon</code>	input	periodic boundaries control key
<code>kmax1</code>	input	dimension of FFT array in the <i>a</i> reciprocal space direction
<code>kmax2</code>	input	dimension of FFT array in the <i>b</i> reciprocal space direction
<code>kmax3</code>	input	dimension of FFT array in the <i>c</i> reciprocal space direction
<code>nospl</code>	input	order of B-spline functions required
<code>fplan</code>	input	integer pointer used for FFT_W routine
<code>bplan</code>	input	integer pointer used for FFT_W routine

reals:

<code>engcpe</code>	output	Coulombic energy
<code>vircpe</code>	output	Coulombic virial
<code>alpha</code>	input	Ewald sum convergence variable
<code>volm</code>	input	volume of the MD cell
<code>epsq</code>	input	relative dielectric constant

integer arrays:

<code>nauxfft</code>	input	buffer required by Cray FFT routine (4)

integer arrays

key1	work	inverse bit address array for first direction (kmaxd)
key2	work	inverse bit address array for second direction (kmaxe)
key3	work	inverse bit address array for third direction (kmaxf)

real arrays:

cell	input	cell vectors (9).
chge	input	atomic electrostatic charge(mxatms)
xxx	input	x coordinate array (mxatms).
yyy	input	y coordinate array (mxatms)
zzz	input	z coordinate array (mxatms)
txx	input	x working array (mxatms).
tyy	input	y working array (mxatms)
tzz	input	z working array (mxatms)
fxx	output	x component of force(mxatms)
fyy	output	y component of force(mxatms)
fzz	output	z component of force(mxatms)
stress	in/out	stress tensor (9)
buffer	work	work space (mxbuff)
csp	work	work space for B-spline calculation (mxspl)
ww1	work	Fourier complex x exponentials (kmaxd)
ww2	work	Fourier complex y exponentials (kmaxe)
ww3	work	Fourier complex z exponentials (kmaxf)
bsp _x	work	B-splines x (mxspme,mxspl)
bsp _y	work	B-splines y (mxspme,mxspl)
bsp _z	work	B-splines z (mxspme,mxspl)
bsd _x	work	B-spline x derivatives (mxspme,mxspl)
bsd _y	work	B-spline y derivatives (mxspme,mxspl)
bsd _z	work	B-spline z derivatives (mxspme,mxspl)
qqc	work	SPME charge array (kmaxd,kmaxe,kmaxf)
qqq	work	FT of qqc array (complex) (kmaxd,kmaxe,kmaxf)
bsc _x	work	B-splines x coefficients (kmaxd)
bsc _y	work	B-splines y coefficients (kmaxe)
bsc _z	work	B-splines z coefficients (kmaxf)
ffttable	work	buffer required by Cray FFT routine (mxftab)

8.1.41.5 Comments

The selection of the different 3D FFT routines is made at compile time (see section 3.2.1).
The default option is DLPFFT3.

8.1.42 EWCORR

8.1.42.1 Header records

```

      subroutine ewcorr
      x  (loglnk,idnode,imcon,mxnode,natms,alpha,engcpe,
      x  epsq,vircpe,lexatm,nexatm,cell,chge,fxx,fyy,fzz,
      x  stress,xdf,xxx,ydf,yyy,zdf,zzz)

c*****
c
c      dl_poly routine to evaluate self interaction correction
c      to ewald sum
c      assuming excluded atom list is for atomic decomposition
c
c      copyright daresbury laboratory june 1996
c      author t.forester june 96
c
c*****

```

8.1.42.2 Function

Calculates the intramolecular corrections to the Ewald sum as defined by the excluded atoms list. For periodic systems with Coulombic forces.

8.1.42.3 Dependencies

ewald3 images

8.1.42.4 Arguments

logicals:

loglnk	input	flag for link cells

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors
natms	input	number of atoms in system

reals:

alpha	input	Ewald convergence parameter
engcpe	output	calculated contribution to electrostatic energy
epsq	input	relative dielectric constant, ϵ_r
vircpe	output	calculated contribution to electrostatic virial

integer arrays:

lexatm	input	list of excluded atoms for atom iatm (msatms,mxexcl)

nexatm	input	last entry in excluded atom list for atom iatm (msatms)
real arrays:		
cell	input	MD cell (9)
chge	input	atomic charges(mxatms)
fxx	output	x component of atomic forces(mxatms)
fyy	output	y component of atomic forces(mxatms)
fzz	output	z component of atomic forces(mxatms)
stress	in/out	stress tensor (9)
xdf	work	x component of interatomic separation vector (mxxdf)
xxx	input	x coordinates(mxatms)
ydf	work	y component of interatomic separation vector (mxxdf)
yyy	input	y coordinates(mxatms)
zdf	work	z component of interatomic separation vector (mxxdf)
zzz	input	z coordinates(mxatms)

8.1.42.5 Comments

The routine EWCORR_NEU should be used in place of this one if neutral groups (charge groups) are in use.

8.1.43 EWCORR_NEU**8.1.43.1 Header records**

```

      subroutine ewcorr_neu
      x  (loglnk,idnode,imcon,mxnode,nneut,alpha,engcpe,
      x  epsq,vircpe,lexatm,nexatm,cell,chge,fxx,fyy,fzz,
      x  stress,xdf,xxx,ydf,yyy,zdf,zzz)

c*****
c
c      dl_poly routine to evaluate self interaction correction
c      to ewald sum
c      assuming excluded atom list is for charge group decomposition
c
c      copyright daresbury laboratory june 1996
c      author t.forester june 96
c
c*****

```

8.1.43.2 Function

Calculates the intramolecular corrections to the Ewald sum as defined by the excluded atoms list. For periodic systems with Coulombic forces.

8.1.43.3 Dependencies

ewald3_neu images

8.1.43.4 Arguments**logicals:**

loglnk	input	flag for link cells

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors
nneut	input	number of neutral groups in system

reals:

alpha	input	Ewald convergence parameter
engcpe	output	calculated contribution to electrostatic energy
epsq	input	relative dielectric constant, ϵ_r
vircpe	output	calculated contribution to electrostatic virial

integer arrays:

lexatm	input	list of excluded atoms for atom iatm (msatms,mxexcl)

nexatm	input	last entry in excluded atom list for atom iatm (msatms)
real arrays:		
cell	input	MD cell (9)
chge	input	atomic charges(mxatms)
fx	output	x component of atomic forces(mxatms)
fy	output	y component of atomic forces(mxatms)
fz	output	z component of atomic forces(mxatms)
stress	in/out	stress tensor (9)
xdf	work	x component of interatomic separation vector (mxxdf)
xxx	input	x coordinates(mxatms)
ydf	work	y component of interatomic separation vector (mxxdf)
yyy	input	y coordinates(mxatms)
zdf	work	z component of interatomic separation vector (mxxdf)
zzz	input	z coordinates(mxatms)

8.1.43.5 Comments

The routine EWCORR should be used in place of this one if an atomistic Verlet list is in use.

8.1.44 EXCLUDE

8.1.44.1 Header records

```

      subroutine exclude
x      (idnode,mxnode,nneut,ntpmls,keybnd,keyang,lexatm,lexsit,
x      lstang,lstbnd,lstcon,lstdih,lstin, nexatm,nexsit,numang,
x      numbonds,numcon,numdih,numinv,numsit,numgrp,listyp,lstgst,
x      numgsit,numshl,lstshl,prmdih)

c
c*****
c
c      dl_poly subroutine for constructing the excluded pair
c      interaction list of the system to be simulated
c
c      keybnd < 0 distance restraint so not excluded
c
c      copyright - daresbury laboratory 1992
c      author      - w. smith          june 1992
c
c      rigid body exclusions added : t.forester nov 1993
c      check on 1..4 scale factors : t.forester feb 1994
c      inversion terms added       : w.smith    jul 1996
c
c*****
c

```

8.1.44.2 Function

Generates lists of excluded non-bonded interactions. Interactions pertaining to a chemical bond, constraint bond, valence angle, dihedral angle or rigid body unit are excluded from the non-bonded interaction list.

8.1.44.3 Dependencies

- ERROR
- WARNING

8.1.44.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors
natms	input	number of atoms in MD cell

ntpmls	input	number of different types of molecule in the system
integer arrays:		
keybnd	input	key for chemical bond potentials (mxtbnd)
keyang	input	key for valence angle potentials (mxtang)
lexatm	output	the excluded interaction list (msatms,mxexcl)
lexsit	output	the excluded interaction list before minimisation (mxsite,mxexcl)
lstang	input	indices of sites in valence angles in the system (mxtang,3)
lstbnd	input	indices of sites in chemical bonds in the system (mxtbnd,2)
lstcon	input	indices of sites in constraint bonds in the system (mxtcon,2)
lstdih	input	indices of sites in dihedral angles in the system (mxt dih,4)
nexatm	output	number of entries in excluded interaction list for the central atom (msatms)
nexsit	output	number of excluded interactions before minimisation (mxsite)
numang	input	number of valence angles in a molecular type (mxtmls)
numbonds	input	number of chemical bonds in a molecular type (mxtmls)
numcon	input	number of constraint bonds in a molecular type (mxtmls)
numdih	input	number of dihedral angles in a molecular type (mxtmls)
nummols	input	number of molecules of each type (mxtmls)
numsit	input	number of sites in a molecular type (mxtmls)
numgrp	input	number of rigid units in a molecular type (mxtmls)
listyp	input	rigid group type (mxungp)
lstgst	input	site indices for a rigid group (mxungp,mxngp)
numgsit	input	number of sites in a rigid group (mxungp)
numshl	input	number of shell model sites in a molecular type (mxtmls)
lstshl	input	indices of shell and core sites (mxtshl,2)
real arrays:		
prmdih	input	parameters for dihedral potentials (mxt dih,mxpdih)

8.1.44.5 Comments

When a chemical bond or valence angle is defined, the atom pairs are placed in the exclude list only if the key (**keybnd** or **keyang**) is non-negative. A negative value flags that the interaction is a 'distance or angular restraint' rather than a chemical bond or valence angle, so the non-bonded interactions (electrostatic and Van-der Waals) between these sites should be evaluated.

The 1..4 atoms of a dihedral angle are also placed in the exclude list. Force fields tend to differ as to how the non-bonded terms for the 1..4 interaction should be treated. Some include both the electrostatic and van der Waals terms, some discard both of these, some retain one and not the other, and some retain all or some of the terms but scaled by a factor (normally in the range (0,1)). The most general way to handle these terms, compatible with a wide range of force fields is to define scaling factors for the electrostatic and van der Waals terms for each 1..4 interaction. The 1..4 interactions are thus *in* the exclude list and the non-bonded terms treated explicitly in the subroutine DIHFRC. In

`SYSDEF` the scaling parameters (`prmdih(i,4)` and `prmdih(i,5)`) are checked so that scaling factors are defined for a given 1..4 pair at most once.

8.1.45 EXCLUDE_ATOM

8.1.45.1 Header records

```

      subroutine exclude_atom
      x  (idnode,mxnode,natms,ntpmls,lexatm,
      x  lexsit,nexatm,nexsit,nummols,numsit)
c
c*****
c
c      dl_poly subroutine for constructing the excluded pair
c      interaction list of the system to be simulated
c      part 2
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith          june 1992
c
c*****
c

```

8.1.45.2 Function

Generates exclusion lists for pair interactions.

8.1.45.3 Dependencies

- none

8.1.45.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	total number of atoms in the MD cell
ntpmls	input	number of unique types of molecule

integer arrays:

lexatm	output	list of interactions to be excluded from the verlet list (msatms,mxexcl)
lexsit	input	the excluded interaction list before minimisation (mxsite,mxexcl)
nexatm	output	number of excluded atoms (msatms)
nexsit	input	number of excluded interactions before minimisation (mxsite)
nummols	input	number of molecules of a given type (mxtmls)
numsit	input	number of sites in a molecule (mxtmls)

real arrays:

8.1.45.5 Comments

8.1.46 EXCLUDE_LINK**8.1.46.1 Header records**

```

      subroutine exclude_link
      x  (idnode,mxnode,natms,ntpmls,lexatm,
      x  lexsit,nexatm,nexsit,nummols,numsit)
c
c*****
c
c      dl_poly subroutine for constructing the excluded pair
c      interaction list of the system to be simulated
c
c      part 2 - link cell implementation
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith             june 1992
c
c*****
c

```

8.1.46.2 Function

Generates exclusion lists for pair interactions for link cell implementation.

8.1.46.3 Dependencies

- none

8.1.46.4 Arguments**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	total number of atoms in the MD cell
ntpmls	input	number of unique types of molecule

integer arrays:

lexatm	output	list of interactions to be excluded from the verlet list (msatms,mxexcl)
lexsit	input	the excluded interaction list before minimisation (mxsite,mxexcl)
nexatm	output	number of excluded atoms (msatms)
nexsit	input	number of excluded interactions before minimisation (mxsite)
nummols	input	number of molecules of a given type (mxtmls)
numsit	input	number of sites in a molecule (mxtmls)

real arrays:

8.1.46.5 Comments

8.1.47 EXCLUDE_RESPA**8.1.47.1 Header records**

```

      subroutine exclude_respa
      x  (idnode,mxnode,natms,ntpmls,keybnd,keyang,lexatm,lexsit,
      x  lstang,lstbnd,lstcon,lstdih,nexatm,nexsit,numang,numbonds,
      x  numcon,numdih,numsit,numshl,lstshl,prmdih)
c
c*****
c
c      dl_poly subroutine for constructing the excluded pair
c      interaction list of the system to be simulated
c
c      keybnd < 0 distance restraint so not excluded
c
c      *** rigid body options removed ***
c
c      copyright - daresbury laboratory 1992
c      'exclude' author    - w. smith          june 1992
c      'respa' version author - t.forester     june 1996
c
c*****
c

```

8.1.47.2 Function

Generates lists of excluded non-bonded interactions. Interactions pertaining to a chemical bond, constraint bond, valence angle or dihedral angle are excluded from the non-bonded interaction list.

8.1.47.3 Dependencies

- ERROR
- WARNING

8.1.47.4 Arguments**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors
natms	input	number of atoms in MD cell
ntpmls	input	number of different types of molecule in the system

integer arrays:

keybnd	input	key for chemical bond potentials (mxtbnd)

keyang	input	key for valence angle potentials (mxtang)
lexatm	output	the excluded interaction list (msatms,mxexcl)
lexsit	output	the excluded interaction list before minimisation (mxsite,mxexcl)
lstang	input	indices of sites in valence angles in the system (mxtang,3)
lstbnd	input	indices of sites in chemical bonds in the system (mxtbnd,2)
lstcon	input	indices of sites in constraint bonds in the system (mxtcon,2)
lst dih	input	indices of sites in dihedral angles in the system (mxt dih,4)
nexatm	output	number of entries in excluded interaction list for the central atom (msatms)
nexsit	output	number of excluded interactions before minimisation (mxsite)
numang	input	number of valence angles in a molecular type (mxtmls)
numbonds	input	number of chemical bonds in a molecular type (mxtmls)
numcon	input	number of constraint bonds in a molecular type (mxtmls)
numdih	input	number of dihedral angles in a molecular type (mxtmls)
nummols	input	number of molecules of each type (mxtmls)
numsit	input	number of sites in a molecular type (mxtmls)
numshl	input	number of shell model sites in a molecular type (mxtmls)
lstshl	input	indices of shell and core sites (mxtshl,2)
real arrays:		
prmdih	input	parameters for dihedral potentials (mxt dih,mxp dih)

8.1.47.5 Comments

This routine differs from EXCLUDE only in that rigid body units are not considered.

8.1.48 EXCLUDENEU

8.1.48.1 Header records

```

      subroutine excludeneu
      x  (idnode,mxnode,nneut,lexatm,lexsit,
      x  neulst,nexatm,nexsit,nummols,numsit)
c
c*****
c
c      dl_poly subroutine for constructing the excluded pair
c      interaction list of the system to be simulated
c      part 2 - neutral group implementation
c
c      copyright - daresbury laboratory      1994
c      author    - t. forester              march 1994
c
c*****
c

```

8.1.48.2 Function

Generates exclusion lists for pair interactions for neutral group implementation.

8.1.48.3 Dependencies

- GSTATE

8.1.48.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
nneut	input	total number of neutral groups

integer arrays:

lexatm	output	list of interactions to be excluded from the verlet list (msatms,mxexcl)
lexsit	input	the excluded interaction list before minimisation (mxsite,mxexcl)
neulst	input	index of first site in charge group (mxneut)
nexatm	output	number of excluded atoms (msatms)
nexsit	input	number of excluded interactions before minimisation (mxsite)
nummols	input	number of molecules of a given type (mxtmls)
numsit	input	number of sites in a molecule (mxtmls)

real arrays:

8.1.48.5 Comments

8.1.49 EXITCOMMS**8.1.49.1 Header records**

```
subroutine exitcomms()
```

```

C*****
C
C      exitcomms: exit from communication harness
C
C      MPI version - t.forester may 1995
C      CPP version - w.smith may 1995
C
C*****

```

8.1.49.2 Function

Exits from communications harness and stops job.

8.1.49.3 Dependencies

- MPI exit calls
- EXIT()

8.1.49.4 Arguments

None.

8.1.49.5 Comments

8.1.50 EXTNFLD

8.1.50.1 Header records

```

      subroutine extnflld
      x  (idnode,imcon,keyfld,mxnode,natms,engfld,virfld,cell,
      x  chge,prmfld,xxx,yyy,zzz,vxx,vyy,vzz,fx,fyy,fzz,weight)

c*****
c
c      dl_poly routine for application of an external field
c
c      replicated data version / block data
c
c      copyright daresbury laboratory 1993
c      author -      t.forester october 1993
c      amended-      t.forester dec 1994
c
c*****

```

8.1.50.2 Function

Applies external field to the system.

8.1.50.3 Dependencies

- ERROR

8.1.50.4 Arguments

integers:

idnode	input	identity of current processor
imcon	input	periodic boundaries key (see IMAGES)
keyfld	input	field key
mxnode	input	total number of processors
natms	input	number of atoms in MD cell

reals:

engfld	output	potential energy of field
virfld	output	field contribution to the virial

real arrays:

cell	input	MD cell vectors (9)
chge	input	atomic charges (mxatms)
prmfld	input	field parameters (mxfld)
xxx	input	x coordinates (mxatms)
yyy	input	y coordinates (mxatms)
zzz	input	z coordinates (mxatms)

vxx	input	x component of velocity (mxatms)
vyy	input	y component of velocity (mxatms)
vzz	input	z component of velocity (mxatms)
fx	in/out	x component of force (mxatms)
fy	in/out	y component of force (mxatms)
fz	in/out	z component of force (mxatms)
weight	input	atomic masses (mxatms)

8.1.50.5 Comments

The available external fields are defined in the table where $\underline{F}$ is the force on an atom, q the charge, m the mass, and $\underline{v}$ the velocity. r is the displacement of the atom from the centre of the MD cell. Others may be added by defining functions for higher values of **keyfld**.

Table 8.4: External Fields

keyfld	potential type	prmfld(1-4)				functional form
1	Electric field	E_x	E_y	E_z		$\underline{F} = q.\underline{E}$
2	Oscillating Shear	A	n			$\underline{F}_x = A\cos(2n\pi.z/L_z)$
3	Continuous Shear	A	z_0			$ z > z_0: \underline{v}_x = (1/2)A(z /z)$
4	Gravitational Field	G_x	G_y	G_z		$\underline{F} = m.\underline{G}$
5	Magnetic Field	H_x	H_y	H_z		$\underline{F} = q(\underline{v} \times \underline{H})$
6	Containing Sphere	A	R_0	n	R_{cut}	$r > R_{\text{cut}}: \underline{F} = A(R_0 - r)^{-n}$

In all but the case of the “Containing sphere” the virial term (**virfld**) is zero as it is independent of particle position. The energy term (**engfld**) is also taken to be zero for lack of an absolute reference point. Note that in general the system will extract energy from the external field so (i) energy is not conserved in the simulation and (ii) a thermostat should be employed when integrating the equations of motion.

8.1.51 FBPFR**8.1.51.1 Header records**

```

      subroutine fbpfr
      x (idnode,mxnode,natms,imcon,rcutfb,engfbp,virfbp,
      x  latinx,ltype,lst,lct,link,lstfbp,ltpfbp,prmfbp,cell,
      x  xxx,yyy,zzz,fx,fy,fz,rcut4b,stress)
c
c*****
c
c      dl_poly subroutine for calculating four body inversion forces
c      arising from the inversion angle between three atoms around a
c      nominated central atom
c
c      Note: the subroutine converts coordinates to reduced units
c      to avoid a call to images.f. The link cell algorithm used
c      here necessitates a parallelepiped cell geometry
c
c      copyright - daresbury laboratory 1996
c      author    - w.smith july 1996
c
c*****
c

```

8.1.51.2 Function

Evaluates energy and forces from four body potentials of the inversion angle type.

8.1.51.3 Dependencies

- DCELL
- ERROR
- GDSUM
- GSTATE
- INVERT

8.1.51.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	total number of atoms in the MD cell

imcon	input	key for periodic boundary conditions
reals:		
rcutfb	output	four body potential cut-off
engfbp	output	four body potential energy
virfbp	output	four body potential virial
integer arrays:		
latinx	work	link cell number (mxatms)
ltype	input	index of atomic types (mxatms)
lst	work	number of atoms in link cell (mxcell)
lct	work	head of chain in link cell (mxcell)
link	work	pointer to next atom in link cell (mxatms)
lstfbp	input	pointer from four body potential to parameter table (mxfbp)
ltpfbp	input	key for three body potentials (mxfbp)
prmfbp	input	three body potential parameters (mxfbp , mxpang)
cell	input	MD cell vectors (9)
xxx	input	x coordinates (mxatms)
yyy	input	y coordinates (mxatms)
zzz	input	z coordinates (mxatms)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
rcut4b	input	three body potential cutoffs (mxfbp)
stress	in/out	stress tensor (9)
real arrays:		

8.1.51.5 Comments

Note this is **not** a general four body potential routine. It is restricted to inversion angle potentials only. Uses the link-cell algorithm to determine the atomic quadruplets. This routine is closely related to the INVFR.C.F routine.

8.1.52 FCAP**8.1.52.1 Header records**

```

      subroutine fcap
      x  (lfcap,idnode,mxnode,natms,fmax,temp,fx,fy,fz)

c*****
c
c      DLPOLY routine for limiting the absolute magnitude of
c      forces. Used in equilibration period only
c
c      copyright daresbury laboratory 1993
c
c      author -      t. forester march 1993
c      amended-      t. forester  sept 1994
c
c*****
c

```

8.1.52.2 Function

Limits the magnitude of the force on atomic sites.

8.1.52.3 Dependencies

- none

8.1.52.4 Arguments**logicals:**

lfcap	input	flag for force capping

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of sites in MD cell

reals:

fmax	input	maximum force permitted (in $kT/\text{\AA}$)
temp	input	simulation temperature

real arrays:

fx	in/out	x component of force (mxatms)
fy	in/out	y component of force (mxatms)
fz	in/out	z component of force (mxatms)

8.1.52.5 Comments

This subroutine is called only during the equilibration period of a simulation. The magnitude of the force on each site is permitted to be at most

$$\|\underline{f}_i\| = k_b T \mathbf{fmax} \quad (8.1)$$

If the atomic force is larger than this it is scaled down to the maximum permitted value. After this all atomic forces are shifted so that the total momentum in the system is conserved (i.e. $\sum_{i=1}^{n_{atoms}} \underline{F}_i = \underline{0}$). Energy is not conserved by force capping.

The principal use of force capping is to facilitate equilibration from initial configurations in which bad atomic overlaps (and hence very large forces) may be present. It is recommended that the value of **fmax** be at least 1000. Values lower than this can result in unphysical configurations being generated because of the effective truncation of the repulsive part of the short-range potentials.

8.1.53 FLSCAN

8.1.53.1 Header records

```

      subroutine fldscan
x     (idnode,mxnode,nfield,ntable,mxtbnd,mxtang,mxtdih,
x     mxngp,mxgrp,mxatms,mxgadm,mxneut,mxteth,mxtet1,mxsvdw,
x     mxvdw,mxn1,mxtbp,mxexcl,mxsite,mxbond,mxcons,mxangl,mxungp,
x     mxdihd,mxtshl,mxshl,mxpmf,mxspmf,mxtinv,mxinv,mxfbp,ngrid,
x     mxtmls,mxtcon,rctbp,rcfbp)
c
c*****
c
c     dl_poly routine for scanning the field file to determine the
c     required parameters
c
c     copyright - daresbury laboratory 1994
c     author    - w. smith november 1994
c
c     $Author: wl $
c     $Date: 1998/10/13 15:48:00 $
c     $Revision: 1.1 $
c     $State: Exp $
c
c*****
c

```

8.1.53.2 Function

Scans the DL_POLY_2 FIELD file to obtain estimates of required DL_POLY_2 array sizes for dynamic array allocation and passes the estimates back to the PARSET subroutine.

8.1.53.3 Dependencies

DBLSTR	ERROR	GETREC
INTSTR	LOWCASE	STRIP

8.1.53.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors
nfield	input	channel number of FIELD file
ntable	input	channel number of TABLE file
mxtbnd	output	max number of bond potentials

mxtang	output	max number of different bond angle potentials
mxt dih	output	max number of different dihedral potentials
mxngp	output	max number of sites in a rigid unit
mxgrp	output	max number of rigid body units in system
mxatms	output	max number of atoms in system
mxgatm	output	max number of sites in rigid unit
mxneut	output	max number of neutral groups + 1
mxteth	output	max number of tethered atom potentials
mxtet1	output	max number of tethered atoms in system
mxsvdw	output	max number of distinct atom types
mxvdw	output	max number of different pair potentials
mxn1	output	max number of sites in a neutral group
mxtbp	output	max number of three body potentials
mxexcl	output	max number of excluded interactions per atom
mxsite	output	max number of molecular sites
mxbond	output	max number of bond potentials in system
mxcons	output	max number constraint bonds in system
mxangl	output	max number of bond angle potentials in system
mxungp	output	max number of unique rigid body units
mxdihd	output	max number of dihedral angle potentials in system
mxtshl	output	max number of core-shell unit types
mxshl	output	max number of core shell units in system
mxpmf	output	max number of atoms in PMF constraints
mxspmf	output	max number of sites in a PMF unit
mxtinv	output	max number of inversion potential types
mxinv	output	max number of inversion potentials in system
mxfbp	output	max number of four body potentials
ngrid	output	max number of grid points in potential arrays
mxtmls	output	max number of molecular types in system
mxtcon	output	max number of specified bondlength constraints
reals:		
rctbp	output	max cutoff for three-body potentials
rcfbp	output	max cutoff for four-body potentials

8.1.53.5 Comments

8.1.54 FORCES

8.1.54.1 Header records

```

      subroutine forces
x  (lmetal,loglnk,lgofr,lzeql,idnode,imcon,keyfce,kmax1,
x  kmax2,kmax3,nhko,nlatt,mxnode,ntpvdw,natms,nstbgr,nstep,
x  nsteql,numrdf,nospl,fplan,bplan,alpha,dlrpot,drewd,engcpe,
x  engsrp,epsq,rcut,rvdw,vircpe,virsrp,volm,ilist,jlist,
x  lentry,lexatm,list,lstvdw,ltpvdw,ltype,nexatm,nauxfft,
x  key1,key2,key3,buffer,cell,chge,ckc,cks,clm,elc,els,emc,
x  ems,enc,ens,erc,fer,fx,fyy,fzz,ggg,rdf,rsqdf,slm,vvv,
x  xdf,xxx,ydf,yyy,zdf,zzz,stress,rho,elrcm,vlrcm,ewlbuf,
x  csp,qqc,txx,tyy,tzz,bsp,bspx,bspy,bspz,bsdx,bsdy,bsdz,qqq,
x  bscx,bscy,bscz,fftable,ww1,ww2,ww3,ahk,zzn,zzd,sss,hon,
x  dhnp,pp,znp,zgc,zgs,crn)
c
c*****
c
c      dl_poly subroutine for calculating interatomic forces
c      using the verlet neighbour list
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1992
c      author      - w. smith march 1992.
c      amended     - t. forester sept 1994
c      amended     - w. smith june 1995 for metal potentialsc
c
c      key:
c      keyfce = odd  ----- short range potentials calculated : srfrce
c      = 0,1  ----- no electrostatics
c      = 2,3  ----- Ewald sum                               : ewald1,2,3
c      = 4,5  ----- distance dependent dielectric           : coul2
c      = 6,7  ----- coulombic                                : coul0
c      = 8,9  ----- truncated and shifted coulombic         : coul4
c      = 10,11 ----- reaction field                          : coul3
c      = 12,13 ----- Smooth Particle Mesh Ewald             : ewald[_spme,2,3]
c      = 14,15 ----- Hautman-Klein-Ewald                   : hkewald1,2,3
c
c*****

```

8.1.54.2 Function

Controls calculation of non-bonded interactions for single time-step simulations.

8.1.54.3 Dependencies

COUL0	COUL1	COUL2	COUL3	COUL4
EWALD_SPME	EWALD1	EWALD2	EWALD3	GDSUM
HKEWALD1	HKEWALD2	HKEWALD3	IMAGES	SCDENS
SRFRCE	SUTTCHEN	RDF0		

8.1.54.4 Arguments**logicals:**

lmetal	input	flag for metal potentials
loglnk	input	flag for link cell algorithm
lgofr	input	flag for construction of rdfs
lzeql	input	flag for equilibration period

integers:

idnode	input	identity of current node
imcon	input	key for periodic boundary conditions
keyfce	input	key for non-bonded force field
kmax1	input	number of reciprocal space vectors in <i>a</i> direction
kmax2	input	number of reciprocal space vectors in <i>b</i> direction
kmax3	input	number of reciprocal space vectors in <i>c</i> direction
nhko	input	required order of HK-Ewald expansion
nlatt	input	lattice search parameter in HK-Ewald
mxnode	input	total number of nodes
ntpdw	output	number of Van der Waals potentials
natms	input	number of atoms in MD cell
nstbgr	input	interval for collection of r.d.f. data
nstep	input	current timestep
nsteql	input	number of equilibration steps
numrdf	in/out	counter for r.d.f. histograms
nospl	input	B-spline interpolation order
fplan	input	integer pointer (FFT)
bplan	input	integer pointer (FFT)

reals:

alpha	input	Ewald convergence parameter
dlrpot	input	distance increment for entries in potential look-up tables
drewd	input	distance increment for entries in Ewald sum look-up tables
engcpe	output	electrostatic energy
engsrp	output	non-bonded potential energy
epsq	input	relative dielectric constant
rcut	input	electrostatic potential cut-off radius
rvdw	input	Van der Waals potential cut-off radius
vircep	output	electrostatic virial

virsrp	output	non-bonded potential virial
volm	input	volume of MD cell
integer arrays:		
ilist	work	neighbourhood list for central site (mxxdf)
jlist	work	auxiliary neighbourhood list (mxxdf)
lentry	input	number of entries in Verlet list (msatms)
lexatm	input	excluded atoms list (msatms,mxexcl)
list	input	the Verlet list (msatms,mxlist)
lstvdw	input	index number of non-bonded potential (mxvdw)
ltpvdw	input	key for non-bonded potentials (mxvdw)
ltype	input	list of atom types (mxatms)
nexatm	input	number of excluded atoms (msatms)
nauxfft	input	FFT flags for FFTW implementation (4)
key1	work	inverse bit address array for first direction (kmaxd)
key2	work	inverse bit address array for second direction (kmaxe)
key3	work	inverse bit address array for third direction (kmaxf)
real arrays:		
buffer	work	work array (mxbuff)
cell	input	MD cell vectors (9)
chge	input	atomic charges (mxatms)
ckc	work	see EWALD1 (msatms)
cks	work	see EWALD1 (msatms)
clm	work	see EWALD1 (msatms)
elc	work	see EWALD1 (msatms,0:1)
els	work	see EWALD1 (msatms,0:1)
emc	work	see EWALD1 (msatms,0:kmaxb)
ems	work	see EWALD1 (msatms,0:kmaxb)
enc	work	see EWALD1 (msatms,0:kmaxc)
ens	work	see EWALD1 (msatms,0:kmaxc)
erc	input	interpolation array of function $\text{erfc}(\alpha r)/r$ (mxgrid)
fer	input	first derivative of interpolation array erc (mxgrid)
fxx	output	x component of force (mxatms)
fyy	output	y component of force (mxatms)
fzz	output	z component of force (mxatms)
ggg	input	interpolation table for non-bonded force (mxgrid,mxvdw)
rdf	in/out	histograms for radial distribution functions (mxrdf,mxvdw)
rsqdf	work	square of intersite distances (mxxdf)
slm	work	see EWALD1 (msatms)
vvv	input	interpolation table for non-bonded energy (mxgrid,mxvdw)
xdf	work	x component of intersite vector (mxxdf)
xxx	input	x coordinates (mxatms)
ydf	work	y component of intersite vector (mxxdf)
yyy	input	y coordinates (mxatms)
zdf	work	z component of intersite vector (mxxdf)

zzz	input	z coordinates (mxatms)
stress	in/out	stress tensor (9)
rho	in/out	Sutton Chen particle density (mxatms)
elrcm	in/out	long range correction to Sutton Chen energy (2)
vlrcm	in/out	long range correction to Sutton Chen virial (2)
ewlbuf	work	storage array for ewald sum evaluation (mxebuf)
csp	work	work space for B-spline calculation (mxspl)
qqc	work	SPME charge array (kmaxd,kmaxe,kmaxf)
txx	input	x working array (mxatms).
tyy	input	y working array (mxatms)
tzz	input	z working array (mxatms)
bspix	work	B-splines x (mxspme,mxspl)
bspix	work	B-splines y (mxspme,mxspl)
bspix	work	B-splines z (mxspme,mxspl)
bsdix	work	B-spline x derivatives (mxspme,mxspl)
bsdix	work	B-spline y derivatives (mxspme,mxspl)
bsdix	work	B-spline z derivatives (mxspme,mxspl)
qqq	work	FT of qqc array (complex) (kmaxd,kmaxe,kmaxf)
bscx	work	complex B-splines x coefficients (kmaxd)
bscy	work	complex B-splines y coefficients (kmaxe)
bscz	work	complex B-splines z coefficients (kmaxf)
ffftable	work	buffer required by Cray FFT routine (mxftab)
ww1	work	Fourier complex x exponentials (kmaxd)
ww2	work	Fourier complex y exponentials (kmaxe)
ww3	work	Fourier complex z exponentials (kmaxf)
ahk	input	coefficients of HK Ewald expansion (0:mxhko)
zzn	work	atom z-coordinate to power 2n (mxxdf)
zzd	work	atom z-coordinate to power 2n-1 (mxxdf)
sss	work	in-plane atomic separation (mxxdf)
hon	input	HK-Ewald real-space convergence function array (mxegrdrd,0:mxhko)
dhn	input	derivative of HK-Ewald convergence function array (mxegrdrd,0:mxhko)
pp	work	polynomial product array (2*mxhko)
znp	work	array of powers of atom z-coordinates (mxhke,0:2*mxhko)
zgc	work	global sums of products of powers of z-coordinate and cosine of $\exp(i.\mathbf{k}.\mathbf{s})$ (0:2*mxhko)
zgs	work	global sums products of powers of z-coordinate and sine of $\exp(i.\mathbf{k}.\mathbf{s})$ (0:2*mxhko)
crn	work	matrix of binomial coefficients (0:mxhko,0:mxhko)

8.1.54.5 Comments

8.1.55 FORCESNEU

8.1.55.1 Header records

```

      subroutine forcesneu
      x  (lgofr,lzeql,idnode,imcon,keyfce,mxnode,natms,
      x  nneut,nstbgr,nstep,nsteql,numrdf,dlrpot,engcpe,
      x  engsrp,epsq,rcut,rvdw,vircpe,virsrp,ilist,jlist,
      x  lentry,lexatm,list,lstfrz,lstneu,lstout,lstvdw,
      x  ltype,nexatm,neulst,buffer,cell,chge,fxx,fyy,
      x  fzz,ggg,rdf,rsqdf,stress,vvv,xxx,yyy,zzz,xdf,
      x  ydf,zdf,txx,tyy,tzz,txt,yyt,zzt,uxx,uyy,uzz)

C*****
C
C      dl_poly subroutine for calculating interatomic forces
C      using the verlet neighbour list
C      neutral group implemenation - no Ewald sum option
C
C      parallel replicated data version
C
C      copyright - daresbury laboratory 1992
C      author    - w. smith march 1992.
C
C      modified  - t. forester april 1993
C      key:
C
C      keyfce = odd  ----- short range potentials calculated : srfrce
C      = 0,1  ----- no electrostatics
C      = 2,3  ----- invalid
C      = 4,5  ----- Distant dependent dielectric           : coul2
C      = 6,7  ----- coulombic                               : coul0
C      = 8,9  ----- invalid
C      = 10,11 ----- reaction field                         : coul3
C
C*****
C

```

8.1.55.2 Function

8.1.55.3 Dependencies

COUL0NEU	COUL2NEU	COUL3NEU	GDSUM
GSTATE	NEUTLST	RDF0NEU	SRFRCENEU
ERROR			

8.1.55.4 Arguments**logicals:**

lgofr	input	flag for construction of rdfs
lzeql	input	flag for equilibration period

integers:

idnode	input	identity of current node
imcon	input	key for periodic boundary conditions
keyfce	input	key for non-bonded force field
mxnode	input	total number of nodes
natms	input	number of atoms in MD cell
nneut	input	number of charge groups in MD cell
nstbgr	input	interval for collection of r.d.f. data
nstep	input	current timestep
nsteql	input	number of equilibration steps
numrdf	in/out	counter for r.d.f. histograms

reals:

dlrpot	input	distance increment for entries in potential look-up tables
engcpe	output	electrostatic energy
engsrp	output	non-bonded potential energy
epsq	input	relative dielectric constant
rcut	input	electrostatic potential cut-off radius
rvdw	input	Van der Waals potential cut-off radius
vircep	output	electrostatic virial
virsrp	output	non-bonded potential virial

integer arrays:

ilist	work	first site of pair interaction (mxxdf)
jlist	work	second site of pair interaction (mxxdf)
lentry	input	number of entries in Verlet list (msatms)
lexatm	input	excluded atoms list (msatms,mxexcl)
list	input	the Verlet list (msatms,mxlist)
lstfrz	input	key for frozen atoms (mxatms)
lstneu	input	index of group to which site belongs (mxatms)
lstout	work	(mxatms)
lstvdw	input	index number of non-bonded potential (mxvdw)
ltype	input	list of atom types (mxatms)
nexatm	input	number of excluded atoms (msatms)
neulst	input	indices of first site in charge group (mxneut)

real arrays:

buffer	work	work array (mxbuff)
cell	input	MD cell vectors (9)
chge	input	atomic charges (mxatms)

<code>fx</code>	output	x component of force (<code>mxatms</code>)
<code>fy</code>	output	y component of force (<code>mxatms</code>)
<code>fz</code>	output	z component of force (<code>mxatms</code>)
<code>gg</code>	input	interpolation table for non-bonded force (<code>mxgrid,mxvdw</code>)
<code>rdf</code>	in/out	histograms for radial distribution functions (<code>mxrdf,mxvdw</code>)
<code>rsqdf</code>	work	square of intersite distances (<code>mxxdf</code>)
<code>stress</code>	in/out	stress tensor (9)
<code>vv</code>	input	interpolation table for non-bonded energy (<code>mxgrid,mxvdw</code>)
<code>xx</code>	input	x coordinates (<code>mxatms</code>)
<code>yy</code>	input	y coordinates (<code>mxatms</code>)
<code>zz</code>	input	z coordinates (<code>mxatms</code>)
<code>xd</code>	work	x component of intersite vector (<code>mxxdf</code>)
<code>yd</code>	work	y component of intersite vector (<code>mxxdf</code>)
<code>zd</code>	work	z component of intersite vector (<code>mxxdf</code>)
<code>tx</code>	work	(<code>mxatms</code>)
<code>ty</code>	work	(<code>mxatms</code>)
<code>tz</code>	work	(<code>mxatms</code>)
<code>xt</code>	work	(<code>mxatms</code>)
<code>yt</code>	work	(<code>mxatms</code>)
<code>zt</code>	work	(<code>mxatms</code>)
<code>ux</code>	work	(<code>mxatms</code>)
<code>uy</code>	work	(<code>mxatms</code>)
<code>uz</code>	work	(<code>mxatms</code>)

8.1.55.5 Comments

8.1.56 FORCESNEU_RESPA**8.1.56.1 Header records**

```

      subroutine forcesneu_respa
      x  (lgr,idnode,imcon,iregn,keyfce,mxnode,mxregn,natms,nneut,
      x  dlrpot,drewd,engcpe,engsrp,epsq,rcut,rvdw,vircpe,
      x  virsrp,ilist,jlist,lentry,lexatm,list,lstfrz,lstneu,
      x  lstout,lstvdw,ltype,neulst,nexatm,buffer,cell,chge,erc,
      x  fer,fx, fyy, fzz,ggg,rdf,rsqdf, stress,txx,tyy,tzz,uxx,uyy,
      x  uzz,vvv,xdf,xxt,xxx,ydf,yyt,yyy,zdf,zzt,zzz)

```

```

C*****
C
C      dl_poly subroutine for calculating interatomic forces
C      using the verlet neighbour list
C      neutral group implemenation for respa implementation
C
C      parallel replicated data version
C
C      copyright - daresbury laboratory 1996
C      author      t.forester      june 1996
C
C*****
C

```

8.1.56.2 Function**8.1.56.3 Dependencies**

COUL0NEU	COUL2NEU	COUL3NEU	ERROR
EWALD2NEU	GIMAX	GSTATE	IMAGES
NEUTLST_RESPA	RDF0NEU	SRFRCENEU	

8.1.56.4 Arguments**logicals:**

lgr	input	flag for construction of rdfs

integers:

idnode	input	identity of current node
imcon	input	key for periodic boundary conditions
iregn	input	identity of current zone
keyfce	input	key for non-bonded force field
mxnode	input	total number of nodes

mxregn	input	total number of zones
natms	input	number of atoms in MD cell
rneut	input	number of charge groups in MD cell
reals:		
dlrpot	input	distance increment for entries in potential look-up tables
drewd	input	distance increment for entries in ewald look-up tables
engcpe	output	electrostatic energy
engsrp	output	non-bonded potential energy
epsq	input	relative dielectric constant
rcut	input	electrostatic potential cut-off radius
rvdw	input	Van der Waals potential cut-off radius
vircep	output	electrostatic virial
virsrp	output	non-bonded potential virial
integer arrays:		
ilist	work	first site of pair interaction (mxxdf)
jlist	work	second site of pair interaction (mxxdf)
lentry	input	number of entries in Verlet list (msatms)
lexatm	input	excluded atoms list (msatms,mxexcl)
list	input	the Verlet list (msatms,mxlist)
lstfrz	input	key for frozen atoms (mxatms)
lstneu	input	index of group to which site belongs (mxatms)
lstout	work	(mxatms)
lstvdw	input	index number of non-bonded potential (mxvdw)
ltype	input	list of atom types (mxatms)
nexatm	input	number of excluded atoms (msatms)
neulst	input	indices of first site in charge group (mxneut)
real arrays:		
buffer	work	work array (mxbuff)
cell	input	MD cell vectors (9)
chge	input	atomic charges (mxatms)
erc	input	interpolation array of function $\text{erfc}(\alpha r)/r$ (mxgrid)
fer	input	first derivative of interpolation array erc (mxgrid)
fx	output	x component of force (mxatms)
fy	output	y component of force (mxatms)
fz	output	z component of force (mxatms)
ggg	input	interpolation table for non-bonded force (mxgrid,mxvdw)
rdf	in/out	histograms for radial distribution functions (mxrdf,mxvdw)
rsqdf	work	square of intersite distances (mxxdf)
stress	in/out	stress tensor (9)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)

uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
vvv	input	interpolation table for non-bonded energy (mxgrid,mxvdw)
xdf	work	x component of intersite vector (mxxdf)
xxt	work	(mxatms)
xxx	input	x coordinates (mxatms)
ydf	work	y component of intersite vector (mxxdf)
yyt	work	(mxatms)
yyy	input	y coordinates (mxatms)
zdf	work	z component of intersite vector (mxxdf)
zzt	work	(mxatms)
zzz	input	z coordinates (mxatms)

8.1.56.5 Comments

8.1.57 FORGEN**8.1.57.1 Header records**

```

      subroutine forgen
      x (ltable,idnode,ntpvdw,dlrpot,rcut,ltpvdw,prmvdw,vvv,ggg)
c
c*****
c
c      dl_poly subroutine for generating potential energy and
c      force arrays for van der waals forces only
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith may 1992.
c      modified for t3d dec 1994
c
c*****
c

```

8.1.57.2 Function

Generates interpolation tables for non-bonded potentials.

8.1.57.3 Dependencies

- ERROR

8.1.57.4 Arguments**logicals:**

ltable	input	flag for use of TABLE file

integers:

idnode	input	identity of current processor
ntpvdw	input	number of non-bonded potentials in system

reals:

dlrpot	output	distance increment in short-range tables
rcut	input	non-bonded potential cut-off

integer arrays:

ltpvdw	input	key for non-bonded potentials (mxvdw)

real arrays:

prmvdw	input	variables for non-bonded potentials (mxvdw,mxpvdw)
vvv	output	interpolation table for non-bonded potentials (mxgrid,mxvdw)
ggg	output	interpolation table for non-bonded force (mxgrid,mxvdw)

8.1.57.5 Comments

The algorithm stores the interpolation tables in arrays **vvv** and **ggg** for energy and force interpolation respectively.

In the default version of FORGEN the arrays **vvv** and **ggg** are tabulated as functions of r . A version with tabulations in r^2 is also available. This is found in the *source* directory with the name FORGEN_RSQ.

This default version is used in conjunction with the subroutines SRFRCE, EWALD2, EWALD4 and DIHFRG, while FORGEN_RSQ is used in conjunction with the subroutines SRFRCE_RSQ, EWALD2_RSQ, EWALD4_RSQ and DIHFRG_RSQ.

Tabulation in r^2 avoids the use of the square root function in evaluation of the non-bonded interactions, and thus typically decreases execution time by 10-15 %. Note that tabulation in r^2 usually requires more grid points (and hence more memory) than tabulation in r . This is to ensure sufficient accuracy is retained at small r .

A guide to the *minimum* number of grid points (**mxgrid**) required for interpolation in r (i.e. using routine FORGEN) to give good energy conservation in a simulation is:

$$\text{mxgrid} \geq 100(\text{rcut}/\text{rmin}) \quad (8.1)$$

where **rmin** is the *smallest* position minimum of the non-bonded potentials in the system. The parameter **mxgrid** is defined in the DL_PARAMS.INC file.

A guide to the *minimum* number of grid points required for interpolation in r^2 (i.e. using the routine FORGEN_RSQ) is:

$$\text{mxgrid} \geq 100(\text{rcut}/\text{rmin})^2 \quad (8.2)$$

where **rmin** is again the smallest position minimum of the non-bonded potentials in the system.

Because of the potential instability in interpolation techniques at small r^2 , interpolation between grid points in the routines SRFRCE_RSQ, EWALD2_RSQ, EWALD4_RSQ and DIHFRG_RSQ is done linearly.

For users in doubt as whether to use FORGEN or FORGEN_RSQ we recommend FORGEN. This is because tabulation in r is less demanding on memory requirements and less prone to inaccuracy should too small a value of **mxgrid**, or too large a value of **rcut**, be used. Tabulation in r is therefore the default option for DL_POLY, r^2 interpolation can be specified at compile time by 'making' the executable with the directive **rsq**. See section 3.2.1.3 for further details.

The other issue of concern to users is the choice of 3 or 4 point interpolation in calculating the potential energy and forces. The relative merits of 3 and 4 point interpolation are as follows: 4 point interpolation may permit a smaller number of grid points to be used in the interpolation tables thus saving on memory requirements. 3 point interpolation is considerable quicker than 4 point interpolation. The choice is therefore between speed and accuracy.

A utility program TABCHK is provided in the DL_POLY *utility* sub-directory to help users choose a sufficiently accurate interpolation scheme for their needs.

8.1.58 FORTAB**8.1.58.1 Header records**

```

      subroutine fortab
      x  (idnode,ntpvdw,ntpatm,mxnode,dlrpot,rcut,engunit,
      x  unqatm,lstvdw,ltpvdw,prmvdw,vvv,ggg,buffer)
c
c*****
c
c      dl_poly subroutine for reading potential energy and
c      force arrays for van der waals forces only
c
c      copyright - daresbury laboratory 1994
c      author    - w. smith march 1994
c
c*****
c

```

8.1.58.2 Function

Reads interpolation tables for non-bonded potentials from the TABLE file.

8.1.58.3 Dependencies

ERROR	GDSUM	GETREC	STRIP

8.1.58.4 Arguments**integers:**

idnode	input	identity of current processor
ntpvdw	input	number of non-bonded potentials in system
ntpatm	input	number of uniques atom types in system
mxnode	input	total number of processors in use

reals:

dlrpot	output	distance increment in short-range tables
rcut	input	non-bonded potential cut-off
engunit	input	unit of energy relative to internal units

character arrays:

unqatm	input	unique atom type names [a8] (mxsite)

integer arrays:

lstvdw	input	pointer from potential index to parameter table (mxvdw)
ltpvdw	input	key for non-bonded potentials (mxvdw)

real arrays:

prmvdw	input	variables for non-bonded potentials (mxvdw,mxpvdw)

vvv	output	interpolation table for non-bonded potentials (mxgrid,mxvdw)
ggg	output	interpolation table for non-bonded force (mxgrid,mxvdw)
buffer	work	(mxbuff)

8.1.58.5 Comments

See section 4.1.5 for a description of the TABLE file.

Note that a version of this routine is available for data tabulated as a function of r^2 (routine FORTAB_RSQ).

8.1.59 FREEZE**8.1.59.1 Header records**

```

      subroutine freeze
      x  (idnode,mxnode,natms,lstfrz,vxx,vyy,vzz,fx,fyy,fzz)

c*****
c
c      dlpoly routine to quench forces and velocities on 'frozen' atoms
c      replicated data version - blocked data
c
c      copyright daresbury laboratory 1994
c      author t.forester nov 1994
c
c*****

```

8.1.59.2 Function

Zeros force and velocity on frozen atoms prior to integration.

8.1.59.3 Dependencies

- none

8.1.59.4 Arguments**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	total number of atoms in the MD cell

integer arrays:

lstfrz	input	key for frozen atoms (mxatms)

real arrays:

vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
fx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)

8.1.59.5 Comments

The arrays are not merged across processors on exit from this routine.

8.1.60 GAUSS**8.1.60.1 Header records**

```

      subroutine gauss(natms,vxx,vyy,vzz)
c
c*****
c
c      dl_poly subroutine for constructing velocity arrays
c      with a gaussian distribution of unit variance.
c
c      based on the method described by Allen and Tildesley in
c      Computer Simulation of Liquids , Clarendon Press 1987 P347
c
c      note - this version uses a universal random number
c      generator, which generates pseudo-random numbers between
c      0 and 1. it is based on the algorithm of marsaglia, zaman
c      and tsang in: stats and prob. lett. 8 (1990) 35-39.
c
c      copyright daresbury laboratory 1992
c      author - w. smith          july 1992
c
c*****
c

```

8.1.60.2 Function

Produces a Gaussian distribution of velocities.

8.1.60.3 Dependencies

- DUNI

8.1.60.4 Arguments

integers:

natms	input	number of atoms in the MD cell

real arrays:

vxx	output	x component of velocity (natms)
vyy	output	y component of velocity (natms)
vzz	output	z component of velocity (natms)

8.1.60.5 Comments

A call to VSCALEG is required to ensure that the systems momentum is zero and to scale the kinetic temperature.

8.1.61 GDSUM

8.1.61.1 Header records

```

      subroutine gdsun(aaa,nnn,bbb)
c
c*****
c
c      dl_poly global summation subroutine for MPI - hypercube assumed
c      double precision version
c
c      copyright - daresbury laboratory 1995
c      author      - w. smith march 1992.
c      MPI version - t.forester may 1995
c      CPP version - w.smith may 1995
c
c*****
c

```

8.1.61.2 Function

Global summation of *real* * 8 arrays across a parallel computer.

8.1.61.3 Dependencies

- standard message send and receive routines
- MYNODE
- NODEDIM

8.1.61.4 Arguments

integers:

nnn in length of input array.

real arrays:

aaa	in/out	name of array to be summed (nnn)
bbb	work	buffer array used for summing (nnn)

8.1.61.5 Comments

This is **not** the Intel routine of the same name and the user is recommended to use the Intel version when running on Intel machines. The code contains a number of options for different parallel environments *viz* the Intel Hypercube, MPI, and the Cray T3D. The appropriate options are selected by c-preprocessing at compile time. With the exception of the MPI version the algorithm is valid for hypercubes only i.e. for 2^D processors.

8.1.62 GETREC**8.1.62.1 Header records**

```

      subroutine getrec(safe,idnode,mxnode,ifile,record)
c
c*****
c
c      dl_poly subroutine to read a character string on one node
c      and broadcast it to all other nodes
c
c      copyright daresbury laboratory 1994
c      author w.smith december 1994
c
c*****
c

```

8.1.62.2 Function

Reads a record (on node 0) and broadcasts information to other nodes.

8.1.62.3 Dependencies

- GISUM
- GSTATE
- GSYNC

8.1.62.4 Arguments**logicals:**

safe	output	error flag

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
ifile	input	input channel number

characters:

record	output	record [a100]

8.1.62.5 Comments

The record is broadcast to the nodes by converting the ascii characters to their integer representations, broadcasting the integers (via a global sum) and converting back to the character representation.

8.1.63 GIMAX

8.1.63.1 Header records

```

      subroutine gimax(aaa,nnn,bbb)
c
c*****
c
c      dl_poly global maximum subroutine for hypercube - MPI version
c      integer version
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith march 1992.
c      MPI version - t.forester may 1995
c      CPP version - w.smith may 1995
c
c*****
c

```

8.1.63.2 Function

Global maximum of integer arrays across a parallel computer.

8.1.63.3 Dependencies

- standard message passing routines
- MYNODE
- NODEDIM

8.1.63.4 Arguments

integers:

nnn	in	length of input array

integer arrays:

aaa	in/out	name of array to be analysed (*)
bbb	work	buffer array used for data transfer (*)

8.1.63.5 Comments

This is **not** the Intel routine of the same name and the user is recommended to use the Intel version when running on Intel machines. The code contains a number of options for different parallel environments *viz* the Intel Hypercube, MPI, and the Cray T3D. The appropriate options are selected by c-preprocessing at compile time. With the exception of the MPI version the algorithm is valid for hypercubes only i.e. for 2^D processors.

8.1.64 GISUM

8.1.64.1 Header records

```

      subroutine gisum(aaa,nnn,bbb)
c
c*****
c
c      dl_poly global summation subroutine for hypercube - MPI version
c      integer version
c
c      copyright - daresbury laboratory 1992
c      author      - w. smith march 1992.
c      MPI version - t.forester may 1995
c      CPP version - w.smith may 1995
c
c*****
c

```

8.1.64.2 Function

Global summation of integer arrays across a parallel computer.

8.1.64.3 Dependencies

- standard message passing routines
- MYNODE
- NODEDIM

8.1.64.4 Arguments

integers:

nnn	in	length of input array

integer arrays:

aaa	in/out	name of array to be summed (*)
bbb	work	buffer array used for summing (*)

8.1.64.5 Comments

This is **not** the Intel routine of the same name and the user is recommended to use the Intel version when running on Intel machines. The code contains a number of options for different parallel environments *viz* the Intel Hypercube, MPI, and the Cray T3D. The appropriate options are selected by c-preprocessing at compile time. With the exception of the MPI version the algorithm is valid for hypercubes only i.e. for 2^D processors.

8.1.65 GSTATE**8.1.65.1 Header records**

```

      subroutine gstate(check)
c
c*****
c
c      dl_poly global status subroutine for intel hypercube
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith      march 1992
c      MPI version - t. forester may 1995
c
c*****
c

```

8.1.65.2 Function

Logical *AND* of variable across all processors.

8.1.65.3 Dependencies

- GISUM

8.1.65.4 Arguments

logicals:

check	in/out	logical variable for global <i>AND</i>

8.1.65.5 Comments

If the variable **check** is **.false.** on any processor a value of **.false.** will be obtained. Users running DL_POLY on Intel platforms may consider using the Intel routine GLAND (global logical *AND*) in place of GSTATE. Note this version relies on use of the subroutine GISUM.

8.1.66 GSYNC**8.1.66.1 Header records**

```
subroutine gsync()
```

```

c*****
c
c    barrier / synchronization routine
c
c    MPI version - t.forester may 1995
c    CPP version - w.smith
c
c*****

```

8.1.66.2 Function

MPI and Cray-T3D mimic of the Intel function of the same name.

8.1.66.3 Dependencies

- global synchronisation call

8.1.66.4 Arguments

none

8.1.66.5 Comments

C-preprocessing is used at compile time to select the appropriate global synchronisation call.

8.1.67 HALF_STEP

8.1.67.1 Header records

```

      subroutine half_step
      x  (lneut,loglnk,idnode,imcon,iregn,keyfce,keyfld,kmax1,kmax2,
      x  kmax3,mxnode,mxregn,natms,nneut,nstep,nsteql,ntangl,ntbond,
      x  ntdihd,ntptbp,ntshl,ntteth,alpha,dlrpot,drewd,engang
      x  engbnd,engcp1,engcp2,engcp3,engdih,engfld,engshl,engsr1,
      x  engsr2,engsr3,engtbp,engtet,epsq,fmax,rcut,rcut1,rcut2,rcuttb,
      x  rvdw,swlen,temp,tstep0,tstep1,tstep2,tstep3,virang,virbnd,
      x  vircp1,vircp2,vircp3,virdih,virfld,virshl,virsr1,virsr2,
      x  virsr3,virtbp,virtet,volm,ilist,jlist,keyang,keybnd,keydih,
      x  keytet,lct,lentry1,lentry2,lentry3,lexatm,link,list1,list2,
      x  list3,listang,listbnd,listdih,listin,listshl,listtet,lst,
      x  lstfrz,lstneu,lsttbp,lstvdw,ltptbp,ltpvdw,ltype,neulst,
      x  nexatm,buffer,cell,chge,ckc,cks,clm,elc,els,emc,ems,enc,ens,
      x  erc,fer,fx1,fx2,fx3,fx0,fy1,fy2,fy3,fy0,fz1,fz2,fz3,fz0,ggg,
      x  prmang,prmbnd,prmdih,prmfld,prmshl,prmtbp,prmtet,rcut3b,rdf,
      x  rsqdf,slm,stres0,stres1,stres2,stres3,txx,tyy,tzz,uxx,uyy,
      x  uzz,vvv,vxx,vyy,vzz,weight,x dab,xdbc,x dcd,xdf,xxs,txt,xxx,
      x  ydab,ydbc,ydcd,ydf,yys,yyt,yyy,zdab,zdbc,zdcd,zdf,zzs,zzt,
      x  zzz)

```

```

C*****
C
C      dl_poly subroutine to advance velocities 1/2 a 'timestep'
C      with forces calculated from each of 4 distance class zones
C      (the zeroeth zone is the high frequency intramolecular
C      interactions)
C
C      copyright daresbury laboratory 1996
C      author   t.forester           june 1996
C
C*****

```

8.1.67.2 Function

Evaluates forces and advances velocities by half a timestep for each of the distance class zones.

8.1.67.3 Dependencies

ANGFRC	BNDFRC	DIHFRC	EWALD1
EWCORR	EWCORR_NEU	EXTNFLD	FREEZE

GDSUM	PAIR_FORCE_SW	RESPA_V	SHLFRC
TETHFRC	THBFRC		

8.1.67.4 Arguments

logicals:

lneut	input	flag for charge group implementation
loglnk	input	flag for link cell algorithm

integers:

idnode	input	identity of current node
imcon	input	key for periodic boundary conditions
iregn	input	identity of current zone
keyfce	input	key for non-bonded force field
keyfld	input	key for external field
kmax1	input	number of reciprocal space vectors in <i>a</i> direction
kmax2	input	number of reciprocal space vectors in <i>b</i> direction
kmax3	input	number of reciprocal space vectors in <i>c</i> direction
mxnode	input	total number of nodes
mxregn	input	total number of zones
natms	input	number of atoms in MD cell
nneut	input	number of charge groups in MD cell
nstep	input	running counter of completed MD cycles
nsteql	input	number of equilibration steps
ntangl	input	total number of valence angles in system
ntbond	input	total number of bonds in system
ntdihd	input	total number of dihedral angles in the system
ntptbp	input	number of three body potentials
ntshl	input	total number of core-shell units in system
ntteth	input	total number of tethered atoms in system

reals:

alpha	input	Ewald sum convergence variable
dlrpot	input	distance increment for short-range potential tabulation
drewd	input	distance increment for entries in Ewald sum look-up tables
engang	output	angular potential energy
engbnd	output	bond potential energy
engcp1	output	electrostatic energy from first zone
engcp2	output	electrostatic energy from second zone
engcp3	output	electrostatic energy from third zone
engdih	output	dihedral potential energy
engfld	output	external field potential energy
engshl	output	shell model potential energy
engsr1	output	non-bonded potential energy from first zone
engsr2	output	non-bonded potential energy from second zone

engsr3	output	non-bonded potential energy from third zone
engtbp	output	three-body potential energy
engtet	output	tethering potential energy
epsq	input	relative dielectric constant
fmax	input	force capping limit
rcut	input	electrostatic potential cut-off radius
rcut1	input	upper cutoff for zone 1
rcut2	input	upper cutoff for zone 2
rcuttb	input	three body potential cut-off
rvdw	input	Van der Waals potential cut-off radius
swlen	input	width of switching region between zones
temp	input	required simulation temperature
tstep0	input	timestep for inner zone
tstep1	input	timestep for first zone
tstep2	input	timestep for second zone
tstep3	input	timestep for third zone
virang	output	angular potential virial
virbnd	output	bond potential virial
vircp1	output	electrostatic virial from first shell
vircp2	output	electrostatic virial from second shell
vircp3	output	electrostatic virial from third shell
virdih	output	dihedral potential virial
virfld	output	external field virial
virshl	output	shell model virial
virsr1	output	non-bonded potential virial from first shell
virsr2	output	non-bonded potential virial from second shell
virsr3	output	non-bonded potential virial from third shell
virtbp	output	three body potential virial
virtet	output	tethering potential virial
volm	input	volume of MD cell
integer arrays:		
ilist	work	neighbourhood list for central site (mxxdf)
jlist	work	(mxxdf)
keyang	input	angle potential key (mxtang)
keybnd	input	chemical bond potential key (mxtbnd)
keydih	input	dihedral potential key (mxtdih)
keytet	input	tethering potential key (mxtdih)
lct	work	(mxcell)
lentry1	input	number of entries in first Verlet list (msatms)
lentry2	input	number of entries in second Verlet list (msatms)
lentry3	input	number of entries in third Verlet list (msatms)
lexatm	input	excluded atoms list (msatms,mxexcl)
link	work	(mxatms)
list1	input	the first Verlet list (msatms,mxlist1)

<code>list2</code>	input	the second Verlet list (<code>msatms,mxlist2</code>)
<code>list3</code>	input	the third Verlet list (<code>msatms,mxlist3</code>)
<code>listang</code>	input	angle potential index list (<code>mxangl,4</code>)
<code>listbnd</code>	input	bond potential index list (<code>mxbond,3</code>)
<code>listdih</code>	input	dihedral angle potential index list (<code>mxdihd,5</code>)
<code>listin</code>	work	(<code>mxatms</code>)
<code>listshl</code>	input	indices for core-shell units on current processor (<code>mxshl,3</code>)
<code>listteth</code>	input	indices of tethered atoms on current processor (<code>mxteth</code>)
<code>lst</code>	work	(<code>mxcell</code>)
<code>lstfrz</code>	input	key for frozen atoms (<code>mxatms</code>)
<code>lstneu</code>	input	index of group to which site belongs (<code>mxatms</code>)
<code>lsttbp</code>	input	pointer from three body potential to parameter table (<code>mxtbp</code>)
<code>ltptbp</code>	input	key for three body potentials (<code>mxtbp</code>)
<code>lstvdw</code>	input	index number of non-bonded potential (<code>mxvdw</code>)
<code>ltpvdw</code>	input	key for non-bonded potentials (<code>mxvdw</code>)
<code>ltype</code>	input	list of atom types (<code>mxatms</code>)
<code>neulst</code>	input	indice of first site in charge group (<code>mxneut</code>)
<code>nexatm</code>	input	number of excluded atoms (<code>msatms</code>)
real arrays:		
<code>buffer</code>	work	work array (<code>mxbuff</code>)
<code>cell</code>	input	MD cell vectors (9)
<code>chge</code>	input	atomic charges (<code>mxatms</code>)
<code>ckc</code>	work	see EWALD1 (<code>msatms</code>)
<code>cks</code>	work	see EWALD1 (<code>msatms</code>)
<code>clm</code>	work	see EWALD1 (<code>msatms</code>)
<code>elc</code>	work	see EWALD1 (<code>msatms,0:1</code>)
<code>emc</code>	work	see EWALD1 (<code>msatms,0:kmaxb</code>)
<code>enc</code>	work	see EWALD1 (<code>msatms,0:kmaxc</code>)
<code>els</code>	work	see EWALD1 (<code>msatms,0:1</code>)
<code>ems</code>	work	see EWALD1 (<code>msatms,0:kmaxb</code>)
<code>ens</code>	work	see EWALD1 (<code>msatms,0:kmaxc</code>)
<code>erc</code>	input	interpolation array of function $\text{erfc}(\alpha r)/r$ (<code>mxgrid</code>)
<code>fer</code>	input	first derivative of interpolation array <code>erc</code> (<code>mxgrid</code>)
<code>fx1</code>	output	x component of force from first shell (<code>mxatms</code>)
<code>fx2</code>	output	x component of force from second shell (<code>mxatms</code>)
<code>fx3</code>	output	x component of force from third shell (<code>mxatms</code>)
<code>fx0</code>	output	x component of force from inner shell (<code>mxatms</code>)
<code>fy1</code>	output	y component of force from first shell (<code>mxatms</code>)
<code>fy2</code>	output	y component of force from second shell (<code>mxatms</code>)
<code>fy3</code>	output	y component of force from third shell (<code>mxatms</code>)
<code>fy0</code>	output	y component of force from inner shell (<code>mxatms</code>)
<code>fz1</code>	output	z component of force from first shell (<code>mxatms</code>)
<code>fz2</code>	output	z component of force from second shell (<code>mxatms</code>)
<code>fz3</code>	output	z component of force from third shell (<code>mxatms</code>)

fz0	output	z component of force from inner shell (mxatms)
ggg	input	interpolation table for non-bonded force (mxgrid,mxvdw)
prmang	input	valence angle potential parameters (mxtang,mxpang)
prmbnd	input	chemical bond potential parameters (mxtbnd,mxpbnd)
prmdih	input	dihedral angle potential parameters (mxt dih,mxp dih)
prmfld	input	external field potential parameters (mxfld)
prmsl	input	core-shell potential parameters (mxtsl)
prmtbp	input	three body potential parameters (mxtbp,mxpang)
prmtet	input	tethering potential parameters (mxteth,mxpbnd)
rcut3b	input	three body potential cutoffs (mxtbp)
rdf	in/out	histograms for radial distribution functions (mxrdf,mxvdw)
rsqdf	work	square of intersite distances (mxxdf)
slm	work	see EWALD1 (msatms)
stres0	in/out	stress tensor from inner zone (9)
stres1	in/out	stress tensor from first zone (9)
stres2	in/out	stress tensor from second zone (9)
stres3	in/out	stress tensor from third zone (9)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
vvv	input	interpolation table for non-bonded energy (mxgrid,mxvdw)
vxx	in/out	x component of velocities (mxatms)
vyy	in/out	y component of velocities (mxatms)
vzz	in/out	z component of velocities (mxatms)
weight	input	atomic mass (mxatms)
xdab	work	(msbad)
xdbc	work	(msbad)
xdcd	work	(msbad)
xdf	work	(mxxdf)
xxs	input	x component of atomic position at $t = 0$ (mxatms)
xxt	work	work (mxatms)
xxx	input	x coordinates (mxatms)
ydab	work	(msbad)
ydbc	work	(msbad)
ydcd	work	(msbad)
ydf	work	(mxxdf)
sys	input	y component of atomic position at $t = 0$ (mxatms)
yyt	work	work (mxatms)
yyy	input	y coordinates (mxatms)
zdab	work	(msbad)
zdbc	work	(msbad)

zdc	work	(msbad)
zdf	work	(mxxdf)
zds	input	z component of atomic position at $t = 0$ (mxatms)
zdt	work	work (mxatms)
zdd	input	z coordinates (mxatms)

8.1.67.5 Comments

This routine is designed to work with DLPOLY_RESPA.

8.1.68 HKEWALD1

8.1.68.1 Header records

```

      subroutine hkewald1
      x  (idnode,mxnode,natms,imcon,nhko,kmax1,kmax2,
      x  engcpe,vircpe,alpha,epsq,cell,ahk,chge,xxx,
      x  yyy,zzz,fx,fyy,fzz,elc,emc,els,ems,ckc,cks,
      x  stress,crn,pp,znp,zgc,zgs,buffer)

C*****
C
C      dl_poly subroutine for calculating coulombic forces in a
C      periodic system using Hautman Klein Ewald method
C
C      parallel replicated data version
C
C      copyright - daresbury laboratory 2000
C      author    - w. smith february 2000
C
C      part 1 - reciprocal space terms (fourier part)
C
C      note - in loop over all k vectors k=2pi(ll/cl,mm/cl)
C      the values of ll and mm are selected so that the symmetry of
C      reciprocal lattice is taken into account i.e. the following
C      rules apply.
C
C      ll ranges over the values 0 to kmax1 only.
C
C      mm ranges over 1 to kmax2 when ll=0 and over
C      -kmax2 to kmax2 otherwise.
C
C*****

```

8.1.68.2 Function

HKEWALD1 calculates of reciprocal space contributions to the Coulombic energy and forces for the Hautman Klein Ewald sum [32]. Applicable to **2D periodic systems only**.

8.1.68.3 Dependencies

- INVERT
- DCELL

- GDSUM
- ERROR
- WARNING

8.1.68.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in the MD cell
imcon	input	periodic boundaries control key
nhko	input	required order of HK-Ewald expansion
kmax1	input	number of k-vectors in the <i>a</i> reciprocal space direction
kmax2	input	number of k-vectors in the <i>b</i> reciprocal space direction

reals:

engcpe	output	Coulombic energy
vircep	output	Coulombic virial
alpha	input	Ewald sum convergence variable
volm	input	volume of the MD cell
epsq	input	relative dielectric constant

real arrays:

cell	input	cell vectors (9).
ahk	input	coefficients of HK Ewald expansion (0:mxhko)
chge	input	atomic electrostatic charge(mxatms)
xxx	input	x coordinate array (mxatms)
yyy	input	y coordinate array (mxatms)
zzz	input	z coordinate array (mxatms)
fxx	output	x component of force(mxatms)
fyy	output	y component of force(mxatms)
fzz	output	z component of force(mxatms)
elc	work	x component of the real part of $\exp(i.\mathbf{k}.\mathbf{s})$ (msatms,0:1)
emc	work	y component of the real part of $\exp(i.\mathbf{k}.\mathbf{s})$ (msatms,0:kmaxb)
els	work	x component of the imag. part of $\exp(i.\mathbf{k}.\mathbf{s})$ (msatms,0:1)
ems	work	y component of the imag. part of $\exp(i.\mathbf{k}.\mathbf{s})$ (msatms,0:kmaxb)
ckc	work	work array for reciprocal space sum (msatms)
cks	work	work array for reciprocal space sum (msatms)
stress	in/out	stress tensor (9)
crn	work	matrix of binomial coefficients (0:mxhko,0:mxhko)
pp	work	polynomial product array (2*mxhko)
znp	work	array of powers of atom z-coordinates (mxhke,0:2*mxhko)
zgc	work	global sums of products of powers of z-coordinate and cosine of $\exp(i.\mathbf{k}.\mathbf{s})$ (0:2*mxhko)
zgs	work	global sums products of powers of z-coordinate

		and sine of $\exp(i.\mathbf{k}.\mathbf{s})$ (0:2*mxhko)
buffer	work	work space (mxbuff)

8.1.68.5 Comments

In order to save memory the second dimension of the work array **elc** and **els** is [0:1]. The higher order terms ($l > 0$) of $\exp(k_a.r)$ are evaluated recursively from the $ll = 1$ and $ll = l - 1$ values and stored in the $(..,0)$ elements of the arrays.

The optimal choice of **alpha** and **kmax** is system dependent - see section 3.3.5 for advice on selection of Ewald sum parameters. Variable **nhko** is usually set to 1.

8.1.69 HKEWALD2

8.1.69.1 Header records

```

      subroutine hkewald2
      x  (idnode,mxnode,nhko,nlatt,imcon,natms,engcpe,
      x  vircpe,drewd,rcut,epsq,cell,chge,ahk,zzn,zzd,
      x  hon,dhn,xxx,yyy,zzz,fx,fyy,fzz,stress,xdf,ydf,
      x  zdf,sss,rsqdf)

c*****
c
c      dl_poly subroutine for calculating real-space contributions to
c      the hautman-klein-ewald electrostatic method
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 2000
c      author    - w. smith may 2000
c
c*****
c

```

8.1.69.2 Function

HKEWALD2 calculates the real space contribution to HK-Ewald energy, virial and forces. For **2D periodic systems** with Coulombic forces.

8.1.69.3 Dependencies

- ERROR
- INVERT

8.1.69.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
nhko	input	required order of HK-Ewald expansion
nlatt	input	required lattice search parameter
imcon	input	periodic boundary condition
natms	input	number of atoms in system

reals:

engcpe	output	calculated contribution to electrostatic energy
vircpe	output	calculated contribution to electrostatic virial

drewd	input	tabulation interval for interpolation tables
rcut	input	cutoff for real space calculation
epsq	input	relative dielectric constant
real arrays:		
chge	input	ionic charges of atoms(mxatms)
ahk	input	coefficients of HK Ewald expansion (0:mxhko)
zzn	work	atom z-coordinate to power 2n (mxxdf)
zzd	work	atom z-coordinate to power 2n-1 (mxxdf)
hon	input	HK-Ewald real-space convergence function array (mxegr d, 0:mxhko)
dhn	input	derivative of HK-Ewald convergence function array (mxegr d, 0:mxhko)
xxx	input	x coordinate array (mxatms)
yyy	input	y coordinate array (mxatms)
zzz	input	z coordinate array (mxatms)
fxx	output	x component of atomic forces(mxatms)
fyy	output	y component of atomic forces(mxatms)
fzz	output	z component of atomic forces(mxatms)
stress	in/out	stress tensor (9)
xdf	work	x component of interatomic separation vector (mxxdf)
ydf	work	y component of interatomic separation vector (mxxdf)
zdf	work	z component of interatomic separation vector (mxxdf)
sss	work	in-plane atomic separation (mxxdf)
rsqdf	input	square of interatomic distance (mxxdf)

8.1.69.5 Comments

The algorithm requires values of the convergence function **hon** and its derivative **dhn**. These are provided as numerical tables. A three point interpolation scheme is used to approximate the functions between tabulation points. The tables are generated in the subroutine **HKGEN**.

Note that the sum over atom pairs includes not only atoms from within the same simulation cell, but also from neighbouring cells (if **nlatt=1**) or next neighbours (if **nlatt=2**). Thus when **nlatt>0**, the cutoff radius becomes effectively $(2*\text{nlatt}+1)*\text{rcut}$, where **rcut** is the simulation cutoff applied to other nonbonded interactions. The sums in **HKEWALD2** also run over the HK-Ewald expansion terms to order **nhko**.

8.1.70 HKEWALD3

8.1.70.1 Header records

```

      subroutine hkewald3
      x  (iatm,ik,engcpe,vircpe,epsq,nexatm,lexatm,
      x  chge,xdf,ydf,zdf,fx,fyy,fzz,stress)
c
c*****
c
c      dl_poly subroutine for calculating exclusion corrections to
c      the hautman-klein-ewald electrostatic method
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 2000
c      author    - w. smith may 2000
c
c*****

```

8.1.70.2 Function

HKEWALD3 calculates corrections to the HK-Ewald method based on the excluded atoms list, which list pairs of atoms that should not interact via a nonbonded potential.

8.1.70.3 Dependencies

- NONE

8.1.70.4 Parameters

integers:

iatm	input	index of primary atom
ik	input	pointer for iatm in excluded atoms list

reals:

engcpe	output	calculated contribution to electrostatic energy
vircpe	output	calculated contribution to electrostatic virial
epsq	input	relative dielectric constant

integer arrays:

nexatm	input	last entry in excluded atom list for atom iatm (msatms)
lexatm	input	list of excluded atoms for atom iatm (msatms,mxexcl)

real arrays:

chge	input	ionic charges of atoms(mxatms)
xdf	work	x component of interatomic separation vector (mxxdf)

ydf	work	y component of interatomic separation vector (mxxdf)
zdf	work	z component of interatomic separation vector (mxxdf)
fx	output	x component of atomic forces(mxatms)
fy	output	y component of atomic forces(mxatms)
fz	output	z component of atomic forces(mxatms)
stress	in/out	stress tensor (9)

8.1.70.5 Comments

The correction applied here is a straightforward extraction of the offending $1/r$ terms.

8.1.71 HKEWALD4**8.1.71.1 Header records**

```

      subroutine hkewald4
      x  (iatm,ik,engcpe,vircpe,engcpl,vircpl,
      x  rcut,epsq,ilist,chge,rsqdf,xdf,ydf,zdf,
      x  fxx,fyy,fzz,flx,fly,flz,stress,stresl)
c
c*****
c
c      dl_poly subroutine for calculating coulombic forces in a
c      periodic system using the hautman-klein-ewald method
c
c      modified to allow direct calculation of primary (short-range)
c      interactions for multiple-time step corrections
c
c      primary neighbours are taken out of the Ewald sum
c      electrostatics are evaluated directly instead
c
c      parallel replicated data version - real space terms
c
c      copyright - daresbury laboratory 2000
c      author    - w. smith july 2000
c
c*****

```

8.1.71.2 Function

Calculates direct real space contributions to the HK-Ewald sum. This routine is used only when all the following conditions are met:

- the multiple timestep algorithm is in use.
- the current timestep corresponds to the first or second step of a multiple timestep iteration.
- the HK-Ewald sum is being used to evaluate the Coulombic potential.
- the system is 2D periodic (a prerequisite for the HK-Ewald sum).

8.1.71.3 Dependencies

- NONE

8.1.71.4 Arguments**integers:**

iatm	input	index of primary atom
ik	input	pointer for iatm in excluded atoms list

reals:

engcpe	output	calculated contribution to electrostatic energy
vircpe	output	calculated contribution to electrostatic virial
engcpl	output	long range contribution to electrostatic energy
vircpl	output	long range contribution to electrostatic virial
rcut	input	applied real-space cut off
epsq	input	relative dielectric constant

integer arrays:

ilist	input	list of atoms interacting with atom iatm (msatms)

real arrays:

chge	input	ionic charges of atoms(mxatms)
rsqdf	input	square of interatomic distance (mxxdf)
xdf	work	x component of interatomic separation vector (mxxdf)
ydf	work	y component of interatomic separation vector (mxxdf)
zdf	work	z component of interatomic separation vector (mxxdf)
fx	output	x component of atomic forces(mxatms)
fy	output	y component of atomic forces(mxatms)
fz	output	z component of atomic forces(mxatms)
flx	output	x component of long range forces(mxatms)
fly	output	y component of long range forces(mxatms)
flz	output	z component of long range forces(mxatms)
stress	in/out	stress tensor (9)
stresl	in/out	long range stress tensor (9)

8.1.71.5 Comments

This subroutine represents a simple sum of Coulomb interactions within the cutoff radius **rcut**. No interpolation is used.

8.1.72 HKGEN

8.1.72.1 Header records

```

      subroutine hkgen
      x  (idnode,nhko,nlatt,alpha,drewd,rcut,ahk,hon,fon,dhn)

c*****
c
c      dl_poly subroutine for generating convergence function
c      arrays for hautman klein ewald method (up to order 3 only)
c
c      copyright - daresbury laboratory 2000
c      author    - w. smith february 2000
c
c*****

```

8.1.72.2 Function

This subroutine calculates the real space convergence function arrays for the HK-Ewald sum.

8.1.72.3 Dependencies

- ERROR
- WARNING

8.1.72.4 Arguments

integers:

idnode	input	identity of current processor
nhko	input	required order of HK-Ewald expansion
nlatt	input	real space lattice search parameter

reals:

alpha	input	Ewald convergence parameter, α
drewd	input	spacing for interpolation arrays
rcut	input	real space cutoff

real arrays:

ahk	output	coefficients of HK Ewald expansion (0:mxhko)
hon	output	HK-Ewald real space convergence function array (mxegr,0:mxhko)
fon	work	work array used in recurrence calculations (i.e. the B_n functions, see below. (mxegr,0:7)
dhn	output	derivative of HK-Ewald convergence function array (mxegr,0:mxhko)

8.1.72.5 Comments

The convergence functions are generated using a recurrence relation for the error function. Noting that $s = (x^2 + y^2)^{1/2}$ we start with the definitions:

$$B_0(s; \alpha) = \operatorname{erf}(\alpha s)/s \quad (8.1)$$

and

$$W(s; \alpha) = \frac{2\alpha}{\sqrt{\pi}} \exp(-\alpha^2 s^2). \quad (8.2)$$

It can then be shown by induction that

$$\frac{\partial}{\partial x} B_n(s; \alpha) = -x B_{n+1}(s; \alpha), \quad (8.3)$$

with

$$B_n(s; \alpha) = \frac{1}{s^2} \{ (2n-1) B_{n-1}(s; \alpha) - (2\alpha^2)^{n-1} W(s; \alpha) \}. \quad (8.4)$$

Rearranging 8.4 gives the following recursion relation:

$$B_{n-1}(s; \alpha) = \{ B_n(s; \alpha) s^2 + (2\alpha^2)^{n-1} W(s; \alpha) \} / (2n-1). \quad (8.5)$$

Thus commencing from $n = 300$ and setting $B_{300}(s; \alpha) = 0$ for all s , the (downwards) recursion relation can be used to generate all $B_n(s; \alpha)$ functions. In fact only the functions from B_0 to B_7 are required by DL_POLY.

It is easy to show that the convergence functions defined by Hautman and Klein [32] can be expressed as short polynomials in s^2 with coefficients that are products of the functions $B_n(s; \alpha)$ and an integer. i.e.

$$h_n(s; \alpha) = \sum_{i=n}^{2n} c_i B_i(s; \alpha) s^2 \quad (8.6)$$

where c_i is some integer. The derivatives of the convergence functions can be expressed in a similar manner.

Note that in DL_POLY we have defined $\alpha = 0.5/\alpha_{HK}$, where α_{HK} is the α parameter defined by Hautman and Klein in [32].

8.1.73 IMAGES

8.1.73.1 Header records

```

      subroutine images
      x (imcon,idnode,mxnode,natms,cell,xxx,yyy,zzz)

c
c*****
c
c      dl_poly subroutine for calculating the minimum image
c      of atom pairs within a specified MD cell
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith march 1992.
c      amended   - t.forester sept 1994
c
c      for
c      imcon=0 no boundary conditions apply
c      imcon=1 standard cubic boundaries apply
c      imcon=2 orthorhombic boundaries apply
c      imcon=3 parallelepiped boundaries apply
c      imcon=4 truncated octahedron boundaries apply
c      imcon=5 rhombic dodecahedron boundaries apply
c      imcon=6 x-y parallelogram boundary conditions : no periodicity in z
c
c      note: in all cases the centre of the cell is at (0,0,0)
c      warning - replicated data version: does not re-merge
c      coordinate arrays
c
c*****
c

```

8.1.73.2 Function

Applies periodic boundary conditions as per Header records above.

8.1.73.3 Dependencies

- none

8.1.73.4 Arguments

integers:

<code>imcon</code>	input	key for periodic boundary conditions
<code>idnode</code>	input	identity of current processor
<code>mxnode</code>	input	total number of processors
<code>natms</code>	input	number of vectors
real arrays:		
<code>cell</code>	input	MD cell vectors (9)
<code>xxx</code>	in/out	x coordinate (*)
<code>yyy</code>	in/out	y coordinate (*)
<code>zzz</code>	in/out	z coordinate (*)

8.1.73.5 Comments

The subroutine uses the intrinsic `NINT` function to find the nearest integer of the reduced vectors. On some machines it may be faster to replace lines of the form

```
x = x - nint(x)
```

with

```
x = x - int(x+sign(0.5d0,x))
```

8.1.74 INITCOMMS**8.1.74.1 Header records**

```
subroutine initcomms()
```

```
c*****
c
c      communication harness initialisation
c
c      MPI version - t.forester may 1995
c      CPP version - w.smith may 1995
c
c*****
```

8.1.74.2 Function

Initialises the parallel environment.

8.1.74.3 Dependencies

- MPI initialization routines.

8.1.74.4 Arguments

None.

8.1.74.5 Comments

The MPI option is selected by C-preprocessing at compile time.

8.1.75 INTLIST

8.1.75.1 Header records

```

      subroutine intlist
      x  (lshmov,idnode,mxnode,natms,nscons,ntangl,ntbond,ntcons,
      x  ntdihd,ntinv,ntpmls,ntteth,ntshl,ntpmf,nspmf,lashap,lishap,
      x  listang,listbnd,listcon,listdih,listinv,listshl,listin,
      x  listme,listot,lstang,lstbnd,lstcon,lstdih,lstinv,lstfrz,
      x  lsttet,lsttet,lstshl,numang,numbonds,numcon,numdih,numinv,
      x  nummols,numsit,numteth,numshl,numpmf,npmf,indpmf,lstpmf,
      x  listpm,lstpmt)

c
c*****
c
c      dl_poly subroutine for constructing the interaction lists
c      for the entire simulated system
c
c      parallel replicated dat version : block data

c      copyright - daresbury laboratory 1992
c      author    - w. smith          july 1992
c      amended   - t.forester        oct 1993
c      amended   - t.forester        dec 1994 : block data
c
c*****
c

```

8.1.75.2 Function

Constructs bonded interaction lists for the entire system.

8.1.75.3 Dependencies

- ERROR
- GSTATE
- PASSCON
- PASSPMF
- SHELLSORT

8.1.75.4 Arguments**logicals:**

lshmov	output	flag for shared bond constraints

integers:

idnode	input	identity of the current processor
mxnode	input	total number of processors
natms	input	number of atoms in MD cell
nscons	output	number of constraint bonds on the current processor
ntangl	output	total number of valence angles in system
ntbond	output	total number of chemical bonds in system
ntcons	output	total number of constraint bonds in system
ntdihd	output	total number of dihedral interactions in system
ntinv	output	total number of inversion interactions in system
ntpmls	input	number of different molecular types
ntteth	input	total number of tethered atoms in system
ntshl	input	total number of core-shell units in system
ntpmf	input	total number of PMF constraints in system
nspmf	input	number of PMF constraints on the current processor

integer arrays:

lashap	output	see PASSCON (mxproc)
lishap	output	see PASSCON (mxlshp)
listang	output	indices for valence angles on current processor (mxangl,4)
listbnd	output	indices for chemical bonds on current processor (mxbond,3)
listcon	output	indices for constraint bonds on current processor (mxcons,3)
listdih	output	indices for dihedral interactions on current processor (mxdihd,5)
listinv	output	indices for inversion interactions on current processor (mxinv,5)
listshl	output	indices for core-shell interactions on current processor (mxshl,3)
listin	output	see PASSCON (mxatms)
listme	output	see PASSCON (mxatms)
listot	output	see PASSCON (mxatms)
lstang	input	indices of sites in valence angles in the system (mxtang,3)
lstbnd	input	indices of sites in chemical bonds in the system (mxtbnd,2)
lstcon	input	indices of sites in constraint bonds in the system (mxtcon,2)
lstdih	input	indices of sites in dihedral interactions in the system (mxt dih,4)
lstinv	input	indices of sites in inversion interactions in the system (mxtinv,4)
lstfrz	input	key for frozen atoms (mxatms)
lsttet	input	indices of tethered sites (mxteth)
listtet	output	indices of tethered atoms on current processor (msteth,2)
lstshl	input	indices of core-shell sites (mxtshl,2)
numang	input	number of valence angles in a molecular type (mxtmls)
numbonds	input	number of chemical bondss in a molecular type (mxtmls)
numcon	input	number of constraint bonds in a molecular type (mxtmls)
numdih	input	number of dihedral interactions in each molecular type (mxtmls)

<code>numinv</code>	input	number of inversion interactions in each molecular type (<code>mxtmls</code>)
<code>nummols</code>	input	number of molecules of each type (<code>mxtmls</code>)
<code>numsit</code>	input	number of sites in each molecular type (<code>mxtmls</code>)
<code>numteth</code>	input	number of tethered sites in each molecular type (<code>mxtmls</code>)
<code>numshl</code>	input	number of core-shell units in each molecular type (<code>mxtmls</code>)
<code>numpmf</code>	input	number of PMF constraints in each molecular type (<code>mxtmls</code>)
<code>npmf</code>	input	number of sites in PMF constraint units (2)
<code>indpmf</code>	input	indices of sites in PMF constraints (<code>mxspmf</code>)
<code>lstpmf</code>	output	indices of atoms in PMF constraints on this processor (<code>mxspmf, mspmf</code>)
<code>listpm</code>	output	see PASSPMF (<code>mxpmf</code>)
<code>lstpmt</code>	output	see PASSPMF (<code>mxpmf</code>)

8.1.75.5 Comments

Need only be called once at the beginning of a job. Each processor constructs its own, independent, interaction lists.

8.1.76 INTSTR**8.1.76.1 Header records**

```
function intstr(word,len,lst)
```

```
c
c*****
c
c      dl_poly function for extracting integers from a
c      character string
c
c      copyright - daresbury laboratory 1994
c      author    - w. smith may 1994.
c
c      parameters:
c      word      - input character string
c      len       - working length of character string
c      lst       - location of space character at end of
c      integer string
c
c*****
c
```

8.1.76.2 Function

Extracts the first integer from a character string.

8.1.76.3 Dependencies

- none

8.1.76.4 Arguments

character arrays:

word	input	character string

integers:

len	input	length of string
lst	output	location of first character after number in the string

8.1.76.5 Comments

8.1.77 INVERT**8.1.77.1 Header records**

```

      subroutine invert(a,b,d)
c
c*****
c
c      dl_poly subroutine to invert a 3 * 3 matrix using cofactors
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith      april 1992
c
c*****
c

```

8.1.77.2 Function

Inverts a 3×3 matrix.

8.1.77.3 Dependencies

- none

8.1.77.4 Arguments

reals:

d	output	determinant of original matrix

real arrays:

a	input	matrix (9)
b	output	inverted matrix (9)

8.1.77.5 Comments

If the matrix is singular, (ie. determinant = 0) the inverse matrix is returned with all entries set to zero.

8.1.78 JACOBI**8.1.78.1 Header records**

```

      subroutine jacobi(a,v,n)
c
c*****
c
c      diagonalisation of real symmetric matrices by jacobi method
c
c      input parameters:
c
c      a(n,n) is the matrix to be diagonalised
c      v(n,n) is the eigenvector matrix
c      n   is the dimension of the matrices
c
c      jacobi processes lower triangle only (upper triangle unchanged)
c
c      variable rho sets absolute tolerance on convergence
c      variable tes is a moving tolerance that diminishes
c      on each pass until at true convergence tes<rho
c
c      copyright daresbury laboratory 1993
c      author w.smith 1993
c
c*****
c

```

8.1.78.2 Function

Diagonalises a real symmetric matrix.

8.1.78.3 Dependencies

- none

8.1.78.4 Arguments

integers:

n input dimension of matrix

real arrays:

a input real symmetric matrix

v output eigenvectors of matrix

8.1.78.5 Comments

8.1.79 KINSTR**8.1.79.1 Header records**

```

      subroutine kinstr
      x  (idnode,mxnode,natms,tstep,vxx,vyy,vzz,
      x  fxx,fyy,fzz,weight,stress)

c*****
c
c      dlpoly routine to calculate the kinetic energy contribution to
c      the stress tensor
c
c      assumes velocities are half-timestep behind forces
c
c      replicated data version / block data
c
c      copyright daresbury laboratory 1994
c      author t.forester may 1994
c      amended t.forester dec 1994 : block data
c
c*****

```

8.1.79.2 Function

Calculates kinetic contribution to the stress tensor.

8.1.79.3 Dependencies

- none

8.1.79.4 Arguments**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	total number of atoms in the MD cell

reals:

tstep	input	simulation time-step, Δt

real arrays:

vxx	input	x component of velocity (mxatms)
vyy	input	y component of velocity (mxatms)
vzz	input	z component of velocity (mxatms)
fxx	input	x component of force (mxatms)
fyy	input	y component of force (mxatms)
fzz	input	z component of force (mxatms)

weight	input	atomic masses (mxatms)
stress	in/out	stress tensor (9)

8.1.79.5 Comments

The velocities are advanced half a timestep by the standard Verlet leapfrog algorithm to calculate the contribution to the stress tensor at time t . The stress tensor is not globally summed on exit from this routine.

8.1.80 LOWCASE**8.1.80.1 Header records**

```
subroutine lowercase(string,length)
```

```
c*****
c
c      DL_POLY routine to lowercase a string of up to 255 characters.
c      Transportable to non-ASCII machines
c
c      copyright daresbury laboratory 1993
c      author    t. forester    july 1993
c
c*****
```

8.1.80.2 Function

Converts a string to lowercase characters.

8.1.80.3 Dependencies

- none

8.1.80.4 Arguments

character:

string	in/out	alphanumeric string

integer:

length	input	string length

8.1.80.5 Comments

8.1.81 LRCMETAL

8.1.81.1 Header records

```

      subroutine lrcmetal
      x  (idnode,imcon,mxnode,natms,ntpatm,engunit,rcut,volm,
      x  lstvdw,ltpvdw,ltype,numtyp,prmvdw,dens,elrcm,vlrcm)

c*****
c
c    DL_POLY subroutine to evaluate long-range corrections to
c    pressure and energy in a periodic metal system.
c
c    copyright daresbury laboratory 1995
c
c    author -      w. smith    june 1995
c
c*****

```

8.1.81.2 Function

Calculates long range correction for Sutton-Chen potentials.

8.1.81.3 Dependencies

- none

8.1.81.4 Arguments

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
natms	input	total number of atoms in the MD cell
ntpatm	input	number of unique types of atoms

reals:

engunit	input	unit of energy relative to internal units
rcut	input	potential cut-off radius
volm	input	volume of MD cell

integer arrays:

lstvdw	input	pointer from potential index to parameter table (mxvdw)
ltpvdw	input	key for non-bonded potentials (mxvdw)
ltype	input	index of atomic types (mxatms)
numtyp	output	number of each atom type (mxsvdw)

real arrays:

prmvdw	input	variables for short-ranged potentials (mxvdw,mxpvdw)

<code>dens</code>	output	particle number density (<code>mxsvdw</code>)
<code>elrcm</code>	output	long range correction to Sutton Chen energy (2)
<code>vlrcm</code>	output	long range correction to Sutton Chen virial (2)

8.1.81.5 Comments

Note the application of the long range correction is not as straightforward as for van der Waals potentials. See chapter 2 section 2.3.2.

8.1.82 LRCORRECT

8.1.82.1 Header records

```

      subroutine lrcorrect
      x  (idnode,imcon,keyfce,mxnode,natms,ntpatm,elrc,engunit,virlrc,
      x  rcut,volm,lstvdw,ltpvdw,ltype,numtyp,numfrz,lstfrz,prmvdw,dens)

c*****
c
c    DL_POLY subroutine to evaluate long-range corrections to
c    pressure and energy in a periodic system.
c
c    copyright daresbury laboratory 1993
c
c    author -      t. forester may 1993
c
c*****

```

8.1.82.2 Function

Calculates long range corrections to the energy and pressure in a periodic system due to van der Waals type potentials.

8.1.82.3 Dependencies

- none

8.1.82.4 Arguments

integers:

idnode	input	identity of current processor
imcon	input	periodic boundary key
keyfce	input	force level key
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ntpatm	output	number of unique atom types

reals:

elrc	output	long range correction to energy
engunit	input	energy unit for input/output
virlrc	output	long range correction to virial
rcut	input	potential cut-off radius
volm	input	volume of the MD cell

integer arrays:

lstvdw	input	pointer from potential index to parameter table (mxvdw)
ltpvdw	input	key for non-bonded potentials (mxvdw)

<code>ltype</code>	input	list of atom types (<code>mxatms</code>)
<code>numtyp</code>	output	number of atoms of given type in simulation (<code>mxsvdw</code>)
<code>numfrz</code>	output	number of frozen atoms of given type in simulation (<code>mxsvdw</code>)
<code>lstfrz</code>	input	frozen atom indicator for all atoms (<code>mxatms</code>)
real arrays:		
<code>prmvdw</code>	input	variables for non-bonded potentials (<code>mxvdw</code> , <code>mxpvdw</code>)
<code>dens</code>	output	number density of atom types (<code>mxsvdw</code>)

8.1.82.5 Comments

The defined potentials must be the same as in FORGEN and FORGEN_RSQ.

8.1.83 MACHINE**8.1.83.1 Header records**

```
subroutine machine(idnode,mxnode)
```

```

c*****
c
c    dl_poly subroutine for obtaining characteristics of
c    the computer on which the program is being run
c
c    copyright daresbury laboratory 1992
c    author - w.smith july 1992
c
c    MPI version - t.forester may 1995
c
c*****

```

8.1.83.2 Function

Returns the processor identity and the number of processors in use.

8.1.83.3 Dependencies

- MYNODE
- NUMNODES

8.1.83.4 Arguments

integers:

idnode	output	identity of current processor
mxnode	output	total number of processors in use

8.1.83.5 Comments

DL_POLY_2 assumes `idnode` takes a value between 0 and `mxnode-1`. The code is C-preprocessed to suit the target computer.

8.1.84 MERGE

8.1.84.1 Header records

```

      subroutine merge(idnode,mxnode,natms,nbuff,xxx,yyy,zzz,buffer)
c
c*****
c
c      dl_poly subroutine for merging coordinate arrays across
c      a number of processors
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith november 1992.
c      MPI version - t. forester may 1995
c      CPP version - w.smith may 1995
c
c*****
c

```

8.1.84.2 Function

Merges arrays across a number of processors. Used, for example, in the Replicated Data integration of the equations of motion where site 'i' is handled on the processor with $idnode = (i * mxnode) / natms$.

8.1.84.3 Dependencies

- ERROR
- standard communication subroutines

8.1.84.4 Arguments

integers:

idnode	input	identity of the current processor
mxnode	input	total number of processors
natms	input	number of elements in arrays to be merged
nbuff	input	size of buffer array

real arrays:

xxx	in/out	first array to be merged (natms)
yyy	in/out	second array to be merged (natms)
zzz	in/out	third array to be merged (natms)
buffer	work	buffer used in data transfer nbuff

8.1.84.5 Comments

In DL_POLY_2 the parallelisation strategy involves assigning contiguous blocks of data to each processor. This is different from the strategy used in DL_POLY_1.1 so subroutines (and particularly this one) should not be mixed between the two versions.

C-preprocessing is used to select the required communication routines. The message passing involves a synchronisation step, message sending and message receiving calls. Compile time options are for the Intel hypercube, MPI, the Cray T3D or a single processor machine.

A version of MERGE is available that offers higher efficiency (i.e. better scaling) on hypercube parallel machines. This is found in the *source* file MERGE.HCUBE.F. Its calling sequence is the same as the original subroutine.

8.1.85 MERGE1

8.1.85.1 Header records

```

      subroutine merge1
      x (idnode,mxnode,natms,lstme,xxx,yyy,zzz,buffer)
c
c*****
c
c      dl_poly subroutine for merging together coordinate arrays
c      across a number of processors during rigid body algorithm
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1993
c      author    - t.forester  november 1993
c      systolic pulse version. T3D t.forester sept 1994
c      MPI version - t.forester may 1995
c      CPP version - w.smith may 1995
c
c*****
c

```

8.1.85.2 Function

Merges indexed arrays across a number of processors. Used, for example, in the Replicated Data integration of the equations of motion involving rigid bodies.

8.1.85.3 Dependencies

- ERROR
- GSTATE
- standard communication subroutines

8.1.85.4 Arguments

integers:

idnode	input	identity of the current processor
mxnode	input	total number of processors
natms	input	number of elements in arrays to be merged

integer arrays:

lstme	input	indices of array “owned” by current processor

real arrays:

xxx	in/out	first array to be merged (natms)

yyy	in/out	second array to be merged (natms)
zzz	in/out	third array to be merged (natms)

8.1.85.5 Comments

This routine is used when the sites “owned” by a processor cannot be guaranteed to be contiguous.

C-preprocessing is used to select the required communication routines. The message passing involves a synchronisation step, message sending and message receiving calls. Compile time options are for the Intel hypercube, MPI, the Cray T3D or a single processor machine.

A version of MERGE1 is available that offers higher efficiency (i.e. better scaling) on hypercube parallel machines. This is found in the *source* file MERGE1.HCUBE.F. Its calling sequence is the same as the original subroutine.

8.1.86 MERGE4

8.1.86.1 Header records

```
subroutine merge4(idnode,mxnode,ngroup,nbuff,q0,q1,q2,q3,buffer)
```

```
C*****
C
C    dl_poly subroutine for merging coordinate arrays across
C    a number of processors
C
C    parallel replicated data version
C
C    copyright - daresbury laboratory 1994
C    author    - t.forester february 1994
C    T3D version - sept 1994 t.forester
C    MPI version - t.forester may 1995
C    CPP version - w.smith may 1995
C
C*****
```

8.1.86.2 Function

Merges 4 arrays across a number of processors. Used, for example, in the Replicated Data integration of the quaternion equations of motion for rigid bodies. Rigid unit 'i' is handled on the processor with $idnode = (i * mxnode) / ngroup$.

8.1.86.3 Dependencies

- ERROR
- standard communication subroutines

8.1.86.4 Arguments

integers:

idnode	input	identity of the current processor
mxnode	input	total number of processors
ngroup	input	number of elements in arrays to be merged
nbuff	input	size of buffer array

real arrays:

q0	in/out	first array to be merged (natms)
q1	in/out	second array to be merged (natms)
q2	in/out	third array to be merged (natms)
q3	in/out	fourth array to be merged (natms)
buffer	work	work space (nbuff)

8.1.86.5 Comments

C-preprocessing is used to select the required communication routines. The message passing involves a synchronisation step, message sending and message receiving calls. Compile time options are for the Intel hypercube, MPI, the Cray T3D or a single processor machine.

A version of MERGE4 is available that offers higher efficiency (i.e. better scaling) on hypercube parallel machines. This is found in the *source* file MERGE4.HCUBE.F. Its calling sequence is the same as the original subroutine.

8.1.87 METGEN**8.1.87.1 Header records**

```

      subroutine metgen
      x  (idnode,ntpvdw,dlrpot,rcut,ltpvdw,prmvdw,vvv,ggg)
c
c*****
c
c      dl_poly subroutine for generating potential energy and
c      force arrays for metal potentials
c
c      copyright - daresbury laboratory 1995
c      author    - w. smith june 1995
c
c*****
c

```

8.1.87.2 Function

Generates potential energy and force arrays for Sutton Chen potentials.

8.1.87.3 Dependencies

- ERROR

8.1.87.4 Arguments**integers:**

idnode	input	identity of current processor
ntpvdw	input	number of short-ranged potentials

reals:

dlrpot	output	distance increment for short-range potential tabulation
rcut	input	potential cut-off

integer arrays:

ltpvdw	input	key for non-bonded potentials (mxvdw)

real arrays:

prmvdw	input	variables for short-ranged potentials (mxvdw,mxpvdw)
vvv	output	interpolation table for non-bonded potentials (mxgrid,mxvdw)
ggg	output	interpolation table for non-bonded force (mxgrid,mxvdw)

8.1.87.5 Comments

Applies to Sutton-Chen potentials only. A version using r^2 interpolation is available as METGEN_RSQ.

8.1.88 MULTIPLE

8.1.88.1 Header records

```

      subroutine multiple
x  (loglnk,lgofr,lzeql,newlst,idnode,imcon,keyfce,nlatt,
x  kmax1,kmax2,kmax3,nhko,multt,mxnode,natms,nstep,nstbgr,
x  nsteql,numrdf,nospl,fplan,bplan,alpha,dlrpot,drewd,
x  engcpe,engsrp,epsq,rcut,rprim,rvdw,vircpe,virsrp,volm,
x  ilist,jlist,lentry,lexatm,list,lstvdw,ltpvdw,ltype,nexatm,
x  nauxfft,key1,key2,key3,buffer,cell,chge,ckc,cks,clm,elc,
x  els,emc,ems,enc,ens,erc,fer,flx,fly,flz,fpz,fpy,fpz,
x  fxx,fyy,fzz,ggg,rdf,rsqdf,slm,stress,vvv,xdf,xxx,ydf,yyy,
x  zdf,zzz,ewlbuf,csp,qqc,txx,tyy,tzz,bsp,bspy,bspz,bsdx,
x  bsdz,bsdz,qqq,bscx,bscy,bscz,fftable,ww1,ww2,ww3,ahk,
x  zzn,zzd,sss,hon,dhn,pp,znp,zgc,zgs,crn)

C*****
C
C  DL_POLY subroutine for multiple time step algorithm
C  reciprocal space calculated on long time steps.
C
C  copyright daresbury laboratory
C
C  author  t. forester,  may 1993
C
C  keyfce = odd  ----- short range potentials calculated : srfrce
C  = 0,1  ----- no electrostatics
C  = 2,3  ----- Ewald sum                      : ewald1,2,3,4
C  = 4,5  ----- distance dependent dielectric    : coul2
C  = 6,7  ----- coulombic                        : coul0
C  = 8,9  ----- truncated and shifted coulombic   : coul4
C  = 10,11 ----- reaction field                   : coul3
C  = 12,13 ----- Smooth Particle Mesh Ewald      : ewald[_spme,2,3,4]
C  = 14,15 ----- Hautman-Klein-Ewald             : hkewald1,2,3,4
C
C*****

```

8.1.88.2 Function

Controls calculation of non-bonded interactions for multiple time-step simulations.

8.1.88.3 Dependencies

COUL0	COUL1	COUL2	COUL3

COUL4	EWALD1	EWALD2	EWALD3
EWALD4	EWALD_SPME	GDSUM	HKEWALD1
HKEWALD2	HKEWALD3	HKEWALD4	IMAGES
PRIMLST	SRFRCE	RDF0	

8.1.88.4 Arguments

logicals:

loglnk	input	flag for link cell algorithm
lgofr	input	flag for construction of radial distribution functions
lzeql	input	flag for equilibration period
newlst	input	flag for updated verlet list

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
keyfce	input	key for non-bonded force field
nlatt	input	lattice search parameter for HK-Ewald
kmax1	input	number of reciprocal space vectors in <i>a</i> direction
kmax2	input	number of reciprocal space vectors in <i>b</i> direction
kmax3	input	number of reciprocal space vectors in <i>c</i> direction
nhko	input	required order of HK-Ewald expansion
multt	input	size of multiple time-step interval
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
nstep	input	step counter
nstbgr	input	start timestep of RDF
nsteql	input	number of equilibration timesteps
numrdf	in/out	RDF configuration counter
nospl	input	order of B-splines in SPME Ewald
fplan	input	integer pointer (FFT)
bplan	input	integer pointer (FFT)

reals:

alpha	input	Ewald convergence parameter
dlrpot	input	distance increment for potential interpolation tables
drewd	input	distance increment for Ewald sum interpolation tables
engcpe	output	electrostatic energy
engsrp	output	non-bonded potential energy
epsq	input	relative dielectric constant
rcut	input	Electrostatic potential cut-off radius
rprim	input	radius of primary cut-off shell
rvdw	input	Van der Waals potential cut-off radius
vircep	output	electrostatic virial
virsrp	output	non-bonded potential virial

volm	input	volume of MD cell
integer arrays:		
ilist	work	neighbourhood list for central site (mxxdf)
jlist	work	auxiliary neighbourhood list (mxxdf)
lentry	input	number of entries in Verlet list (msatms)
lexatm	input	excluded atoms list (msatms,mxexcl)
list	input	the Verlet list (msatms,mxlist)
lstvdw	input	index number of non-bonded potential (mxvdw)
ltype	input	list of atom types (mxatms)
nexatm	input	number of excluded atoms (msatms)
nauxfft	input	FFT flags for FFTW implementation (4)
key1	work	inverse bit address array for first direction (kmaxd)
key2	work	inverse bit address array for second direction (kmaxe)
key3	work	inverse bit address array for third direction (kmaxf)
real arrays:		
buffer	work	work array (mxbuff)
cell	input	MD cell vectors (9)
chge	input	atomic charges (mxatms)
ckc	work	see EWALD1 (msatms)
cks	work	see EWALD1 (msatms)
clm	work	see EWALD1 (msatms)
elc	work	see EWALD1 (msatms,0:1)
els	work	see EWALD1 (msatms,0:1)
emc	work	see EWALD1 (msatms,0:kmaxb)
ems	work	see EWALD1 (msatms,0:kmaxb)
enc	work	see EWALD1 (msatms,0:kmaxc)
ens	work	see EWALD1 (msatms,0:kmaxc)
erc	input	interpolation table for Ewald sum real space energy
fer	input	interpolation table for Ewald sum real space force
flx	work	x component of time derivative of secondary force (mxatms)
fly	work	y component of time derivative of secondary force (mxatms)
flz	work	z component of time derivative of secondary force (mxatms)
fpx	work	x component of secondary force (mxatms)
fpv	work	y component of secondary force (mxatms)
fpz	work	z component of secondary force (mxatms)
fxv	output	x component of force (mxatms)
fvy	output	y component of force (mxatms)
fzv	output	z component of force (mxatms)
ggg	input	interpolation table for non-bonded force (mxgrid,mxvdw)
rdf	input	histograms for radial distribution functions (mxrdf,mxvdw)
rsqdf	work	square of intersite distance (mxxdf)
slm	work	see EWALD1 (msatms)
stress	in/out	stress tensor (9)
vvv	input	interpolation table for non-bonded energy (mxgrid,mxvdw)

xdf	work	x component of intersite vector (mxxdf)
xxx	input	x coordinates (mxatms)
ydf	work	y component of intersite vector (mxxdf)
yyy	input	y coordinates (mxatms)
zdf	work	z component of intersite vector (mxxdf)
zzz	input	z coordinates (mxatms)
ewlbuf	work	storage array for ewald sum evaluation (mxebuf)
csp	work	work space for B-spline calculation (mxspl)
qqc	work	SPME charge array (kmaxd,kmaxe,kmaxf)
txx	input	x working array (mxatms).
tyy	input	y working array (mxatms)
tzz	input	z working array (mxatms)
bspx	work	B-splines x (mxspme,mxspl)
bspy	work	B-splines y (mxspme,mxspl)
bspz	work	B-splines z (mxspme,mxspl)
bsdx	work	B-spline x derivatives (mxspme,mxspl)
bsdy	work	B-spline y derivatives (mxspme,mxspl)
bsdz	work	B-spline z derivatives (mxspme,mxspl)
qqq	work	FT of qqc array (complex) (kmaxd,kmaxe,kmaxf)
bscx	work	complex B-splines x coefficients (kmaxd)
bscy	work	complex B-splines y coefficients (kmaxe)
bscz	work	complex B-splines z coefficients (kmaxf)
fftttable	work	buffer required by Cray FFT routine (mxftab)
ww1	work	Fourier complex x exponentials (kmaxd)
ww2	work	Fourier complex y exponentials (kmaxe)
ww3	work	Fourier complex z exponentials (kmaxf)
ahk	input	coefficients of HK Ewald expansion (0:mxhko)
zzn	work	atom z-coordinate to power 2n (mxxdf)
zzd	work	atom z-coordinate to power 2n-1 (mxxdf)
sss	work	in-plane atomic separation (mxxdf)
hon	input	HK-Ewald real-space convergence function array (mxegrnd,0:mxhko)
dhn	input	derivative of HK-Ewald convergence function array (mxegrnd,0:mxhko)
pp	work	polynomial product array (2*mxhko)
znp	work	array of powers of atom z-coordinates (mxhke,0:2*mxhko)
zgc	work	global sums of products of powers of z-coordinate and cosine of $\exp(i.\mathbf{k.s})$ (0:2*mxhko)
zgs	work	global sums products of powers of z-coordinate and sine of $\exp(i.\mathbf{k.s})$ (0:2*mxhko)
crn	work	matrix of binomial coefficients (0:mxhko,0:mxhko)

8.1.88.5 Comments

MULTIPLE uses the linear interpolation method of Streett and Tildesely [39]. Secondary forces and energies are evaluated on the first and second steps of the multiple timestep cycle. The time derivative of the secondary contributions are evaluated numerically and used to estimate the value of the secondary contributions on subsequent timesteps. In subsequent timesteps only pair interaction within the primary cutoff sphere are evaluated.

The stability of the algorithm depends upon the choice of `multtt` and `rprim`. Checks on energy conservation etc, in a simulation should be made to ensure appropriate parameters are used.

If the value of `multtt` is less than 3 a single timestep algorithm will result.

8.1.89 MULTIPLE_NSQ**8.1.89.1 Header records**

```

      subroutine multiple_nsq
x  (lnsq,lgofr,lzeql,newlst,idnode,imcon,keyfce,
x  multt,mxnode,natms,nstep,nstbgr,nsteql,numrdf,
x  delr,dlrpot,engcpe,engsrp,engcp3,epsq,rcut,
x  rprim,rvdw,vircpe,virsrp,vircp3,ilist,lentry,
x  list,lstvdw,ltpvdw,ltype,buffer,cell,chge,flx,fly,
x  flz,fxz,fyz,fzz,fpz,ggg,rdf,rsqdf,vvv,xdx,
x  xxx,ydx,yyy,zdx,zzz,stress)

C*****
C
C      DL_POLY subroutine for multiple time step algorithm
C      to be used with all-pairs option
C
C      flx,fly,flz : forces from electrostatics from r > rcut
C      fpz,fpy,fpz : forces from electrostatics from rprim < r <= rcut
C      fxx,fyy,fzz : total force
C
C      copyright daresbury laboratory 1993
C
C      author t. forester, may 1993
C
C      keyfce = odd  ----- short range potentials calculated : srfrce
C      = 0,1  ----- no electrostatics
C      Ewald sum --- not used
C      = 4,5  ----- Distant dependent dielectric           : coul2
C      = 6,7  ----- coulombic                               : coul0
C      truncated and shifted coulombic -- not used
C      reaction field - not used
C
C*****

```

8.1.89.2 Function

Controls calculation of non-bonded interactions for multiple time-step simulations when all pair interactions are to be calculated explicitly.

8.1.89.3 Dependencies

COUL0	COUL2	GDSUM	IMAGES
PRIMLST	SRFRCE	RDF0	ERROR

8.1.89.4 Arguments**logicals:**

lnsq	input	flag for “all-pairs” algorithm
lgofr	input	flag for construction of radial distribution functions
lzeql	input	flag for equilibration period
newlst	input	flag for updated verlet list

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
keyfce	input	key for non-bonded force field
multt	input	size of multiple time-step interval
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
nstep	input	step counter
nstbgr	input	start timestep of RDF
nsteql	input	number of equilibration timesteps
numrdf	in/out	RDF configuration counter

reals:

delr	input	width of verlet border
dlrpot	input	distance increment for potential interpolation tables
engcpe	output	electrostatic energy
engsrp	output	non-bonded potential energy
engcp3	input	electrostatic energy from long-range interactions
epsq	input	relative dielectric constant
rcut	input	Electrostatic potential cut-off radius
rprim	input	radius of primary cut-off shell
rvdw	input	Van der Waals potential cut-off radius
vircep	output	electrostatic virial
virsrp	output	non-bonded potential virial
vircp3	input	electrostatic virial from long-range interactions

integer arrays:

ilist	work	neighbourhood list for central site (mxxdf)
lentry	input	number of entries in Verlet list (msatms)
list	input	the Verlet list (msatms,mxlist)
lstvdw	input	index number of non-bonded potential (mxvdw)
ltpvdw	input	key for non-bonded potentials (mxvdw)
ltype	input	list of atom types (mxatms)

real arrays:

buffer	work	work array (mxbuff)
cell	input	MD cell vectors (9)
chge	input	atomic charges (mxatms)

flx	work	x component of forces from $r > r_{\text{cut}}$ (mxatms)
fly	work	y component of forces from $r > r_{\text{cut}}$ (mxatms)
flz	work	z component of forces from $r > r_{\text{cut}}$ (mxatms)
fx	output	x component of force (mxatms)
fy	output	y component of force (mxatms)
fz	output	z component of force (mxatms)
fp	work	x component of secondary force (mxatms)
fp	work	y component of secondary force (mxatms)
fp	work	z component of secondary force (mxatms)
gg	input	interpolation table for non-bonded force (mxgrid,mxvdw)
rd	input	histograms for radial distribution functions (mxrdf,mxvdw)
rsqdf	work	square of intersite distance (mxxdf)
vv	input	interpolation table for non-bonded energy (mxgrid,mxvdw)
xd	work	x component of intersite vector (mxxdf)
xx	input	x coordinates (mxatms)
yd	work	y component of intersite vector (mxxdf)
yy	input	y coordinates (mxatms)
zd	work	z component of intersite vector (mxxdf)
zz	input	z coordinates (mxatms)
stress	in/out	stress tensor (9)

8.1.89.5 Comments

Secondary forces and energies (from $r_{\text{prim}} < r < r_{\text{cut}}$) are evaluated on the first step of the multiple timestep cycle and assumed to be constant until they are updated. In subsequent timesteps only pair interaction within the primary cutoff sphere are evaluated.

The stability of the algorithm depends upon the choice of **multt**, **rprim** and **rcut**. Checks on energy conservation etc, in a simulation should be made to ensure appropriate parameters are used.

8.1.90 MULTIPLNEU

8.1.90.1 Header records

```

      subroutine multipleneu
      x  (lgofr,lzeql,newlst,idnode,imcon,keyfce,multt,
      x  mxnode,natms,nneut,nstbgr,nstep,nsteql,numrdf,
      x  delr,dlrpot,engcpe,engsrp,epsq,rprim,rcut,rvdw,
      x  vircpe,virsrp,ilist,jlist,lentry,lexatm,list,
      x  lstout,lstvdw,ltype,nexatm,neulst,lstneu,lstfrz,
      x  buffer,cell,chge,fx,fy,fz,flx,fly,flz,fpx,
      x  fpy,fpz,ggg,rdf,rsqdf,vvv,xdf,xxx,ydf,yyy,zdf,
      x  zzz,stress,txx,tyy,tzz,txt,yyt,zzt,uxx,
      x  uyy,uzz)

c
c*****
c
c      dl_poly subroutine for calculating interatomic forces
c      using the verlet neighbour list
c      neutral group implementation - no Ewald sum option
c      multiple timestep option
c
c      parallel replicated data version
c
c      fpx,fpy,fpz : forces from electrostatics from rprim < r <= rcut
c      fxx,fyy,fzz : total force
c
c      copyright daresbury laboratory april 1994
c      author - t. forester april 1993
c      key:
c
c      keyfce = odd  ----- short range potentials calculated : srfrce
c      = 0,1  ----- no electrostatics
c      = 2,3  ----- invalid
c      = 4,5  ----- Distant dependent dielectric           : coul2
c      = 6,7  ----- coulombic                               : coul0
c      = 8,9  ----- invalid
c      = 10,11 ----- reaction field                         : coul3
c
c*****
c

```

8.1.90.2 Function

Controls calculation of non-bonded interactions for multiple time-step simulations when a group based cutoff scheme is in use.

8.1.90.3 Dependencies

COUL0NEU	COUL2NEU	COUL3NEU	GDSUM
GSTATE	NEUTLST	PRNEULST	SRFRCENEU
RDF0NEU			

8.1.90.4 Arguments

logicals:

lgofr	input	flag for construction of radial distribution functions
lzeql	input	flag for equilibration period
newlst	input	flag for updated verlet list

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
keyfce	input	key for non-bonded force field
multt	input	size of multiple time-step interval
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
nneut	input	number of charge groups in system
nstbgr	input	start timestep of RDF
nstep	input	step counter
nsteql	input	number of equilibration timesteps
numrdf	in/out	RDF configuration counter

reals:

delr	input	width of verlet border
dlrpot	input	distance increment for potential interpolation tables
engcpe	output	electrostatic energy
engsrp	output	non-bonded potential energy
epsq	input	relative dielectric constant
rprim	input	radius of primary cut-off shell
rcut	input	Electrostatic potential cut-off radius
rvdw	input	Van der Waals potential cut-off radius
vircep	output	electrostatic virial
virsrp	output	non-bonded potential virial

integer arrays:

ilist	work	indices of first site in pair interactions (mxxdf)
jlist	work	indices of second site in pair interactions (mxxdf)
lentry	input	number of entries in Verlet list (msatms)

lexatm	input	excluded atoms list (msatms,mxexcl)
list	input	the Verlet list (msatms,mxlist)
lstout	work	work space (mxatms)
lstvdw	input	index number of non-bonded potential (mxvdw)
ltype	input	list of atom types (mxatms)
nexatm	input	number of excluded atoms (msatms)
neulst	input	indice of first site in charge group (mxneut)
lstneu	input	index of group to which site belongs (mxatms)
lstfrz	input	key for frozen atoms (mxatm)
real arrays:		
buffer	work	work array (mxbuff)
cell	input	MD cell vectors (9)
chge	input	atomic charges (mxatms)
fxx	output	x component of force (mxatms)
fyy	output	y component of force (mxatms)
fzz	output	z component of force (mxatms)
flx	work	x component of time derivative of secondary forces (mxatms)
fly	work	y component of time derivative of secondary forces (mxatms)
flz	work	z component of time derivative of secondary forces (mxatms)
fpx	work	x component of secondary force (mxatms)
fpv	work	y component of secondary force (mxatms)
fpz	work	z component of secondary force (mxatms)
ggg	input	interpolation table for non-bonded force (mxgrid,mxvdw)
rdf	input	histograms for radial distribution functions (mxrdf,mxvdw)
rsqdf	work	square of intersite distance (mxxdf)
vvv	input	interpolation table for non-bonded energy (mxgrid,mxvdw)
xdf	work	x component of intersite vector (mxxdf)
xxx	input	x coordinates (mxatms)
ydf	work	y component of intersite vector (mxxdf)
yyy	input	y coordinates (mxatms)
zdf	work	z component of intersite vector (mxxdf)
zzz	input	z coordinates (mxatms)
stress	in/out	stress tensor (9)
txx	work	work space (mxatms)
tyy	work	work space (mxatms)
tzz	work	work space (mxatms)
txx	work	work space (mxatms)
tyy	work	work space (mxatms)
tzz	work	work space (mxatms)
uxx	work	work space (mxatms)
uyy	work	work space (mxatms)
uzz	work	work space (mxatms)

8.1.90.5 Comments

The algorithm is essentially the same as used in MULTIPLE except that a group based cutoff scheme is used in place of an atomistic based cutoff scheme. The shifted coulombic potential options is not compatible with a group based scheme. The Ewald sum is more efficiently evaluated with an atomistic cutoff scheme.

8.1.91 MYNODE**8.1.91.1 Header records**

```
integer function mynode()
```

```
c*****
c
c      INTEL like routine to determine identity of processing node
c
c      MPI version - t.forester may 1995
c
c*****
```

8.1.91.2 Function

Determines identity of node in MPI environment.

8.1.91.3 Dependencies

- MPI_COMM_RANK

8.1.91.4 Arguments

None.

8.1.91.5 Comments

C-preprocessing is used at compile time to select the appropriate parallel harness.

8.1.92 NEUTLST**8.1.92.1 Header records**

```

      subroutine neutlst
      x  (newlst,lchk,isn,imcon,idnode,mxnode,ineu,ia,ll,rcut,
      x  nexatm,lexatm,list,lentry,ilist,jlist,ifox,jfox,
      x  lstfrz,neulst,cell,xxx,yyy,zzz,xdf,ydf,zdf,rsqdf,
      x  txx,tyy,tzz,xxt,yyt,zzt,uxx,uyy,uzz)

c*****
c
c      dlpoly routine to create pair lists for neutral group
c      implementations.
c      loops over group ineu
c
c      replicated data version
c
c      copyright daresbury laboratory 1994
c      author t.forester march 1994
c
c      isn = -1 => secondary neighbours
c      isn =  1 => primary neighbours - must contain excld interactions
c
c*****

```

8.1.92.2 Function

Creates list of pair interactions from charge group interactions.

8.1.92.3 Dependencies

- IMAGES

8.1.92.4 Arguments**logicals:**

newlst	input	flag for construction of Verlet list
lchk	output	flag for safe completion of list

integers:

isn	input	key for primary/secondary shell
imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	number of nodes in use
ineu	input	identity of central group

ia	input	local identity of central group
ll	output	length of pair list
reals:		
rcut	input	potential cut-off
integer arrays:		
nexatm	input	number of excluded atoms for each site (msatms)
lexatm	input	list of interactions to be excluded from the verlet list (msatms,mxexcl)
list	input	the group based Verlet neighbourhood list (msatms,mxlist)
lentry	input	number of neighbours in Verlet neighbourhood list (msatms)
ilist	output	index of first site in pair interaction (mxxdf)
jlist	output	index of second site in pair interaction (mxxdf)
ifox	work	work array (mxatms)
jfox	work	work array (mxatms)
lstfrz	input	key for frozen atoms (mxatms)
neulst	input	indice of first site in charge group (mxneut)
real arrays:		
cell	input	MD cell vectors (9)
xxx	input	x coordinates (mxatms)
yyy	input	y coordinates (mxatms)
zzz	input	z coordinates (mxatms)
xdf	output	x coordinate of interatomic vector (mxxdf)
ydf	output	y coordinate of interatomic vector (mxxdf)
zdf	output	z coordinate of interatomic vector (mxxdf)
rsqdf	output	square of interatomic distances (mxxdf)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)

8.1.92.5 Comments

8.1.93 NEUTLST_RESPA**8.1.93.1 Header records**

```

      subroutine neutlst_respa
      x  (newlst,lchk,isn,imcon,idnode,mxnode,ineu,ia,ll,rcut,
      x  nexatm,lexatm,list,lentry,ilist,jlist,
      x  lstfrz,neulst,cell,xxx,yyy,zzz,xdf,ydf,zdf,rsqdf,
      x  txx,tyy,tzz,xxt,yyt,zzt,uxx,uyy,uzz)

c*****
c
c      dlpoly routine to create pair lists for neutral group
c      implementations.
c      loops over group ineu
c      test on group centre - group centre + correction for group size.
c
c      replicated data version
c
c      copyright daresbury laboratory 1994
c      author t.forester march 1994
c      modified from neutlst - t.forester june 1996
c
c*****

```

8.1.93.2 Function

Creates list of pair interactions from charge group interactions.

8.1.93.3 Dependencies

- IMAGES

8.1.93.4 Arguments**logicals:**

newlst	input	flag for construction of Verlet list
lchk	output	flag for safe completion of list

integers:

isn	input	key for excluded interactions
imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	number of nodes in use
ineu	input	identity of central group
ia	input	local identity of central group
ll	output	length of pair list

reals:		
rcut	input	potential cut-off
integer arrays:		
nexatm	input	number of excluded atoms for each site (msatms)
lexatm	input	list of interactions to be excluded from the verlet list (msatms , mxexcl)
list	input	the group based Verlet neighbourhood list (msatms ,*)
lentry	input	number of neighbours in Verlet neighbourhood list (msatms)
ilist	output	index of first site in pair interaction (mxxdf)
jlist	output	index of second site in pair interaction (mxxdf)
lstfrz	input	key for frozen atoms (mxatms)
neulst	input	indice of first site in charge group (mxneut)
real arrays:		
cell	input	MD cell vectors (9)
xxx	input	x coordinates (mxatms)
yyy	input	y coordinates (mxatms)
zzz	input	z coordinates (mxatms)
xdf	output	x coordinate of interatomic vector (mxxdf)
ydf	output	y coordinate of interatomic vector (mxxdf)
zdf	output	z coordinate of interatomic vector (mxxdf)
rsqdf	output	square of interatomic distances (mxxdf)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)

8.1.93.5 Comments

8.1.94 NODDIM**8.1.94.1 Header records**

```
integer function nodedim()
```

```
c*****
c
c      calculate dimension of hypercube
c
c      MPI version - t.forester may 1995
c
c*****
```

8.1.94.2 Function

Calculates hypercube dimension in serial and MPI environments.

8.1.94.3 Dependencies

- ERROR
- MPI_COMM_SIZE

8.1.94.4 Arguments

None

8.1.94.5 Comments

This mimics the INTEL routine of the same name. Use the INTEL routine if you happen to be running on an INTEL. This routine will abort if the parallel machine is not a hypercube.

8.1.95 NPT_B0**8.1.95.1 Header records**

```

      subroutine npt_b0
      x  (idnode,imcon,mxnode,natms,ntpatm,elrc,engke,virlrc,press,
      x  taup,taut,sigma,tstep,virtot,volm,lstfrz,buffer,cell,dens,
      x  fxx,fyy,fzz,vxx,vyy,vzz,weight,xdf,xxx,ydf,yyy,zdf,zzz,
      x  vx1,vy1,vz1,stress,eta,dens0)

c*****
c
c      dl_poly subroutine for integrating newtonian equations of
c      motion in molecular dynamics - verlet leapfrog with Berendsen
c      thermostat amd isotropic pressure control
c      isothermal compressibility (beta) set to that of liquid water
c      = 0.007372 dlpoly units
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1993/1999
c      author    -    t. forester december 1993
c      adapted - atomic systems w.smith oct 1999
c
c*****

```

8.1.95.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion subject to a Berendsen thermostat and isotropic barostat. This version is for systems without constraint bonds or rigid bodies.

8.1.95.3 Dependencies

- GDSUM
- IMAGES
- MERGE

8.1.95.4 Arguments**integers:**

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use

natms	input	number of atoms in MD cell
ntpatm	input	number of unique types of atoms
reals:		
elrc	in/out	long range correction to van der Waals energy
engke	output	kinetic energy
virirc	in/out	long range correction to van der Waals virial
press	input	external pressure, P_{ext}
taup	input	barostat time constant, τ_P
taut	input	thermostat time constant, τ_T
sigma	input	required system kinetic energy
tstep	input	simulation time-step, Δt
virtot	input	virial from non-constraint terms
volm	in/out	volume of MD cell
integer arrays:		
lstfrz	input	key for frozen atoms (mxatms)
real arrays:		
buffer	work	see MERGE (mxbuff)
cell	in/out	MD cell vectors (9)
dens	in/out	particle number density (mxsvdw)
fx	in/out	x component of force (mxatms)
fy	in/out	y component of force (mxatms)
fz	in/out	z component of force (mxatms)
vx	in/out	x component of velocity (mxatms)
vy	in/out	y component of velocity (mxatms)
vz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xdf	work	x component of intersite vector (mxxdf)
xxx	in/out	x coordinates (mxatms)
ydf	work	y component of intersite vector (mxxdf)
yyy	in/out	y coordinates (mxatms)
zdf	work	z component of intersite vector (mxxdf)
zzz	in/out	z coordinates (mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)
eta	output	cell scaling tensor (9)
dens0	work	(mxsvdw)

8.1.95.5 Comments

Uses a leapfrog integration scheme (see NVE_1 for general comments).

The pressure is adjusted by isotropic changes to the MD cell vectors at each step. The

scale factor, S , is

$$S = \left(1 + \frac{\beta \Delta t}{\tau_P} (\mathcal{P} - P_{\text{ext}})\right)^{1/3} \quad (8.1)$$

where β is the isothermal compressibility (taken to be that of liquid water) and $\mathcal{P}$ is the instantaneous pressure in the system. Choice of τ_P depends on the actual β of the system. In water based systems a value of τ_P in the range (0.5, 5) ps is normally appropriate.

Temperature is controlled by the Berendsen thermostat. At each step the velocities are scaled by χ where:

$$\chi = \left(1 + \frac{\Delta t}{\tau_T} \left(\frac{T}{\mathcal{T}} - 1\right)\right)^{1/2} \quad (8.2)$$

where τ_T is the thermostat relaxation time, $\mathcal{T}$ the instantaneous temperature and T the specified system temperature. Usually a value of τ_T in the range (0.1, 1) ps works well.

The Berendsen thermostat and barostat do not conserve energy.

The barostat and thermostat require several iterations in order to accurately predict the current temperature and pressure. The variable `maxit` controls the number of thermostat/barostat iterations used. This is set to 4 in this version, but may be changed by editing the source code and recompiling.

8.1.96 NPT_B1**8.1.96.1 Header records**

```

      subroutine npt_b1
      x  (safe,lshmov,idnode,imcon,mxnode,natms,ntpatm,nscons,
      x  ntcons,elrc,engke,virlrc,press,taup,taut,sigma,tolnce,
      x  tstep,virtot,vircon,volm,lashap,lishap,listcon,listme,
      x  lstfrz,listot,buffer,cell,dens,dxt,dxx,dyt,dyy,dzt,dzz,
      x  fxx,fyy,fzz,prmcon,txx,tyy,tzz,uxx,uyy,uzz,vxx,vyy,vzz,
      x  weight,xdf,xxt,xxx,ydf,yyt,yyy,zdf,zzt,zzz,vx1,vy1,vz1,
      x  stress,eta,dens0)

c*****
c
c      dl_poly subroutine for integrating newtonian equations of
c      motion in molecular dynamics - verlet leapfrog with Berendsen
c      thermostat and isotropic pressure control
c      isothermal compressibility (beta) set to that of liquid water
c      = 0.007372 dlpoly units
c
c      parallel replicated data version
c
c      for systems using bond CONSTRAINTS. Frozen atoms feb 1994
c
c      copyright - daresbury laboratory 1993
c      author    -    t. forester december 1993
c
c*****
c

```

8.1.96.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion subject to a Berendsen thermostat and isotropic barostat.

8.1.96.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- RDSHAKE_1

8.1.96.4 Arguments**logicals:**

safe	output	flag for convergence of SHAKE
lshmov	input	flag for shared constraint bonds

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ntpatm	input	number of unique types of atoms
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system

reals:

elrc	in/out	long range correction to van der Waals energy
engke	output	kinetic energy
virirc	in/out	long range correction to van der Waals virial
press	input	external pressure, P_{ext}
taup	input	barostat time constant, τ_P
taut	input	thermostat time constant, τ_T
sigma	input	required system kinetic energy
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
virtot	input	virial from non-constraint terms
vircon	output	constraint virial
volm	in/out	volume of MD cell

integer arrays:

lashap	input	see RDSHAKE_1 (mxproc)
lishap	input	see RDSHAKE_1 (mxlshp)
listcon	input	indices for constraint bonds (mxcons ,3)
listme	input	see RDSHAKE_1 (mxatms)
lstfrz	input	key for frozen atoms (mxatms)
listot	input	see RDSHAKE_1 (mxatms)

real arrays:

buffer	work	see MERGE (mxbuff)
cell	in/out	MD cell vectors (9)
dens	in/out	particle number density (mxsvdw)
dxt	work	see RDSHAKE_1 (mxcons)
dxs	work	see RDSHAKE_1 (mxcons)
dys	work	see RDSHAKE_1 (mxcons)
dyy	work	see RDSHAKE_1 (mxcons)
dzt	work	see RDSHAKE_1 (mxcons)
dzz	work	see RDSHAKE_1 (mxcons)
fxs	in/out	x component of force (mxatms)

fyf	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
prmcon	input	constraint distances (mxtcon)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xdf	work	x component of intersite vector (mxxdf)
xxt	work	(mxatms)
xxx	in/out	x coordinates (mxatms)
ydf	work	y component of intersite vector (mxxdf)
yyt	work	(mxatms)
yyy	in/out	y coordinates (mxatms)
zdf	work	z component of intersite vector (mxxdf)
zzt	work	(mxatms)
zzz	in/out	z coordinates (mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)
eta	output	cell scaling tensor (9)
dens0	work	(mxsvdw)

8.1.96.5 Comments

Uses a leapfrog integration scheme (see NVE_1 for general comments).

The pressure is adjusted by isotropic changes to the MD cell vectors at each step. The scale factor, S , is

$$S = \left(1 + \frac{\beta \Delta t}{\tau_P} (\mathcal{P} - P_{\text{ext}})\right)^{1/3} \quad (8.3)$$

where β is the isothermal compressibility (taken to be that of liquid water) and $\mathcal{P}$ is the instantaneous pressure in the system. Choice of τ_P depends on the actual β of the system. In water based systems a value of τ_P in the range (0.5, 5) ps is normally appropriate.

Temperature is controlled by the Berendsen thermostat. At each step the velocities are scaled by χ where:

$$\chi = \left(1 + \frac{\Delta t}{\tau_T} \left(\frac{T}{\mathcal{T}} - 1\right)\right)^{1/2} \quad (8.4)$$

where τ_T is the thermostat relaxation time, $\mathcal{T}$ the instantaneous temperature and T the specified system temperature. Usually a value of τ_T in the range (0.1, 1) ps works well.

The Berendsen thermostat and barostat do not conserve energy. The thermostat and barostat are applied concurrently with a call to `RDSHAKE_1` to handle bond constraints using SHAKE. The variable `maxit` controls the number of thermostat/barostat iterations used. This is set to 5 for systems with bond constraints and set to 4 otherwise. This may be changed by editing the source code and recompiling.

8.1.97 NPT_B3**8.1.97.1 Header records**

```

      subroutine npt_b3
      x  (safe,lshmov,idnode,imcon,mxnode,natms,ntpatm,nscons,
      x  ntcons,elrc,engke,virlrc,press,taup,taut,sigma,tolnce,
      x  tstep,vircon,volm,lashap,lishap,listcon,listme,
      x  lstfrz,listot,buffer,cell,dens,dxt,dxx,dyt,dyy,dzt,dzz,
      x  fxx,fyy,fzz,prmcon,txx,tyy,tzz,uxx,uyy,uzz,vxx,vyy,vzz,
      x  weight,xdf,xxt,xxx,ydf,yyt,yyy,zdf,zzt,zzz,vx1,vy1,vz1,
      x  stress,eta,dens0)

C*****
C
C      dl_poly subroutine for integrating newtonian equations of
C      motion in molecular dynamics - verlet leapfrog with Berendsen
C      thermostat and anisotropic pressure control
C      isothermal compressibility (beta) set to that of liquid water
C      = 0.007372 dlpoly units
C
C      parallel replicated data version
C
C      for systems using bond CONSTRAINTS. Frozen atoms feb 1994
C
C      copyright - daresbury laboratory 1993
C      author    -    t. forester december 1993
C
C*****
C

```

8.1.97.2 Function

The verlet leapfrog algorithm for integrating Newton's equations of motion subject to an anisotropic Berendsen barostat and thermostat. That is, both cell shape and cell volume are allowed to vary.

8.1.97.3 Dependencies

- ERROR
- GDSUM
- IMAGES
- MERGE

- RDSHAKE_1

8.1.97.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
lshmov	input	flag for shared bond constraints

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ntpatm	input	number of unique types of atoms
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system

reals:

elrc	in/out	long range correction to van der Waals energy
engke	output	kinetic energy
virlrc	in/out	long range correction to van der Waals virial
press	input	external pressure, P_{ext}
taup	input	barostat time constant, τ_P
taut	input	thermostat time constant, τ_T
sigma	input	required system kinetic energy
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircon	input	constraint virial
volm	in/out	volume of MD cell

integer arrays:

lashap	input	see RDSHAKE_1 (mxproc)
lishap	input	see RDSHAKE_1 (mxlshp)
listcon	input	indices for constraint bonds (mxcons, 3)
listme	input	see RDSHAKE_1 (mxatms)
lstfrz	input	key for frozen atoms (mxatms)
listot	input	see RDSHAKE_1 (mxatms)

real arrays:

buffer	work	see MERGE (mxbuff)
cell	in/out	MD cell vectors (9)
dens	in/out	particle number density (mxsvdw)
dxt	work	see RDSHAKE_1 (mxcons)
dxx	work	see RDSHAKE_1 (mxcons)
dxt	work	see RDSHAKE_1 (mxcons)
dyy	work	see RDSHAKE_1 (mxcons)
dzt	work	see RDSHAKE_1 (mxcons)
dzz	work	see RDSHAKE_1 (mxcons)

fx	in/out	x component of force (mxatms)
fy	in/out	y component of force (mxatms)
fz	in/out	z component of force (mxatms)
prmcon	input	constraint distances (mxtcon)
tx	work	(mxatms)
ty	work	(mxatms)
tz	work	(mxatms)
ux	work	(mxatms)
uy	work	(mxatms)
uz	work	(mxatms)
vx	in/out	x component of velocity (mxatms)
vy	in/out	y component of velocity (mxatms)
vz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xd	work	(mxxdf)
xt	work	(mxatms)
xx	in/out	x coordinates (mxatms)
yd	work	(mxxdf)
yt	work	(mxatms)
yy	in/out	y coordinates (mxatms)
zd	work	(mxxdf)
zt	work	(mxatms)
zz	in/out	z coordinates (mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)
eta	output	cell scaling tensor (9)
dens0	work	(mxsvdw)

8.1.97.5 Comments

Uses a leapfrog integration scheme (see NVE_1 for general comments).

The pressure is adjusted by anisotropic changes to the MD cell vectors. The changes to the cell tensor $\hat{H}$ are given by

$$\Delta\hat{H} = \hat{\eta}.\hat{H} \quad (8.5)$$

where the components of $\hat{\eta}$ are given by

$$\eta_{\mu\nu} = \frac{\beta\Delta t}{\tau_P} \left(\frac{\sigma_{\mu\nu}}{V} - \delta_{\mu\nu} P_{\text{ext}} \right) \quad (8.6)$$

and V is the cell volume and $\delta_{\mu\nu}$ is the Kronecker delta. β is the isothermal compressibility (taken to be that of liquid water) and $\hat{\sigma}$ the instantaneous stress tensor in the system.

Choice of τ_P depends on the actual β of the system. In water based systems a value of τ_P in the range (0.5, 5) ps is normally appropriate.

Temperature is controlled by the Berendsen thermostat. The velocities are scaled by χ where:

$$\chi = \left(1 + \frac{\Delta t}{\tau_T} \left(\frac{T}{\mathcal{T}} - 1\right)\right)^{1/2} \quad (8.7)$$

where τ_T is the thermostat relaxation time, $\mathcal{T}$ the instantaneous temperature and T the specified system temperature. Usually a value of τ_T in the range (0.1, 1) ps works well.

The Berendsen thermostat and barostat do not conserve energy. The thermostat and barostat are applied concurrently with a call to RDSHAKE_1 to handle bond constraints using SHAKE. The variable `maxit` controls the number of thermostat/barostat iterations used. This is set to 5 for systems with bond constraints and set to 4 otherwise. This may be changed by editing the source code and recompiling.

8.1.98 NPT_H0**8.1.98.1 Header records**

```

      subroutine npt_h0
      x  (idnode,imcon,mxnode,natms,ntpatm,chip,chit,conint,consv,elrc,
      x  engke,virlrc,press,taup,taut,sigma,temp,tstep,virtot,volm,
      x  lstfrz,buffer,cell,dens,fx,fy,fz,vx,vy,vz,weight,xdf,
      x  xxx,ydf,yyy,zdf,zzz,vx1,vy1,vz1,stress,eta,dens0)

c*****
c
c      dl_poly subroutine for integrating newtonian equations of
c      motion in molecular dynamics - verlet leapfrog with Hoover
c      thermostat + piston.
c
c      reference: Melchionna, Ciccotti and Holian,
c      Mol Phys 1993, 78, p533
c
c      parallel replicated data version
c
c      for systems using bond constraints (using atomic pressure)
c
c      copyright daresbury laboratory 1995/1999
c      author      -      s. melchionna    april 1995
c      and          -      t. forester      april 1995
c      adapted     -      w.smith for atomic systems oct 1999
c
c*****

```

8.1.98.2 Function

The verlet leapfrog algorithm for integrating Newton's equations of motion subject to a Nosé-Hoover thermostat and isotropic barostat. This version is for systems without constraint bonds or rigid bodies.

8.1.98.3 Dependencies

- GDSUM
- IMAGES
- MERGE

8.1.98.4 Arguments

integers:

<code>idnode</code>	input	identity of current processor
<code>imcon</code>	input	key for periodic boundary conditions
<code>mxnode</code>	input	total number of processors in use
<code>natms</code>	input	number of atoms in MD cell
<code>ntpatm</code>	input	number of unique types of atoms
reals:		
<code>chip</code>	in/out	length scaling variable
<code>chit</code>	in/out	velocity scaling variable
<code>conint</code>	in/out	time integral of scaling variables
<code>consv</code>	in/out	scaling variables contribution to extended Hamiltonian
<code>elrc</code>	in/out	long range correction to van der Waals energy
<code>engke</code>	output	kinetic energy
<code>virirc</code>	in/out	long range correction to van der Waals virial
<code>press</code>	input	external pressure, P_{ext}
<code>taup</code>	input	barostat time constant, τ_P
<code>taut</code>	input	thermostat time constant, τ_T
<code>sigma</code>	input	required system kinetic energy
<code>temp</code>	input	required system temperature
<code>tstep</code>	input	simulation time-step, Δt
<code>virtot</code>	input	virial
<code>volm</code>	in/out	volume of MD cell
integer arrays:		
<code>lstfrz</code>	input	key for frozen atoms (<code>mxatms</code>)
real arrays:		
<code>buffer</code>	work	see MERGE (<code>mxbuff</code>)
<code>cell</code>	in/out	MD cell vectors (9)
<code>dens</code>	in/out	particle number density (<code>mxsvdw</code>)
<code>fxx</code>	in/out	x component of force (<code>mxatms</code>)
<code>fyy</code>	in/out	y component of force (<code>mxatms</code>)
<code>fzz</code>	in/out	z component of force (<code>mxatms</code>)
<code>vxx</code>	in/out	x component of velocity (<code>mxatms</code>)
<code>vyy</code>	in/out	y component of velocity (<code>mxatms</code>)
<code>vzz</code>	in/out	z component of velocity (<code>mxatms</code>)
<code>weight</code>	input	atomic masses (<code>mxatms</code>)
<code>xdf</code>	work	x component of intersite vector (<code>mxxdf</code>)
<code>xxx</code>	in/out	x coordinates (<code>mxatms</code>)
<code>ydf</code>	work	y component of intersite vector (<code>mxxdf</code>)
<code>yyy</code>	in/out	y coordinates (<code>mxatms</code>)
<code>zdf</code>	work	z component of intersite vector (<code>mxxdf</code>)
<code>zzz</code>	in/out	z coordinates (<code>mxatms</code>)
<code>vx1</code>	work	(<code>msatms</code>)
<code>vy1</code>	work	(<code>msatms</code>)
<code>vz1</code>	work	(<code>msatms</code>)
<code>stress</code>	in/out	stress tensor (9)

eta	output	cell scaling tensor (9)
dens0	work	(mxsvdw)

8.1.98.5 Comments

Uses a leapfrog integration scheme (see NVE_1 for general comments).

The pressure is adjusted by isotropic changes to the MD cell vectors. At each step cell vectors are scaled by S , where

$$S = \exp(\Delta t \cdot \eta) \quad (8.8)$$

where the time derivative of η is given by

$$\frac{d\eta}{dt} = \frac{(\mathcal{P} - P_{\text{ext}})V}{Nk_B T \tau_P^2} \quad (8.9)$$

where V is the system volume, T the required system temperature, $\mathcal{P}$ the instantaneous atomic pressure and N the number of atoms in the system.

Temperature is controlled by the Nosé-Hoover thermostat. The value of the friction coefficient, χ for the thermostat is found from the first-order differential equation:

$$\frac{d\chi}{dt} = \frac{1}{\tau_T^2} \left(\frac{\mathcal{T}}{T} - 1 \right) \quad (8.10)$$

where $\mathcal{T}$ is the instantaneous temperature. Usually a value of τ_T in the range (0.5, 2) ps works well but this may be system dependent.

The extended Hamiltonian of the system is a conserved quantity and this may be used as a check on the choice of τ_T and τ_P .

The barostat and thermostat require several iterations in order to accurately predict the current temperature and pressure. This is controlled by the variable **maxit** which is set to 4 in this version, but may be changed by editing the source code and recompiling.

8.1.99 NPT_H1**8.1.99.1 Header records**

```

      subroutine npt_h1
      x  (safe,lshmov,idnode,imcon,mxnode,natms,ntpatm,nscons,
      x  ntcons,chip,chit,conint,consv,elrc,engke,virlrc,press,
      x  taup,taut,sigma,temp,tolnce,tstep,virtot,vircon,volm,
      x  lashap,lishap,listcon,listme,lstfrz,listot,buffer,
      x  cell,dens,dxt,dxx,dyt,dyy,dzt,dzz,fxx,fyy,fzz,prmcon,
      x  txx,tyy,tzz,uxx,uyy,uzz,vxx,vyy,vzz,weight,xdf,xxt,
      x  xxx,ydf,yyt,yyy,zdf,zzt,zzz,vx1,vy1,vz1,stress,eta,dens0)

C*****
C
C      dl_poly subroutine for integrating newtonian equations of
C      motion in molecular dynamics - verlet leapfrog with Hoover
C      thermostat + piston.
C
C      reference: Melchionna, Ciccotti and Holian,
C      Mol Phys 1993, 78, p533
C
C      parallel replicated data version
C
C      for systems using bond constraints (using atomic pressure)
C
C      copyright daresbury laboratory 1995
C      author      -      s. melchionna      april 1995
C      and         -      t. forester        april 1995
C
C*****
C

```

8.1.99.2 Function

The verlet leapfrog algorithm for integrating Newton's equations of motion subject to a Nosé-Hoover thermostat and isotropic barostat.

8.1.99.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- RDSHAKE_1

8.1.99.4 Arguments**logicals:**

safe	output	flag for convergence of SHAKE
lshmov	input	flag for shared bond constraints

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ntpatm	input	number of unique types of atoms
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system

reals:

chip	in/out	length scaling variable
chit	in/out	velocity scaling variable
conint	in/out	time integral of scaling variables
consv	in/out	scaling variables contribution to extended Hamiltonian
elrc	in/out	long range correction to van der Waals energy
engke	output	kinetic energy
virlrc	in/out	long range correction to van der Waals virial
press	input	external pressure, P_{ext}
taup	input	barostat time constant, τ_P
taut	input	thermostat time constant, τ_T
sigma	input	required system kinetic energy
temp	input	required system temperature
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
virtot	input	virial from non-constraint terms
vircon	output	constraint virial
volm	in/out	volume of MD cell

integer arrays:

lashap	input	see RDSHAKE_1 (mxproc)
lishap	input	see RDSHAKE_1 (mxlshp)
listcon	input	indices for constraint bonds (mxcons , 3)
listme	input	see RDSHAKE_1 (mxatms)
lstfrz	input	key for frozen atoms (mxatms)
listot	input	see RDSHAKE_1 (mxatms)

real arrays:

buffer	work	see MERGE (mxbuff)
cell	in/out	MD cell vectors (9)
dens	in/out	particle number density (mxsvdw)
dxt	work	see RDSHAKE_1 (mxcons)
dxx	work	see RDSHAKE_1 (mxcons)

dyt	work	see RDSHAKE_1 (mxcons)
dyy	work	see RDSHAKE_1 (mxcons)
dzt	work	see RDSHAKE_1 (mxcons)
dzz	work	see RDSHAKE_1 (mxcons)
fx	in/out	x component of force (mxatms)
fy	in/out	y component of force (mxatms)
fz	in/out	z component of force (mxatms)
prmcon	input	constraint distances (mxtcon)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xdf	work	x component of intersite vector (mxxdf)
xxt	work	(mxatms)
xxx	in/out	x coordinates (mxatms)
ydf	work	y component of intersite vector (mxxdf)
yyt	work	(mxatms)
yyy	in/out	y coordinates (mxatms)
zdf	work	z component of intersite vector (mxxdf)
zzt	work	(mxatms)
zzz	in/out	z coordinates (mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)
eta	output	cell scaling tensor (9)
dens0	work	(mxsvdw)

8.1.99.5 Comments

Uses a leapfrog integration scheme (see NVE_1 for general comments).

The pressure is adjusted by isotropic changes to the MD cell vectors. At each step cell vectors are scaled by S , where

$$S = \exp(\Delta t \cdot \eta) \quad (8.11)$$

where the time derivative of η is given by

$$\frac{d\eta}{dt} = \frac{(\mathcal{P} - P_{\text{ext}})V}{Nk_B T \tau_P^2} \quad (8.12)$$

where V is the system volume, T the required system temperature, $\mathcal{P}$ the instantaneous atomic pressure and N the number of atoms in the system.

Temperature is controlled by the Nosé-Hoover thermostat. The value of the friction coefficient, χ for the thermostat is found from the first-order differential equation:

$$\frac{d\chi}{dt} = \frac{1}{\tau_T^2} \left(\frac{\mathcal{T}}{T} - 1 \right) \quad (8.13)$$

where $\mathcal{T}$ is the instantaneous temperature. Usually a value of τ_T in the range (0.5, 2) ps works well but this may be system dependent.

The extended Hamiltonian of the system is a conserved quantity and this may be used as a check on the choice of τ_T and τ_P .

This subroutine also includes a call to `RDSHAKE_1` to handle bond constraints by the SHAKE method. The barostat and thermostat require several iterations in order to accurately predict the current temperature and pressure. This is controlled by the variable `maxit` which is set to 5 for systems with bond constraints and set to 4 otherwise.

8.1.100 NPT_H3**8.1.100.1 Header records**

```

      subroutine npt_h3
      x  (safe,lshmov,idnode,imcon,mxnode,natms,ntpatm,nscons,
      x  ntcons,chit,conint,consv,elrc,engke,virlrc,press,
      x  taup,taut,sigma,temp,tolnce,tstep,vircon,volm,
      x  lashap,lishap,listcon,listme,lstfrz,listot,buffer,
      x  cell,dens,dxt,dxx,dyt,dyy,dzt,dzz,fxx,fyy,fzz,prmcon,
      x  txx,tyy,tzz,uxx,uyy,uzz,vxx,vyy,vzz,weight,xdf,xxt,
      x  xxx,ydf,yyt,yyy,zdf,zzt,zzz,vx1,vy1,vz1,stress,eta,dens0)

C*****
C
C      dl_poly subroutine for integrating newtonian equations of
C      motion in molecular dynamics - verlet leapfrog with Hoover
C      thermostat + piston.
C
C      Parrinello- Rahman type : changing cell shape.
C
C      reference: Melchionna, Ciccotti and Holian,
C      Mol Phys 1993, 78, p533
C
C      parallel replicated data version
C
C      for systems using bond constraints (using atomic pressure)
C
C      copyright daresbury laboratory 1995
C      author      -      t. forester      june 1995
C
C*****
C

```

8.1.100.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion. The system is coupled to a Nosé-Hoover thermostat and the Melchionna variant of the anisotropic Hoover barostat.

8.1.100.3 Dependencies

- ERROR
- GDSUM

- IMAGES
- MERGE
- RDSHAKE_1

8.1.100.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
lshmov	input	flag for shared bond constraints

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ntpatm	input	number of unique types of atoms
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system

reals:

chit	in/out	velocity scaling variable
conint	in/out	time integral of scaling variables
consv	in/out	scaling variables contribution to extended Hamiltonian
elrc	in/out	long range correction to van der Waals energy
engke	output	kinetic energy
virlrc	in/out	long range correction to van der Waals virial
press	input	external pressure, P_{ext}
taup	input	barostat time constant, τ_P
taut	input	thermostat time constant, τ_T
sigma	input	required system kinetic energy
temp	input	required system temperature
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircon	input	constraint virial
volm	in/out	volume of MD cell

integer arrays:

lashap	input	see RDSHAKE_1 (mxproc)
lishap	input	see RDSHAKE_1 (mxlshp)
listcon	input	indices for constraint bonds (mxcons , 3)
listme	input	see RDSHAKE_1 (mxatms)
lstfrz	input	key for frozen atoms (mxatms)
listot	input	see RDSHAKE_1 (mxatms)

real arrays:

buffer	work	see MERGE (mxbuff)
cell	in/out	MD cell vectors (9)

dens	in/out	particle number density (mxsvdw)
dxt	work	see RDSHAKE_1 (mxcons)
dxs	work	see RDSHAKE_1 (mxcons)
dys	work	see RDSHAKE_1 (mxcons)
dyt	work	see RDSHAKE_1 (mxcons)
dzt	work	see RDSHAKE_1 (mxcons)
dzt	work	see RDSHAKE_1 (mxcons)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
prmcon	input	constraint distances (mxtcon)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xdf	work	(mxxdf)
xxt	work	(mxatms)
xxx	in/out	x coordinates (mxatms)
ydf	work	(mxxdf)
yyt	work	(mxatms)
yyy	in/out	y coordinates (mxatms)
zdf	work	(mxxdf)
zzt	work	(mxatms)
zzz	in/out	z coordinates (mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor, $\hat{\sigma}$ (9)
eta	in/out	cell scaling tensor, $\hat{\eta}$ (9)
dens0	work	(mxsvdw)

8.1.100.5 Comments

Uses a leapfrog integration scheme (see NVE_1 for general comments). This subroutine must be compiled with the STRESS option turned on. The pressure is adjusted by anisotropic changes to the MD cell vectors. The changes to the cell tensor $\hat{H}$ are given by

$$\Delta\hat{H} = \exp(\Delta t.\hat{\eta}) \quad (8.14)$$

where the time derivative of the tensor $\hat{\eta}$ is given by

$$\frac{d\hat{\eta}}{dt} = \frac{(\hat{\sigma} - \hat{\delta}P_{\text{ext}})V}{Nk_B T\tau_P^2} \quad (8.15)$$

where $\hat{\sigma}$ is the instantaneous atomic stress tensor, $\hat{\delta}$ the Kronecker delta, V is the system volume, N the number of atoms in the system, and T the required system temperature. Choice of τ_P is system dependent but normally a value in the range (1, 10) ps is appropriate.

Temperature is controlled by the Nosé-Hoover thermostat. The value of the friction coefficient, χ for the thermostat is found from the first-order differential equation:

$$\frac{d\chi}{dt} = \frac{1}{\tau_T^2} \left(\frac{\mathcal{T}}{T} - 1 \right) \quad (8.16)$$

where $\mathcal{T}$ is the instantaneous temperature. Usually a value of τ_T in the range (0.5, 2) ps works well but this may be system dependent.

The extended Hamiltonian of the system is a conserved quantity and this may be used as a check on the choice of τ_T and τ_P .

This subroutine also includes a call to `RDSHAKE_1` to handle bond constraints by the SHAKE method. The barostat and thermostat require several iterations in order to accurately predict the current temperature and pressure. This is controlled by the variable `maxit` which is set to 5 for systems with bond constraints and set to 4 otherwise.

8.1.101 NPTQ_B1

8.1.101.1 Header records

```

      subroutine nptq_b1
      x  (safe,safeq,lshmov,imcon,idnode,mxnode,natms,ngroup,nscons,
      x  ntcons,ntfree,ntpatm,elrc,engke,engrot,virlrc,press,
      x  quattol,sigma,taup,taut,tolnce,tstep,virtot,vircom,
      x  vircon,volm,lstme,lishap,lashap,lstfrz,listcon,
      x  listme,lstfre,lstgtp,lstrgd,listot,numgsit,cell,
      x  dens,gcmx1,gcmz1,dxx,dyy,dzz,dxt,dyt,dzt,
      x  gvz1,gvz1,gvz1,uxx,uyy,uzz,prmcon,xdx,ydx,zdx,omx,
      x  omy,omz,omx1,omy1,omz1,opx,opy,opz,oqx,oqy,oqz,
      x  q0,q1,q2,q3,qn0,qn1,qn2,qn3,gxx,gyy,gzz,xxx,yyy,
      x  zzz,vxx,vyy,vzz,gvxx,gvyy,gvzz,fx,fy,fz,tqx,tqy,
      x  tqz,weight,rotinx,rotiny,rotinz,gcmx,gcmz,gcmz,gmass,
      x  buffer,xxt,yyt,zzt,txx,tyy,tzz,vx1,vy1,vz1,stress,eta)

C*****
C
C      dlpoly routine to integrate rigid body equations of motion
C      using implicit leapfrog quaternion algorithm
C      plus bond constraints - PROVIDED rigid body sites
C      and constraint sites do not coincide.
C
C      dl_poly subroutine for integrating newtonian equations of
C      motion in molecular dynamics - verlet leapfrog with Berendsen
C      thermostat and barostat.
C
C      parallel replicated data version : block data
C
C      for systems using bond constraints
C
C      parallel replicated data version : block data
C
C      tqx,tqy,tqz = torque in lab fixed frame (input)
C      omx,omy,omz =angular velocity in body fixed frame (principal axis)
C      rotinx,y,z = rotational inertia in body fixed frame
C
C      copyright daresbury laboratory 1993
C      author      t.forester october 1993
C      amended     t.forester dec 1994 : block data
C
C*****

```

8.1.101.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion for systems containing rigid bodies. The system is coupled to a Berendsen thermostat and isotropic barostat. Implicit quaternion equations are used for the rigid body equations of motion.

8.1.101.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- MERGE1
- RDSHAKE_1

8.1.101.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
safeq	output	flag for convergence of quaternion algorithm
lshmov	input	flag for shared bond constraints

integers:

imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ngrp	input	number of rigid body units in system
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system
ntfree	input	number of sites in system not in rigid body units
ntpatm	input	number of unique types of atoms

reals:

elrc	in/out	long range correction to van der Waals energy
engke	output	kinetic energy
engrot	output	rotational kinetic energy
virlrc	in/out	long range correction to van der Waals virial
press	input	external pressure, P_{ext}
quattol	input	tolerance for quaternion algorithm
sigma	input	required system kinetic energy
taup	input	barostat time constant, τ_P
taut	input	thermostat time constant, τ_T
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt

<code>virtot</code>	input	virial from non-constraint terms
<code>vircom</code>	output	molecule to c.o.m. virial correction
<code>vircon</code>	output	constraint virial
<code>volm</code>	in/out	volume of MD cell
integer arrays:		
<code>lstme</code>	input	indice list for MERGE1 (<code>mxatms</code>)
<code>lishap</code>	input	see RDSHAKE_1 (<code>mxlshp</code>)
<code>lashap</code>	input	see RDSHAKE_1 (<code>mxproc</code>)
<code>lstfrz</code>	input	key for frozen atoms (<code>mxatms</code>)
<code>listcon</code>	input	indices for constraint bonds (<code>mxcons,3</code>)
<code>listme</code>	input	see RDSHAKE_1 (<code>mxatms</code>)
<code>lstfre</code>	input	indices of non-rigid unit sites handled by current processor (<code>mxatms</code>)
<code>lstgtp</code>	input	pointer from rigid unit index to rigid unit type (<code>mxgrp</code>)
<code>lstrgd</code>	input	indices of rigid unit sites handled by current processor (<code>mxgadm</code>)
<code>listot</code>	input	see RDSHAKE_1 (<code>mxatms</code>)
<code>numgsit</code>	input	number of sites in a rigid unit type (<code>mxungp</code>)
real arrays:		
<code>cell</code>	in/out	MD cell vectors (9)
<code>dens</code>	in/out	particle number density (<code>mxsvdw</code>)
<code>gcmx1</code>	work	(<code>msgrp</code>)
<code>gcmy1</code>	work	(<code>msgrp</code>)
<code>gcmz1</code>	work	(<code>msgrp</code>)
<code>dx</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dy</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dzz</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dxt</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dyt</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dzt</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>gvx1</code>	work	(<code>msgrp</code>)
<code>gvy1</code>	work	(<code>msgrp</code>)
<code>gvz1</code>	work	(<code>msgrp</code>)
<code>uxx</code>	work	(<code>mxatms</code>)
<code>uyy</code>	work	(<code>mxatms</code>)
<code>uzz</code>	work	(<code>mxatms</code>)
<code>prmcon</code>	input	constraint distances (<code>mxtcon</code>)
<code>xdf</code>	work	(<code>mxxdf</code>)
<code>ydf</code>	work	(<code>mxxdf</code>)
<code>zdf</code>	work	(<code>mxxdf</code>)
<code>omx</code>	in/out	x component of angular velocity (<code>mxgrp</code>)
<code>omy</code>	in/out	y component of angular velocity (<code>mxgrp</code>)
<code>omz</code>	in/out	z component of angular velocity (<code>mxgrp</code>)
<code>omx1</code>	work	(<code>msgrp</code>)

omy1	work	(msgrp)
omz1	work	(msgrp)
opx	work	(msgrp)
opy	work	(msgrp)
opz	work	(msgrp)
oqx	work	(msgrp)
oqy	work	(msgrp)
oqz	work	(msgrp)
q0	in/out	first component of quaternion vector (mxgrp)
q1	in/out	second component of quaternion vector (mxgrp)
q2	in/out	third component of quaternion vector (mxgrp)
q3	in/out	fourth component of quaternion vector (mxgrp)
qn0	work	(msgrp)
qn1	work	(msgrp)
qn2	work	(msgrp)
qn3	work	(msgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
tqx	work	x component of torque on rigid unit (msgrp)
tqy	work	y component of torque on rigid unit (msgrp)
tqz	work	z component of torque on rigid unit (msgrp)
weight	input	atomic masses (mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)
gcmx	in/out	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	in/out	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	in/out	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	input	rigid unit mass (mxungp)
buffer	work	(mxbuff)
xxt	work	(mxatms)

yyt	work	(mxatms)
zzt	work	(mxatms)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)
eta	output	cell scaling tensor (9)

8.1.101.5 Comments

See NPT_B1 for general comments on the Berendsen barostat and thermostat.

Finchams implicit quaternion leapfrog algorithm is used to solve the rigid body equations of motion. This subroutine does not allow a rigid body to be involved in bond constraints with an atom or another rigid body. Should this be required use the routine NPTQ_B2 instead.

8.1.102 NPTQ_B2

8.1.102.1 Header records

```

      subroutine nptq_b2
      x  (safe,safeq,lshmov,imcon,idnode,mxnode,natms,ngroup,nscons,
      x  ntcons,ntfree,ntpatm,elrc,engke,engrot,virlrc,press,
      x  quattol,sigma,taup,taut,tolnce,tstep,vircom,vircon,
      x  virtot,volm,lstme,lishap,lashap,lstfrz,listcon,listme,
      x  lstfre,lstcsit,lstbod,lstgtp,lstrgd,listot,numgsit,cell,
      x  gcmx1,gcmx2,gcmx3,dens,dxx,dyy,dzz,dxt,dyt,dzt,uxx,uyy,
      x  uzz,prmcon,omx,omy,omz,opx,opy,opz,oqx,oqy,oqz,q0,q1,q2,
      x  q3,qn0,qn1,qn2,qn3,gxx,gyy,gzz,xxx,yyy,zzz,vxx,vyy,vzz,
      x  gvxx,gvyy,gvzz,fxx,fyy,fzz,tqx,tqy,tqz,weight,rotinx,
      x  rotiny,rotinz,gcmx,gcmx2,gcmx3,gmass,buffer,txt,yyt,zzt,
      x  txx,tyy,tzz,vx1,vy1,vz1,stress,gvx1,gvx2,gvx3,omy1,omy2,omy3,
      x  omz1,redmass,eta)

C*****
C
C      dlpoly routine to integrate rigid body equations of motion
C      using implicit leapfrog quaternion algorithm
C      plus bond constraints - rigid body sites and constraint sites
C      may coincide.
C
C      dl_poly subroutine for integrating newtonian equations of
C      motion in molecular dynamics - verlet leapfrog with Berendsen
C      thermostat and barostat.
C
C      parallel replicated data version : block data
C
C      tqx,tqy,tqz = torque in lab fixed frame (input)
C      omx,omy,omz = angular velocity in body fixed frame (principal axis)
C      rotinx,y,z = rotational inertia in body fixed frame
C
C      copyright daresbury laboratory 1995
C      author      t.forester june 1995
C
C*****

```

8.1.102.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion for systems containing rigid bodies. The rigid bodies may be linked by constraint bonds. The system is coupled to a Berendsen thermostat and isotropic barostat Implicit quaternion equations

are used for the rigid body equations of motion.

8.1.102.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- MERGE1
- MERGE4
- QSHAKE

8.1.102.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
safeq	output	flag for convergence of quaternion algorithm
lshmov	input	flag for shared bond constraints

integers:

imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ngrp	input	number of rigid body units in system
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system
ntfree	input	number of sites in system not in rigid body units
ntpatm	input	number of unique types of atoms

reals:

elrc	in/out	long range correction to van der Waals energy
engke	output	kinetic energy
engrot	output	rotational kinetic energy
virlrc	in/out	long range correction to van der Waals virial
press	input	external pressure, P_{ext}
quattol	input	tolerance for quaternion algorithm
sigma	input	required system kinetic energy
taup	input	barostat time constant, τ_P
taut	input	thermostat time constant, τ_T
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircom	output	molecule to c.o.m. virial correction
vircon	output	constraint virial

virtot	input	virial from non-constraint terms
volm	in/out	volume of MD cell
integer arrays:		
lstme	input	indice list for MERGE1 (mxatms)
lishap	input	see QSHAKE (mxlshp)
lashap	input	see QSHAKE (mxproc)
lstfrz	input	key for frozen atoms (mxatms)
listcon	input	indices for constraint bonds (mxcons ,3)
listme	input	see QSHAKE (mxatms)
lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
lstcsit	input	indices of rigid unit sites in constraints (2*mxcons)
lstbod	input	pointer from site index to rigid unit index (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgadm)
listot	input	see QSHAKE (mxatms)
numgsit	input	number of sites in a rigid unit type (mxungp)
real arrays:		
cell	in/out	MD cell vectors (9)
gcmx1	work	(msgrp)
gcmy1	work	(msgrp)
gcmz1	work	(msgrp)
dens	in/out	particle number density (mxsvdw)
dx	work	see QSHAKE (mxcons)
dy	work	see QSHAKE (mxcons)
dzz	work	see QSHAKE (mxcons)
dxt	work	see QSHAKE (mxcons)
dzt	work	see QSHAKE (mxcons)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
prmcon	input	constraint distances (mxtcon)
omx	in/out	x component of angular velocity (mxgrp)
omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
opx	work	(msgrp)
opy	work	(msgrp)
opz	work	(msgrp)
oqx	work	(msgrp)
oqy	work	(msgrp)
oqz	work	(msgrp)
q0	in/out	first component of quaternion vector (mxgrp)

q1	in/out	second component of quaternion vector (mxgrp)
q2	in/out	third component of quaternion vector (mxgrp)
q3	in/out	fourth component of quaternion vector (mxgrp)
qn0	work	(msgrp)
qn1	work	(msgrp)
qn2	work	(msgrp)
qn3	work	(msgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
txx	work	x component of torque on rigid unit (msgrp)
txy	work	y component of torque on rigid unit (msgrp)
txz	work	z component of torque on rigid unit (msgrp)
weight	input	atomic masses (mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)
gcmx	in/out	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	in/out	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	in/out	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	input	rigid unit mass (mxungp)
buffer	work	(mxbuff)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)

gvx1	work	(msgrp)
gvy1	work	(msgrp)
gvz1	work	(msgrp)
omx1	work	(msgrp)
omy1	work	(msgrp)
omz1	work	(msgrp)
redmass	work	effective reduced mass along constraint bond (mxcons)
eta	output	cell scaling tensor (9)

8.1.102.5 Comments

See NPT_B1 for general comments on the Berendsen thermostat and barostat. This subroutine allows a rigid body to be involved in bond constraints with an atom or another rigid body. Should this not be required, the routine NPTQ_B1 is more efficient.

8.1.103 NPTQ_B3

8.1.103.1 Header records

```

      subroutine nptq_b3
x   (safe,safeq,lshmov,imcon,idnode,mxnode,natms,ngroup,nscons,
x   ntcons,ntfree,ntpatm,elrc,engke,engrot,virlrc,press,
x   quattol,sigma,taup,taut,tolnce,tstep,vircom,vircon,volm,
x   lstme,lishap,lashap,lstfrz,listcon,listme,lstfre,lstgtp,
x   lstrgd,listot,numgsit,cell,dens,gcmx1,gcmx1,gcmz1,dxx,
x   dyy,dzz,dxt,dyt,dzt,gvx1,gvy1,gvz1,uxx,uyy,uzz,prmcon,
x   xdf,ydf,zdf,omx,omy,omz,omx1,omy1,omz1,opx,opy,opz,oqx,
x   oxy,oqz,q0,q1,q2,q3,qn0,qn1,qn2,qn3,gxx,gyy,gzz,xxx,yyy,
x   zzz,vxx,vyy,vzz,gvxx,gvyv,gvzz,fxx,fyy,fzz,tqx,tqy,
x   tqz,weight,rotinx,rotiny,rotinz,gcmx,gcmx,gcmz,gmass,
x   buffer,xxt,yyt,zzt,txx,tyy,tzz,vx1,vy1,vz1,stress,eta)

C*****
C
C   dlpoly routine to integrate rigid body equations of motion
C   using implicit leapfrog quaternion algorithm
C   plus bond constraints - PROVIDED rigid body sites
C   and constraint sites do not coincide.
C
C   dl_poly subroutine for integrating newtonian equations of
C   motion in molecular dynamics - verlet leapfrog with Berendsen
C   thermostat and barostat.
C   isothermal compressibility (beta) set to that of liquid water
C   = 0.007372 dlpoly units
C
C   parallel replicated data version : block data
C
C   for systems using bond constraints
C
C   parallel replicated data version : block data
C
C   tqx,tqy,tqz = torque in lab fixed frame (input)
C   omx,omy,omz =angular velocity in body fixed frame (principal axis)
C   rotinx,y,z = rotational inertia in body fixed frame
C
C   copyright daresbury laboratory 1993
C   author      t.forester october 1993
C   amended     t.forester dec 1994 : block data
C

```

C*****

8.1.103.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion for systems containing rigid bodies. The system is coupled to a Berendsen thermostat and anisotropic barostat. Implicit quaternion equations are used for the rigid body equations of motion.

8.1.103.3 Dependencies

- ERROR
- GDSUM
- IMAGES
- MERGE
- MERGE1
- RDSHAKE_1

8.1.103.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
safeq	output	flag for convergence of quaternion algorithm
lshmov	input	flag for shared bond constraints

integers:

imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ngrp	input	number of rigid body units in system
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system
ntfree	input	number of sites in system not in rigid body units
ntpatm	input	number of unique types of atoms

reals:

elrc	in/out	long range correction to van der Waals energy
engke	output	kinetic energy
engrot	output	rotational kinetic energy
virlrc	in/out	long range correction to van der Waals virial
press	input	external pressure, P_{ext}
quattol	input	tolerance for quaternion algorithm
sigma	input	required system kinetic energy

taup	input	barostat time constant, τ_P
taut	input	thermostat time constant, τ_T
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircom	output	molecule to c.o.m. virial correction
vircon	output	constraint virial
volm	in/out	volume of MD cell
integer arrays:		
lstme	input	indice list for MERGE1 (mxatms)
lishap	input	see RDSHAKE_1 (mxlshp)
lashap	input	see RDSHAKE_1 (mxproc)
lstfrz	input	key for frozen atoms (mxatms)
listcon	input	indices for constraint bonds (mxcons ,3)
listme	input	see RDSHAKE_1 (mxatms)
lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgadm)
listot	input	see RDSHAKE_1 (mxatms)
numgsit	input	number of sites in a rigid unit type (mxungp)
real arrays:		
cell	in/out	MD cell vectors (9)
dens	in/out	particle number density (mxsvdw)
gcmx1	work	(msggrp)
gcmy1	work	(msggrp)
gcmz1	work	(msggrp)
dxx	work	see RDSHAKE_1 (mxcons)
dyy	work	see RDSHAKE_1 (mxcons)
dzz	work	see RDSHAKE_1 (mxcons)
dxt	work	see RDSHAKE_1 (mxcons)
dyt	work	see RDSHAKE_1 (mxcons)
dzt	work	see RDSHAKE_1 (mxcons)
gvx1	work	(msggrp)
gvy1	work	(msggrp)
gvz1	work	(msggrp)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
prmcon	input	constraint distances (mxtcon)
xdf	work	(mxxdf)
ydf	work	(mxxdf)
zdf	work	(mxxdf)
omx	in/out	x component of angular velocity (mxgrp)

omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
omx1	work	(msgrp)
omy1	work	(msgrp)
omz1	work	(msgrp)
opx	work	(msgrp)
opy	work	(msgrp)
opz	work	(msgrp)
oqx	work	(msgrp)
oqy	work	(msgrp)
oqz	work	(msgrp)
q0	in/out	first component of quaternion vector (mxgrp)
q1	in/out	second component of quaternion vector (mxgrp)
q2	in/out	third component of quaternion vector (mxgrp)
q3	in/out	fourth component of quaternion vector (mxgrp)
qn0	work	(msgrp)
qn1	work	(msgrp)
qn2	work	(msgrp)
qn3	work	(msgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
tqx	work	x component of torque on rigid unit (msgrp)
tqy	work	y component of torque on rigid unit (msgrp)
tqz	work	z component of torque on rigid unit (msgrp)
weight	input	atomic masses (mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)
gcmx	in/out	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	in/out	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	in/out	z coordinate of rigid unit c.o.m. (mxgrp)

gmass	input	rigid unit mass (mxungp)
buffer	work	(mxbuff)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)
eta	output	cell scaling tensor (9)

8.1.103.5 Comments

See NPT_B3 for general comments on the Berendsen thermostat and anisotropic barostat.

Finchams implicit quaternion leapfrog algorithm is used to solve the rigid body equations of motion. This subroutine does not allow a rigid body to be involved in bond constraints with an atom or another rigid body. Should this be required use the routine NPTQ_B4.

8.1.104 NPTQ_B4

8.1.104.1 Header records

```

      subroutine nptq_b4
x  (safe,safeq,lshmov,imcon,idnode,mxnode,natms,ngroup,nscons,
x  ntcons,ntfree,ntpatm,elrc,engke,engrot,virlrc,press,
x  quattol,sigma,taup,taut,tolnce,tstep,vircom,vircon,volm,
x  lstme,lishap,lashap,lstfrz,listcon,listme,lstfre,lstgtp,
x  lstrgd,listot,lstcsit,lstbod,numgsit,cell,dens,gcmx1,
x  gcmy1,gcmz1,dxx,dyy,dzz,dxt,dyt,dzt,gvx1,gvx1,gvz1,uxx,
x  uyy,uzz,prmcon,omx,omy,omz,omx1,omy1,omz1,opx,opy,opz,
x  oqx,oqy,oqz,q0,q1,q2,q3,qn0,qn1,qn2,qn3,gxx,gyy,gzz,xxx,
x  yyy,zzz,vxx,vyy,vzz,gvxx,gvyy,gvzz,fx,fyy,fzz,tqx,tqy,
x  tqz,weight,rotinx,rotiny,rotinz,gcmx,gcmy,gcmz,gmass,
x  buffer,xxt,yyt,zzt,txx,tyy,tzz,vx1,vy1,vz1,stress,redmass,eta)

C*****
C
C      dlpoly routine to integrate rigid body equations of motion
C      using implicit leapfrog quaternion algorithm
C      plus bond constraints. Rigid body sites and constraint sites may
C      coincide.
C
C      dl_poly subroutine for integrating newtonian equations of
C      motion in molecular dynamics - verlet leapfrog with Berendsen
C      thermostat and barostat.
C
C      parallel replicated data version : block data
C
C      for systems using bond constraints
C
C      tqx,tqy,tqz = torque in lab fixed frame (input)
C      omx,omy,omz =angular velocity in body fixed frame (principal axis)
C      rotinx,y,z = rotational inertia in body fixed frame
C
C      copyright daresbury laboratory 1995
C      author      t.forester june 1995
C
C*****

```

8.1.104.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion for systems containing rigid bodies. The Rigid bodies may be linked by constraint bonds. The system is

coupled to a Berendsen thermostat and anisotropic barostat. Implicit quaternion equations are used for the rigid body equations of motion.

8.1.104.3 Dependencies

- ERROR
- GDSUM
- IMAGES
- MERGE
- MERGE1
- MERGE4
- QSHAKE

8.1.104.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
safeq	output	flag for convergence of quaternion algorithm
lshmov	input	flag for shared bond constraints

integers:

imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ngrp	input	number of rigid body units in system
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system
ntfree	input	number of sites in system not in rigid body units
ntpatm	input	number of unique types of atoms

reals:

elrc	in/out	long range correction to van der Waals energy
engke	output	kinetic energy
engrot	output	rotational kinetic energy
virirc	in/out	long range correction to van der Waals virial
press	input	external pressure, P_{ext}
quattol	input	tolerance for quaternion algorithm
sigma	input	required system kinetic energy
taup	input	barostat time constant, τ_P
taut	input	thermostat time constant, τ_T
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt

vircom	output	molecule to c.o.m. virial correction
vircon	output	constraint virial
volm	in/out	volume of MD cell
integer arrays:		
lstme	input	indice list for MERGE1 (mxatms)
lishap	input	see QSHAKE (mxlshp)
lashap	input	see QSHAKE (mxproc)
lstfrz	input	key for frozen atoms (mxatms)
listcon	input	indices for constraint bonds (mxcons ,3)
listme	input	see QSHAKE (mxatms)
lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgadm)
listot	input	see QSHAKE (mxatms)
lstcsit	input	indices of rigid unit sites in constraints (2*mxcons)
lstbod	input	pointer from site index to rigid unit index (mxatms)
numgsit	input	number of sites in a rigid unit type (mxungp)
real arrays:		
cell	in/out	MD cell vectors (9)
dens	in/out	particle number density (mxsvdw)
gcmx1	work	(msgrp)
gcmy1	work	(msgrp)
gcmz1	work	(msgrp)
dxs	work	see QSHAKE (mxcons)
dys	work	see QSHAKE (mxcons)
dzs	work	see QSHAKE (mxcons)
dxt	work	see QSHAKE (mxcons)
dys	work	see QSHAKE (mxcons)
dzt	work	see QSHAKE (mxcons)
gvx1	work	(msgrp)
gvy1	work	(msgrp)
gvz1	work	(msgrp)
uxs	work	(mxatms)
uys	work	(mxatms)
uzs	work	(mxatms)
prmcon	input	constraint distances (mxtcon)
omx	in/out	x component of angular velocity (mxgrp)
omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
omx1	work	(msgrp)
omy1	work	(msgrp)
omz1	work	(msgrp)

opx	work	(msgrp)
opy	work	(msgrp)
opz	work	(msgrp)
oqx	work	(msgrp)
oqy	work	(msgrp)
oqz	work	(msgrp)
q0	in/out	first component of quaternion vector (mxgrp)
q1	in/out	second component of quaternion vector (mxgrp)
q2	in/out	third component of quaternion vector (mxgrp)
q3	in/out	fourth component of quaternion vector (mxgrp)
qn0	work	(msgrp)
qn1	work	(msgrp)
qn2	work	(msgrp)
qn3	work	(msgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
txx	work	x component of torque on rigid unit (msgrp)
txy	work	y component of torque on rigid unit (msgrp)
txz	work	z component of torque on rigid unit (msgrp)
weight	input	atomic masses (mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)
gcmx	in/out	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	in/out	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	in/out	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	input	rigid unit mass (mxungp)
buffer	work	(mxbuff)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)

txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)
redmass	work	effective reduced mass along constraint bond (mxcons)
eta	output	cell scaling tensor (9)

8.1.104.5 Comments

See NPT_B3 for general comments on the Berendsen thermostat and anisotropic barostat. This subroutine allows a rigid body to be involved in bond constraints with an atom or another rigid body. Should this not be required, the routine NPTQ_B3 is more efficient.

8.1.105 NPTQ_H1

8.1.105.1 Header records

```

      subroutine nptq_h1
      x (safe,safeq,lshmov,imcon,idnode,mxnode,natms,ngroup,nscons,
      x  ntcons,ntfree,ntpatm,chip,chit,consv,conint,elrc,engke,
      x  engrot,virlrc,press,quattol,sigma,taup,taut,temp,tolnce,
      x  tstep,virtot,vircom,vircon,volm,lstme,lishap,lashap,
      x  lstfrz,listcon,listme,lstfre,lstgtp,lstrgd,listot,
      x  numgsit,cell,dens,dxx,dyy,dzz,dxt,dyt,dzt,gcmx1,gcmx1,
      x  gcmz1,gvx1,gyv1,gvz1,uxx,uyy,uzz,prmcon,xdf,ydf,zdf,omx,
      x  omy,omz,omx1,omy1,omz1,opx,opy,opz,oqx,oqy,oqz,q0,q1,q2,
      x  q3,qn0,qn1,qn2,qn3,gxx,gyy,gzz,xxx,yyy,zzz,vxx,vyy,vzz,
      x  gvxx,gvyy,gvzz,fx,fyy,fzz,tqx,tqy,tqz,weight,rotinx,
      x  rotiny,rotinz,gcmx,gcmx,gcmz,gmass,buffer,txt,yyt,zzt,
      x  txx,tyy,tzz,vx1,vy1,vz1,stress,eta)

C*****
C
C      dlpoly routine to integrate rigid body equations of motion
C      using implicit leapfrog quaternion algorithm
C      plus bond constraints - PROVIDED rigid body sites
C      and constraint sites do not coincide.
C
C      dl_poly subroutine for integrating newtonian equations of
C      motion in molecular dynamics - verlet leapfrog with Hoover
C      thermostat and barostat (Melchionna et al variant)
C
C      parallel replicated data version : block data
C
C      for systems using bond constraints
C
C      parallel replicated data version : block data
C
C      tqx,tqy,tqz = torque in lab fixed frame (input)
C      omx,omy,omz =angular velocity in body fixed frame (principal axis)
C      rotinx,y,z = rotational inertia in body fixed frame
C
C      copyright daresbury laboratory 1993
C      author      t.forester october 1993
C      amended     t.forester dec 1994 : block data
C
C*****

```

8.1.105.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion for systems containing rigid bodies. The system is coupled to a Nosé-Hoover thermostat and the Melchionna variant of the Hoover barostat. Implicit quaternion equations are used for the rigid body equations of motion.

8.1.105.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- MERGE1
- RDSHAKE_1

8.1.105.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
safeq	output	flag for convergence of quaternion algorithm
lshmov	input	flag for shared bond constraints

integers:

imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ngrp	input	number of rigid body units in system
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system
ntfree	input	number of sites in system not in rigid body units
ntpatm	input	number of unique types of atoms

reals:

chip	in/out	length scaling variable
chit	in/out	velocity scaling variable
consv	in/out	scaling variables contribution to extended Hamiltonian
conint	in/out	time integral of scaling variables
elrc	in/out	long range correction to van der Waals energy
engke	output	kinetic energy
engrot	output	rotational kinetic energy
virirc	in/out	long range correction to van der Waals virial
press	input	external pressure
quattol	input	tolerance for quaternion algorithm

<code>sigma</code>	input	required system kinetic energy
<code>taup</code>	input	barostat time constant, τ_P
<code>taut</code>	input	thermostat time constant, τ_T
<code>temp</code>	input	required system temperature
<code>tolnce</code>	input	tolerance for SHAKE algorithm
<code>tstep</code>	input	simulation time-step, Δt
<code>virtot</code>	input	virial from non-constraint terms
<code>vircom</code>	output	molecule to c.o.m. virial correction
<code>vircon</code>	output	constraint virial
<code>volm</code>	in/out	volume of MD cell
integer arrays:		
<code>lstme</code>	input	indice list for MERGE1 (<code>mxatms</code>)
<code>lishap</code>	input	see RDSHAKE_1 (<code>mxlshp</code>)
<code>lashap</code>	input	see RDSHAKE_1 (<code>mxproc</code>)
<code>lstfrz</code>	input	key for frozen atoms (<code>mxatms</code>)
<code>listcon</code>	input	indices for constraint bonds (<code>mxcons</code> , 3)
<code>listme</code>	input	see RDSHAKE_1 (<code>mxatms</code>)
<code>lstfre</code>	input	indices of non-rigid unit sites handled by current processor (<code>mxatms</code>)
<code>lstgtp</code>	input	pointer from rigid unit index to rigid unit type (<code>mxgrp</code>)
<code>lstrgd</code>	input	indices of rigid unit sites handles by current processor (<code>mxgadm</code>)
<code>listot</code>	input	see RDSHAKE_1 (<code>mxatms</code>)
<code>numgsit</code>	input	number of sites in a rigid unit type (<code>mxungp</code>)
real arrays:		
<code>cell</code>	in/out	MD cell vectors (9)
<code>dens</code>	in/out	particle number density (<code>mxsvdw</code>)
<code>dx</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dy</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dzz</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dxt</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dyt</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dzt</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>gcmx1</code>	work	(<code>msgrp</code>)
<code>gcmy1</code>	work	(<code>msgrp</code>)
<code>gcmz1</code>	work	(<code>msgrp</code>)
<code>gvx1</code>	work	(<code>msgrp</code>)
<code>gvy1</code>	work	(<code>msgrp</code>)
<code>gvz1</code>	work	(<code>msgrp</code>)
<code>uxx</code>	work	(<code>mxatms</code>)
<code>uyy</code>	work	(<code>mxatms</code>)
<code>uzz</code>	work	(<code>mxatms</code>)
<code>prmcon</code>	input	constraint distances (<code>mxtcon</code>)
<code>xdf</code>	work	(<code>mxxdf</code>)

ydf	work	(mxxdf)
zdf	work	(mxxdf)
omx	in/out	x component of angular velocity (mxgrp)
omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
omx1	work	(msgrp)
omy1	work	(msgrp)
omz1	work	(msgrp)
opx	work	(msgrp)
opy	work	(msgrp)
opz	work	(msgrp)
oqx	work	(msgrp)
oqy	work	(msgrp)
oqz	work	(msgrp)
q0	in/out	first component of quaternion vector (mxgrp)
q1	in/out	second component of quaternion vector (mxgrp)
q2	in/out	third component of quaternion vector (mxgrp)
q3	in/out	fourth component of quaternion vector (mxgrp)
qn0	work	(msgrp)
qn1	work	(msgrp)
qn2	work	(msgrp)
qn3	work	(msgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
tqx	work	x component of torque on rigid unit (msgrp)
tqy	work	y component of torque on rigid unit (msgrp)
tqz	work	z component of torque on rigid unit (msgrp)
weight	input	atomic masses (mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)

gcmx	in/out	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	in/out	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	in/out	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	input	rigid unit mass (mxungp)
buffer	work	(mxbuff)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)
eta	output	cell scaling tensor (9)

8.1.105.5 Comments

See **NPT_H1** for general comments on the barostat and thermostat.

Fincham's implicit quaternion leapfrog algorithm is used to solve the rigid body equations of motion. This subroutine does not allow a rigid body to be involved in bond constraints with an atom or another rigid body. Should this be required use the routine **NPTQ_H2** instead.

8.1.106 NPTQ_H2

8.1.106.1 Header records

```

      subroutine nptq_h2
x  (safe,safeq,lshmov,imcon,idnode,mxnode,natms,ngroup,nscons,
x  ntcons,ntfree,ntpatm,chip,chip,consv,conint,elrc,engke,
x  engrot,virlrc,press,quattol,sigma,taup,taut,temp,tolnce,
x  timestep,vircom,vircon,virtot,volm,lstme,lishap,lashap,
x  lstfrz,listcon,listme,lstfre,lstcsit,lstbod,lstgtp,lstrgd,
x  listot,numgsit,cell,gcmx1,gcmx1,gcmz1,dens,dxx,dyy,dzz,
x  dxt,dyt,dzt,uxx,uyy,uzz,prmcon,omx,omy,omz,opx,opy,opz,
x  oqx,oqy,oqz,q0,q1,q2,q3,qn0,qn1,qn2,qn3,gxx,gyy,gzz,xxx,
x  yyy,zzz,vxx,vyy,vzz,gvxx,gvyy,gvzz,fxx,fyy,fzz,tqx,tqy,
x  tqz,weight,rotinx,rotiny,rotinz,gcmx,gcmx,gcmz,gmass,
x  buffer,xxt,yyt,zzt,txx,tyy,tzz,vx1,vy1,vz1,stress,gvx1,
x  gvy1,gvz1,omx1,omy1,omz1,redmass,eta)

C*****
C
C      dlpoly routine to integrate rigid body equations of motion
C      using implicit leapfrog quaternion algorithm
C      plus bond constraints - rigid body sites and constraint sites
C      may coincide.
C
C      dl_poly subroutine for integrating newtonian equations of
C      motion in molecular dynamics - verlet leapfrog with Hoover
C      thermostat and barostat.
C
C      parallel replicated data version : block data
C
C      tqx,tqy,tqz = torque in lab fixed frame (input)
C      omx,omy,omz = angular velocity in body fixed frame (principal axis)
C      rotinx,y,z = rotational inertia in body fixed frame
C
C      copyright daresbury laboratory 1995
C      author      t.forester june 1995
C
C*****

```

8.1.106.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion for systems containing rigid bodies. The rigid bodies may be linked by constraint bonds. The system is coupled to a Nosé-Hoover thermostat and Melchionna variant of the Hoover barostat.

Implicit quaternion equations are used for the rigid body equations of motion.

8.1.106.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- MERGE1
- MERGE4
- QSHAKE

8.1.106.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
safeq	output	flag for convergence of quaternion algorithm
lshmov	input	flag for shared bond constraints

integers:

imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ngrp	input	number of rigid body units in system
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system
ntfree	input	number of sites in system not in rigid body units
ntpatm	input	number of unique types of atoms

reals:

chip	in/out	length scaling variable
chit	in/out	velocity scaling variable
consv	in/out	scaling variables contribution to extended Hamiltonian
conint	in/out	time integral of scaling variables
elrc	in/out	long range correction to van der Waals energy
engke	output	kinetic energy
engrot	output	rotational kinetic energy
virlrc	in/out	long range correction to van der Waals virial
press	input	external pressure
quattol	input	tolerance for quaternion algorithm
sigma	input	required system kinetic energy
taup	input	barostat time constant, τ_P
taut	input	thermostat time constant, τ_T

temp	input	required system temperature
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircom	output	molecule to c.o.m. virial correction
vircon	output	constraint virial
virtot	input	virial from non-constraint terms
volm	in/out	volume of MD cell
integer arrays:		
lstme	input	indice list for MERGE1 (mxatms)
lishap	input	see QSHAKE (mxlshp)
lashap	input	see QSHAKE (mxproc)
lstfrz	input	key for frozen atoms (mxatms)
listcon	input	indices for constraint bonds (mxcons ,3)
listme	input	see QSHAKE (mxatms)
lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
lstcsit	input	indices of rigid unit sites in constraints (2*mxcons)
lstbod	input	pointer from site index to rigid unit index (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgadm)
listot	input	see QSHAKE (mxatms)
numgsit	input	number of sites in a rigid unit type (mxungp)
real arrays:		
cell	in/out	MD cell vectors (9)
gcmx1	work	(msgrp)
gcmy1	work	(msgrp)
gcmz1	work	(msgrp)
dens	in/out	particle number density (mxsvdw)
dxx	work	see QSHAKE (mxcons)
dyy	work	see QSHAKE (mxcons)
dzz	work	see QSHAKE (mxcons)
dxt	work	see QSHAKE (mxcons)
dyt	work	see QSHAKE (mxcons)
dzt	work	see QSHAKE (mxcons)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
prmcon	input	constraint distances (mxtcon)
omx	in/out	x component of angular velocity (mxgrp)
omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
opx	work	(msgrp)
opy	work	(msgrp)

opz	work	(msgrp)
oqx	work	(msgrp)
oqy	work	(msgrp)
oqz	work	(msgrp)
q0	in/out	first component of quaternion vector (mxgrp)
q1	in/out	second component of quaternion vector (mxgrp)
q2	in/out	third component of quaternion vector (mxgrp)
q3	in/out	fourth component of quaternion vector (mxgrp)
qn0	work	(msgrp)
qn1	work	(msgrp)
qn2	work	(msgrp)
qn3	work	(msgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
tqx	work	x component of torque on rigid unit (msgrp)
tqy	work	y component of torque on rigid unit (msgrp)
tqz	work	z component of torque on rigid unit (msgrp)
weight	input	atomic masses (mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)
gcmx	in/out	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	in/out	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	in/out	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	input	rigid unit mass (mxungp)
buffer	work	(mxbuff)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
txx	work	(mxatms)
tyy	work	(mxatms)

tzz	work	(mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)
gvx1	work	(msgrp)
gvy1	work	(msgrp)
gvz1	work	(msgrp)
omx1	work	(msgrp)
omy1	work	(msgrp)
omz1	work	(msgrp)
redmass	work	effective reduced mass along constraint bond (mxcons)
eta	output	cell scaling tensor (9)

8.1.106.5 Comments

See NPT_H1 for general comments on the Hoover type barostat and thermostat.

Fincham's implicit quaternion leapfrog algorithm is used to solve the rigid body equations of motion. This subroutine allows a rigid body to be involved in bond constraints with an atom or another rigid body. Should this not be required the routine NPTQ_H1 is more efficient.

8.1.107 NPTQ_H3

8.1.107.1 Header records

```

      subroutine nptq_h3
x  (safe,safeq,lshmov,imcon,idnode,mxnode,natms,ngroup,nscons,
x  ntcons,ntfree,ntpatm,chit,conint,consv,elrc,engke,engrot,
x  virirc,press,quattol,sigma,taup,taut,temp,tolnce,tstep,
x  vircom,vircon,volm,lstme,lishap,lashap,lstfrz,listcon,
x  listme,lstfre,lstgtp,lstrgd,listot,numgsit,cell,dens,
x  gcmx1,gcmx2,gcmx3,dxx,dyy,dzz,dxt,dyt,dzt,gvx1,gvx2,gvx3,
x  uxx,uyy,uzz,prmcon,xdf,ydf,zdf,omx,omy,omz,omx1,omy1,
x  omz1,opx,opy,opz,oqx,oqy,oqz,q0,q1,q2,q3,qn0,qn1,qn2,qn3,
x  gxx,gyy,gzz,xxx,yyy,zzz,vxx,vyy,vzz,gvxx,gvyy,gvzz,fx,
x  fyy,fzz,tqx,tqy,tqz,weight,rotinx,rotiny,rotinz,gcmx,
x  gcmx2,gcmx3,gmass,buffer,txt,tyt,ztz,txx,txy,txz,vy1,vy2,
x  vz1,stress,eta)

C*****
C
C      dlpoly routine to integrate rigid body equations of motion
C      using implicit leapfrog quaternion algorithm
C      plus bond constraints - PROVIDED rigid body sites
C      and constraint sites do not coincide.
C
C      dl_poly subroutine for integrating newtonian equations of
C      motion in molecular dynamics - verlet leapfrog with Hoover like
C      thermostat and barostat.
C
C      parallel replicated data version : block data
C
C      for systems using bond constraints
C
C      tqx,tqy,tqz = torque in lab fixed frame (input)
C      omx,omy,omz =angular velocity in body fixed frame (principal axis)
C      rotinx,y,z = rotational inertia in body fixed frame
C
C      copyright daresbury laboratory 1995
C      author      t.forester june 1995
C
C*****

```

8.1.107.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion for systems containing rigid bodies. The system is coupled to a Nosé-Hoover thermostat and the Melchionna variant of the Hoover anisotropic barostat. Implicit quaternion equations are used for the rigid body equations of motion.

8.1.107.3 Dependencies

- ERROR
- GDSUM
- IMAGES
- MERGE
- MERGE1
- RDSHAKE_1

8.1.107.4 Arguments

logicals:

<code>safe</code>	output	flag for convergence of SHAKE
<code>safeq</code>	output	flag for convergence of quaternion algorithm
<code>lshmov</code>	input	flag for shared bond constraints

integers:

<code>imcon</code>	input	key for periodic boundary conditions
<code>idnode</code>	input	identity of current processor
<code>mxnode</code>	input	total number of processors in use
<code>natms</code>	input	number of atoms in MD cell
<code>ngrp</code>	input	number of rigid body units in system
<code>nscons</code>	input	number of constraints on current processor
<code>ntcons</code>	input	number of constraints in system
<code>ntfree</code>	input	number of sites in system not in rigid body units
<code>ntpatm</code>	input	number of unique types of atoms

reals:

<code>chit</code>	in/out	velocity scaling variable
<code>conint</code>	in/out	time integral of scaling variables
<code>consv</code>	in/out	scaling variables contribution to extended Hamiltonian
<code>elrc</code>	in/out	long range correction to van der Waals energy
<code>engke</code>	output	kinetic energy
<code>engrot</code>	output	rotational kinetic energy
<code>virlrc</code>	in/out	long range correction to van der Waals virial
<code>press</code>	input	external pressure
<code>quattol</code>	input	tolerance for quaternion algorithm

<code>sigma</code>	input	required system kinetic energy
<code>taup</code>	input	barostat time constant, τ_P
<code>taut</code>	input	thermostat time constant, τ_T
<code>temp</code>	input	required system temperature
<code>tolnce</code>	input	tolerance for SHAKE algorithm
<code>tstep</code>	input	simulation time-step, Δt
<code>vircom</code>	output	molecule to c.o.m. virial correction
<code>vircon</code>	output	constraint virial
<code>volm</code>	in/out	volume of MD cell
integer arrays:		
<code>lstme</code>	input	indice list for MERGE1 (<code>mxatms</code>)
<code>lishap</code>	input	see RDSHAKE_1 (<code>mxlshp</code>)
<code>lashap</code>	input	see RDSHAKE_1 (<code>mxproc</code>)
<code>lstfrz</code>	input	key for frozen atoms (<code>mxatms</code>)
<code>listcon</code>	input	indices for constraint bonds (<code>mxcons</code> ,3)
<code>listme</code>	input	see RDSHAKE_1 (<code>mxatms</code>)
<code>lstfre</code>	input	indices of non-rigid unit sites handled by current processor (<code>mxatms</code>)
<code>lstgtp</code>	input	pointer from rigid unit index to rigid unit type (<code>mxgrp</code>)
<code>lstrgd</code>	input	indices of rigid unit sites handles by current processor (<code>mxgadm</code>)
<code>listot</code>	input	see RDSHAKE_1 (<code>mxatms</code>)
<code>numgsit</code>	input	number of sites in a rigid unit type (<code>mxungp</code>)
real arrays:		
<code>cell</code>	in/out	MD cell vectors (9)
<code>dens</code>	in/out	particle number density (<code>mxsvdw</code>)
<code>gcmx1</code>	work	(<code>msgrp</code>)
<code>gcmy1</code>	work	(<code>msgrp</code>)
<code>gcmz1</code>	work	(<code>msgrp</code>)
<code>dx</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dy</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dzz</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dxt</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dzt</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>gvx1</code>	work	(<code>msgrp</code>)
<code>gvy1</code>	work	(<code>msgrp</code>)
<code>gvz1</code>	work	(<code>msgrp</code>)
<code>uxx</code>	work	(<code>mxatms</code>)
<code>uyy</code>	work	(<code>mxatms</code>)
<code>uzz</code>	work	(<code>mxatms</code>)
<code>prmcon</code>	input	constraint distances (<code>mxtcon</code>)
<code>xdf</code>	work	(<code>mxxdf</code>)
<code>ydf</code>	work	(<code>mxxdf</code>)

zdf	work	(mxxdf)
omx	in/out	x component of angular velocity (mxgrp)
omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
omx1	work	(msgrp)
omy1	work	(msgrp)
omz1	work	(msgrp)
opx	work	(msgrp)
opy	work	(msgrp)
opz	work	(msgrp)
oqx	work	(msgrp)
oqy	work	(msgrp)
oqz	work	(msgrp)
q0	in/out	first component of quaternion vector (mxgrp)
q1	in/out	second component of quaternion vector (mxgrp)
q2	in/out	third component of quaternion vector (mxgrp)
q3	in/out	fourth component of quaternion vector (mxgrp)
qn0	work	(msgrp)
qn1	work	(msgrp)
qn2	work	(msgrp)
qn3	work	(msgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
txx	work	x component of torque on rigid unit (msgrp)
txy	work	y component of torque on rigid unit (msgrp)
txz	work	z component of torque on rigid unit (msgrp)
weight	input	atomic masses (mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)
gcmx	in/out	x coordinate of rigid unit c.o.m. (mxgrp)

gcmx	in/out	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	in/out	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	input	rigid unit mass (mxungp)
buffer	work	(mxbuff)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)
eta	in/out	cell scaling tensor (9)

8.1.107.5 Comments

See NPT_H3 for general comments on the thermostat and anisotropic barostat and on compilation.

Fincham's implicit quaternion leapfrog algorithm is used to solve the rigid body equations of motion. This subroutine does not allow a rigid body to be involved in bond constraints with an atom or another rigid body. Should this be required use the routine NPTQ_H4 instead.

8.1.108 NPTQ_H4

8.1.108.1 Header records

```

      subroutine nptq_h4
      x  (safe,safeq,lshmov,imcon,idnode,mxnode,natms,ngroup,nscons,
      x  ntcons,ntfree,ntpatm,chit,conint,consv,elrc,engke,engrot,
      x  virirc,press,quattol,sigma,taup,taut,temp,tolnce,tstep,
      x  vircom,vircon,volm,lstme,lishap,lashap,lstfrz,listcon,
      x  listme,lstfre,lstgtp,lstrgd,listot,lstcsit,lstbod,numgsit,
      x  cell,dens,gcmx1,gcmx2,gcmx3,dxx,dyy,dzz,dxt,dyt,dzt,gvx1,
      x  gvy1,gvz1,uxx,uyy,uzz,prmcon,omx,omy,omz,omx1,
      x  omy1,omz1,opx,opy,opz,oqx,oqy,oqz,q0,q1,q2,q3,qn0,qn1,qn2,
      x  qn3,gxx,gyy,gzz,xxx,yyy,zzz,vxx,vyy,vzz,gvxx,gvyy,gvzz,
      x  fxx,fyy,fzz,tqx,tqy,tqz,weight,rotinx,rotiny,rotinz,gcmx,
      x  gcmx2,gcmx3,buffer,xxt,yyt,zzt,txx,tyy,tzz,vx1,vy1,
      x  vz1,stress,eta,redmass)

C*****
C
C      dlpoly routine to integrate rigid body equations of motion
C      using implicit leapfrog quaternion algorithm
C      plus bond constraints. Rigid body sites and constraint sites may
C      coincide.
C
C      dl_poly subroutine for integrating newtonian equations of
C      motion in molecular dynamics - verlet leapfrog with Hoover like
C      thermostat and barostat.
C
C      parallel replicated data version : block data
C
C      for systems using bond constraints
C
C      tqx,tqy,tqz = torque in lab fixed frame (input)
C      omx,omy,omz =angular velocity in body fixed frame (principal axis)
C      rotinx,y,z = rotational inertia in body fixed frame
C
C      copyright daresbury laboratory 1995
C      author      t.forester june 1995
C
C*****

```

8.1.108.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion for systems containing rigid bodies. The rigid bodies may be linked by constraint bonds. The system is coupled to a Nosé-Hoover thermostat and Melchionna variant of the Hoover anisotropic barostat. Implicit quaternion equations are used for the rigid body equations of motion.

8.1.108.3 Dependencies

- ERROR
- GDSUM
- IMAGES
- MERGE
- MERGE1
- MERGE4
- QSHAKE

8.1.108.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
safeq	output	flag for convergence of quaternion algorithm
lshmov	input	flag for shared bond constraints

integers:

imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ngrp	input	number of rigid body units in system
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system
ntfree	input	number of sites in system not in rigid body units
ntpatm	input	number of unique types of atoms

reals:

chit	in/out	velocity scaling variable
conint	in/out	time integral of scaling variables
consv	in/out	scaling variables contribution to extended Hamiltonian
elrc	in/out	long range correction to van der Waals energy
engke	output	kinetic energy
engrot	output	rotational kinetic energy
virirc	in/out	long range correction to van der Waals virial

press	input	external pressure
quattol	input	tolerance for quaternion algorithm
sigma	input	required system kinetic energy
taup	input	barostat time constant, τ_P
taut	input	thermostat time constant, τ_T
temp	input	required system temperature
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircom	output	molecule to c.o.m. virial correction
vircon	output	constraint virial
volm	in/out	volume of MD cell
integer arrays:		
lstme	input	indice list for MERGE1 (mxatms)
lishap	input	see QSHAKE (mxlshp)
lashap	input	see QSHAKE (mxproc)
lstfrz	input	key for frozen atoms (mxatms)
listcon	input	indices for constraint bonds (mxcons ,3)
listme	input	see QSHAKE (mxatms)
lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgatk)
listot	input	see QSHAKE (mxatms)
lstcsit	input	indices of rigid unit sites in constraints (2*mxcons)
lstbod	input	pointer from site index to rigid unit index (mxatms)
numgsit	input	number of sites in a rigid unit type (mxungp)
real arrays:		
cell	in/out	MD cell vectors (9)
dens	in/out	particle number density (mxsvdw)
gcmx1	work	(msgrp)
gcmy1	work	(msgrp)
gcmz1	work	(msgrp)
dxx	work	see QSHAKE (mxcons)
dyy	work	see QSHAKE (mxcons)
dzz	work	see QSHAKE (mxcons)
dxt	work	see QSHAKE (mxcons)
dxt	work	see QSHAKE (mxcons)
dzt	work	see QSHAKE (mxcons)
gvx1	work	(msgrp)
gvy1	work	(msgrp)
gvz1	work	(msgrp)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
prmcon	input	constraint distances (mxtcon)

omx	in/out	x component of angular velocity (mxgrp)
omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
omx1	work	(msgrp)
omy1	work	(msgrp)
omz1	work	(msgrp)
opx	work	(msgrp)
opy	work	(msgrp)
opz	work	(msgrp)
oqx	work	(msgrp)
oqy	work	(msgrp)
oqz	work	(msgrp)
q0	in/out	first component of quaternion vector (mxgrp)
q1	in/out	second component of quaternion vector (mxgrp)
q2	in/out	third component of quaternion vector (mxgrp)
q3	in/out	fourth component of quaternion vector (mxgrp)
qn0	work	(msgrp)
qn1	work	(msgrp)
qn2	work	(msgrp)
qn3	work	(msgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
txx	work	x component of torque on rigid unit (msgrp)
txy	work	y component of torque on rigid unit (msgrp)
txz	work	z component of torque on rigid unit (msgrp)
weight	input	atomic masses (mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)
gcmx	in/out	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	in/out	y coordinate of rigid unit c.o.m. (mxgrp)

<code>gcmz</code>	in/out	z coordinate of rigid unit c.o.m. (<code>mxgrp</code>)
<code>gmass</code>	input	rigid unit mass (<code>mxungp</code>)
<code>buffer</code>	work	(<code>mxbuff</code>)
<code>xxt</code>	work	(<code>mxatms</code>)
<code>yyt</code>	work	(<code>mxatms</code>)
<code>zzt</code>	work	(<code>mxatms</code>)
<code>txx</code>	work	(<code>mxatms</code>)
<code>tyy</code>	work	(<code>mxatms</code>)
<code>tzz</code>	work	(<code>mxatms</code>)
<code>vx1</code>	work	(<code>msatms</code>)
<code>vy1</code>	work	(<code>msatms</code>)
<code>vz1</code>	work	(<code>msatms</code>)
<code>stress</code>	in/out	stress tensor (9)
<code>eta</code>	in/out	cell scaling tensor (9)
<code>redmass</code>	work	effective reduced mass along constraint bond (<code>mxcons</code>)

8.1.108.5 Comments

See `NPT_H3` for general comments on the thermostat and anisotropic barostat.

Fincham's implicit quaternion leapfrog algorithm is used to solve the rigid body equations of motion. This subroutine allows a rigid body to be involved in bond constraints with an atom or another rigid body. Should this not be required the routine `NPTQ_H3` is more efficient.

8.1.109 NST_B0**8.1.109.1 Header records**

```

      subroutine nst_b0
      x  (idnode,imcon,mxnode,natms,ntpatm,elrc,engke,virlrc,press,
      x  taup,taut,sigma,tstep,volm,lstfrz,buffer,cell,dens,fx,ffy,
      x  fzz,vxx,vyy,vzz,weight,xd,xxx,yd,yyy,zd,zzz,vx1,vy1,vz1,
      x  stress,eta,dens0)

c*****
c
c      dl_poly subroutine for integrating newtonian equations of
c      motion in molecular dynamics - verlet leapfrog with Berendsen
c      thermostat and anisotropic pressure control
c      isothermal compressibility (beta) set to that of liquid water
c      = 0.007372 dlpoly units
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1993/1999
c      author    -    t. forester december 1993
c      adapted   - w. smith for atomic systems nov 1999
c
c*****

```

8.1.109.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion subject to a Berendsen thermostat and anisotropic barostat. This version is for systems without constraint bonds or rigid bodies.

8.1.109.3 Dependencies

- ERROR
- GDSUM
- IMAGES
- MERGE

8.1.109.4 Arguments**integers:**

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions

mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ntpatm	input	number of unique types of atoms
reals:		
elrc	in/out	long range correction to van der Waals energy
engke	output	kinetic energy
virlrc	in/out	long range correction to van der Waals virial
press	input	external pressure, P_{ext}
taup	input	barostat time constant, τ_P
taut	input	thermostat time constant, τ_T
sigma	input	required system kinetic energy
v tstep	input	simulation time-step, Δt
volm	in/out	volume of MD cell
integer arrays:		
lstfrz	input	key for frozen atoms (mxatms)
real arrays:		
buffer	work	see MERGE (mxbuff)
cell	in/out	MD cell vectors (9)
dens	in/out	particle number density (mxsvdw)
fx	in/out	x component of force (mxatms)
fy	in/out	y component of force (mxatms)
fz	in/out	z component of force (mxatms)
vx	in/out	x component of velocity (mxatms)
vy	in/out	y component of velocity (mxatms)
vz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xdf	work	x component of intersite vector (mxxdf)
xxx	in/out	x coordinates (mxatms)
ydf	work	y component of intersite vector (mxxdf)
yyy	in/out	y coordinates (mxatms)
zdf	work	z component of intersite vector (mxxdf)
zzz	in/out	z coordinates (mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)
eta	output	cell scaling tensor (9)
dens0	work	(mxsvdw)

8.1.109.5 Comments

Uses a leapfrog integration scheme (see NVE_1 for general comments).

The pressure is adjusted by anisotropic changes to the MD cell vectors. The changes to

the cell tensor $\hat{H}$ are given by

$$\Delta\hat{H} = \hat{\eta} \cdot \hat{H} \quad (8.17)$$

where the components of $\hat{\eta}$ are given by

$$\eta_{\mu\nu} = \frac{\beta\Delta t}{\tau_P} \left(\frac{\sigma_{\mu\nu}}{V} - \delta_{\mu\nu} P_{\text{ext}} \right) \quad (8.18)$$

and V is the cell volume and $\delta_{\mu\nu}$ is the Kronecker delta. β is the isothermal compressibility (taken to be that of liquid water) and $\hat{\sigma}$ the instantaneous stress tensor in the system. Choice of τ_P depends on the actual β of the system. In water based systems a value of τ_P in the range (0.5, 5) ps is normally appropriate.

Temperature is controlled by the Berendsen thermostat. The velocities are scaled by χ where:

$$\chi = \left(1 + \frac{\Delta t}{\tau_T} \left(\frac{T}{\mathcal{T}} - 1 \right) \right)^{1/2} \quad (8.19)$$

where τ_T is the thermostat relaxation time, $\mathcal{T}$ the instantaneous temperature and T the specified system temperature. Usually a value of τ_T in the range (0.1, 1) ps works well.

The Berendsen thermostat and barostat do not conserve energy.

The barostat and thermostat require several iterations in order to accurately predict the current temperature and pressure. This is controlled by the variable `maxit` which is set to 4 in this version. It may be changed by editing and recompiling the subroutine.

8.1.110 NST_H0**8.1.110.1 Header records**

```

      subroutine nst_h0
      x  (idnode,imcon,mxnode,natms,ntpatm,chit,conint,consv,
      x  elrc,engke,virlrc,press,taup,taut,sigma,temp,tstep,volm,
      x  lstfrz,buffer,cell,dens,fx,fy,fz,vx,vy,vz,weight,
      x  xdf,xxx,ydf,yyy,zdf,zzz,vx1,vy1,vz1,stress,eta,dens0)

C*****
C
C      dl_poly subroutine for integrating newtonian equations of
C      motion in molecular dynamics - verlet leapfrog with Hoover
C      thermostat + piston.
C
C      Parrinello- Rahman type : changing cell shape.
C
C      reference: Melchionna, Ciccotti and Holian,
C      Mol Phys 1993, 78, p533
C
C      parallel replicated data version
C
C      copyright daresbury laboratory 1995/1999
C      author      - t. forester      june 1995
C      adapted     - w. smith atomic systems nov 1999
C
C*****

```

8.1.110.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion. The system is coupled to a Nosé-Hoover thermostat and the Melchionna variant of the anisotropic Hoover barostat. This version is for systems without constraint bonds or rigid bodies.

8.1.110.3 Dependencies

- ERROR
- GDSUM
- IMAGES
- MERGE

8.1.110.4 Arguments**integers:**

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ntpatm	input	number of unique types of atoms

reals:

chit	in/out	velocity scaling variable
conint	in/out	time integral of scaling variables
consv	in/out	scaling variables contribution to extended Hamiltonian
elrc	in/out	long range correction to van der Waals energy
engke	output	kinetic energy
virlrc	in/out	long range correction to van der Waals virial
press	input	external pressure, P_{ext}
taup	input	barostat time constant, τ_P
taut	input	thermostat time constant, τ_T
sigma	input	required system kinetic energy
temp	input	required system temperature
tstep	input	simulation time-step, Δt
volm	in/out	volume of MD cell

integer arrays:

lstfrz	input	key for frozen atoms (mxatms)

real arrays:

buffer	work	see MERGE (mxbuff)
cell	in/out	MD cell vectors (9)
dens	in/out	particle number density (mxsvdw)
fx	in/out	x component of force (mxatms)
fy	in/out	y component of force (mxatms)
fz	in/out	z component of force (mxatms)
vx	in/out	x component of velocity (mxatms)
vy	in/out	y component of velocity (mxatms)
vz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xdf	work	(mxxdf)
xxx	in/out	x coordinates (mxatms)
ydf	work	(mxxdf)
yyy	in/out	y coordinates (mxatms)
zdf	work	(mxxdf)
zzz	in/out	z coordinates (mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)

stress	in/out	stress tensor, $\hat{\sigma}$ (9)
eta	in/out	cell scaling tensor, $\hat{\eta}$ (9)
dens0	work	(mxsvdw)

8.1.110.5 Comments

Uses a leapfrog integration scheme (see NVE_1 for general comments). This subroutine must be compiled with the STRESS option turned on. The pressure is adjusted by anisotropic changes to the MD cell vectors. The changes to the cell tensor $\hat{H}$ are given by

$$\Delta \hat{H} = \exp(\Delta t \cdot \hat{\eta}) \quad (8.20)$$

where the time derivative of the tensor $\hat{\eta}$ is given by

$$\frac{d\hat{\eta}}{dt} = \frac{(\hat{\sigma} - \delta P_{\text{ext}})V}{Nk_B T \tau_P^2} \quad (8.21)$$

where $\hat{\sigma}$ is the instantaneous atomic stress tensor, δ the Kronecker delta, V is the system volume, N the number of atoms in the system, and T the required system temperature. Choice of τ_P is system dependent but normally a value in the range (1, 10) ps is appropriate.

Temperature is controlled by the Nosé-Hoover thermostat. The value of the friction coefficient, χ for the thermostat is found from the first-order differential equation:

$$\frac{d\chi}{dt} = \frac{1}{\tau_T^2} \left(\frac{\mathcal{T}}{T} - 1 \right) \quad (8.22)$$

where $\mathcal{T}$ is the instantaneous temperature. Usually a value of τ_T in the range (0.5, 2) ps works well but this may be system dependent.

The extended Hamiltonian of the system is a conserved quantity and this may be used as a check on the choice of τ_T and τ_P .

The barostat and thermostat require several iterations in order to accurately predict the current temperature and pressure. This is controlled by the variable **maxit** which is set to 4 in this version. It may be changed by editing and recompiling the subroutine.

8.1.111 NUMNODES**8.1.111.1 Header records**

```
integer function numnodes()
```

```
c*****
c
c      calculate number of nodes
c
c      MPI version - t.forester may 1995
c
c*****
```

8.1.111.2 Function

Calculates number of processors in use in serial and MPI environments.

8.1.111.3 Dependencies

- MPI_COMM_SIZE

8.1.111.4 Arguments

None.

8.1.111.5 Comments

This is a mimic of the INTEL routine of the same name. C preprocessing is used at compile time to select the appropriate parallel environment. Use the Intel routine if running on Intel hardware.

8.1.112 NVE_0**8.1.112.1 Header records**

```

      subroutine nve_0
      x      (idnode,mxnode,natms,imcon,tstep,engke,weight,cell,
      x      xxx,yyy,zzz,vxx,vyy,vzz,fxx,fyy,fzz,buffer,stress)

c
c*****
c
c      dl_poly subroutine for integrating newtonian equations of
c      motion in molecular dynamics - verlet leapfrog.
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1992
c      author      - w. smith march 1992.
c
c*****
c

```

8.1.112.2 Function

The verlet algorithm for integrating Newton's equations of motion when there are *no bond constraints* specified in the system.

8.1.112.3 Dependencies:

- IMAGES
- MERGE
- GDSUM

8.1.112.4 Arguments:**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	total number of sites in the simulated system.
imcon	input	periodic boundary condition key. See IMAGES

reals:

tstep	input	simulation time step, Δt .
engke	output	kinetic energy

real arrays:

weight	input	atomic mass array (mxatms).
cell	input	cell vectors (9). See IMAGES.
xxx	in/out	x coordinate array (mxatms).
yyy	in/out	y coordinate array (mxatms)
zzz	in/out	z coordinate array (mxatms)
vxx	in/out	x velocity array (mxatms)
vyx	in/out	y velocity array (mxatms)
vzz	in/out	z velocity array (mxatms)
fx	input	x component of force array (mxatms)
fy	input	y component of force array (mxatms)
fz	input	z component of force array (mxatms)
buffer	work	buffer array (mxbuff). See MERGE
stress	in/out	stress tensor (9)

8.1.112.5 Comments

Generates trajectories in the microcanonical ensemble (NVE). The total energy (kinetic plus potential) of the system should be conserved.

On calling this subroutine **xxx**, **yyy**, **zzz** hold values corresponding to t ; **vxx**, **vyx**, **vzz** hold values corresponding to $t - \frac{1}{2}\Delta t$; **fx**, **fy** and **fz** hold values corresponding to t

On leaving the routine **xxx**, **yyy**, **zzz** hold values corresponding to $t + \Delta t$, and **vxx**, **vyx** and **vzz** hold values corresponding to $t + \frac{1}{2}\Delta t$.

8.1.113 NVE_1**8.1.113.1 Header records**

```

      subroutine nve_1
      x  (safe,lshmov,idnode,imcon,mxnode,natms,nscons,ntcons,
      x  engke,tolnce,tstep,vircon,lashap,lishap,listcon,
      x  listme,listot,lstfrz,buffer,cell,dxt,dxx,dyt,
      x  dyy,dzt,dzz,fx,fyy,fzz,prmcon,txx,tyy,tzz,uxx,
      x  uyy,uzz,vxx,vyy,vzz,weight,xdf,xxt,xxx,ydf,yyt,
      x  yyy,zdf,zzt,zzz,stress)
c
c*****
c
c      dl_poly subroutine for integrating newtonian equations of
c      motion in molecular dynamics. Verlet leapfrog With RD-SHAKE
c
c      parallel replicated data version : block data
c
c      copyright - daresbury laboratory 1992
c      author      - w. smith august 1992.
c      amended     - t.forester sept 1994
c      amended     - t.forester dec 1994 : block data
c
c*****

```

8.1.113.2 Function

The verlet leapfrog algorithm for integrating Newton's equations of motion.

8.1.113.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- RDSHAKE_1

8.1.113.4 Arguments**logicals:**

safe	output	flag for convergence of SHAKE
lshmov	input	flag for shared bond constraints

integers:

idnode	input	identity of current processor

<code>imcon</code>	input	key for periodic boundary conditions
<code>mxnode</code>	input	total number of processors in use
<code>natms</code>	input	number of atoms in MD cell
<code>nscons</code>	input	number of constraints on current processor
<code>ntcons</code>	input	number of constraints in system
reals:		
<code>engke</code>	output	kinetic energy
<code>tolnce</code>	input	tolerance for SHAKE algorithm
<code>tstep</code>	input	simulation time-step, Δt
<code>vircon</code>	output	constraint virial
integer arrays:		
<code>lashap</code>	input	see RDSHAKE_1 (<code>mxproc</code>)
<code>lishap</code>	input	see RDSHAKE_1 (<code>mxlshp</code>)
<code>listcon</code>	input	indices for constraint bonds (<code>mxcons</code> ,3)
<code>listme</code>	input	see RDSHAKE_1 (<code>mxatms</code>)
<code>listot</code>	input	see RDSHAKE_1 (<code>mxatms</code>)
<code>lstfrz</code>	input	key for frozen atoms (<code>mxatms</code>)
real arrays:		
<code>buffer</code>	work	see MERGE (<code>mxbuff</code>)
<code>cell</code>	input	MD cell vectors (9)
<code>dxt</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dxs</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dys</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dyy</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dzt</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dzz</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>fxs</code>	in/out	x component of force (<code>mxatms</code>)
<code>fys</code>	in/out	y component of force (<code>mxatms</code>)
<code>fzs</code>	in/out	z component of force (<code>mxatms</code>)
<code>prmcon</code>	input	constraint distances (<code>mxtcon</code>)
<code>txs</code>	work	(<code>mxatms</code>)
<code>tyy</code>	work	(<code>mxatms</code>)
<code>tzz</code>	work	(<code>mxatms</code>)
<code>uxs</code>	work	(<code>mxatms</code>)
<code>uys</code>	work	(<code>mxatms</code>)
<code>uzz</code>	work	(<code>mxatms</code>)
<code>vxs</code>	in/out	x component of velocity (<code>mxatms</code>)
<code>vys</code>	in/out	y component of velocity (<code>mxatms</code>)
<code>vzs</code>	in/out	z component of velocity (<code>mxatms</code>)
<code>weight</code>	input	atomic masses (<code>mxatms</code>)
<code>xdf</code>	work	x component of intersite vector (<code>mxxdf</code>)
<code>xxt</code>	work	(<code>mxatms</code>)
<code>xxx</code>	in/out	x coordinates (<code>mxatms</code>)
<code>ydf</code>	work	y component of intersite vector (<code>mxxdf</code>)

yyt	work	(mxatms)
yyy	in/out	y coordinates (mxatms)
zdf	work	z component of intersite vector (mxxdf)
zzt	work	(mxatms)
zzz	in/out	z coordinates (mxatms)
stress	in/out	stress tensor (9)

8.1.113.5 Comments

Generates trajectories in the microcanonical ensemble (NVE). The total energy (kinetic plus potential) of the system should be conserved.

The subroutine employs the leapfrog algorithm: On entry **xxx**, **yyy**, **zzz** and **fx**, **fyy**, **fzz** are at time t , while **vxx**, **vy**, **vzz** are at time $t - \frac{1}{2}\Delta t$. On exit the positions and velocities have been advanced to time $t + \Delta t$ and $t + \frac{1}{2}\Delta t$ respectively.

This subroutine also includes a call to **RDSHAKE_1** to handle bond constraints by the iterative SHAKE method.

8.1.114 NVT_B0**8.1.114.1 Header records**

```

      subroutine nvt_b0
      x  (idnode,imcon,natms,mxnode,engke,sigma,tstep,taut,
      x  buffer,cell,fx,fy,fz,vx,vy,vz,weight,xxx,yyy,zzz,stress)

c
c*****
c
c      dl_poly subroutine for integrating newtonian equations of
c      motion in molecular dynamics - verlet leapfrog with Berendsen
c      thermostat.
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1993
c      author    - t. forester      july 1993
c
c*****
c

```

8.1.114.2 Function

The verlet leapfrog algorithm for integrating Newton's equations of motion subject to the Berendsen thermostat.

8.1.114.3 Dependencies:

- IMAGES
- MERGE
- GDSUM

8.1.114.4 Arguments:**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors in use
imcon	input	periodic boundary condition key. See IMAGES
natms	input	total number of sites in the simulated system.

reals:

engke	output	kinetic energy
sigma	input	required kinetic energy

tstep	input	simulation time step.
taut	input	thermostat decay constants
real arrays:		
buffer	work	work array (mxbuff)
cell	input	cell vectors (9). See IMAGES.
fx	input	x component of force (mxatms)
fy	input	y component of force (mxatms)
fz	input	z component of force (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
stress	in/out	stress tensor (9)

8.1.114.5 Comments

The Berendsen thermostat pushes the system temperature towards the required value by the scaling of atomic velocities. Energy is not conserved and the generated trajectories do not correspond to either the microcanonical (NVE) or the canonical (NVT) ensemble. The canonical ensemble is generated by the Nosé-Hoover thermostat (see NVT_H0).

For general comments of the Verlet leapfrog integrator see NVE_0.

8.1.115 NVEQ_1**8.1.115.1 Header records**

```

      subroutine nveq_1
x  (safe,safeq,lshmov,imcon,idnode,mxnode,natms,ngroup,nscons,
x  ntcons,ntfree,engke,engrot,quattol,tolnce,tstep,vircom,
x  vircon,lstme,lishap,lashap,lstfrz,listcon,listme,lstfre,
x  lstgtp,lstrgd,listot,numgsit,cell,gcmx1,gcmx1,gcmz1,dxx,
x  dyy,dzz,dxt,dyt,dzt,uxx,uyy,uzz,prmcon,omx,omy,omz,opx,
x  opy,opz,oqx,oqy,oqz,q0,q1,q2,q3,qn0,qn1,qn2,qn3,gxx,gyy,
x  gzz,xxx,yyy,zzz,vxx,vyy,vzz,gvxx,gvyy,gvzz,fxx,fyy,fzz,
x  tqx,tqy,tqz,weight,rotinx,rotiny,rotinz,gcmx,gcmx,gcmz,
x  gmass,buffer,xxt,yyt,zzt,txx,tyy,tzz,vx1,vy1,vz1,stress)

c*****
c
c      dlpoly routine to integrate rigid body equations of motion
c      using implicit leapfrog quaternion algorithm
c      plus bond constraints - PROVIDED rigid body sites
c      and constraint sites do not coincide.
c
c      parallel replicated data version : block data
c
c      tqx,tqy,tqz = torque in lab fixed frame (input)
c      omx,omy,omz = angular velocity in body fixed frame (principal axis)
c      rotinx,y,z = rotational inertia in body fixed frame
c
c      copyright daresbury laboratory 1993
c      author      t.forester october 1993
c      amended     t.forester dec 1994 : block data
c
c*****

```

8.1.115.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion for systems containing rigid bodies. Implicit quaternion equations are used.

8.1.115.3 Dependencies

- GDSUM
- IMAGES
- MERGE

- MERGE1
- RDSHAKE_1

8.1.115.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
safeq	output	flag for convergence of quaternion algorithm
lshmov	input	flag for shared bond constraints

integers:

imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ngrp	input	number of rigid body units in system
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system
ntfree	input	number of sites in system not in rigid body units

reals:

engke	output	kinetic energy
engrot	output	rotational kinetic energy
quattol	input	tolerance for quaternion algorithm
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircom	output	molecule to c.o.m. virial correction
vircon	output	constraint virial

integer arrays:

lstme	input	indice list for MERGE1 (mxatms)
lishap	input	see RDSHAKE_1 (mxlshp)
lashap	input	see RDSHAKE_1 (mxproc)
lstfrz	input	key for frozen atoms (mxatms)
listcon	input	indices for constraint bonds (mxcons , 3)
listme	input	see RDSHAKE_1 (mxatms)
lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgadm)
listot	input	see RDSHAKE_1 (mxatms)
numgsit	input	number of sites in a rigid unit type (mxungp)

real arrays:

cell	input	MD cell vectors (9)
gcmx1	work	(msggrp)

gcmy1	work	(msgrp)
gcmz1	work	(msgrp)
dxs	work	see RDSHAKE_1 (mxcons)
dyy	work	see RDSHAKE_1 (mxcons)
dzz	work	see RDSHAKE_1 (mxcons)
dxt	work	see RDSHAKE_1 (mxcons)
dzt	work	see RDSHAKE_1 (mxcons)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
prmcon	input	constraint distances (mxtcon)
omx	in/out	x component of angular velocity (mxgrp)
omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
opx	work	(msgrp)
opy	work	(msgrp)
opz	work	(msgrp)
oqx	work	(msgrp)
oqy	work	(msgrp)
oqz	work	(msgrp)
q0	in/out	first component of quaternion vector (mxgrp)
q1	in/out	second component of quaternion vector (mxgrp)
q2	in/out	third component of quaternion vector (mxgrp)
q3	in/out	fourth component of quaternion vector (mxgrp)
qn0	work	(msgrp)
qn1	work	(msgrp)
qn2	work	(msgrp)
qn3	work	(msgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
fxs	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)

tqx	work	x component of torque on rigid unit (msgrp)
tqy	work	y component of torque on rigid unit (msgrp)
tqz	work	z component of torque on rigid unit (msgrp)
weight	input	atomic masses (mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)
gcmx	in/out	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	in/out	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	in/out	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	input	rigid unit mass (mxungp)
buffer	work	(mxbuff)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)

8.1.115.5 Comments

Generates trajectories in the microcanonical ensemble (NVE). The total energy (kinetic plus potential) of the system should be conserved. See NVE_1 for general comments.

Fincham's implicit quaternion leapfrog algorithm is used to solve the rigid body equations of motion. This subroutine does not allow a rigid body to be involved in bond constraints with an atom or another rigid body. Should this be required use the routine NVEQ_2 instead.

8.1.116 NVEQ_2**8.1.116.1 Header records**

```

      subroutine nveq_2
x  (safe,safeq,lshmov,imcon,idnode,mxnode,natms,ngroup,nscons,
x  ntcons,ntfree,engke,engrot,quattol,tolnce,tstep,vircom,
x  vircon,lstme,lishap,lashap,lstfrz,listcon,listme,lstfre,
x  lstcsit,lstbod,lstgtp,lstrgd,listot,numgsit,cell,gcmx1,
x  gcmy1,gcmz1,dxx,dyy,dzz,dxt,dyt,dzt,uxx,uyy,uzz,prmcon,
x  omx,omy,omz,opx,opy,opz,oqx,oqy,oqz,q0,q1,q2,q3,qn0,qn1,
x  qn2,qn3,gxx,gyy,gzz,xxx,yyy,zzz,vxx,vyy,vzz,gvxx,gvyy,
x  gvzz,fxx,fyy,fzz,tqx,tqy,tqz,weight,rotinx,rotiny,rotinz,
x  gcmx,gcmy,gcmz,gmass,buffer,txt,yyt,zzt,txx,tyy,tzz,vx1,
x  vy1,vz1,stress,gvx1,gvx1,gvz1,omx1,omy1,omz1,redmass)

C*****
C
C      dlpoly routine to integrate rigid body equations of motion
C      using implicit leapfrog quaternion algorithm
C      plus bond constraints - rigid body sites and constraint sites
C      may coincide.
C
C      parallel replicated data version : block data
C
C      tqx,tqy,tqz = torque in lab fixed frame (input)
C      omx,omy,omz = angular velocity in body fixed frame (principal axis)
C      rotinx,y,z = rotational inertia in body fixed frame
C
C      copyright daresbury laboratory 1995
C      author      t.forester june 1995
C
C*****

```

8.1.116.2 Function

The verlet leapfrog algorithm for integrating Newton's equations of motion for systems containing rigid bodies. Implicit quaternion equations are used.

8.1.116.3 Dependencies

- GDSUM
- IMAGES
- MERGE

- MERGE1
- MERGE4
- QSHAKE

8.1.116.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
safeq	output	flag for convergence of quaternion algorithm
lshmov	input	flag for shared bond constraints

integers:

imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ngrp	input	number of rigid body units in system
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system
ntfree	input	number of sites in system not in rigid body units

reals:

engke	output	kinetic energy
engrot	output	rotational kinetic energy
quattol	input	tolerance for quaternion algorithm
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircom	output	molecule to c.o.m. virial correction
vircon	output	constraint virial

integer arrays:

lstme	input	indice list for MERGE1 (mxatms)
lishap	input	see QSHAKE (mxlshp)
lashap	input	see QSHAKE (mxproc)
lstfrz	input	key for frozen atoms (mxatms)
listcon	input	indices for constraint bonds (mxcons ,3)
listme	input	see QSHAKE (mxatms)
lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
lstcsit	input	indices of rigid unit sites in constraints (2*mxcons)
lstbod	input	pointer from site index to rigid unit index (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgadm)
listot	input	see QSHAKE (mxatms)
numgsit	input	number of sites in a rigid unit type (mxungp)

real arrays:

cell	input	MD cell vectors (9)
gcmx1	work	(msgrp)
gcmy1	work	(msgrp)
gcmz1	work	(msgrp)
dxx	work	see QSHAKE (mxcons)
dyy	work	see QSHAKE (mxcons)
dzz	work	see QSHAKE (mxcons)
dxt	work	see QSHAKE (mxcons)
dyt	work	see QSHAKE (mxcons)
dzt	work	see QSHAKE (mxcons)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
prmcon	input	constraint distances (mxtcon)
omx	in/out	x component of angular velocity (mxgrp)
omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
opx	work	(msgrp)
opy	work	(msgrp)
opz	work	(msgrp)
oqx	work	(msgrp)
oqy	work	(msgrp)
oqz	work	(msgrp)
q0	in/out	first component of quaternion vector (mxgrp)
q1	in/out	second component of quaternion vector (mxgrp)
q2	in/out	third component of quaternion vector (mxgrp)
q3	in/out	fourth component of quaternion vector (mxgrp)
qn0	work	(msgrp)
qn1	work	(msgrp)
qn2	work	(msgrp)
qn3	work	(msgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)

fx	in/out	x component of force (mxatms)
fy	in/out	y component of force (mxatms)
fz	in/out	z component of force (mxatms)
tx	work	x component of torque on rigid unit (msgrp)
ty	work	y component of torque on rigid unit (msgrp)
tz	work	z component of torque on rigid unit (msgrp)
weight	input	atomic masses (mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)
gcmx	in/out	x coordinate of rigid unit c.o.m. (mxgrp)
gcmx	in/out	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	in/out	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	input	rigid unit mass (mxungp)
buffer	work	(mxbuff)
xxt	work	(mxatms)
yxt	work	(mxatms)
zzt	work	(mxatms)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)
gvx1	work	(msgrp)
gvy1	work	(msgrp)
gvz1	work	(msgrp)
omx1	work	(msgrp)
omy1	work	(msgrp)
omz1	work	(msgrp)
redmass	work	effective reduced mass along constraint bond (mxcons)

8.1.116.5 Comments

Generates trajectories in the microcanonical ensemble (NVE). The total energy (kinetic plus potential) of the system should be conserved. See NVE_1 for general comments.

Fincham's implicit quaternion leapfrog algorithm is used to solve the rigid body equations of motion. This subroutine allows a rigid body to be involved in a bond constraints with an atom or another rigid body. Should this not be required the routine NVEQ_1 is more efficient.

8.1.117 NVT_B0**8.1.117.1 Header records**

```

      subroutine nvt_b0
      x  (idnode,mxnode,imcon,natms,engke,sigma,tstep,taut,
      x  buffer,cell,fx,fy,fz,vx,vy,vz,weight,xxx,yyy,zzz,stress)

c
c*****
c
c      dl_poly subroutine for integrating newtonian equations of
c      motion in molecular dynamics - verlet leapfrog with Berendsen
c      thermostat.
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1993
c      author    - t. forester      july 1993
c
c*****
c

```

8.1.117.2 Function

Integrates equations of motion subject to the Berendsen thermostat. Assumes no bond constraints are specified for the system.

8.1.117.3 Dependencies

- GDSUM
- IMAGES
- MERGE

8.1.117.4 Arguments**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors in use
imcon	input	periodic boundary condition key. See IMAGES
natms	input	total number of atoms in MD cell.

reals:

engke	output	kinetic energy
sigma	input	system kinetic energy for specified temperature

tstep	input	simulation timestep, Δt
taut	input	thermostat relaxation time
real arrays:		
buffer	work	buffer array (mxbuff). See MERGE
cell	input	cell vectors (9). See IMAGES.
fx	input	x component of force (mxatms)
fy	input	y component of force (mxatms)
fz	input	z component of force (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
stress	in/out	stress tensor (9)

8.1.117.5 Comments

The integration scheme is the Verlet leapfrog algorithm. Temperature is controlled by the Berendsen thermostat. The velocities are scaled by χ where:

$$\chi = \left(1 + \frac{\Delta t}{\tau} \left(\frac{T}{\mathcal{T}} - 1\right)\right)^{1/2} \quad (8.1)$$

where τ is the thermostat relaxation time, $\mathcal{T}$ the instantaneous temperature and T the specified system temperature. Usually a value of $\tau \approx 1$ ps works well.

The Berendsen thermostat does not conserve energy nor does it generate trajectories in the canonical (NVT) ensemble.

8.1.118 NVT_B1**8.1.118.1 Header records**

```

      subroutine nvt_b1
      x  (safe,lshmov,idnode,imcon,mxnode,natms,nscons,ntcons,
      x  engke,taut,sigma,tolnce,tstep,vircon,lashap,
      x  lishap,listcon,listme,lstfrz,listot,buffer,
      x  cell,dxt,dxx,dyt,dyy,dzt,dzz,fx,fyy,fzz,prmcon,
      x  txx,tyy,tzz,uxx,uyy,uzz,vxx,vyy,vzz,weight,xdf,
      x  xxt,xxx,ydf,yyt,yyy,zdf,zzt,zzz,vx1,vy1,vz1,stress)

C*****
C
C    dl_poly subroutine for integrating newtonian equations of
C    motion in molecular dynamics - verlet leapfrog with Berendsen
C    thermostat.
C
C    parallel replicated data version : block data
C
C    for systems using bond constraints
C
C    copyright - daresbury laboratory 1993
C    author    -   t. forester   may 1993
C    amended : t.forester sept 1994
C    amended : t.forester dec 1994 : block data
C
C*****
C

```

8.1.118.2 Function

Integrates equations of motion subject to the Berendsen thermostat and bond constraints.

8.1.118.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- RDSHAKE_1

8.1.118.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
lshmov	input	flag for shared bond constraints
integers:		
idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system
reals:		
engke	output	kinetic energy
taut	input	thermostat relaxation time, τ_T
sigma	input	desired kinetic energy
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircon	output	constraint virial
integer arrays:		
lashap	input	see RDSHAKE_1 (mxproc)
lishap	input	see RDSHAKE_1 (mxlshp)
listcon	input	indices for constraint bonds (mxcons , 3)
listme	input	see RDSHAKE_1 (mxatms)
lstfrz	input	key for frozen atoms (mxatms)
listot	input	see RDSHAKE_1 (mxatms)
real arrays:		
buffer	work	see MERGE (mxbuff)
cell	input	MD cell vectors (9)
dxt	work	see RDSHAKE_1 (mxcons)
dxs	work	see RDSHAKE_1 (mxcons)
dys	work	see RDSHAKE_1 (mxcons)
dyy	work	see RDSHAKE_1 (mxcons)
dzt	work	see RDSHAKE_1 (mxcons)
dzz	work	see RDSHAKE_1 (mxcons)
fxs	in/out	x component of force (mxatms)
fys	in/out	y component of force (mxatms)
fzs	in/out	z component of force (mxatms)
prmcon	input	constraint distances (mxtcon)
txs	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
uxs	work	(mxatms)
uys	work	(mxatms)
uzz	work	(mxatms)
vxs	in/out	x component of velocity (mxatms)
vys	in/out	y component of velocity (mxatms)

vzz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xdf	work	x component of intersite vector (mxxdf)
xxt	work	(mxatms)
xxx	in/out	x coordinates (mxatms)
ydf	work	y component of intersite vector (mxxdf)
yyt	work	(mxatms)
yyy	in/out	y coordinates (mxatms)
zdf	work	z component of intersite vector (mxxdf)
zzt	work	(mxatms)
zzz	in/out	z coordinates (mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)

8.1.118.5 Comments

The integration scheme is the Verlet leapfrog algorithm. Temperature is controlled by the Berendsen thermostat. At each step the velocities are scaled by χ where:

$$\chi = \left(1 + \frac{\Delta t}{\tau_T} \left(\frac{T}{\mathcal{T}} - 1\right)\right)^{1/2} \quad (8.2)$$

where τ_T is the thermostat relaxation time, $\mathcal{T}$ the instantaneous temperature and T the specified system temperature. Usually a value of τ_T in the range (0.1, 1) ps works well.

The Berendsen thermostat does not conserve energy nor does it generate trajectories in the canonical (NVT) ensemble.

The thermostat needs two iterations to converge as the SHAKE algorithm affects the final velocities. Note that SHAKE need only be applied on the first iteration since subsequently the velocities of atoms in a constraint bond (relative to the bond C.O.M. velocity) are orthogonal to the bond itself and scaling of velocities does not alter this.

8.1.119 NVT_E0**8.1.119.1 Header records**

```

      subroutine nvt_e0
      x      (idnode,imcon,natms,mxnode,engke,sigma,tstep,
      x      buffer,cell,fx,fy,fz,vx,vy,vz,weight,xxx,yyy,zzz,
      x      stress)

c
c*****
c
c      dl_poly subroutine for integrating newtonian equations of
c      motion in molecular dynamics - verlet leapfrog with Evans
c      thermostat.
c      Comp. Phys. reports 1, 299, (1984)
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1993
c      author    - t. forester    feb 1993
c
c*****
c

```

8.1.119.2 Function

The verlet algorithm for integrating Newton's equations of motion subject to Gaussian constraints on the temperature. The thermostat is due to Evans. This routine is called only when *no bond constraints* are specified in the system.

8.1.119.3 Dependencies:

- IMAGES
- MERGE
- GDSUM

8.1.119.4 Arguments:**integers:**

idnode	input	identity of node on the cube.
imcon	input	periodic boundary condition key. See IMAGES
natms	input	total number of atoms in MD cell.
mxnode	input	total number of nodes on the cube

reals:

engke	output	kinetic energy
sigma	input	system kinetic energy for specified temperature
tstep	input	simulation time step, Δt .

real arrays:

buffer	work	buffer array (mxbuff). See MERGE
cell	input	cell vectors (9). See IMAGES.
fx	input	x component of force array (mxatms)
fy	input	y component of force array (mxatms)
fz	input	z component of force array (mxatms)
vx	in/out	x velocity array (mxatms)
vy	in/out	y velocity array (mxatms)
vz	in/out	z velocity array (mxatms)
weight	input	atomic mass array (mxatms).
xx	in/out	x coordinate array (mxatms).
yy	in/out	y coordinate array (mxatms)
zz	in/out	z coordinate array (mxatms)
stress	in/out	stress tensor (9)

8.1.119.5 Comments

Temperature is constrained to the specified system temperature by Gaussian constraints.

8.1.120 NVT_E1**8.1.120.1 Header records**

```

      subroutine nvt_e1
      x  (safe,lshmov,idnode,imcon,mxnode,natms,nscons,ntcons,
      x  engke,sigma,tolnce,tstep,vircon,lashap,lishap,
      x  listcon,listme,listot,lstfrz,buffer,cell,dxt,
      x  dxx,dyt,dyy,dzt,dzz,fxx,fyy,fzz,prmcon,txx,tyy,
      x  tzz,uxx,uyy,uzz,vxx,vyy,vzz,weight,xdf,xxt,xxx,
      x  ydf,yyt,yyy,zdf,zzt,zzz,stress)

C*****
C
C      dl_poly subroutine for integrating newtonian equations of
C      motion in molecular dynamics - verlet leapfrog with Evans
C      thermostat.
C      Comp. Phys. reports 1, 299, (1984)
C
C      parallel replicated data version : block data
C
C      for systems using bond CONSTRAINTS.
C
C      copyright - daresbury laboratory 1993
C      author      -      t. forester      july 1993
C      amended     -      t. forester      sept 1994
C      amended     -      t. forester      dec 1994 : block data
C
C*****
C

```

8.1.120.2 Function

Verlet leapfrog integration of equations of motion using both bond constraints and Gaussian temperature constraints.

8.1.120.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- RDSHAKE_1

8.1.120.4 Arguments**logicals:**

safe	output	flag for convergence of SHAKE
lshmov	input	flag for shared bond constraints

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
nscons	input	number of constraints on current processor
ntcons	input	total number of constraints in system

reals:

engke	output	kinetic energy
sigma	input	system kinetic energy
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircon	output	constraint virial

integer arrays:

lashap	input	see RDSHAKE_1 (mxproc)
lishap	input	see RDSHAKE_1 (mxlshp)
listcon	input	indices for constraint bonds (mxcons ,3)
listme	input	see RDSHAKE_1 (mxatms)
listot	input	see RDSHAKE_1 (mxatms)
lstfrz	input	key for frozen atoms (mxatms)

real arrays:

buffer	work	see MERGE (mxbuff)
cell	input	MD cell vectors (9)
dxt	work	see RDSHAKE_1 (mxcons)
dxx	work	see RDSHAKE_1 (mxcons)
dxt	work	see RDSHAKE_1 (mxcons)
dyy	work	see RDSHAKE_1 (mxcons)
dzt	work	see RDSHAKE_1 (mxcons)
dzz	work	see RDSHAKE_1 (mxcons)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
prmcon	input	constraint distances (mxtcon)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)

vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
weight	input	mass (mxatms)
xdf	work	x component of intersite vector (mxxdf)
xxt	work	(mxatms)
xxx	in/out	x coordinates (mxatms)
ydf	work	y component of intersite vector (mxxdf)
yxt	work	(mxatms)
yyy	in/out	y coordinates (mxatms)
zdf	work	z component of intersite vector (mxxdf)
zzt	work	(mxatms)
zzz	in/out	z coordinates (mxatms)
stress	in/out	stress tensor (9)

8.1.120.5 Comments

Bond constraints and temperature constraints are solved iteratively.

8.1.121 NVT_H0**8.1.121.1 Header records**

```

      subroutine nvt_h0
      x  (idnode,mxnode,imcon,natms,chit,conint,consv,engke,
      x  taut,sigma,tstep,buffer,cell,fxx,fyy,fzz,vxx,vyy,
      x  vzz,weight,xxx,yyy,zzz,uxx,uyy,uzz,stress)

c
c*****
c
c      dl_poly subroutine for integrating newtonian equations of
c      motion in molecular dynamics - verlet leapfrog with Hoover
c      thermostat.
c      Phys. Rev A 31, 1695 (1985)
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1993
c      author    - t. forester      feb 1993
c      amended   - t. forester      sept 1994
c
c*****
c

```

8.1.121.2 Function

Integrates equations of motion subject to the Hoover thermostat. Assumes no bond constraints are specified for the system.

8.1.121.3 Dependencies

- GDSUM
- IMAGES
- MERGE

8.1.121.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
imcon	input	periodic boundary condition key. See IMAGES
natms	input	total number of atoms in MD cell.

reals:

chit	in/out	friction coefficient for thermostat
conint	in/out	integral of chit for conserved quantity
consv	out	kinetic plus potential energy of thermostat
engke	output	kinetic energy
taut	input	reciprocal of thermostating rate
sigma	input	system kinetic energy for specified temperature
tstep	input	simulation time step

real arrays:

buffer	work	buffer array (mxbuff). See MERGE
cell	input	cell vectors (9). See IMAGES.
fx	input	x component of force array (mxatms)
fy	input	y component of force array (mxatms)
fz	input	z component of force array (mxatms)
vxx	in/out	x velocity array (mxatms)
vyy	in/out	y velocity array (mxatms)
vzz	in/out	z velocity array (mxatms)
weight	input	atomic masses (mxatms)
xxx	in/out	x coordinate array (mxatms).
yyy	in/out	y coordinate array (mxatms)
zzz	in/out	z coordinate array (mxatms)
ux	work	x component of new velocity (mxatms)
uy	work	y component of new velocity (mxatms)
uz	work	z component of new velocity (mxatms)
stress	in/out	stress tensor (9)

8.1.121.5 Comments

Verlet leapfrog implementation of the Nosé-Hoover thermostat. The value of the friction coefficient, χ for the thermostat is found from the first-order differential equation:

$$\dot{\chi} = \frac{1}{\tau^2}(\mathcal{T}/T - 1) \quad (8.3)$$

where τ is the reciprocal of the thermostating rate, $\mathcal{T}$ the instantaneous temperature and T the specified system temperature. Usually a value of $\tau = 1\text{ps}$ works well but this may be system dependent.

Overall the conserved quantity is, to within a constant, the Helmholtz free-energy of the system. The contributions of the thermostat to this are placed in the variable **consv** and passed back to the main program.

8.1.122 NVT_H1**8.1.122.1 Header records**

```

      subroutine nvt_h1
      x (safe,lshmov,idnode,imcon,mxnode,natms,nscons,ntcons,
      x  chit,consv,conint,engke,taut,sigma,tolnce,tstep,vircon,
      x  lashap,lishap,listcon,listme,listot,lstfrz,buffer,
      x  cell,dxt,dxx,dyt,dyy,dzt,dzz,fx,fyy,fzz,prmcon,txx,
      x  tyy,tzz,uxx,uyy,uzz,vxx,vyy,vzz,weight,xdf,xxt,xxx,
      x  ydf,yyt,yyy,zdf,zzt,zzz,vx1,vy1,vz1,stress)

C*****
C
C      dl_poly subroutine for integrating newtonian equations of
C      motion in molecular dynamics - verlet leapfrog with Hoover
C      thermostat.
C
C      parallel replicated data version : block data
C
C      for systems using bond constraints
C
C      copyright - daresbury laboratory 1993
C      author    -   t. forester    may 1993
C      amended   -   t.forester    sept 1994
C      amended   -   t.forester    dec 1994 : block data
C
C*****
C

```

8.1.122.2 Function

Integrates equations of motion using both bond constraints and the Hoover thermostat.

8.1.122.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- RDSHAKE_1

8.1.122.4 Arguments

logicals:

<code>safe</code>	output	flag for convergence of SHAKE
<code>lshmov</code>	input	flag for shared bond constraints
integers:		
<code>idnode</code>	input	identity of current processor
<code>imcon</code>	input	key for periodic boundary conditions
<code>mxnode</code>	input	total number of processors in use
<code>natms</code>	input	number of atoms in MD cell
<code>nscons</code>	input	number of constraints on current processor
<code>ntcons</code>	input	total number of constraints in system
reals:		
<code>chit</code>	in/out	friction coefficient for heat bath
<code>consv</code>	out	kinetic plus potential energy of thermostat
<code>conint</code>	in/out	integral of chit for conserved quantity
<code>engke</code>	output	kinetic energy
<code>taut</code>	input	thermostat time constant, τ_T
<code>sigma</code>	input	system kinetic energy
<code>tolnce</code>	input	tolerance for SHAKE algorithm
<code>tstep</code>	input	simulation time-step, Δt
<code>vircon</code>	output	constraint virial
integer arrays:		
<code>lashap</code>	input	see RDSHAKE_1 (<code>mxproc</code>)
<code>lishap</code>	input	see RDSHAKE_1 (<code>mxlshp</code>)
<code>listcon</code>	input	indices for constraint bonds (<code>mxcons</code> , 3)
<code>listme</code>	input	see RDSHAKE_1 (<code>mxatms</code>)
<code>listot</code>	input	see RDSHAKE_1 (<code>mxatms</code>)
<code>lstfrz</code>	input	key for frozen atoms (<code>mxatms</code>)
real arrays:		
<code>buffer</code>	work	see MERGE (<code>mxbuff</code>)
<code>cell</code>	input	MD cell vectors (9)
<code>dxt</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dxs</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dys</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dyy</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dzt</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dzz</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>fxs</code>	in/out	x component of force (<code>mxatms</code>)
<code>fys</code>	in/out	y component of force (<code>mxatms</code>)
<code>fzs</code>	in/out	z component of force (<code>mxatms</code>)
<code>prmcon</code>	input	constraint distances (<code>mxtcon</code>)
<code>txs</code>	work	(<code>mxatms</code>)
<code>tys</code>	work	(<code>mxatms</code>)
<code>tzs</code>	work	(<code>mxatms</code>)
<code>uxs</code>	work	(<code>mxatms</code>)
<code>uys</code>	work	(<code>mxatms</code>)

uzz	work	(mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xdf	work	x component of intersite vector (mxxdf)
xxt	work	(mxatms)
xxx	in/out	x coordinates (mxatms)
ydf	work	y component of intersite vector (mxxdf)
yyt	work	(mxatms)
yyy	in/out	y coordinates (mxatms)
zdf	work	z component of intersite vector (mxxdf)
zzt	work	(mxatms)
zzz	in/out	z coordinates (mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)

8.1.122.5 Comments

Temperature is controlled by the Nosé-Hoover thermostat. The value of the friction coefficient, χ for the thermostat is found from the first-order differential equation:

$$\dot{\chi} = \frac{1}{\tau_T^2} \left(\frac{\mathcal{T}}{T} - 1 \right) \quad (8.4)$$

where $\mathcal{T}$ is the instantaneous temperature. Usually a value of τ_T in the range (0.5, 2) ps works well but this may be system dependent.

The extended Hamiltonian of the system is a conserved quantity and this may be used as a check on the choice of τ_T .

This subroutine also includes a call to `RDSHAKE_1` to handle bond constraints by the iterative SHAKE method. The SHAKE routine need only be applied on the first thermostat iteration. Thereafter the velocities of atoms in a constraint bond are orthogonal to the bond and the velocity scaling of the thermostat does not alter this. Currently 4 thermostat iterations are used if constraint bonds are present, 3 if they are not. This may be changed by altering the parameter `maxit` in the code and recompiling.

8.1.123 NVT_H1**8.1.123.1 Header records**

```

      subroutine nvt_h1
x  (safe,lshmov,idnode,imcon,mxnode,natms,nscons,ntcons,
x  chit,consv,conint,engke,taut,sigma,tolnce,tstep,vircon,
x  lashap,lishap,listcon,listme,listot,lstfrz,buffer,
x  cell,dxt,dxx,dyt,dyy,dzt,dzz,fx,fyy,fzz,prmcon,txx,
x  tyy,tzz,uxx,uyy,uzz,vxx,vyy,vzz,weight,xdf,xxt,xxx,
x  ydf,yyt,yyy,zdf,zzt,zzz,vx1,vy1,vz1,stress)

C*****
C
C    dl_poly subroutine for integrating newtonian equations of
C    motion in molecular dynamics - verlet leapfrog with Hoover
C    thermostat.
C
C    parallel replicated data version : block data
C
C    for systems using bond constraints
C
C    copyright - daresbury laboratory 1993
C    author    -    t. forester    may 1993
C    amended   -    t.forester    sept 1994
C    amended   -    t.forester    dec 1994 : block data
C
C*****
C

```

8.1.123.2 Function

Integrates equations of motion using both bond constraints and the Hoover thermostat.

8.1.123.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- RDSHAKE_1

8.1.123.4 Arguments

logicals:

<code>safe</code>	output	flag for convergence of SHAKE
<code>lshmov</code>	input	flag for shared bond constraints
integers:		
<code>idnode</code>	input	identity of current processor
<code>imcon</code>	input	key for periodic boundary conditions
<code>mxnode</code>	input	total number of processors in use
<code>natms</code>	input	number of atoms in MD cell
<code>nscons</code>	input	number of constraints on current processor
<code>ntcons</code>	input	total number of constraints in system
reals:		
<code>chit</code>	in/out	friction coefficient for heat bath
<code>consv</code>	out	kinetic plus potential energy of thermostat
<code>conint</code>	in/out	integral of chit for conserved quantity
<code>engke</code>	output	kinetic energy
<code>taut</code>	input	thermostat time constant, τ_T
<code>sigma</code>	input	system kinetic energy
<code>tolnce</code>	input	tolerance for SHAKE algorithm
<code>tstep</code>	input	simulation time-step, Δt
<code>vircon</code>	output	constraint virial
integer arrays:		
<code>lashap</code>	input	see RDSHAKE_1 (<code>mxproc</code>)
<code>lishap</code>	input	see RDSHAKE_1 (<code>mxlshp</code>)
<code>listcon</code>	input	indices for constraint bonds (<code>mxcons</code> , 3)
<code>listme</code>	input	see RDSHAKE_1 (<code>mxatms</code>)
<code>listot</code>	input	see RDSHAKE_1 (<code>mxatms</code>)
<code>lstfrz</code>	input	key for frozen atoms (<code>mxatms</code>)
real arrays:		
<code>buffer</code>	work	see MERGE (<code>mxbuff</code>)
<code>cell</code>	input	MD cell vectors (9)
<code>dxt</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dxs</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dys</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dyy</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dzt</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>dzz</code>	work	see RDSHAKE_1 (<code>mxcons</code>)
<code>fxs</code>	in/out	x component of force (<code>mxatms</code>)
<code>fys</code>	in/out	y component of force (<code>mxatms</code>)
<code>fzs</code>	in/out	z component of force (<code>mxatms</code>)
<code>prmcon</code>	input	constraint distances (<code>mxtcon</code>)
<code>txs</code>	work	(<code>mxatms</code>)
<code>tyy</code>	work	(<code>mxatms</code>)
<code>tzz</code>	work	(<code>mxatms</code>)
<code>uxs</code>	work	(<code>mxatms</code>)
<code>uyy</code>	work	(<code>mxatms</code>)

uzz	work	(mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xdf	work	x component of intersite vector (mxxdf)
xxt	work	(mxatms)
xxx	in/out	x coordinates (mxatms)
ydf	work	y component of intersite vector (mxxdf)
yyt	work	(mxatms)
yyy	in/out	y coordinates (mxatms)
zdf	work	z component of intersite vector (mxxdf)
zzt	work	(mxatms)
zzz	in/out	z coordinates (mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)

8.1.123.5 Comments

Temperature is controlled by the Nosé-Hoover thermostat. The value of the friction coefficient, χ for the thermostat is found from the first-order differential equation:

$$\dot{\chi} = \frac{1}{\tau_T^2} \left(\frac{\mathcal{T}}{T} - 1 \right) \quad (8.5)$$

where $\mathcal{T}$ is the instantaneous temperature. Usually a value of τ_T in the range (0.5, 2) ps works well but this may be system dependent.

The extended Hamiltonian of the system is a conserved quantity and this may be used as a check on the choice of τ_T .

This subroutine also includes a call to RDSHAKE_1 to handle bond constraints by the iterative SHAKE method. The SHAKE routine need only be applied on the first thermostat iteration. Thereafter the velocities of atoms in a constraint bond are orthogonal to the bond and the velocity scaling of the thermostat does not alter this. Currently 4 thermostat iterations are used if constraint bonds are present, 3 if they are not. This may be changed by altering the parameter **maxit** in the code and recompiling.

8.1.124 NVTQ_B1

8.1.124.1 Header records

```

      subroutine nvtq_b1
x  (safe,safeq,lshmov,imcon,idnode,mxnode,natms,ngroup,nscons,
x  ntcons,ntfree,engke,engrot,quattol,sigma,taut,tolnce,
x  tstep,vircom,vircon,lstme,lishap,lashap,lstfrz,listcon,
x  listme,lstfre,lstgtp,lstrgd,listot,numgsit,cell,gcmx1,
x  gcmy1,gcmz1,gvx1,gyy1,gvz1,dxx,dyy,dzz,dxt,dyt,dzt,uxx,
x  uyy,uzz,prmcon,xdx,ydy,zdz,omx,omy,omz,omx1,omy1,omz1,
x  opx,opy,opz,oqx,oqy,oqz,q0,q1,q2,q3,qn0,qn1,qn2,qn3,gxx,
x  gyy,gzz,xxx,yyy,zzz,vxx,vyy,vzz,gvxx,gvyy,gvzz,fxx,fyy,
x  fzz,tqx,tqy,tqz,weight,rotinx,rotiny,rotinz,gcmx,gcmy,
x  gcmz,gmass,buffer,txt,yyt,zzt,txx,tyy,tzz,vx1,vy1,vz1,
x  stress)

C*****
C
C  dlpoly routine to integrate rigid body equations of motion
C  using implicit leapfrog quaternion algorithm
C  plus bond constraints - PROVIDED rigid body sites
C  and constraint sites do not coincide.
C
C  dl_poly subroutine for integrating newtonian equations of
C  motion in molecular dynamics - verlet leapfrog with Berendsen
C  thermostat.
C
C  parallel replicated data version : block data
C
C  for systems using bond constraints
C
C  parallel replicated data version : block data
C
C  tqx,tqy,tqz = torque in lab fixed frame (input)
C  omx,omy,omz = angular velocity in body fixed frame (principal axis)
C  rotinx,y,z = rotational inertia in body fixed frame
C
C  copyright daresbury laboratory 1993
C  author      t.forester october 1993
C  amended     t.forester dec 1994 : block data
C
C*****

```

8.1.124.2 Function

The verlet leapfrog algorithm for integrating Newton's equations of motion for systems containing rigid bodies with Berendsen thermostat. Implicit quaternion equations are used for the rigid body equations of motion.

8.1.124.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- MERGE1
- RDSHAKE_1

8.1.124.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
safeq	output	flag for convergence of quaternion algorithm
lshmov	input	flag for shared bond constraints

integers:

imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ngrp	input	number of rigid body units in system
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system
ntfree	input	number of sites in system not in rigid body units

reals:

engke	output	kinetic energy
engrot	output	rotational kinetic energy
quattol	input	tolerance for quaternion algorithm
sigma	input	required system kinetic energy
taut	input	thermostat time constant, τ_T
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircom	output	molecule to c.o.m. virial correction
vircon	output	constraint virial

integer arrays:

lstme	input	indice list for MERGE1 (mxatms)
lishap	input	see RDSHAKE_1 (mxlshp)

lashap	input	see RDSHAKE_1 (mxproc)
lstfrz	input	key for frozen atoms (mxatms)
listcon	input	indices for constraint bonds (mxcons ,3)
listme	input	see RDSHAKE_1 (mxatms)
lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgatk)
listot	input	see RDSHAKE_1 (mxatms)
numgsit	input	number of sites in a rigid unit type (mxungp)
real arrays:		
cell	input	MD cell vectors (9)
gcmx1	work	(msgrp)
gcmy1	work	(msgrp)
gcmz1	work	(msgrp)
gvx1	work	(msgrp)
gvy1	work	(msgrp)
gvz1	work	(msgrp)
dx	work	see RDSHAKE_1 (mxcons)
dy	work	see RDSHAKE_1 (mxcons)
dz	work	see RDSHAKE_1 (mxcons)
dxt	work	see RDSHAKE_1 (mxcons)
dxt	work	see RDSHAKE_1 (mxcons)
dzt	work	see RDSHAKE_1 (mxcons)
ux	work	(mxatms)
uy	work	(mxatms)
uz	work	(mxatms)
prmcon	input	constraint distances (mxtcon)
xdf	work	(mxxdf)
ydf	work	(mxxdf)
zdf	work	(mxxdf)
omx	in/out	x component of angular velocity (mxgrp)
omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
omx1	work	(msgrp)
omy1	work	(msgrp)
omz1	work	(msgrp)
opx	work	(msgrp)
opy	work	(msgrp)
opz	work	(msgrp)
oqx	work	(msgrp)
oqy	work	(msgrp)
oqz	work	(msgrp)

q0	in/out	first component of quaternion vector (mxgrp)
q1	in/out	second component of quaternion vector (mxgrp)
q2	in/out	third component of quaternion vector (mxgrp)
q3	in/out	fourth component of quaternion vector (mxgrp)
qn0	work	(msgrp)
qn1	work	(msgrp)
qn2	work	(msgrp)
qn3	work	(msgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
tqx	work	x component of torque on rigid unit (msgrp)
tqy	work	y component of torque on rigid unit (msgrp)
tqz	work	z component of torque on rigid unit (msgrp)
weight	input	atomic masses (mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)
gcmx	in/out	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	in/out	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	in/out	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	input	rigid unit mass (mxungp)
buffer	work	(mxbuff)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)

stress in/out stress tensor (9)

8.1.124.5 Comments

See NVT_B1 for general comments on the Berendsen thermostat.

Fincham's implicit quaternion leapfrog algorithm is used to solve the rigid body equations of motion. This subroutine does not allow a rigid body to be involved in bond constraints with an atom or another rigid body. Should this be required use the routine NVTQ_B2 instead.

8.1.125 NVTQ_B2**8.1.125.1 Header records**

```

      subroutine nvtq_b2
x  (safe,safeq,lshmov,imcon,idnode,mxnode,natms,ngroup,nscons,
x  ntcons,ntfree,engke,engrot,quattol,sigma,taut,tolnce,
x  tstep,vircom,vircon,lstme,lishap,lashap,lstfrz,listcon,
x  listme,lstfre,lstcsit,lstbod,lstgtp,lstrgd,listot,
x  numgsit,cell,gcmx1,gcmx2,gcmz1,dxx,dyy,dzz,dxt,dyt,dzt,
x  uxx,uyy,uzz,prmcon,omx,omy,omz,opx,opy,opz,oqx,oqy,oqz,
x  q0,q1,q2,q3,qn0,qn1,qn2,qn3,gxx,gyy,gzz,xxx,yyy,zzz,vxx,
x  vyy,vzz,gvxx,gvyy,gvzz,fx,fy,fz,tqx,tqy,tqz,weight,
x  rotinx,rotiny,rotinz,gcmx,gcmx2,gcmz,gmass,buffer,txt,tyt,
x  zzt,txx,tyy,tzz,vx1,vy1,vz1,stress,gvx1,gvy1,gvz1,omx1,
x  omy1,omz1,redmass)

c*****
c
c  dlpoly routine to integrate rigid body equations of motion
c  using implicit leapfrog quaternion algorithm
c  plus bond constraints - rigid body sites and constraint sites
c  may coincide.
c
c  dl_poly subroutine for integrating newtonian equations of
c  motion in molecular dynamics - verlet leapfrog with Berendsen
c  thermostat.
c
c  parallel replicated data version : block data
c
c  tqx,tqy,tqz = torque in lab fixed frame (input)
c  omx,omy,omz = angular velocity in body fixed frame (principal axis)
c  rotinx,y,z = rotational inertia in body fixed frame
c
c  copyright daresbury laboratory 1995
c  author      t.forester june 1995
c
c*****

```

8.1.125.2 Function

The verlet leapfrog algorithm for integrating Newton's equations of motion for systems containing linked rigid bodies with Berendsen thermostat. Implicit quaternion equations are used for the rigid body equations of motion.

8.1.125.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- MERGE1
- MERGE4
- QSHAKE

8.1.125.4 Arguments**logicals:**

safe	output	flag for convergence of SHAKE
safeq	output	flag for convergence of quaternion algorithm
lshmov	input	flag for shared bond constraints

integers:

imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ngrp	input	number of rigid body units in system
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system
ntfree	input	number of sites in system not in rigid body units

reals:

engke	output	kinetic energy
engrot	output	rotational kinetic energy
quattol	input	tolerance for quaternion algorithm
sigma	input	required system kinetic energy
taut	input	thermostat time constant, τ_T
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircom	output	molecule to c.o.m. virial correction
vircon	output	constraint virial

integer arrays:

lstme	input	indice list for MERGE1 (mxatms)
lishap	input	see QSHAKE (mxlshp)
lashap	input	see QSHAKE (mxproc)
lstfrz	input	key for frozen atoms (mxatms)
listcon	input	indices for constraint bonds (mxcons , 3)
listme	input	see QSHAKE (mxatms)

lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
lstcsit	input	indices of rigid unit sites in constraints (2*mxcons)
lstbod	input	pointer from site index to rigid unit index (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgadm)
listot	input	see QSHAKE (mxatms)
numgsit	input	number of sites in a rigid unit type (mxungp)
real arrays:		
cell	input	MD cell vectors (9)
gcmx1	work	(msgrp)
gcmy1	work	(msgrp)
gcmz1	work	(msgrp)
dx	work	see QSHAKE (mxcons)
dy	work	see QSHAKE (mxcons)
dz	work	see QSHAKE (mxcons)
dx	work	see QSHAKE (mxcons)
dy	work	see QSHAKE (mxcons)
dz	work	see QSHAKE (mxcons)
ux	work	(mxatms)
uy	work	(mxatms)
uz	work	(mxatms)
prmcon	input	constraint distances (mxtcon)
omx	in/out	x component of angular velocity (mxgrp)
omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
opx	work	(msgrp)
opy	work	(msgrp)
opz	work	(msgrp)
oqx	work	(msgrp)
oqy	work	(msgrp)
oqz	work	(msgrp)
q0	in/out	first component of quaternion vector (mxgrp)
q1	in/out	second component of quaternion vector (mxgrp)
q2	in/out	third component of quaternion vector (mxgrp)
q3	in/out	fourth component of quaternion vector (mxgrp)
qn0	work	(msgrp)
qn1	work	(msgrp)
qn2	work	(msgrp)
qn3	work	(msgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)

xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
txx	work	x component of torque on rigid unit (msgrp)
txy	work	y component of torque on rigid unit (msgrp)
txz	work	z component of torque on rigid unit (msgrp)
weight	input	atomic masses (mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)
gcmx	in/out	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	in/out	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	in/out	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	input	rigid body unit mass (mxungp)
buffer	work	(mxbuff)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)
gvx1	work	(msgrp)
gvy1	work	(msgrp)
gvz1	work	(msgrp)
omx1	work	(msgrp)
omy1	work	(msgrp)
omz1	work	(msgrp)
redmass	work	effective reduced mass along constraint bond (mxcons)

8.1.125.5 Comments

See NVT_B1 for general comments on the Berendsen thermostat.

Fincham's implicit quaternion leapfrog algorithm is used to solve the rigid body equations of motion. This subroutine allows a rigid body to be involved in bond constraints with an atom or another rigid body. Should this not be required the routine NVTQ_B1 is more efficient.

8.1.126 NVTQ_H1

8.1.126.1 Header records

```

      subroutine nvtq_h1
x  (safe,safeq,lshmov,imcon,idnode,mxnode,natms,ngroup,nscons,
x  ntcons,ntfree,chit,consv,conint,engke,engrot,quattol,
x  sigma,taut,tolnce,tstep,vircom,vircon,lstme,lishap,
x  lashap,lstfrz,listcon,listme,lstfre,lstgtp,lstrgd,
x  listot,numgsit,cell,gcmx1,gcmx2,gcmz1,dxx,dyy,dzz,
x  dxt,dyt,dzt,uxx,uyy,uzz,prmcon,xdf,ydf,zdf,omx,omy,
x  omz,omx1,omy1,omz1,opx,opy,opz,oqx,oqy,oqz,q0,q1,q2,q3,
x  qn0,qn1,qn2,qn3,gxx,gyy,gzz,xxx,yyy,zzz,vxx,vyy,vzz,gvxx,
x  gvyy,gvzz,fxx,fyy,fzz,tqx,tqy,tqz,weight,rotinx,rotiny,
x  rotinz,gcmx,gcmx2,gcmz,gmass,buffer,xxt,yyt,zzt,txx,
x  tyy,tzz,gvx1,gvx2,gvz1,vx1,vx2,vz1,stress)

C*****
C
C      dlpoly routine to integrate rigid body equations of motion
C      using implicit leapfrog quaternion algorithm
C      plus bond constraints - PROVIDED rigid body sites
C      and constraint sites do not coincide.
C
C      dl_poly subroutine for integrating newtonian equations of
C      motion in molecular dynamics - verlet leapfrog with Hoover
C      thermostat.
C
C      for systems using bond constraints
C
C      parallel replicated data version : block data
C
C      tqx,tqy,tqz = torque in lab fixed frame (input)
C      omx,omy,omz =angular velocity in body fixed frame (principal axis)
C      rotinx,y,z = rotational inertia in body fixed frame
C
C      copyright daresbury laboratory 1993
C      author      t.forester october 1993
C      amended     t.forester dec 1994 : block data
C
C*****

```

8.1.126.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion for systems containing rigid bodies. The system is coupled to a Nosé-Hoover thermostat.

8.1.126.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- MERGE1
- RDSHAKE_1

8.1.126.4 Arguments

logicals:

<code>safe</code>	output	flag for convergence of SHAKE
<code>safeq</code>	output	flag for convergence of quaternion algorithm
<code>lshmov</code>	input	flag for shared bond constraints

integers:

<code>imcon</code>	input	key for periodic boundary conditions
<code>idnode</code>	input	identity of current processor
<code>mxnode</code>	input	total number of processors in use
<code>natms</code>	input	number of atoms in MD cell
<code>ngrp</code>	input	number of rigid body units in system
<code>nscons</code>	input	number of constraints on current processor
<code>ntcons</code>	input	number of constraints in system
<code>ntfree</code>	input	number of sites in system not in rigid body units

reals:

<code>chit</code>	in/out	velocity scaling variable
<code>consv</code>	output	scaling variables contribution to extended Hamiltonian
<code>conint</code>	in/out	time integral of scaling variables
<code>engke</code>	output	kinetic energy
<code>engrot</code>	output	rotational kinetic energy
<code>quattol</code>	input	tolerance for quaternion algorithm
<code>sigma</code>	input	required system kinetic energy
<code>taut</code>	input	thermostat time constant, τ_T
<code>tolnce</code>	input	tolerance for SHAKE algorithm
<code>tstep</code>	input	simulation time-step, Δt
<code>vircom</code>	output	molecule to c.o.m. virial correction
<code>vircon</code>	output	constraint virial

integer arrays:

lstme	input	indice list for MERGE1 (mxatms)
lishap	input	see RDSHAKE_1 (mxlshp)
lashap	input	see RDSHAKE_1 (mxproc)
lstfrz	input	key for frozen atoms (mxatms)
listcon	input	indices for constraint bonds (mxcons ,3)
listme	input	see RDSHAKE_1 (mxatms)
lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgatk)
listot	input	see RDSHAKE_1 (mxatms)
numgsit	input	number of sites in a rigid unit type (mxungp)
real arrays:		
cell	input	MD cell vectors (9)
gcmx1	work	(msgrp)
gcmy1	work	(msgrp)
gcmz1	work	(msgrp)
dxx	work	see RDSHAKE_1 (mxcons)
dyy	work	see RDSHAKE_1 (mxcons)
dzz	work	see RDSHAKE_1 (mxcons)
dxt	work	see RDSHAKE_1 (mxcons)
dyt	work	see RDSHAKE_1 (mxcons)
dzt	work	see RDSHAKE_1 (mxcons)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
prmcon	input	constraint distances (mxtcon)
xdf	work	(mxxdf)
ydf	work	(mxxdf)
zdf	work	(mxxdf)
omx	in/out	x component of angular velocity (mxgrp)
omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
omx1	work	(msgrp)
omy1	work	(msgrp)
omz1	work	(msgrp)
opx	work	(msgrp)
opy	work	(msgrp)
opz	work	(msgrp)
oqx	work	(msgrp)
oqy	work	(msgrp)
oqz	work	(msgrp)
q0	in/out	first component of quaternion vector (mxgrp)

q1	in/out	second component of quaternion vector (mxgrp)
q2	in/out	third component of quaternion vector (mxgrp)
q3	in/out	fourth component of quaternion vector (mxgrp)
qn0	work	(msgrp)
qn1	work	(msgrp)
qn2	work	(msgrp)
qn3	work	(msgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
txx	work	x component of torque on rigid unit (msgrp)
txy	work	y component of torque on rigid unit (msgrp)
txz	work	z component of torque on rigid unit (msgrp)
weight	input	atomic masses (mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)
gcmx	in/out	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	in/out	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	in/out	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	input	rigid unit mass (mxungp)
buffer	work	(mxbuff)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
gvx1	work	(msgrp)
gvy1	work	(msgrp)
gvz1	work	(msgrp)
vx1	work	(msatms)

vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)

8.1.126.5 Comments

See NVT_H1 for general comments on the Nosé-Hoover thermostat.

Fincham's implicit quaternion leapfrog algorithm is used to solve the rigid body equations of motion. This subroutine does not allow a rigid body to be involved in bond constraints with an atom or another rigid body. Should this be required use the routine NVTQ_H2 instead.

8.1.127 NVTQ_H2**8.1.127.1 Header records**

```

      subroutine nvtq_h2
x  (safe,safeq,lshmov,imcon,idnode,mxnode,natms,ngroup,nscons,
x  ntcons,ntfree,conint,consv,chit,engke,engrot,quattol,
x  sigma,taut,tolnce,tstep,vircom,vircon,lstme,lishap,
x  lashap,lstfrz,listcon,listme,lstfre,lstcsit,lstbod,
x  lstgtp,lstrgd,listot,numgsit,cell,gcmx1,gcmx1,gcmz1,dxx,
x  dyy,dzz,dxt,dyt,dzt,uxx,uyy,uzz,prmcon,omx,omy,omz,opx,
x  opy,opz,oqx,oqy,oqz,q0,q1,q2,q3,qn0,qn1,qn2,qn3,gxx,gyy,
x  gzz,xxx,yyy,zzz,vxx,vyy,vzz,gvxx,gvyy,gvzz,fxx,fyy,fzz,
x  tqx,tqy,tqz,weight,rotinx,rotiny,rotinz,gcmx,gcmx,gcmz,
x  gmass,buffer,xxt,yyt,zzt,txx,tyy,tzz,vx1,vy1,vz1,stress,
x  gvxx1,gvyy1,gvzz1,omx1,omy1,omz1,redmass)

c*****
c
c  dlpoly routine to integrate rigid body equations of motion
c  using implicit leapfrog quaternion algorithm
c  plus bond constraints - rigid body sites and constraint sites
c  may coincide.
c
c  dl_poly subroutine for integrating newtonian equations of
c  motion in molecular dynamics - verlet leapfrog with Hoover
c  thermostat.
c
c  parallel replicated data version : block data
c
c  tqx,tqy,tqz = torque in lab fixed frame (input)
c  omx,omy,omz = angular velocity in body fixed frame (principal axis)
c  rotinx,y,z = rotational inertia in body fixed frame
c
c  copyright daresbury laboratory 1995
c  author      t.forester june 1995
c
c*****

```

8.1.127.2 Function

The Verlet leapfrog algorithm for integrating Newton's equations of motion for systems containing linked rigid bodies with a Nosé-Hoover thermostat. Implicit quaternion equations are used for the rigid body equations of motion.

8.1.127.3 Dependencies

- GDSUM
- IMAGES
- MERGE
- MERGE1
- MERGE4
- QSHAKE

8.1.127.4 Arguments**logicals:**

safe	output	flag for convergence of SHAKE
safeq	output	flag for convergence of quaternion algorithm
lshmov	input	flag for shared bond constraints

integers:

imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
ngrp	input	number of rigid body units in system
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system
ntfree	input	number of sites in system not in rigid body units

reals:

conint	in/out	time integral of scaling variables
consv	output	scaling variables contribution to extended Hamiltonian
chit	in/out	velocity scaling variable
engke	output	kinetic energy
engrot	output	rotational kinetic energy
quattol	input	tolerance for quaternion algorithm
sigma	input	required system kinetic energy
taut	input	thermostat time constant, τ_T
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircom	output	molecule to c.o.m. virial correction
vircon	output	constraint virial

integer arrays:

lstme	input	indice list for MERGE1 (mxatms)
lishap	input	see QSHAKE (mxlshp)
lashap	input	see QSHAKE (mxproc)

lstfrz	input	key for frozen atoms (mxatms)
listcon	input	indices for constraint bonds (mxcons ,3)
listme	input	see QSHAKE (mxatms)
lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
lstcsit	input	indices of rigid unit sites in constraints (2* mxcons)
lstbod	input	pointer from site index to rigid unit index (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgadm)
listot	input	see QSHAKE (mxatms)
numgsit	input	number of sites in a rigid unit type (mxungp)
real arrays:		
cell	input	MD cell vectors (9)
gcmx1	work	(msgrp)
gcmy1	work	(msgrp)
gcmz1	work	(msgrp)
dxs	work	see QSHAKE (mxcons)
dys	work	see QSHAKE (mxcons)
dzs	work	see QSHAKE (mxcons)
dxt	work	see QSHAKE (mxcons)
dys	work	see QSHAKE (mxcons)
dzt	work	see QSHAKE (mxcons)
uxs	work	(mxatms)
uys	work	(mxatms)
uzs	work	(mxatms)
prmcon	input	constraint distances (mxtcon)
omx	in/out	x component of angular velocity (mxgrp)
omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
opx	work	(msgrp)
opy	work	(msgrp)
opz	work	(msgrp)
oqx	work	(msgrp)
oxy	work	(msgrp)
ozx	work	(msgrp)
q0	in/out	first component of quaternion vector (mxgrp)
q1	in/out	second component of quaternion vector (mxgrp)
q2	in/out	third component of quaternion vector (mxgrp)
q3	in/out	fourth component of quaternion vector (mxgrp)
qn0	work	(msgrp)
qn1	work	(msgrp)
qn2	work	(msgrp)
qn3	work	(msgrp)

gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
tqx	work	x component of torque on rigid unit (msgrp)
tqy	work	y component of torque on rigid unit (msgrp)
tqz	work	z component of torque on rigid unit (msgrp)
weight	input	atomic masses (mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)
gcmx	in/out	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	in/out	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	in/out	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	input	rigid unit mass (mxungp)
buffer	work	(mxbuff)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
vx1	work	(msatms)
vy1	work	(msatms)
vz1	work	(msatms)
stress	in/out	stress tensor (9)
gvx1	work	(msgrp)
gvy1	work	(msgrp)
gvz1	work	(msgrp)
omx1	work	(msgrp)
omy1	work	(msgrp)
omz1	work	(msgrp)
redmass	work	effective reduced mass along constraint bond (mxcons)

8.1.127.5 Comments

See NVT_H1 for general comments on the Nosé-Hoover thermostat.

Fincham's implicit quaternion leapfrog algorithm is used to solve the rigid body equations of motion. This subroutine allows a rigid body to be involved in bond constraints with an atom or another rigid body. Should this not be required the routine NVTQ_H1 is more efficient.

8.1.128 PAIR_FORCES_SW**8.1.128.1 Header records**

```

      subroutine pair_force_sw
      x  (lgr,lneut,idnode,imcon,iregn,keyfce,mxnode,mxregn,
      x  natms,nneut,numrdf,dlrpot,drewd,engcpe,engsrp,epsq,
      x  rcut,rcuta,rcutb,rvdw,swlen,vircpe,virsrp,ilist,jlist,
      x  lentry,lexatm,list,lstfrz,lstneu,lstout,lstvdw,ltpvdw,
      x  ltype,neulst,nexatm,buffer,cell,chge,erc,fer,fxx,fyy,
      x  fzz,ggg,rdf,rsqdf,stress,txx,tyy,tzz,uxx,uyy,uzz,vvv,
      x  xdf,xxt,xxx,ydf,yyt,yyy,zdf,zzt,zzz)

c*****
c
c      dlpoly subroutine to control calculation of pair interactions
c      Cubic switching functions may be applied at the boundaries
c      between multiple-time step zones.
c
c      copyright daresbury laboratory 1996
c      author t.forester          june 1996
c
c*****

```

8.1.128.2 Function

Controls calculation of non-bonded interactions for multiple time-step simulations.

8.1.128.3 Dependencies

COUL0	COUL0_SW	COUL2	COUL2_SW
COUL3	COUL3_SW	COUL4	EWALD2
EWALD2_SW	FORCESNEU_RESPA	GDSUM	IMAGES
SRFRCE	SRFRCE_SW	RDF0	RDF0_SW

8.1.128.4 Arguments**logicals:**

lgr	input	flag for construction of rdfs
lneut	input	flag for charge group implementation

integers:

idnode	input	identity of current node
imcon	input	key for periodic boundary conditions
iregn	input	identity of current zone
keyfce	input	key for non-bonded force field

<code>mxnode</code>	input	total number of nodes
<code>mxregn</code>	input	total number of zones
<code>natms</code>	input	number of atoms in MD cell
<code>nneut</code>	input	number of charge groups in MD cell
<code>numrdf</code>	in/out	running counter of configurations used for RDFs.
reals:		
<code>dlrpot</code>	input	distance increment for entries in potential look-up tables
<code>drewd</code>	input	distance increment for entries in Ewald sum look-up tables
<code>engcpe</code>	output	electrostatic energy
<code>engsrp</code>	output	non-bonded potential energy
<code>epsq</code>	input	relative dielectric constant
<code>rcut</code>	input	electrostatic potential cut-off radius
<code>rcuta</code>	input	upper cutoff for zone 1
<code>rcutb</code>	input	upper cutoff for zone 2
<code>rvdw</code>	input	Van der Waals potential cut-off radius
<code>swlen</code>	input	width of switching region between zones
<code>vircep</code>	output	electrostatic virial
<code>virsrp</code>	output	non-bonded potential virial
<code>volm</code>	input	volume of MD cell
integer arrays:		
<code>ilist</code>	work	neighbourhood list for central site (<code>mxxdf</code>)
<code>jlist</code>	work	neighbourhood list for second site (<code>mxxdf</code>)
<code>lentry</code>	input	number of entries in Verlet list (<code>msatms</code>)
<code>lexatm</code>	input	excluded atoms list (<code>msatms,mxexcl</code>)
<code>list</code>	input	the Verlet list (<code>msatms,*</code>)
<code>lstfrz</code>	input	key for frozen atoms (<code>mxatms</code>)
<code>lstneu</code>	input	index of group to which site belongs (<code>mxatms</code>)
<code>lstout</code>	work	(<code>mxatms</code>)
<code>lstvdw</code>	input	index number of non-bonded potential (<code>mxvdw</code>)
<code>ltpvdw</code>	input	key for non-bonded potentials (<code>mxvdw</code>)
<code>ltype</code>	input	list of atom types (<code>mxatms</code>)
<code>nexatm</code>	input	number of excluded atoms (<code>msatms</code>)
<code>neulst</code>	input	indices of first site in charge group (<code>mxneut</code>)
real arrays:		
<code>buffer</code>	work	work array (<code>mxbuff</code>)
<code>cell</code>	input	MD cell vectors (9)
<code>chge</code>	input	atomic charges (<code>mxatms</code>)
<code>erc</code>	input	interpolation array of function $\text{erfc}(\alpha r)/r$ (<code>mxgrid</code>)
<code>fer</code>	input	first derivative of interpolation array <code>erc</code> (<code>mxgrid</code>)
<code>fxx</code>	output	x component of force (<code>mxatms</code>)
<code>fyy</code>	output	y component of force (<code>mxatms</code>)
<code>fzz</code>	output	z component of force (<code>mxatms</code>)
<code>ggg</code>	input	interpolation table for non-bonded force (<code>mxgrid,mxvdw</code>)
<code>rdf</code>	in/out	histograms for radial distribution functions (<code>mxrdf,mxvdw</code>)

rsqdf	work	square of intersite distances (mxxdf)
stress	in/out	stress tensor (9)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
vvv	input	interpolation table for non-bonded energy (mxgrid,mxvdw)
xdf	work	x component of intersite vector (mxxdf)
xxt	work	work (mxatms)
xxx	input	x coordinates (mxatms)
ydf	work	y component of intersite vector (mxxdf)
yyt	work	work (mxatms)
yyy	input	y coordinates (mxatms)
zdf	work	z component of intersite vector (mxxdf)
zzt	work	work (mxatms)
zzz	input	z coordinates (mxatms)

8.1.128.5 Comments

8.1.129 PARLINK**8.1.129.1 Header records**

```

      subroutine parlink
      x  (newlst,natms,idnode,mxnode,imcon,rcut,delr,
      x  lct,link,lexatm,nexatm,lentry,list,lstfrz,
      x  cell,xxx,yyy,zzz,uxx,uyy,uzz,buffer)
c
c*****
c
c      dl_poly subroutine for constructing the verlet neighbour
c      list based on link-cell method.
c      frozen atoms taken into account
c
c      to be used with the link version of exclude :exclude_link
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1993
c      author    - t. forester september 1993.
c
c*****
c

```

8.1.129.2 Function

Constructs distributed Verlet neighbour list from a link cell search. Brode-Alhrichs ordering of pairs is imposed.

8.1.129.3 Dependencies

- DCELL
- ERROR
- GSTATE
- IMAGES
- INVERT
- MERGE

8.1.129.4 Arguments

logicals:

newlst input flag for construction of Verlet list

integers:

natms	input	total number of atoms in the MD cell
idnode	input	identity of current processor
mxnode	input	total number of processors in use
imcon	input	key for periodic boundary conditions

reals:

rcut	input	Potential cut-off
delr	input	width of verlet border

integer arrays:

lct	work	link cell head of chain (mxcell)
link	work	pointer to next atom in link cell (mxatms)
lexatm	input	list of interactions to be excluded from the verlet list (msatms , mxexcl)
nexatm	input	number of excluded atoms for each site (msatms)
lentry	output	number of neighbours in Verlet neighbourhood list (msatms)
list	output	the Verlet neighbourhood list (msatms , mxlist)
lstfrz	input	key for frozen atoms (mxatms)

real arrays:

cell	input	MD cell vectors (9)
xxx	input	x coordinates (mxatms)
yyy	input	y coordinates (mxatms)
zzz	input	z coordinates (mxatms)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
buffer	work	(mxbuff)

8.1.129.5 Comments

See PARLST for general comments on the Brode-Alhrichs algorithm. The maximum number of link cells is given by **mxcell** in the file DL_PARAMS.INC.

The link cell algorithm can only be used when the MD cell is a parallelepiped or non-periodic *i.e.* for **imcon** = 0, 1, 2, 3 or 6. Mock cell vectors are created when there are no periodic boundaries (**imcon** = 0) or when the system is not periodic in the Z direction (**imcon** = 6).

There ratio of the potential cutoff to the link cell length may between 1 and 5. This is controlled by the variable **irat** which is hard wired to 1, as this has proved optimal in parallel applications. Users may wish to uncomment the code that allows **irat** to vary.

8.1.130 PARLINKNEU**8.1.130.1 Header records**

```

      subroutine parlinkneu
      x  (newlst,lneut,natms,nneut,idnode,mxnode,imcon,rcut,delr,
      x  lentry,lct,link,lstneu,list,lstfrz,neulst,cell,xxx,yyy,
      x  zzz,uxx,uyy,uzz,buffer)

c
c*****
c
c      dl_poly subroutine for constructing the verlet neighbour
c      list based on link-cell method with neutral groups
c      frozen atoms taken into account
c
c      to be used with the link version of exclude :excludeneu_link
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1996
c      author    - t. forester january 1996.
c
c*****
c

```

8.1.130.2 Function

Constructs distributed Verlet neighbour list from a link cell search. Brode-Alhrichs ordering of charge group pairs is imposed.

8.1.130.3 Dependencies

- DCELL
- ERROR
- GIMAX
- GSTATE
- IMAGES
- INVERT
- MERGE
- SHELLSORT

8.1.130.4 Arguments

logicals:

newlst	input	flag for construction of Verlet list
lneut	input	flag for neutral (charge) group scheme

integers:

natms	input	total number of atoms in the MD cell
nneut	input	total number of charge groups in the MD cell
idnode	input	identity of current processor
mxnode	input	total number of processors in use
imcon	input	key for periodic boundary conditions

reals:

rcut	input	Potential cut-off
delr	input	width of verlet border

integer arrays:

lentry	output	number of neighbours in Verlet neighbourhood list (msatms)
lct	work	link cell head of chain (mxcell)
link	work	pointer to next atom in link cell (mxatms)
lstneu	input	pointer from atoms to charge group (mxatms)
list	output	the Verlet neighbourhood list (msatms,mxlist)
lstfrz	input	key for frozen atoms (mxatms)
neulst	input	index of first site in cahрге groups (mxneut)
cell	input	MD cell vectors (9)
xxx	input	x coordinates (mxatms)
yyy	input	y coordinates (mxatms)
zzz	input	z coordinates (mxatms)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
buffer	work	(mxbuff)

real arrays:

8.1.130.5 Comments

See PARLST for general comments on the Brode-Alhrichs algorithm. The maximum number of link cells is given by **mxcell** in the file DL_PARAMS.INC.

The link cell algorithm can only be used when the MD cell is a parallelepiped or non-periodic *i.e.* for **imcon** = 0, 1, 2, 3 or 6. Mock cell vectors are created when there are no periodic boundaries (**imcon** = 0) or when the system is not periodic in the Z direction (**imcon** = 6).

8.1.131 PARLST

8.1.131.1 Header records

```

      subroutine parlst
      x  (newlst,natms,idnode,mxnode,imcon,rcut,delr,lexatm,nexatm,
      x  noxatm,lentry,list,lstfrz,cell,xxx,yyy,zzz,xdf,ydf,zdf)
c
c*****
c
c      dl_poly subroutine for constructing the verlet neighbour
c      list based on the brode-ahlrchs decomposition
c      frozen atoms taken into account
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1992
c      author    - w.smith    march 1992
c      modified  - t.forester october 1993
c
c*****
c

```

8.1.131.2 Function

Constructs the distributed Verlet neighbourhood list based on the Brode-Ahlrichs decomposition.

8.1.131.3 Dependencies

- ERROR
- GSTATE
- IMAGES

8.1.131.4 Arguments

logicals:

newlst	input	flag for construction of a new verlet list

integers:

natms	input	number of atoms in the MD cell
idnode	input	identity of current processor
mxnode	input	total number of processors in use
imcon	input	control variable for periodic boundary conditions

reals:

rcut	input	potential cut off radius
delr	input	width of the outer shell of the verlet list
integer arrays:		
lexatm	input	list of interactions to be excluded from the verlet list (msatms , mxexcl)
nexatm	input	number of excluded atoms for each site (msatms)
noxatm	work	running counter for number of excluded atoms found during the construction of the verlet list (msatms)
lentry	output	number of neighbours in Verlet neighbourhood list (msatms)
list	output	the Verlet neighbourhood list (msatms , mxlist)
lstfrz	input	key for frozen atoms (mxatms)
real arrays:		
cell	input	MD cell vectors (9)
xxx	input	x coordinate (mxatms)
yyy	input	y coordinate (mxatms)
zzz	input	z coordinate (mxatms)
xdf	work	x component of intersite vector (mxxdf)
ydf	work	y component of intersite vector (mxxdf)
zdf	work	z component of intersite vector (mxxdf)

8.1.131.5 Comments

The sites in the cell are distributed evenly across all processors. Thus the Verlet list on any particular processor will contain pairs of indices unique to the processor. The Brode-Ahlich decomposition arranges the pairs of interactions so that the number of neighbours for each central site is reasonably uniform.

msatms must be at least $(\text{natms} + \text{mxnode} - 1) / \text{mxnode}$. **msatms** is defined in the file **DL_PARAMS.INC**.

The number of neighbours for any particular site increases as $(\text{rcut} + \text{delr})^3$. The number of neighbours must be less than or equal to **mxlist**. **mxlist** is defined in **DL_PARAMS.INC**.

8.1.132 PARLST_NSQ**8.1.132.1 Header records**

```

      subroutine parlst_nsq
      x  (newlst,natms,idnode,mxnode,imcon,engcpe,epsq,rcut,
      x  vircpe,lexatm,nexatm,noxatm,lentry,list,lstfrz,
      x  cell,xxx,yyy,zzz,xdf,ydf,zdf,fx,fyy,fzz,chge,stress)
c
c*****
c
c      dl_poly subroutine for constructing the verlet neighbour
c      list based on the brode-ahlrchs decomposition .
c      Evaluates Coulombic interactions for everything outside verlet
c      shell - as part of 'secondary' forces.
c      frozen atom option included
c
c      to be used with multiple_nsq
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1994
c      author    - t. forester march 1994
c
c      stress tensor : t.forester may 1994
c
c*****
c

```

8.1.132.2 Function**8.1.132.3 Dependencies**

- ERROR
- GSTATE
- IMAGES

8.1.132.4 Arguments**logicals:**

newlst	input	flag for construction of a new verlet list

integers:

natms	input	number of atoms in the MD cell
idnode	input	identity of current processor

<code>mxnode</code>	input	total number of processors in use
<code>imcon</code>	input	control variable for periodic boundary conditions
reals:		
<code>engcpe</code>	output	Coulombic energy
<code>epsq</code>	input	relative dielectric constant, ϵ_r
<code>rcut</code>	input	potential cut off radius
<code>vircpe</code>	output	Coulombic virial
integer arrays:		
<code>lexatm</code>	input	list of interactions to be excluded from the verlet list (<code>msatms</code> , <code>mxexcl</code>)
<code>nexatm</code>	input	number of excluded atoms for each site (<code>msatms</code>)
<code>noxatm</code>	work	running counter for number of excluded atoms found during the construction of the verlet list (<code>msatms</code>)
<code>lentry</code>	output	number of neighbours in Verlet neighbourhood list (<code>msatms</code>)
<code>list</code>	output	the Verlet neighbourhood list (<code>msatms</code> , <code>mxlist</code>)
<code>lstfrz</code>	input	key for frozen atoms (<code>mxatms</code>)
real arrays:		
<code>cell</code>	input	MD cell vectors (9)
<code>xxx</code>	input	x coordinate (<code>mxatms</code>)
<code>yyy</code>	input	y coordinate (<code>mxatms</code>)
<code>zzz</code>	input	z coordinate (<code>mxatms</code>)
<code>xdf</code>	work	x component of intersite vector (<code>mxxdf</code>)
<code>ydf</code>	work	y component of intersite vector (<code>mxxdf</code>)
<code>zdf</code>	work	z component of intersite vector (<code>mxxdf</code>)
<code>fxx</code>	in/out	x component of force (<code>mxatms</code>)
<code>fyy</code>	in/out	y component of force (<code>mxatms</code>)
<code>fzz</code>	in/out	z component of force (<code>mxatms</code>)
<code>chge</code>	input	electrostatic charge (<code>mxatms</code>)
<code>stress</code>	in/out	stress tensor (9)

8.1.132.5 Comments

See PARLST for a description of the Brode-Ahrlrichs algorithm and choice of parameters. In this variant, the electrostatic interactions for pairs outside the Verlet shell are evaluated explicitly. In this way an “all-pairs” calculation is invoked. The more distant interactions are updated only when the pair list is updated. This algorithm is designed for use with MULTIPLE_NSQ on non-periodic charged systems.

8.1.133 PARLST_RESPA**8.1.133.1 Header records**

```

      subroutine parlst_respa
      x  (newlst,natms,idnode,mxnode,imcon,rcut,rcut2,
      x  rcut1,delr,swlen,lexatm,nexatm,noxatm,lentry3,lentry2,
      x  lentry1,list3,list2,list1,lstfrz,cell,xxx,yyy,zzz,
      x  xdf,ydf,zdf)

c
c*****
c
c      dl_poly subroutine for constructing the verlet neighbour
c      list based on the brode-ahlrichs decomposition
c      frozen atoms taken into account
c
c      list constructed on basis of centre - centre  distances
c      plus the size of each group.
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1996
c      author    - t.forester   june    1996

c*****
c
c

```

8.1.133.2 Function

Constructs the distributed Verlet neighbourhood list based on the Brode-Ahlrichs decomposition and subdivides it into 3 multiple time step zones.

8.1.133.3 Dependencies

ERROR GSTATE IMAGES

8.1.133.4 Arguments**logicals:**

newlst input flag for construction of a new verlet list

integers:

natms input number of atoms in the MD cell

idnode input identity of current processor

<code>mxnode</code>	input	total number of processors in use
<code>imcon</code>	input	control variable for periodic boundary conditions
reals:		
<code>rcut</code>	input	potential cut off radius
<code>rcut2</code>	input	upper cut off of second zone
<code>rcut1</code>	input	upper cut off of first zone
<code>delr</code>	input	width of the outer shell of the verlet list
<code>swlen</code>	input	width of switching region between zones
integer arrays:		
<code>lexatm</code>	input	list of interactions to be excluded from the verlet list (<code>msatms,mxexcl</code>)
<code>nexatm</code>	input	number of excluded atoms for each site (<code>msatms</code>)
<code>noxatm</code>	work	running counter for number of excluded atoms found during the construction of the verlet list (<code>msatms</code>)
<code>lentry3</code>	output	number of neighbours in third Verlet neighbourhood list (<code>msatms</code>)
<code>lentry2</code>	output	number of neighbours in second Verlet neighbourhood list (<code>msatms</code>)
<code>lentry1</code>	output	number of neighbours in first Verlet neighbourhood list (<code>msatms</code>)
<code>list3</code>	output	the third Verlet neighbourhood list (<code>msatms,mxlist3</code>)
<code>list2</code>	output	the second Verlet neighbourhood list (<code>msatms,mxlist2</code>)
<code>list1</code>	output	the third Verlet neighbourhood list (<code>msatms,mxlist1</code>)
<code>lstfrz</code>	input	key for frozen atoms (<code>mxatms</code>)
real arrays:		
<code>cell</code>	input	MD cell vectors (9)
<code>xxx</code>	input	x coordinate (<code>mxatms</code>)
<code>yyy</code>	input	y coordinate (<code>mxatms</code>)
<code>zzz</code>	input	z coordinate (<code>mxatms</code>)
<code>xdf</code>	work	x component of intersite vector (<code>mxxdf</code>)
<code>ydf</code>	work	y component of intersite vector (<code>mxxdf</code>)
<code>zdf</code>	work	z component of intersite vector (<code>mxxdf</code>)

8.1.133.5 Comments

See PARLST for a description of the Brode-Alhrichs algorithm. If `swlen` is non-zero then there will be some overlap between the multiple-time step zones and hence some atom pairs will appear in more than one of `list1`, `list2`, `list3`.

8.1.134 PARNEULST

8.1.134.1 Header records

```

      subroutine parneulst
      x  (newlst,lneut,nneut,idnode,mxnode,imcon,rcut,
      x  delr,lentry,list,lstfrz,neulst,cell,xxx,yyy,zzz,
      x  xdf,ydf,zdf)
c
c*****
c
c      dl_poly subroutine for constructing the verlet neighbour
c      list based on the brode-ahlrachs decomposition
c      group based implementation
c      frozen atoms taken into account
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1992
c      author    - t.forester   april   1994
c
c*****
c

```

8.1.134.2 Function

Generates distributed Verlet neighbourhood list based on the Brode-Ahlrichs decomposition. The list consists of interacting *groups* as opposed to *atoms*.

8.1.134.3 Dependencies

- ERROR
- GSTATE
- ERROR

8.1.134.4 Arguments

logicals:

newlst	input	flag for construction of a new verlet list
lneut	input	flag for neutral group implementation

integers:

nneut	input	number of charge groups in system
idnode	input	identity of current processor

<code>mxnode</code>	input	total number of processors in use
<code>imcon</code>	input	control variable for periodic boundary conditions
reals:		
<code>rcut</code>	input	potential cut off radius
<code>delr</code>	input	width of the outer shell of the verlet list
integer arrays:		
<code>lentry</code>	output	number of neighbours in Verlet neighbourhood list (<code>msatms</code>)
<code>list</code>	output	the Verlet neighbourhood list (<code>msatms,mxlist</code>)
<code>lstfrz</code>	input	key for frozen atoms (<code>mxatms</code>)
<code>neulst</code>	input	indice of first site in charge group (<code>mxneut</code>)
real arrays:		
<code>cell</code>	input	MD cell vectors (9)
<code>xxx</code>	input	x coordinate (<code>mxatms</code>)
<code>yyy</code>	input	y coordinate (<code>mxatms</code>)
<code>zzz</code>	input	z coordinate (<code>mxatms</code>)
<code>xdf</code>	work	x component of intersite vector (<code>mxxdf</code>)
<code>ydf</code>	work	y component of intersite vector (<code>mxxdf</code>)
<code>zdf</code>	work	z component of intersite vector (<code>mxxdf</code>)

8.1.134.5 Comments

For general comments on the Brode-Ahlrichs algorithm see comments on `PARLST`. In this routine groups rather than atoms are assigned to the list. Thus the parameter `msatms` should be at least $(\text{nneut} + \text{mxnode} - 1) / \text{mxnode}$. `msatms` is defined in the file `DL_PARAMS.INC`. The “excluded” interactions are taken care of when the groups are unpacked - see `NEUTLST`.

8.1.135 PARNEULST_RESPA

8.1.135.1 Header records

```

      subroutine parneulst_respa
      x  (newlst,lneut,nneut,idnode,mxnode,imcon,rcut,rcut2,rcut1,
      x  delr,swlen,lentry3,lentry2,lentry1,list3,list2,list1,
      x  lstfrz,neulst,cell,xxx,yyy,zzz,xdf,ydf,zdf,txt,yyt,zzt,
      x  txx,tyy,tzz,buffer,uxx)

c
c*****
c
c      dl_poly subroutine for constructing the verlet neighbour
c      list based on the brode-ahlrichs decomposition
c      frozen atoms taken into account
c      3 distance clases
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1996
c      author    - t.forester    june    1996
c
c*****
c

```

8.1.135.2 Function

Generates distributed Verlet neighbourhood list based on the Brode-Ahlrichs decomposition. The list consists of interacting *groups* as opposed to *atoms*, and the list is subdivided into three distance classes.

8.1.135.3 Dependencies

- ERROR
- GSTATE
- ERROR

8.1.135.4 Arguments

logicals:

newlst	input	flag for construction of a new verlet list
lneut	input	flag for neutral group implementation

integers:

nneut	input	number of charge groups in system
idnode	input	identity of current processor
mxnode	input	total number of processors in use
imcon	input	control variable for periodic boundary conditions
reals:		
rcut	input	potential cut off radius
rcut2	input	upper cut off radius for second zone
rcut1	input	upper cut off radius for first zone
delr	input	width of the outer shell of the verlet list
swlen	input	width of switching region between zones
integer arrays:		
lentry3	output	number of neighbours in third Verlet neighbourhood list (msatms)
lentry2	output	number of neighbours in second Verlet neighbourhood list (msatms)
lentry1	output	number of neighbours in first Verlet neighbourhood list (msatms)
list3	output	the third Verlet neighbourhood list (msatms,mxlist3)
list2	output	the second Verlet neighbourhood list (msatms,mxlist2)
list1	output	the third Verlet neighbourhood list (msatms,mxlist1)
lstfrz	input	key for frozen atoms (mxatms)
neulst	input	indice of first site in charge group (mxneut)
real arrays:		
cell	input	MD cell vectors (9)
xxx	input	x coordinate (mxatms)
yyy	input	y coordinate (mxatms)
zzz	input	z coordinate (mxatms)
xdf	work	x component of intersite vector (mxxdf)
ydf	work	y component of intersite vector (mxxdf)
zdf	work	z component of intersite vector (mxxdf)
xxt	work	(mxatms)
xxt	work	(mxatms)
zzt	work	(mxatms)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
buffer	work	(mxbuff)
uxx	work	(mxatms)

8.1.135.5 Comments

For general comments on the Brode-Ahlrichs algorithm see `co PARLST`. In this routine groups rather than atoms are assigned to the list. Thus the parameter **msatms** should be at least $(\text{nneut} + \text{mxnode} - 1) / \text{mxnode}$. **msatms** is defined in the file `DL_PARAMS.INC`. The “excluded” interactions are taken care of when the groups are unpacked - see `NEUTLST`.

If **swlen** is non-zero then there will be some overlap between the multiple-time step

zones and hence some atom pairs will appear in more than one of `list1`, `list2`, `list3`. The assignments are based on the group centre - centre distances and the width of each group.

8.1.136 PARSET

This description applies to the subroutine PARSET in DL_POLY_2 Version 2.11 and subsequent versions and not to the utility program of the same name.

8.1.136.1 Header records

```
subroutine parset(idnode,mxnode,buffer)
```

```
c
c*****
c
c    dl_poly subroutine to determine required array sizes for
c    allocation of memory manager
c
c    author - w.smith june 1997
c    copyright daresbury laboratory 1997
c
c*****
c
```

8.1.136.2 Function

Calculates estimates of required DL_POLY_2 array sizes for dynamic array allocation and stores the computed values in the /params/ COMMON block.

8.1.136.3 Dependencies

```
DCELL          CFGSCAN          CONSCAN
FLDSCAN
```

8.1.136.4 Arguments

```
integers:
  idnode      input      identity of current processor
  mxnode      input      total number of processors
real arrays:
  buffer      work        (10)
```

8.1.136.5 Comments

See description of the DL_PARAMS.INC file 8.1.1 for a description of the contents of the /params/ COMMON block.

8.1.137 PASSCON**8.1.137.1 Header records**

```

      subroutine passcon
      x  (lshmov,idnode,mxnode,natms,nscons,lashap,lishap,listme,
      x   listin,listot,listcon,lstfrz)
c
c*****
c
c      dl_poly subroutine for passing information about bond
c      constraints between nodes
c
c      parallel replicated data version assuming direct node-node
c      connection (i.e. this version may be intel specific)
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith august 1992.
c      MPI version t.forester may 1995
c      CPP version - w.smith may 1995
c
c*****
c

```

8.1.137.2 Function

Passes information about bond constraints between processors. This routine is only called during the initialisation of the system for the MD simulation and only when constraint dynamics are required.

8.1.137.3 Dependencies

- message send and receive routines
- ERROR
- GISUM
- GSTATE
- GSYNC

8.1.137.4 Arguments

logicals:

lshmov output flag for shared bond constraints

integers:

<code>idnode</code>	input	identity of current processor
<code>mxnode</code>	input	total number of processors
<code>natms</code>	input	number of atoms in MD cell
<code>nscons</code>	input	number of constraints handled on current processor
integer arrays:		
<code>lashap</code>	output	“last atom” marker (<code>mxproc</code>)
<code>lishap</code>	output	“shared atoms” list (<code>mxlshp</code>)
<code>listme</code>	output	number of constraints involving atom that are processed on this processor (<code>mxatms</code>)
<code>listin</code>	work	number of constraints involving atom that are processed by communicating processor (<code>mxatms</code>)
<code>listot</code>	output	number of processors with constraints involving atom (<code>mxatms</code>)
<code>listcon</code>	input	bond constraint bookkeeping list (<code>mxcons,3</code>)
<code>lstfrz</code>	input	key for frozen atoms (<code>mxatms</code>)

8.1.137.5 Comments

Abandon hope all who enter here - You mess with this routine at your own peril.

C-preprocessing is used to select the appropriate communication routines.

8.1.138 PASSPMF**8.1.138.1 Header records**

```

      subroutine passpmf
      x (idnode,mxnode,natms,nspmf,listpm,listin,lstpmt,lstpmf,npmf)
c
c*****
c
c      dl_poly subroutine for passing information about PMF
c      constraints between nodes
c
c      parallel replicated data version assuming direct node-node
c      connection (i.e. this version may be intel specific)
c
c      copyright - daresbury laboratory 1995
c      author    - t.forester august 1995.
c
c*****
c

```

8.1.138.2 Function

Passes information about Potential of Mean Force (PMF) constraints between processors.

8.1.138.3 Dependencies

- ERROR
- GISUM

8.1.138.4 Arguments**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors
natms	input	number of atoms in MD cell
nspmf	input	number of PMF constraints on the current processor

integer arrays:

listpm	output	flag for atoms in PMF constraints (mxpmf)
listin	work	(mxatms)
lstpmt	output	global PMF sites register (mxpmf)
lstpmf	input	local PMF sites register (mxspmf,mspmf)
npmf	input	number of sites in PMF constraint units (2)

8.1.138.5 **Comments**

8.1.139 PASSQUAT**8.1.139.1 Header records**

```

      subroutine passquat
      x  (lcnb,idnode,mxnode,natms,ngroup,nscons,ntpmls,listin,
      x  listcon,lstrgd,lstout,lstcsit,lstgtp,nummols,numgrp,numgsit)
c
c*****
c
c      dl_poly subroutine for passing information about rigid body
c      atoms involved in bond constraints between nodes
c
c      parallel replicated data version assuming direct node-node
c      connection
c
c      copyright - daresbury laboratory 1995
c      author    - t. forester december 1995.
c
c*****
c

```

8.1.139.2 Function**8.1.139.3 Dependencies**

- ERROR
- GISUM
- GSTATE

8.1.139.4 Arguments**logicals:**

lcnb	output	flag for rigid bodies with bond constraints

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	total number of atoms in the MD cell
ngroup	input	number of rigid body units in system
nscons	input	number of constraints on current processor
ntpmls	output	number of unique types of molecule

integer arrays:

listin	work	(mxatms)
listcon	input	indices for constraint bonds (mxcons,3)

<code>lstrgd</code>	input	indices of rigid unit sites handles by current processor (<code>mxgadm</code>)
<code>lstout</code>	work	(<code>mxatms</code>)
<code>lstcsit</code>	output	indices of rigid unit sites in constraints (<code>2*mxcons</code>)
<code>lstgtp</code>	input	pointer from rigid unit index to rigid unit type (<code>mxgrp</code>)
<code>nummols</code>	input	number of molecules of a given type (<code>mxtmls</code>)
<code>numgrp</code>	output	number of rigid units in a molecule (<code>mxtmls</code>)
<code>numgsit</code>	input	number of sites in a rigid unit type (<code>mxungp</code>)

8.1.139.5 Comments

Don't mess with this routine!

8.1.140 PMF_1**8.1.140.1 Header records**

```

      subroutine pmf_1
      x  (safe,safep,lshmov,idnode,imcon,mxnode,natms,nscons,
      x  ntcons,nspmf,ntpmf,engke,prmpmf,tolnce,tstep,vircon,
      x  virpmf,lstpmf,npmf,lashap,lishap,listcon,listpm,lstpmt,
      x  listme,listot,lstfrz,buffer,cell,dxt,dxx,dyt,
      x  dyy,dzt,dzz,pmfwght,fx,fyy,fzz,prmcon,txx,tyy,tzz,
      x  uxx,uyy,uzz,vxx,vyy,vzz,weight,xdf,xxt,xxx,ydf,yyt,
      x  yyy,zdf,zzt,zzz,stress)
c
c*****
c
c      dl_poly subroutine for integrating newtonian equations of
c      motion in molecular dynamics. Verlet leapfrog With RD-SHAKE
c      and PMF_SHAKE - for potential of mean force calculations.
c
c      parallel replicated data version : block data
c      adapted from dl_poly routine nve_1.f
c
c      copyright - daresbury laboratory 1995
c      author - t.forester aug 1995
c
c*****
c

```

8.1.140.2 Function

As per header record. Generates trajectories in the microcanonical ensemble subject to bond constraints and Potential of Mean Force (PMF) constraints.

8.1.140.3 Dependencies

- MERGE
- GDSUM
- IMAGES
- PMF_SHAKE
- RDSHAKE_1

8.1.140.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
safep	output	flag for convergence of PMF constraints
lshmov	input	flag for shared bond constraints

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
natms	input	total number of atoms in the MD cell
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system
nspmf	input	number of PMF constraints on the current processor
ntpmf	input	total number of PMF constraints in system

reals:

engke	output	kinetic energy
prmpmf	input	PMF constraint distance
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircon	output	constraint virial
virpmf	output	virial from PMF constraint terms

integer arrays:

lstpmf	output	indices of atoms in PMF constraints on this processor (mxspmf , mshpmf)
nmpmf	input	number of sites in PMF constraint units (2)
lashap	input	see RDSHAKE_1 (mxproc)
lishap	input	see RDSHAKE_1 (mxlshp)
listcon	input	indices for constraint bonds (mxcons , 3)
listpm	input	indices of sites in PMF constraints on current processor (mxpmf)
lstpmt	input	indices of all sites involved in PMF constraints (mxpmf)
listme	input	see RDSHAKE_1 (mxatms)
listot	input	see RDSHAKE_1 (mxatms)
lstfrz	input	key for frozen atoms (mxatms)

real arrays:

buffer	work	(mxbuff)
cell	input	MD cell vectors (9)
dxt	work	see RDSHAKE_1 (mxcons)
dxs	work	see RDSHAKE_1 (mxcons)
dys	work	see RDSHAKE_1 (mxcons)
dyy	work	see RDSHAKE_1 (mxcons)
dzt	work	see RDSHAKE_1 (mxcons)
dzz	work	see RDSHAKE_1 (mxcons)

pmfwght	input	weighting for MPF sites (mxspmf)
fx	in/out	x component of force (mxatms)
fy	in/out	y component of force (mxatms)
fz	in/out	z component of force (mxatms)
prmcon	input	constraint distances (mxtcon)
tx	work	(mxatms)
ty	work	(mxatms)
tz	work	(mxatms)
ux	work	(mxatms)
uy	work	(mxatms)
uz	work	(mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xdf	work	(mxxdf)
xxt	work	(mxatms)
xxx	in/out	x coordinates (mxatms)
ydf	work	(mxxdf)
yxt	work	(mxatms)
yyy	in/out	y coordinates (mxatms)
zdf	work	(mxxdf)
zzt	work	(mxatms)
zzz	in/out	z coordinates (mxatms)
stress	in/out	stress tensor (9)

8.1.140.5 Comments

This subroutine is a modification of NVE_1. Constant temperature / constant pressure / rigid body versions may be obtained by modifying the appropriate integration routine.

8.1.141 PMF_SHAKE**8.1.141.1 Header records**

```

      subroutine pmf_shake
      x  (safep,idnode,mxnode,imcon,nspmf,tstep,tolnce,
      x  prmpmf,virpmf,npmf,lstpmf,listpm,lstpmt,dxt,
      x  dyt,dzt,cell,pmfwght,pmfnrm,xxx,yyy,zzz,xxt,
      x  yyt,zzt,stress,summas,buffer,dxp,dyp,dzp)

C*****
C
C      dlpoly constraint subroutine for potential of mean force calc.
C      accumulates constraint force to maintain reaction coordinate
C
C      copyright daresbury laboratory 1995
C      author t.forester august 1995
C
C*****

```

8.1.141.2 Function

Enforces the constraint condition in a Potential of Mean Force calculation.

8.1.141.3 Dependencies

- ERROR
- GDSUM
- IMAGES
- SPLICE

8.1.141.4 Arguments**logicals:**

safep	output	flag for convergence of PMF SHAKE

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
imcon	input	key for periodic boundary conditions
nspmf	input	number of PMF constraints on the current processor

reals:

tstep	input	simulation time-step, Δt
tolnce	input	tolerance for SHAKE algorithm

<code>prmpmf</code>	input	PMF constraint distance
<code>virpmf</code>	output	virial from PMF constraint terms
integer arrays:		
<code>npmf</code>	input	number of sites in PMF constraint units (2)
<code>lstpmf</code>	input	indices of atoms in PMF constraints on this processor (<code>mxspmf</code> , <code>mshpmf</code>)
<code>listpm</code>	input	see PASSPMF (<code>mxpmf</code>)
<code>lstpmt</code>	input	see PASSPMF (<code>mxpmf</code>)
real arrays:		
<code>dxt</code>	work	(<code>mxcons</code>)
<code>dzt</code>	work	(<code>mxcons</code>)
<code>dzt</code>	work	(<code>mxcons</code>)
<code>cell</code>	input	MD cell vectors (9)
<code>pmfwght</code>	input	weighting for PMF sites (<code>mxspmf</code>)
<code>pmfnrm</code>	input	normalising factors for PMF wieghts (2)
<code>xxx</code>	in/out	x coordinates (<code>mxatms</code>)
<code>yyy</code>	in/out	y coordinates (<code>mxatms</code>)
<code>zzz</code>	in/out	z coordinates (<code>mxatms</code>)
<code>xxt</code>	work	(<code>mxatms</code>)
<code>yyt</code>	work	(<code>mxatms</code>)
<code>zzt</code>	work	(<code>mxatms</code>)
<code>stress</code>	in/out	stress tensor (9)
<code>summas</code>	input	mass of PMF constraint units (2)
<code>buffer</code>	work	(<code>mxbuff</code>)
<code>dxp</code>	input	x component of PMF vector before unconstrained integration (<code>mxcons</code>)
<code>dyp</code>	input	y component of PMF vector before unconstrained integration (<code>mxcons</code>)
<code>dzp</code>	input	z component of PMF vector before unconstrained integration (<code>mxcons</code>)

8.1.141.5 Comments

8.1.142 PRIMLST

8.1.142.1 Header records

```

      subroutine primlst
      x  (idnode,mxnode,natms,imcon,rprim,lentry,
      x  list,cell,xxx,yyy,zzz,xdf,ydf,zdf)
c
c*****
c
c      dlpoly routine to split interaction list into primary and
c      secondary neighbours for use with multiple timestep method
c
c      copyright daresbury laborartory 1993
c
c      author - t. forester february 1993
c
c*****
c

```

8.1.142.2 Function

Classifies sites in the verlet list as either 'primary' or 'secondary' neighbours of the central site for use with multiple time-step methods. Primary neighbours are flagged by a negative entry in the verlet list.

8.1.142.3 Dependencies

- IMAGES

8.1.142.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors
natms	input	number of atoms in the MD cell
imcon	input	key for periodic boundary conditions

reals:

rprim	input	radius of the primary neighbourhood shell

integer arrays:

lentry	input	number of neighbours in the Verlet list (msatms)
list	in/out	the Verlet neighbourhood list (msatms,mxlist)

real arrays:

cell	input	MD cell vectors (9).
xxx	input	x coordinate array (mxatms).
yyy	input	y coordinate array (mxatms)

zzz	input	z coordinate array (mxatms)
xdf	work	x component of intersite vector (mxxdf)
ydf	work	y component of intersite vector (mxxdf)
zdf	work	z component of intersite vector (mxxdf)

8.1.142.5 Comments

8.1.143 PRNEULST**8.1.143.1 Header records**

```

      subroutine prneulst
      x  (newlst,imcon,idnode,mxnode,nneut,rprim,
      x  list,lentry,neulst,cell,xxx,yyy,zzz,xdf,ydf,zdf)

C*****
C
C      dlpoly routine to partition neutral group list into
C      primary and secondary groups
C      loops over group ineu
C
C      replicated data version
C
C      copyright daresbury laboratory 1994
C      author t.forester april 1994
C
C*****

```

8.1.143.2 Function

Partitions groups based Verlet list into primary and secondary neighbours. Secondary neighbours are flagged by a negative index.

8.1.143.3 Dependencies

- ERROR
- GIMAX
- GSTATE
- IMAGES

8.1.143.4 Arguments**logicals:**

newlst	input	flag for construction of Verlet list

integers:

imcon	input	key for periodic boundary conditions
idnode	input	identity of current processor
mxnode	input	total number of processors in use
nneut	input	number of charge groups in system

reals:

<code>rprim</code>	input	primary shell cut-off
integer arrays:		
<code>list</code>	in/out	the Verlet neighbourhood list (<code>msatms,mxlist</code>)
<code>lentry</code>	input	number of neighbours in Verlet neighbourhood list (<code>msatms</code>)
<code>neulst</code>	input	indice of first site in charge group (<code>mxneut</code>)
real arrays:		
<code>cell</code>	input	MD cell vectors (9)
<code>xxx</code>	input	x coordinates (<code>mxatms</code>)
<code>yyy</code>	input	y coordinates (<code>mxatms</code>)
<code>zzz</code>	input	z coordinates (<code>mxatms</code>)
<code>xdf</code>	work	(<code>mxxdf</code>)
<code>ydf</code>	work	(<code>mxxdf</code>)
<code>zdf</code>	work	(<code>mxxdf</code>)

8.1.143.5 Comments

8.1.144 QSHAKE**8.1.144.1 Header records**

```

      subroutine qshake
      x  (newstep,safe,lshmov,idnode,imcon,mxnode,natms,
      x  nscons,tolnce,tstep,vircon,lashap,lishap,listcon,
      x  listme,listot,lstfrz,lstbod,lstgtp,lstcsit,
      x  buffer,cell,dxt,dxx,dyt,dyy,dzt,dzz,prmcon,txx,
      x  tyy,tzz,weight,xxt,xxx,yyt,yyy,zzt,zzz,fx,fy,
      x  fzz,stress,q0,q1,q2,q3,gxx,gyy,gzz,gmass,rotinx,
      x  rotiny,rotinz,redmass)
c
c*****
c
c      dl_poly subroutine for applying bond constraint corrections after
c      atomic integration. Assumes rigid bodies connected by constraints
c      If this is not so use rdshake_1 instead
c      Must be used in conjunction with integration algorithms
c
c      copyright - daresbury laboratory 1995
c      author    - t. forester june 1995
c
c*****
c

```

8.1.144.2 Function

Applies bond constraint corrections to atomic forces after “unconstrained” integration. This routine has the capability to handle rigid bodies linked by constraint bonds.

8.1.144.3 Dependencies

- GDSUM
- GSTATE
- IMAGES
- SHMOVE
- SPLICE

8.1.144.4 Arguments

logicals:

newstep input flag for first pass of constraint cycle

safe	output	flag for convergence of SHAKE
lshmov	input	flag for shared atoms in bond constraints
integers:		
idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
nscons	input	number of constraints on current node
reals:		
tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircon	output	constraint virial
integer arrays:		
lashap	input	“last-atom” marker for inter-processor communications (mxproc)
lishap	input	“shared atom” list (mxlshp)
listcon	input	Bond constraint bookkeeping list (mxcons , 3)
listme	input	“shared atoms” register (mxatms)
listot	input	global “shared atoms” register (mxatms)
lstfrz	input	key for frozen atoms (mxatms)
lstbod	input	pointer from site index to rigid unit index (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstcsit	input	indices of rigid unit sites in constraints (2*mxcons)
real arrays:		
buffer	work	work array (mxbuff)
cell	input	MD cell vectors (9)
dxt	work	x component of bond vector during SHAKE iteration (mxcons)
dxx	input	x component of bond vector before unconstrained integration (mxcons)
dyt	work	y component of bond vector during SHAKE iteration (mxcons)
dyy	input	y component of bond vector before unconstrained integration (mxcons)
dzt	work	z component of bond vector during SHAKE iteration (mxcons)
dzz	input	z component of bond vector before unconstrained integration (mxcons)
prmcon	input	constraint distances (mxtcon)
txx	work	x component of coordinate transfer buffer (mxatms)
tyy	work	y component of coordinate transfer buffer (mxatms)
tzz	work	z component of coordinate transfer buffer (mxatms)
weight	input	atomic masses (mxatms)
xxt	work	x component of SHAKE correction (mxatms)

xxx	input	x coordinates (mxatms)
yyt	work	y component of SHAKE correction (mxatms)
yyy	input	y coordinates (mxatms)
zzt	work	z component of SHAKE correction (mxatms)
zzz	input	z coordinates (mxatms)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
stress	in/out	stress tensor (9)
q0	input	first component of quaternion vector (mxgrp)
q1	input	second component of quaternion vector (mxgrp)
q2	input	third component of quaternion vector (mxgrp)
q3	input	fourth component of quaternion vector (mxgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
gmass	input	rigid unit mass (mxungp)
rotinx	input	rotational inertia about x local body axis (mxungp,2)
rotiny	input	rotational inertia about y local body axis (mxungp,2)
rotinz	input	rotational inertia about z local body axis (mxungp,2)
redmass	work	effective reduced mass along constraint bond (mxcons)

8.1.144.5 Comments

This subroutine computes the SHAKE corrections to the *forces* after unconstrained integration. The corrections to velocity and positions are obtained by *reintegrating* the equations of motion with the new forces.

The routine RDSHAKE_1 performs the SHAKE correction to *positions* when the system does not contain rigid bodies linked by bond constraints.

8.1.145 QUATBOOK**8.1.145.1 Header records**

```

      subroutine quatbook
      x  (idnode,imcon,mxnode,natms,ngroup,ntpmls,ntfree,
      x  degfre,degrot,numgsit,numgrp,nummols,numsit,
      x  lstme,lstfrz,listyp,lstfre,lstgot,lstgtp,
      x  lstgst,lstrgd,lstbod,xxx,yyy,zzz,weight,q0,q1,
      x  q2,q3,cell,buffer,gxx,gyy,gzz,gcmx,gcmx,gcmz,
      x  xxt,yyt,zzt,txx,tyy,tzz,gmass,rotinx,rotiny,
      x  rotinz)

C*****
C
C      DLPOLY subroutine for setting up bookkeeping for rigid bodies
C
C      parallel replicated data version : block data
C
C      copyright daresbury laboratory 1993
C      author      t.forester october 1993
C      amended     t.forester dec 1994 : block data
C
C*****

```

8.1.145.2 Function

Bookkeeping for rigid body units: finds centre of mass, principal axis system, moments of inertia and orientation (represented by a quaternion vector).

8.1.145.3 Dependencies

ERROR	GDSUM	GSTATE	IMAGES
INVERT	JACOBI	MERGE	MERGE4
SHELLSORT			

8.1.145.4 Arguments**integers:**

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
natms	input	total number of atoms in the MD cell
ngroup	output	number of rigid body units in system
ntpmls	input	number of unique types of molecule

ntfree	output	number of sites in system not in rigid body units
reals:		
degfre	output	degrees of freedom
degrot	output	rotational degrees of freedom
integer arrays:		
numgsit	input	number of sites in a rigid unit type (mxungp)
numgrp	input	number of rigid units in a molecule (mxtmls)
nummols	input	number of molecules of a given type (mxtmls)
numsit	input	number of sites in a molecule (mxtmls)
lstme	output	indice list for MERGE1 (mxatms)
lstfrz	input	key for frozen atoms (mxatms)
listyp	input	rigid unit type (mxungp)
lstfre	output	indices of non-rigid unit sites handled by current processor (mxatms)
lstgot	work	(mxatms)
lstgtp	output	pointer from rigid unit index to rigid unit type (mxgrp)
lstgst	input	site indices for rigid bodies (mxungp,mxngp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgadm)
lstbod	output	pointer from atom index to rigid unit index (mxatms)
real arrays:		
xxx	input	x coordinates (mxatms)
yyy	input	y coordinates (mxatms)
zzz	input	z coordinates (mxatms)
weight	input	atomic masses (mxatms)
q0	output	first component of quaternion vector (mxgrp)
q1	output	second component of quaternion vector (mxgrp)
q2	output	third component of quaternion vector (mxgrp)
q3	output	fourth component of quaternion vector (mxgrp)
cell	input	MD cell vectors (9)
buffer	work	(mxbuff)
gxx	output	x component of site in local body frame (mxungp,mxngp)
gyy	output	y component of site in local body frame (mxungp,mxngp)
gzz	output	z component of site in local body frame (mxungp,mxngp)
gcmx	output	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	output	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	output	z coordinate of rigid unit c.o.m. (mxgrp)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
gmass	output	rigid unit mass (mxungp)

rotinx	output	rotational inertia about x local body axis (mxungp)
rotiny	output	rotational inertia about y local body axis (mxungp)
rotinz	output	rotational inertia about z local body axis (mxungp)

8.1.145.5 Comments

The principal axes are assigned to molecular types with the components of the rotational inertia tensor $\underline{\underline{\mathbf{I}}}$ obeying: $I_{xx} \geq I_{yy} \geq I_{zz}$ (to insure all processors converge on the same axes system). Quaternions are deduced from the atomic positions. By convention the first component of the quaternion vector, **q0** is taken to be positive. A check is made that the quaternion representation reproduces the atomic positions accurately. If this is not so a call to ERROR will occur.

8.1.146 QUATQNCH**8.1.146.1 Header records**

```

      subroutine quatqnc
      x  (idnode,imcon,mxnode,natms,ngroup,lstgtp,lstrgd,numgsit,
      x  lstme,gxx,gyy,gzz,buffer,cell,xxt,yyt,zzt,gcmx,gcmx,
      x  gcmz,gmass,gvxx,gvyy,gvzz,q0,q1,q2,q3,omx,omy,omz,
      x  rotinx,rotiny,rotinz,vxx,vyy,vzz,weight,xxx,yyy,zzz)

c*****
c
c      dlpoly routine to convert atomic velocities to rigid body
c      c.o.m. and angular velocity
c
c      parallel replicated data version : block data
c
c      copyright daresbury laboratory 1993.
c      author   - t.forester nov 1993.
c      author   - t.forester dec 1994 : block data.
c
c*****

```

8.1.146.2 Function

Calculates rigid unit centre of mass and rotational velocity from atomic contributions.

8.1.146.3 Dependencies

- IMAGES
- MERGE
- MERGE1

8.1.146.4 Arguments**integers:**

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors
natms	output	total number of atoms in the MD cell
ngroup	input	number of rigid body units in system

integer arrays:

lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current

		processor (mxgadm)
numgsit	input	number of sites in a rigid unit type (mxungp)
lstme	work	indice list for MERGE1 (mxatms)
real arrays:		
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
buffer	work	(mxbuff)
cell	input	MD cell vectors (9)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
gcmx	input	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	input	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	input	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	input	rigid unit mass (mxungp)
gvxx	output	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	output	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	output	z component of rigid unit c.o.m. velocity (mxgrp)
q0	input	first component of quaternion vector (mxgrp)
q1	input	second component of quaternion vector (mxgrp)
q2	input	third component of quaternion vector (mxgrp)
q3	input	fourth component of quaternion vector (mxgrp)
omx	output	x component of angular velocity (mxgrp)
omy	output	y component of angular velocity (mxgrp)
omz	output	z component of angular velocity (mxgrp)
rotinx	input	rotational inertia about x local body axis (mxungp)
rotiny	input	rotational inertia about y local body axis (mxungp)
rotinz	input	rotational inertia about z local body axis (mxungp)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xxx	input	x coordinates (mxatms)
yyy	input	y coordinates (mxatms)
zzz	input	z coordinates (mxatms)

8.1.146.5 Comments

8.1.147 QUENCH**8.1.147.1 Header records**

```

      subroutine quench
      x  (imcon,idnode,mxnode,natms,nscons,tolnce,
      x  lashap,lishap,listcon,listme,listot,
      x  buffer,cell,dxt,dyt,dzt,uxx,uyy,uzz,vxx,vyy,vzz,
      x  weight,xxt,xxx,yyt,yyy,zzt,zzz)
c
c*****
c
c      dl_poly subroutine for quenching the bond energies in the
c      initial structure of a molecule defined by constraints
c
c      copyright - daresbury laboratory 1992
c      author w.smith november 1992
c
c*****
c

```

8.1.147.2 Function

Removes velocity component along direction of constraint bonds.

8.1.147.3 Dependencies

- ERROR
- GSTATE
- IMAGES
- SHMOVE
- SPLICE

8.1.147.4 Arguments**integers:**

imcon	input	key for periodic boundary conditions
idnode	input	identity of current node
mxnode	input	total number of nodes
natms	input	number of atoms in MD cell
nscons	input	number of constraints on current node

reals:

tolnce	input	tolerance for SHAKE algorithm

integer arrays:

lashap	input	see RDSHAKE_1 (mxproc)
lishap	input	see RDSHAKE_1 (mxlshp)
listcon	input	indices for constraint bonds (mxcons,3)
listme	input	see RDSHAKE_1 (mxatms)
listot	input	see RDSHAKE_1 (mxatms)

real arrays:

buffer	work	see MERGE (mxbuff)
cell	input	MD cell vectors (9)
dxt	work	see RDSHAKE_1 (mxcons)
dyt	work	see RDSHAKE_1 (mxcons)
dzt	work	see RDSHAKE_1 (mxcons)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
weight	input	mass (mxatms)
xxt	work	(mxatms)
xxx	input	x coordinates (mxatms)
yyt	work	(mxatms)
yyy	input	y coordinates (mxatms)
zzt	work	(mxatms)
zzz	input	z coordinates (mxatms)

8.1.147.5 Comments

8.1.148 RDF0**8.1.148.1 Header records**

```

      subroutine rdf0
      x (iatm,ik,rcut,ilist,ltype,lstvdw,rdf,rsqdf)
c
c*****
c
c      dl_poly subroutine for accumulating statistic for radial
c      distribution functions.
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1994
c      author    - t. forester    march 1994
c
c*****
c

```

8.1.148.2 Functions

Accumulates histograms for radial distribution functions.

8.1.148.3 Dependencies

- none

8.1.148.4 Arguments**integers:**

iatm	input	index of central site
ik	input	number of neighbours

reals:

rcut	input	non-bonded potential cutoff

integer arrays:

ilist	input	index of neighbours around central site (mxxdf)
ltype	input	index of atomic types (mxatms)
lstvdw	input	key for type of non-bonded potential (mxvdw)

real arrays:

rdf	in/out	histograms for rdfs (mxrdf , mxvdw)
rsqdf	input	square of interatomic distances (mxxdf)

8.1.148.5 Comments

The actual rdfs are calculated in RDF1 from data collected in this routine.

8.1.149 RDF0_SW**8.1.149.1 Header records**

```

      subroutine rdf0_sw
      x (iatm,ik,rcut,rcut2,rcut1,ilist,ltype,lstvdw,rdf,rsqdf)
c
c*****
c
c      dl_poly subroutine for accumulating statistic for radial
c      distribution functions.
c      double precision accumulators
c      version for multiple time step algorithm with
c      overlapping switching regions
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1996
c      author    - t. forester      june 1996
c
c*****
c

```

8.1.149.2 Functions

Accumulates histograms for radial distribution functions in multiple timestep algorithm.

8.1.149.3 Dependencies

- none

8.1.149.4 Arguments**integers:**

iatm	input	index of central site
ik	input	number of neighbours

reals:

rcut	input	non-bonded potential cutoff
rcut2	input	upper cutoff for region
rcut1	input	lower cutoff for region

integer arrays:

ilist	input	index of neighbours around central site (mxrdf)
ltype	input	index of atomic types (mxatms)
lstvdw	input	key for type of non-bonded potential (mxvdw)

real arrays:

rdf	in/out	histograms for rdfs (mxrdf,mxvdw)

rsqdf input square of interatomic distances (**mxxdf**)

8.1.149.5 Comments

The actual rdfs are calculated in RDF1 from data collected in this routine.

8.1.150 RDF0NEU**8.1.150.1 Header records**

```

      subroutine rdf0neu
      x  (ik,rcut,ilist,jlist,ltype,lstvdw,rdf,rsqdf)
c
c*****
c
c      dl_poly subroutine for accumulating statistic for radial
c      distribution functions.
c      double precision accumulators
c      neutral group implementation
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1994
c      author    - t. forester    march 1994
c      amended   t. forester    april 1994
c
c*****
c

```

8.1.150.2 Functions

Accumulates histograms for radial distribution functions - charge group implementation.

8.1.150.3 Dependencies

- none

8.1.150.4 Arguments**integers:**

ik	input	number of pairs

reals:

rcut	input	non-bonded potential cutoff

integer arrays:

ilist	input	first index of pair (mxxdf)
jlist	input	second index of pair (mxxdf)
ltype	input	index of atomic types (mxatms)
lstvdw	input	key for type of non-bonded potential (mxvdw)

real arrays:

rdf	in/out	histograms for rdfs (mxrdf,mxvdw)
rsqdf	input	square of interatomic distances (mxxdf)

8.1.150.5 Comments

The actual rdfs are calculated in RDF1 from data collected in this routine.

8.1.151 RDF1**8.1.151.1 Header records**

```

      subroutine rdf1
      x  (lpgr,cfgname,unqatm,idnode,mxnode,ntpatm,ntpvdw,numrdf,
      x  rcut,volm,lstvdw,dens,rdf,buffer)

c
c*****
c
c      dl_poly subroutine for calculating radial distribution functions
c      from accumulated data.
c      double precision version
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1994
c      author    - t. forester    march 1994
c
c*****
c

```

8.1.151.2 Functions

Calculates and prints radial distribution functions and integral coordination numbers.

8.1.151.3 Dependencies

- GDSUM

8.1.151.4 Arguments**logicals:**

lpgr	input	flag for printing radial distribution functions

characters:

cfgname	input	simulation header [a80]
unqatm	input	atomic type names [a8] (mxsite)

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
ntpatm	input	number of types of atoms
ntpvdw	input	number of non-bonded potentials
numrdf	input	number of timesteps used for rdf statistics

reals:

rcut	input	non-bonded potential cutoff

volm	input	volume of MD cell
integer arrays:		
lstvdw	input	key for type of non-bonded potential (mxvdw)
real arrays:		
dens	input	number density of atomic sites (mxsvdw)
rdf	input	histograms for rdfs (mxrdf, mxvdw)
buffer	work	work (mxbuff)

8.1.151.5 Comments

For non periodic systems only the number of neighbours, $n(r)$, has meaning since an arbitrary value needs to be assigned to the volume in order to compute the radial distribution functions. The average number of atoms of type B within a radius r of atoms of type A is given by

$$n(r) = 4\pi \int_0^r g(r) \rho_B r^2 dr \quad (8.1)$$

8.1.152 RDSHAKE_1**8.1.152.1 Header records**

```

      subroutine rdshake_1
      x  (safe,lshmov,idnode,imcon,mxnode,natms,nscons,
      x  tolnc,tstep,vircon,lashap,lishap,listcon,listme,
      x  listot,lstfrz,buffer,cell,dxt,dxx,dyt,dyy,dzt,dzz,
      x  prmcon,txx,tyy,tzz,weight,xxt,xxx,yyt,yyy,zzt,zzz,
      x  stress)
c
c*****
c
c      dl_poly subroutine for applying bond constraint corrections after
c      atomic integration.
c      Must be used in conjunction with integration algorithms
c
c      copyright - daresbury laboratory 1994
c      author    - w. smith august 1992.
c      t. forester march 1994.
c
c*****
c

```

8.1.152.2 Function

Applies bond constraint corrections to atomic positions after “unconstrained” integration.

8.1.152.3 Dependencies

- ERROR
- GDSUM
- GSTATE
- IMAGES
- SHMOVE
- SPLICE

8.1.152.4 Arguments

logicals:

safe	output	flag for convergence of SHAKE
lshmov	input	flag for shared bond constraints

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
nscons	input	number of constraints on current node

reals:

tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircon	output	constraint virial

integer arrays:

lashap	input	“last-atom” marker for inter-processor communications (mxproc)
lishap	input	“shared atom” list (mxlshp)
listcon	input	Bond constraint bookkeeping list (mxcons ,3)
listme	input	“shared atoms” register (mxatms)
listot	input	global “shared atoms” register (mxatms)
lstfrz	input	key for frozen atoms (mxatms)

real arrays:

buffer	work	work array (mxbuff)
cell	input	MD cell vectors (9)
dxt	work	x component of bond vector during SHAKE iteration (mxcons)
dxx	input	x component of bond vector before unconstrained integration (mxcons)
dyt	work	y component of bond vector during SHAKE iteration (mxcons)
dyy	input	y component of bond vector before unconstrained integration (mxcons)
dzt	work	z component of bond vector during SHAKE iteration (mxcons)
dzz	input	z component of bond vector before unconstrained integration (mxcons)
prmcon	input	constraint distances (mxtcon)
txx	work	x component of coordinate transfer buffer (mxatms)
tyy	work	y component of coordinate transfer buffer (mxatms)
tzz	work	z component of coordinate transfer buffer (mxatms)
weight	input	atomic masses (mxatms)
xxt	work	x component of SHAKE correction (mxatms)
xxx	in/out	x coordinates (mxatms)
yyt	work	y component of SHAKE correction (mxatms)
yyy	in/out	y coordinates (mxatms)
zzt	work	z component of SHAKE correction (mxatms)
zzz	in/out	z coordinates (mxatms)
stress	in/out	stress tensor (9)

8.1.152.5 Comments

This subroutine computes the SHAKE corrections to the *position* along with the constant virial after unconstrained integration . The corrections to velocity and force are done in the integration routine *e.g.* NVE_1.

The routine QSHAKE performs the SHAKE correction to the *forces* when the system contains rigid bodies linked by bond constraints.

8.1.153 RESPA_R**8.1.153.1 Header records**

```

      subroutine respa_r
      x  (safe,lshmov,imcon,idnode,mxnode,natms,nscons,ntcons,
      x  tolnc,tstep,vircon,lshap,lashap,lstfrz,listcon,listme,
      x  listot,buffer,cell,dxt,dxx,dyt,dyy,dzt,dzz,fx,fyy,fzz,
      x  prmcon,stress,txx,tyy,tzz,uxx,uyy,uzz,vxx,vyy,vzz,weight,
      x  xdf,xxt,xxx,ydf,yyt,yyy,zdf,zzt,zzz)

C*****
C
C      dlpolym routine to integrate atomistic equations of motion
C      using RESPA algorithm with bond constraints
C
C      parallel replicated data version : block data
C
C      copyright daresbury laboratory 1996
C      author      t.forester june 1996
C
C*****

```

8.1.153.2 Function

The RESPA algorithm for integrating positions from Newton's equations of motion. Thermostats and barostats are disabled.

8.1.153.3 Dependencies

ERROR	GDSUM	IMAGES	MERGE
RDSHAKE_1			

8.1.153.4 Arguments**logicals:**

safe	output	flag for convergence of SHAKE
lshmov	input	flag for shared bond constraints

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
nscons	input	number of constraints on current processor
ntcons	input	number of constraints in system

reals:

tolnce	input	tolerance for SHAKE algorithm
tstep	input	simulation time-step, Δt
vircon	output	constraint virial

integer arrays:

lashap	input	see RDSHAKE_1 (mxproc)
lishap	input	see RDSHAKE_1 (mxlshp)
listcon	input	indices for constraint bonds (mxcons , 3)
listme	input	see RDSHAKE_1 (mxatms)
listot	input	see RDSHAKE_1 (mxatms)
lstfrz	input	key for frozen atoms (mxatms)

real arrays:

buffer	work	see MERGE (mxbuff)
cell	input	MD cell vectors (9)
dxt	work	see RDSHAKE_1 (mxcons)
dxx	work	see RDSHAKE_1 (mxcons)
dxt	work	see RDSHAKE_1 (mxcons)
dyy	work	see RDSHAKE_1 (mxcons)
dzt	work	see RDSHAKE_1 (mxcons)
dzz	work	see RDSHAKE_1 (mxcons)
fx	in/out	x component of force (mxatms)
fy	in/out	y component of force (mxatms)
fz	in/out	z component of force (mxatms)
prmcon	input	constraint distances (mxtcon)
stress	in/out	stress tensor (9)
tx	work	(mxatms)
ty	work	(mxatms)
tz	work	(mxatms)
ux	work	(mxatms)
uy	work	(mxatms)
uz	work	(mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)
xdf	work	(mxxdf)
xxt	work	(mxatms)
xxx	in/out	x coordinates (mxatms)
ydf	work	(mxxdf)
yxt	work	(mxatms)
yyy	in/out	y coordinates (mxatms)
zdf	work	(mxxdf)
zzt	work	(mxatms)
zzz	in/out	z coordinates (mxatms)

8.1.153.5 Comments

Uses a RESPA integration scheme. This routine must be used in conjunction with the velocity integration routine RESPA_V.

As supplied this routine does not cater for

- Rigid bodies (except those defined using bond constraints)
- Barostats or thermostats
- Potential of mean force (pmf) constraints.

8.1.154 RESPA_V**8.1.154.1 Header records**

```

      subroutine respa_v
      x  (idnode,imcon,mxnode,natms,tstep,
      x  fxx,fyy,fzz,vxx,vyy,vzz,weight)

c
c*****
c
c      dl_poly subroutine for advancing velocities in
c      the RESPA formulation of molecular dynamics.
c
c      parallel replicated data version : block data
c
c      copyright - daresbury laboratory 1995
c      author    - t.forester feb 1995
c
c*****
c

```

8.1.154.2 Function

The RESPA algorithm for integrating Newton's equations of motion. Thermostats and barostats are disabled.

8.1.154.3 Dependencies

ERROR	GDSUM	IMAGES	MERGE
RDSHAKE_1			

8.1.154.4 Arguments**integers:**

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell

reals:

tstep	input	simulation time-step, Δt

real arrays:

fxx	input	x component of force (mxatms)
fyy	input	y component of force (mxatms)
fzz	input	z component of force (mxatms)

vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
weight	input	atomic masses (mxatms)

8.1.154.5 Comments

Uses a RESPA integration scheme. This routine must be used in conjunction with the coordinate integration routine RESPA_V.

As supplied this routine does not cater for

- Rigid bodies (except those defined using bond constraints)
- Barostats or thermostats
- Potential of mean force (pmf) constraints.

8.1.155 RESULT**8.1.155.1 Header records**

```

      subroutine result
      x  (lgofr,lpgr,lzden,cfgrname,atmnam,unqatm,idnode,imcon,
      x  keyens,mxnode,natms,nzden,nstep,ntpatm,ntpvdw,numacc,
      x  numrdf,chip,chit,conint,rcut,tstep,volm,lstvdw,buffer,
      x  cell,dens,eta,fx,fy,fzz,ravval,rdf,ssqval,stkval,
      x  stpval,sumval,vxx,vyy,vzz,xxx,yyy,zumval,zzz,xx0,yy0,
      x  zz0,zdens,xs,ys,zs)
c
c*****
c
c      dl_poly subroutine for writing simulation summary and
c      saving the restart data
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith dec 1992.
c
c
c*****
c

```

8.1.155.2 Function

Writes out final results to OUTPUT file and arranges storing of final configuration.

8.1.155.3 Dependencies

- REVIVE
- RDF1
- ZDEN1
- TIMCHK

8.1.155.4 Arguments**logicals:**

lgofr	input	flag for calculating radial distribution functions
lpgr	input	flag for printing radial distribution functions
lzden	input	flag for calculating z density profile

characters:

cfgrname	input	configuration name [a80]
atmnam	input	site labels [a8] (mxatms)

unqatm	input	unique atom labels [a8] (mxsite)
integers:		
idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
keyens	input	ensemble key
mxnode	input	total number of processors in use
natms	input	number of atoms in MD cell
nzden	input	number of configurations used for z density profiles
nstep	input	number of completed MD steps
ntpatm	input	number of unique atom types
ntpvdw	input	number of non-bonded potentials
numacc	input	number of steps used to accumulate statistics
numrdf	input	number of configurations used for rdf histograms
reals:		
chip	input	friction coefficient for Hoover barostat
chit	input	friction coefficient for Hoover thermostat
conint	input	time integral of friction coefficient
rcut	input	radial cut off
tstep	input	simulation time-step, Δt
volm	input	volume of cell
integer arrays:		
lstvdw	input	pointer from potential index to parameter table (mxvdw)
real arrays:		
buffer	work	work array (mxbuff)
cell	input	MD cell vectors (9)
dens	input	number density of unique atom types (mxsvdw)
eta	input	cell scaling tensor (9)
fxx	input	x component of force (mxatms)
fyy	input	y component of force (mxatms)
fzz	input	z component of force (mxatms)
ravval	input	rolling averages (mxnstk)
rdf	input	radial distribution histograms (mxrdf,mxvdw)
ssqval	in/out	rms fluctuations (mxnstk)
stkval	input	statistics file for rolling averages (mxstak,mxnstk)
stpval	input	statistics file (mxnstk)
sumval	input	statistics file (mxnstk)
vxx	input	x component of velocity (mxatms)
vyy	input	y component of velocity (mxatms)
vzz	input	z component of velocity (mxatms)
xxx	input	x coordinate (mxatms)
yyy	input	y coordinate (mxatms)
zumval	input	statistics file (mxnstk)
zzz	input	z coordinate (mxatms)
xx0	input	x component of atomic displacement (mxatms)

yy0	input	y component of atomic displacement (mxatms)
zz0	input	z component of atomic displacement (mxatms)
zdens	input	z density histograms (mxrdf,mxsvdw)
xxs	input	x component of atomic position at time origin (mxatms)
yyS	input	y component of atomic position at time origin (mxatms)
zzs	input	z component of atomic position at time origin (mxatms)

8.1.155.5 Comments

8.1.156 REVIVE**8.1.156.1 Header records**

```

      subroutine revive
      x  (lgofr,lzden,cfpname,atmnam,idnode,imcon,mxnode,
      x  natms,nstep,nzden,numacc,numrdf,chip,chit,conint,
      x  tstep,buffer,cell,fx,fy,fz,ravval,rdf,ssqval,
      x  stkval,stpval,sumval,vxx,vyy,vzz,xxx,yyy,zumval,
      x  zzz,xx0,yy0,zz0,zdens,xs,ys,zs,eta)
c
c*****
c
c      dl_poly subroutine for writing restart files at job termination
c      or at selected intervals in simulation
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith dec 1992.
c
c*****
c

```

8.1.156.2 Function

Dumps restart data to disk i.e. writes the REVIVE file.

8.1.156.3 Dependencies

- ERROR
- GDSUM
- MERGE

8.1.156.4 Arguments**logicals:**

lgofr	input	flag for radial distribution functions
lzden	input	flag for calculating z density profile

characters:

cfpname	input	configuration name [a80]
atmnam	input	atomic labels [a8] (mxatms)

integers:

idnode	input	identity of current processor
imcon	input	periodic boundary key
mxnode	input	total number of processors

<code>natms</code>	input	number of atoms in MD cell
<code>nstep</code>	input	time-step counter
<code>nzden</code>	input	number of configurations used for z density profiles
<code>numacc</code>	input	number of steps used for statistics
<code>numrdf</code>	input	number of steps used for rdfs
reals:		
<code>chip</code>	input	friction coefficient for Hoover barostat
<code>chit</code>	input	friction coefficient for Hoover thermostat
<code>conint</code>	input	time integral of <code>chit</code> and <code>chip</code>
<code>tstep</code>	input	simulation time-step, Δt
real arrays:		
<code>buffer</code>	work	work array (<code>mxbuff</code>)
<code>cell</code>	input	cell vectors (9)
<code>fx</code>	input	x component of force (<code>mxatms</code>)
<code>fy</code>	input	y component of force (<code>mxatms</code>)
<code>fz</code>	input	z component of force (<code>mxatms</code>)
<code>ravval</code>	input	statistics array (<code>mxnstk</code>)
<code>rdf</code>	input	histograms of rdfs (<code>mxrdf, mxvdw</code>)
<code>ssqval</code>	input	statistics array (<code>mxnstk</code>)
<code>stkval</code>	input	statistics array (<code>mxstak, mxnstk</code>)
<code>stpval</code>	input	statistics array (<code>mxnstk</code>)
<code>sumval</code>	input	statistics array (<code>mxnstk</code>)
<code>vxx</code>	input	x component of velocity (<code>mxatms</code>)
<code>vyy</code>	input	y component of velocity (<code>mxatms</code>)
<code>vzz</code>	input	z component of velocity (<code>mxatms</code>)
<code>xxx</code>	input	x coordinate (<code>mxatms</code>)
<code>yyy</code>	input	y coordinate (<code>mxatms</code>)
<code>zumval</code>	input	statistics array (<code>mxnstk</code>)
<code>zzz</code>	input	z coordinate (<code>mxatms</code>)
<code>xx0</code>	input	increment in x coordinate (<code>mxatms</code>)
<code>yy0</code>	input	increment in y coordinate (<code>mxatms</code>)
<code>zz0</code>	input	increment in z coordinate (<code>mxatms</code>)
<code>zdens</code>	input	z density histograms (<code>mxrdf, mxsvdw</code>)
<code>xxs</code>	input	x component of atomic position at time origin (<code>mxatms</code>)
<code>yyz</code>	input	y component of atomic position at time origin (<code>mxatms</code>)
<code>zzs</code>	input	z component of atomic position at time origin (<code>mxatms</code>)
<code>eta</code>	input	cell scaling tensor (9)

8.1.156.5 Comments

8.1.157 SCDENS**8.1.157.1 Header records**

```

      subroutine scdens
      x  (idnode,imcon,mxnode,ntpvdw,natms,rvdw,dlrpot,engmet,rsrden,
      x  ilist,lentry,list,lstvdw,ltpvdw,ltype,buffer,cell,rho,vvv,
      x  rsqdf,xdf,xxx,ydf,yyy,zdf,zzz,elrcm)
c
c*****
c
c      dl_poly subroutine for calculating local density in metals
c      using the verlet neighbour list and sutton-chen potentials
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1995
c      author    - w. smith june 1995
c
c*****
c

```

8.1.157.2 Function

Calculates local density for Sutton-Chen potentials and part of the energy of interaction.

8.1.157.3 Dependencies

- DENLOC
- GDSUM
- IMAGES

8.1.157.4 Arguments**integers:**

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
ntpvdw	input	number of Van der Waals potentials
natms	input	total number of atoms in the MD cell

reals:

rvdw	input	potential cut-off radius
dlrpot	input	distance increment for short-range potential tabulation

engmet	output	potential energy
rsrden	output	reciprocal square root of Sutton Chen density
integer arrays:		
ilist	input	index of neighbours around central site (mxxdf)
lentry	input	number of neighbours in Verlet neighbourhood list (msatms)
list	input	the Verlet neighbourhood list (msatms , mxlist)
lstvdw	input	pointer from potential index to parameter table (mxvdw)
ltpvdw	input	key for non-bonded potentials (mxvdw)
ltype	input	index of atomic types (mxatms)
real arrays:		
buffer	work	(mxbuff)
cell	input	MD cell vectors (9)
rho	output	Sutton Chen particle density (mxatms)
vvv	input	interpolation table for non-bonded potentials (mxgrid , mxvdw)
rsqdf	work	square of interatomic distances (mxxdf)
xdf	work	x component of interatomic vector (mxxdf)
xxx	input	x coordinates (mxatms)
ydf	work	y component of interatomic vector (mxxdf)
yyy	input	y coordinates (mxatms)
zdf	work	z component of interatomic vector (mxxdf)
zzz	input	z coordinates (mxatms)
elrcm	input	long range correction to Sutton Chen energy (2)

8.1.157.5 Comments

This routine only calculates the density part of the Sutton-Chen interaction. The remainder is calculated in SUTTCHEM.

8.1.158 SCL_CSUM**8.1.158.1 Header records**

```
subroutine scl_csum(nnn,tot,aaa)
```

```
c*****
c
c      dl_poly subroutine to calculate the scalar sum of the elements
c      of a complex array
c
c      copyright daresbury laboratory 1998
c      author w.smith july 1998
c
c*****
```

8.1.158.2 Function

This routine performs a scalar sum on the contents of a complex array, and returns the value.

8.1.158.3 Dependencies

- NONE

8.1.158.4 Arguments**integers:**

nnn	input	number of array elements required

complex:

tot	output	complex result

complex arrays:

aaa	input	complex array (*)

8.1.158.5 Comments

It is used to avoid the loop nesting needed for multidimensional arrays.

8.1.159 SET_BLOCK**8.1.159.1 Header records**

```
subroutine set_block(nnn,ccc,aaa)
```

```
c*****
c
c      dl_poly subroutine to initialise an array to a single value
c
c      copyright daresbury laboratory 1998
c      author w.smith july 1998
c
c*****
```

8.1.159.2 Function

This routine initialises any given real array to some constant value specified in the calling sequence.

8.1.159.3 Dependencies

- NONE

8.1.159.4 Arguments

integers:

nnn	input	number of array elements required

reals:

ccc	input	required constant value

complex arrays:

aaa	output	array to be initialised (*)

8.1.159.5 Comments

It is used to avoid the loop nesting needed for multidimensional arrays.

8.1.160 SHELLSORT**8.1.160.1 Header records**

```
subroutine shellsort(n,list)
```

```
c*****
c
c      dlpoly shell sort routine.
c      Sorts an array of integers into ascending order
c
c      copyright daresbury laboratory 1993
c      author - t.forester   november 1993
c
c*****
```

8.1.160.2 Function

Sorts elements of an integer array into ascending order.

8.1.160.3 Dependencies

- none

8.1.160.4 Arguments

integers:

n	input	number of items to be sorted

integer arrays:

list	in/out	array to be sorted (*)

8.1.160.5 Comments

Uses the Shellsort algorithm.

8.1.161 SHLFRC**8.1.161.1 Header records**

```

      subroutine shlfrc
      x  (idnode,imcon,mxnode,ntshl,engshl,virshl,listshl,cell,
      x  fxx,fyy,fzz,prmschl,xxx,yyy,zzz,xdx,ydy,zdz,stress)
c
c*****
c
c      dl_poly subroutine for calculating shell model spring energy and
c      force terms in molecular dynamics.
c
c      copyright - daresbury laboratory 1994
c      author    - w. smith          july 1994
c
c*****
c

```

8.1.161.2 Function

Calculates the the core-shell spring energy in shell model potential.

8.1.161.3 Dependencies

- ERROR
- GDSUM
- IMAGES

8.1.161.4 Arguments**integers:**

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors in use
ntshl	input	total number of core-shell units in system

reals:

engshl	output	core-shell energy
virshl	output	core-shell virial

integer arrays:

listshl	input	indices for core-shell interactions on current processor (mxshl,3)

real arrays:

cell	input	MD cell vectors (9)
fxx	in/out	x component of force (mxatms)

fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
prmshl	input	core-shell potential parameters (mxtshl)
xxx	input	x coordinates (mxatms)
yyy	input	y coordinates (mxatms)
zzz	input	z coordinates (mxatms)
xdf	work	(mxxdf)
ydf	work	(mxxdf)
zdf	work	(mxxdf)
stress	in/out	stress tensor (9)

8.1.161.5 Comments

8.1.162 SHLMERGE**8.1.162.1 Header records**

```

      subroutine shlmerge
      x (idnode,mxnode,ntshl,listshl,xxx,yyy,zzz,buffer)
c
c*****
c
c      dl_poly subroutine for merging core-shell velocity data
c      to restore data replication on all nodes
c
c      copyright - daresbury laboratory 1993
c      author    - w. smith february 1993
c      MPI version - w. smith june 1995
c      CPP version - w. smith june 1995
c
c*****
c

```

8.1.162.2 Function

Merges core-shell (velocity) arrays across all processors after application of SHLQNCH.

8.1.162.3 Dependencies

- standard send/receive communication routines
- ERROR
- GSYNC

8.1.162.4 Arguments**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors in use
ntshl	input	total number of core-shell units in system

integer arrays:

listshl	input	indices for core-shell interactions on current processor (mxshl,3)

real arrays:

xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
buffer	work	(mxbuff)

8.1.162.5 Comments

8.1.163 SHLQNC**8.1.163.1 Header records**

```

      subroutine shlqnc
      x (idnode,mxnode,ntshl,temp,listshl,weight,vxx,vyy,vzz,buffer)
c
c*****
c
c      dl_poly subroutine for quenching the internal bond energies
c      in ions defined by shell model
c
c      copyright - daresbury laboratory 1994
c      author w.smith july 1994
c
c*****
c

```

8.1.163.2 Function

Scales core-shell spring temperature to 1 K.

8.1.163.3 Dependencies

- SHLMERGE

8.1.163.4 Arguments**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors in use
ntshl	input	total number of core-shell units in system

reals:

temp	input	required system temperature

integer arrays:

listshl	input	indices for core-shell interactions on current processor (mxshl,3)

real arrays:

weight	input	atomic masses (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
buffer	work	(mxbuff)

8.1.163.5 **Comments**

8.1.164 SHMOVE**8.1.164.1 Header records**

```

      subroutine shmove
      x  (idnode,mxnode,natms,lashap,lishap,xxt,yyt,zzt,
      x  txx,tyy,tzz,buffer)
c
c*****
c
c      dl_poly subroutine for passing coordinate updates between
c      nodes during the shake iteration cycle
c
c      parallel replicated data algorithm
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith august 1992.
c      MPI version - t.forester may 1995
c      CPP version - w.smith may 1995
c
c*****
c

```

8.1.164.2 Function

Passes coordinate updates between nodes during the SHAKE iteration cycle.

8.1.164.3 Dependencies

- standard send/receive communication routines
- GSYNC

8.1.164.4 Arguments**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors
natms	input	number of atoms in MD cell

integer arrays:

lashap	input	Outgoing transfer buffer (mxproc)
lishap	input	Incoming transfer buffer (mxlshp)

real arrays:

xxt	in/out	increment to x coordinate (mxatms)
yyt	in/out	increment to y coordinate (mxatms)
zzt	in/out	increment to z coordinate (mxatms)

txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
buffer	work	communications buffer (mxbuff)

8.1.164.5 Comments

C pre-processing is required to select the appropriate communication routines. Messing with this routine may seriously damage the health of DL_POLY_2 .

8.1.165 SIMDEF**8.1.165.1 Header records**

```

      subroutine simdef
      x  (lfcap,lgofr,lnsq,loptim,lpgr,ltraj,ltscal,lzeql,lzden,
      x  sysname,idnode,mxnode,intsta,istray,keyens,keyfce,keyres,
      x  keytrj,kmax1,kmax2,kmax3,multt,nstack,nstbgr,nstbpo,
      x  nstbts,nsteql,nstraj,nstrun,alpha,delr,epsq,fmax,
      x  press,quattol,rcut,rprim,rvdw,taup,taut,temp,timcls,
      x  timjob,tolnce,tstep)
c
c*****
c
c      dl_poly subroutine for reading in the simulation control
c      parameters
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith july 1992.
c
c      modified
c      author    - t. forester      may 1993
c      amended   - t.forester      sept 1994 - dl_poly_1.1
c      amended   - t.forester      nov 1994 - macro version
c      amended   - w.smith         dec 1994 - t3d adaptation
c
c*****
c

```

8.1.165.2 Function

Reads in control variables for the job from the CONTROL file.

8.1.165.3 Dependencies

- DCELL
- ERROR
- GETREC
- LOWCASE
- STRIP

8.1.165.4 Arguments**logicals:**

lfcap	output	flag for capping forces during equilibration
lgofr	output	flag for calculating radial distribution functions
lnsq	output	flag for “all-pairs” algorithm
loptim	output	flag for structure optimization
lpgr	output	flag for printing radial distribution functions
ltraj	output	flag for dumping trajectory data
ltscal	output	flag for temperature scaling
lzeql	output	flag for accumulating statistics during equilibration
lzden	output	flag for calculating z density profile

characters:

sysname	output	system name [a80]

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
intsta	output	interval for accumulating statistics
istraj	output	interval for dumping trajectories
keyens	output	key for selecting ensemble
keyfce	output	key for non-bonded force field
keyres	output	key for restart of MD run
keytrj	output	key for trajectory information
kmax1	output	number of reciprocal space vectors in <i>a</i> direction
kmax2	output	number of reciprocal space vectors in <i>b</i> direction
kmax3	output	number of reciprocal space vectors in <i>c</i> direction
multt	output	size of multiple timestep interval
nstack	output	size of stack for rolling averages
nstbgr	output	interval for collecting rdf data
nstbpo	output	interval for printing out statistical data
nstbts	output	interval for temperature scaling in equilibration mode
nsteql	output	number of timesteps for equilibration
nstraj	output	timestep at which dumping of trajectory information begins
nstrun	output	number of timesteps for MD run

reals:

alpha	output	Ewald convergence variable
delr	output	border width for Verlet list
epsq	output	relative dielectric constant
fmax	output	force capping limit
press	output	simulation pressure
quattol	output	tolerance for quaternion algorithm
rcut	output	(electrostatic) potential cut-off radius
rprim	output	radius of primary cut-off shell
rvdw	output	Van der Waals potential cut-off radius

taup	output	coupling parameter for barostat
taut	output	coupling parameter for thermostat
temp	output	simulation temperature
timcls	output	time, in seconds, allocated for closing the job
timjob	output	total time, in seconds, allocated for the job
tolnce	output	tolerance for SHAKE algorithm
tstep	output	simulation timestep, Δt

8.1.165.5 Comments

See section 4.1.1 for a description of the CONTROL file.

8.1.166 SIMDEF_RESPA**8.1.166.1 Header records**

```

      subroutine simdef_respa
      x  (lfcap,lgofr,loptim,lpgr,ltraj,ltscal,lzeql,lzden,
      x  sysname,idnode,mxnode,intsta,istray,keyens,keyfce,keyres,
      x  keytrj,kmax1,kmax2,kmax3,mult0,mult1,mult2,nstack,nstbgr,
      x  nstbpo,nstbts,nsteql,nstraj,nstrun,alpha,delr,epsq,fmax,
      x  press,rcut,rcut1,rcut2,rvdw,temp,timcls,
      x  timjob,tolnce,tstep,swlen)

c
c*****
c
c      dl_poly subroutine for reading in the simulation control
c      parameters - respa version
c
c      copyright - daresbury laboratory 1996
c      simdef author   - w. smith july 1992.
c
c      modified for respa
c      t.forester                june 1996
c
c*****
c

```

8.1.166.2 Function

Reads in control variables for the job from the CONTROL file.

8.1.166.3 Dependencies

DCELL	ERROR	GETREC	LOWCASE
STRIP			

8.1.166.4 Arguments**logicals:**

lfcap	output	flag for capping forces during equilibration
lgofr	output	flag for calculating radial distribution functions
loptim	output	flag for structure optimization
lpgr	output	flag for printing radial distribution functions
ltraj	output	flag for dumping trajectory data
ltscal	output	flag for temperature scaling

lzeql	output	flag for accumulating statistics during equilibration
lzden	output	flag for calculating z density profile
characters:		
sysname	output	system name [a80]
integers:		
idnode	input	identity of current processor
mxnode	input	total number of processors in use
intsta	output	interval for accumulating statistics
istraj	output	interval for dumping trajectories
keyens	output	key for selecting ensemble
keyfce	output	key for non-bonded force field
keyres	output	key for restart of MD run
keytrj	output	key for trajectory information
kmax1	output	number of reciprocal space vectors in <i>a</i> direction
kmax2	output	number of reciprocal space vectors in <i>b</i> direction
kmax3	output	number of reciprocal space vectors in <i>c</i> direction
mult0	output	multiple timestep factor for inner shell
mult1	output	multiple timestep factor for first shell
mult2	output	multiple timestep factor for second shell
nstack	output	size of stack for rolling averages
nstbgr	output	interval for collecting rdf data
nstbpo	output	interval for printing out statistical data
nstbts	output	interval for temperature scaling in equilibration mode
nsteql	output	number of timesteps for equilibration
nstraj	output	timestep at which dumping of trajectory information begins
nstrun	output	number of timesteps for MD run
reals:		
alpha	output	Ewald convergence variable
delr	output	border width for Verlet list
epsq	output	relative dielectric constant
fmax	output	force capping limit
press	output	simulation pressure (not active)
rcut	output	(electrostatic) potential cut-off radius
rcut1	output	radius of first shell cut-off
rcut2	output	radius of second shell cut-off
rvdw	output	Van der Waals potential cut-off radius
temp	output	simulation temperature
timcls	output	time, in seconds, allocated for closing the job
timjob	output	total time, in seconds, allocated for the job
tolnce	output	tolerance for SHAKE algorithm
tstep	output	simulation timestep, Δt
swlen	output	switching length

8.1.166.5 Comments

See section 4.1.1 for a description of the CONTROL file. This subroutine is a modified version of SIMDEF for use with the RESPA integration scheme.

8.1.167 SPL_CEXP**8.1.167.1 Header records**

```
subroutine spl_cexp(ndiv1,ndiv2,ndiv3,ww1,ww2,ww3)
```

```
c*****
c
c      dl-poly routine to create complex exponential arrays for
c      b-splines
c
c      copyright daresbury laboratory 1998
c
c      author w smith oct 1998
c
c*****
```

8.1.167.2 Function

This routine calculates all the complex exponentials required by the SPME method.

8.1.167.3 Dependencies

- NONE

8.1.167.4 Arguments

integers:

ndiv1	input	number of x array elements required
ndiv2	input	number of y array elements required
ndiv3	input	number of z array elements required

complex arrays:

ww1	input	x complex exponential array (*)
ww2	input	y complex exponential array (*)
ww3	input	z complex exponential array (*)

8.1.167.5 Comments

8.1.168 SPLICE**8.1.168.1 Header records**

```

      subroutine splice
      x (idnode,mxnode,natms,listme,listot,xxx,yyy,zzz,buffer)
c
c*****
c
c      dl_poly subroutine for splicing together coordinate arrays
c      across a number of processors during shake algorithm
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1993
c      author    - w. smith      march 1993
c
c      second version of splice
c
c*****
c

```

8.1.168.2 Function

Updates coordinates on all processors of those sites involved in constraint bonds once the SHAKE cycle has been completed.

8.1.168.3 Dependencies

- ERROR
- GDSUM

8.1.168.4 Arguments**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors
natms	input	number of sites involved in constraints

integer arrays:

listme	input	“shared atoms” register (mxatms)
listot	input	global “shared atoms” register (mxatms)

real arrays:

xxx	in/out	x coordinate array (natms)
yyy	in/out	y coordinate array (natms)
zzz	in/out	z coordinate array (natms)

`buffer` `work` work array for global sum (`mxbuff`)

8.1.168.5 Comments

The arrays `listme` and `listot` are created in PASSCON. Only sites involved in bond constraints are updated.

8.1.169 SRFRCE**8.1.169.1 Header records**

```

      subroutine srfrce
      x  (iatm,ik,engsrp,virsrp,rcut,dlrpot,ilist,ltype,lstvdw,
      x  ltpvdw,rsqdf,xdf,ydf,zdf,fx,fyy,fzz,vvv,ggg,stress)
c
c*****
c
c      dl_poly subroutine for calculating short range force and
c      potential energy terms using verlet neighbour list
c
c      parallel replicated data version
c
c      3 point interpolation
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith      march 1992
c      T3D optimised version. t.forester july 1994
c
c      version 3
c      author    - t. forester    june 1993
c      stress tensor added t.forester may 1994
c
c*****
c

```

8.1.169.2 Function

Calculates non-bonded forces and potential energies using the Verlet neighbour list.

8.1.169.3 Dependencies

- none

8.1.169.4 Arguments**integers:**

iatm	input	index of the central site
ik	input	number of entries in the Verlet list

reals:

engsrp	output	the non-bonded potential energy
virsrp	output	the non-bonded virial
rcut	input	the non-bonded potential cutoff
dlrpot	input	tabulation interval for interpolation tables

integer arrays:

ilist	input	the Verlet neighbour list (mxlist)
ltype	input	atom type (mxatms)
lstvdw	input	index number of non-bonded potential (mxvdw)
ltpvdw	input	key for non-bonded potentials (mxvdw)

real arrays:

rsqdf	input	square of inter site vector length (mxxdf)
xdf	input	x coordinate for inter site vector (mxxdf)
ydf	input	y coordinate for inter site vector (mxxdf)
zdf	input	z coordinate for inter site vector (mxxdf)
fx	output	x component of force (mxatms)
fy	output	y component of force (mxatms)
fz	output	z component of force (mxatms)
vvv	input	interpolation table for potential energy (mxgrid,mxvdw)
ggg	input	interpolation table for force (mxgrid,mxvdw)
stress	in/out	stress tensor (9)

8.1.169.5 Comments

The algorithm requires the interpolation arrays **vvv** and **ggg**. The arrays are generated in the subroutine **FORGEN**.

In the default version of **SRFRCE** a three point interpolation scheme is used to approximate the functions between tabulation points and the arrays **vvv** and **ggg** are tabulated as functions of r . Other versions using different interpolation schemes are available. These include tabulations in r^2 or with 4-point interpolation formulas. These are indicated in the *source* directory by the additional qualifiers **_RSQ** and **_4PT** attached to the file names. See the notes on subroutine **FORGEN** for a discussion of relative merits of the different interpolation schemes.

8.1.170 SRFRCE_SW**8.1.170.1 Header records**

```

      subroutine srfrce_sw
      x  (iatm,ik,iregn,mxregn,engsrp,virsrp,rcut2,rcut1,dlrpot,swlen,
      x  ilist,ltype,lstvdw,ltpvdw,rsqdf,xdf,ydf,zdf,fx,fy,fz,
      x  vvv,ggg,stress)
c
c*****
c
c      dl_poly subroutine for calculating short range force and
c      potential energy terms using verlet neighbour list
c      with switching function
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1996
c      author    - w. smith      march 1992
c      switching functions
c      author    - t. forester   june 1996
c
c*****
c

```

8.1.170.2 Function

Calculates non-bonded forces and potential energies using the Verlet neighbour list and a cubic switching function.

8.1.170.3 Dependencies

- none

8.1.170.4 Arguments**integers:**

iatm	input	index of the central site
ik	input	number of entries in the Verlet list
iregn	input	identity of current zone
mxregn	input	total number of zones

reals:

engsrp	output	the non-bonded potential energy
virsrp	output	the non-bonded virial
rcut1	input	lower cut off radius R_1
rcut2	input	upper cut off radius R_2

swlen	input	switching length λ
dlrpot	input	tabulation interval for interpolation tables
integer arrays:		
ilist	input	the Verlet neighbour list (mxlist)
ltype	input	atom type (mxatms)
lstvdw	input	index number of non-bonded potential (mxvdw)
ltpvdw	input	key for non-bonded potentials (mxvdw)
real arrays:		
rsqdf	input	square of inter site vector length (mxxdf)
xdf	input	x coordinate for inter site vector (mxxdf)
ydf	input	y coordinate for inter site vector (mxxdf)
zdf	input	z coordinate for inter site vector (mxxdf)
fx	in/out	x component of force (mxatms)
fy	in/out	y component of force (mxatms)
fz	in/out	z component of force (mxatms)
vvv	input	interpolation table for potential energy (mxgrid,mxvdw)
ggg	input	interpolation table for force (mxgrid,mxvdw)
stress	in/out	stress tensor (9)

8.1.170.5 Comments

See SRFRCE for comments on the interpolation schemes used to calculate the short-ranged potentials. Note only the 3-point interpolation scheme is supplied See COUL0_SW for comments on the switching functions.

8.1.171 SRFRCEU**8.1.171.1 Header records**

```

      subroutine srfrceuu
      x  (ik,engsrp,virsrp,dlrpot,rcut,ilist,jlist,ltype,
      x  lstvdw,rsqdf,xdx,ydx,zdx,fx,fx,fy,fz,vv,gg,ss)
c
c*****
c
c      dl_poly subroutine for calculating short range force and
c      potential energy terms using verlet neighbour list
c      neutral group implementation
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith          march 1992
c
c      neutral groups
c      author    - t. forester      march 1994
c
c*****
c

```

8.1.171.2 Function

Calculates non-bonded forces and potential energies using the Verlet neighbour list in the charge group implementation.

8.1.171.3 Dependencies

- none

8.1.171.4 Arguments**integers:**

ik	input	number of entries in the Verlet list

reals:

engsrp	output	the non-bonded potential energy
virsrp	output	the non-bonded virial
dlrpot	input	tabulation interval for interpolation tables
rcut	input	the non-bonded potential cutoff

integer arrays:

ilist	input	indices of first site in pairs (mxlist)
jlist	input	indices of second site in pairs (mxlist)

<code>ltype</code>	input	atom type (<code>mxatms</code>)
<code>lstvdw</code>	input	index number of non-bonded potential (<code>mxvdw</code>)
real arrays:		
<code>rsqdf</code>	input	square of inter site vector length (<code>mxxdf</code>)
<code>xdf</code>	input	x coordinate for inter site vector (<code>mxxdf</code>)
<code>ydf</code>	input	y coordinate for inter site vector (<code>mxxdf</code>)
<code>zdf</code>	input	z coordinate for inter site vector (<code>mxxdf</code>)
<code>fx</code>	output	x component of force (<code>mxatms</code>)
<code>fy</code>	output	y component of force (<code>mxatms</code>)
<code>fz</code>	output	z component of force (<code>mxatms</code>)
<code>vvv</code>	input	interpolation table for potential energy (<code>mxgrid,mxvdw</code>)
<code>ggg</code>	input	interpolation table for force (<code>mxgrid,mxvdw</code>)
<code>stress</code>	in/out	stress tensor (9)

8.1.171.5 Comments

The algorithm uses a 3 point interpolation scheme.

8.1.172 STATIC**8.1.172.1 Header records**

```

      subroutine static
      x  (lzeql,cfname,idnode,intsta,imcon,keyens,natms,nstack,nstep,
      x  nsteql,ntpatm,numacc,mxnode,cons,degfre,degrot,engang,
      x  engbnd,engcpe,engdih,engke,engrot,engsrp,engunit,stpcfg,
      x  stpeng,stpeth,stpprs,stptmp,stpvir,stpvol,tstep,virbnd,
      x  vircom,vircon,vircpe,virsrp,engfld,virfld,engtbp,virtbp,
      x  virpmf,virshl,engshl,engtet,virtet,degshl,shlke,virang,
      x  width,ltype,numtyp,buffer,cell,chge,fx,ffy,fzz,ravval,
      x  ssqval,stkval,stpval,sumval,vxx,vyy,vzz,xxx,yyy,zzz,zumval,
      x  xx0,yy0,zz0,weight,stress)
c
c*****
c
c      dl_poly subroutine for accumulating periodic data during the
c      molecular dynamics simulation and computing the rolling averages
c
c      copyright daresbury laboratory 1992
c
c      author - w. smith      august 1992
c
c*****
c

```

8.1.172.2 Function

Accumulates periodic data during the simulations and computes rolling averages.

8.1.172.3 Dependencies

- DCELL
- DIFFSN0
- DIFFSN1
- ERROR

8.1.172.4 Arguments

logicals:

lzeql	input	flag for accumulating statistics during equilibration

characters:

cfname	input	configuration name

integers:

idnode	input	identity of current processor
intsta	input	interval for collection of statistics
imcon	input	key for periodic boundary conditions
keyens	input	key for integration algorithm (ensemble)
natms	input	number of sites in the MD cell
nstack	input	number of stacked variables
nstep	input	number of MD time steps completed
nsteql	input	number of equilibration steps required
ntpatm	input	number of unique atom types
numacc	in/out	number of time steps used for statistics
mxnode	input	total number of processors in use

reals:

consv	input	conserved quantity
degfre	input	thermodynamic degrees of freedom
degrot	input	rotational degrees of freedom
engang	input	current valence angle energy
engbnd	input	current chemical bond energy
engcpe	input	current coulombic energy
engdih	input	current dihedral interaction energy
engke	input	current kinetic energy
engrot	input	rotational kinetic energy
engsrp	input	current short-range potential energy
engunit	input	energy unit
stpcfg	output	current total configurational energy
stpeng	output	current total energy
stpeth	output	current enthalpy
stpprs	output	current pressure
stptmp	output	current temperature
stpvir	output	current virial
stpvol	output	current cell volume
tstep	input	MD timestep, Δt
virbnd	input	current chemical bond virial
vircom	input	molecule to c.o.m. virial correction
vircon	input	constraint virial
vircoe	input	current coulombic virial
virsrp	input	current short-range potential virial
engfld	input	energy due to external field
virfld	input	virial due to external field
engtbp	input	three body potential energy
virtbp	input	three body potential virial
virpmf	input	virial from PMF constraint terms
virshl	input	core-shell virial
engshl	input	core-shell potential energy

engtet	input	tethering energy
virtet	input	tethering virial
degshl	input	core-shell psuedo degrees of freedom
shlke	input	core-shell kinetic energy
virang	input	valence angle virial
width	output	half smallest perpendicular width of MD cell
integer arrays:		
ltype	input	atomic type (mxatms)
numtyp	input	number atoms of given type (mxsvdw)
real arrays:		
buffer	work	(mxbuff)
cell	input	MD cell vectors (9)
chge	input	atomic charges (mxtams)
fxx	input	x component of force (mxatms)
fyy	input	y component of force (mxatms)
fzz	input	z component of force (mxatms)
ravval	in/out	rolling averages (mxnstk)
ssqval	in/out	sum of squares of statistical data (mxnstk)
stkval	in/out	statistical data (mxstak,mxnstk)
stpval	output	current values of statistical data (mxnstk)
sumval	in/out	sum of statistical data (mxnstk)
vxx	input	x component of velocity (mxatms)
vyy	input	y component of velocity (mxatms)
vzz	input	z component of velocity (mxatms)
xxx	input	x coordinates (mxatms)
yyy	input	y coordinates (mxatms)
zzz	input	z coordinates (mxatms)
zumval	in/out	sum of rolling average statistical data (mxnstk)
xx0	in/out	x displacement from start of simulation (mxatms)
yy0	in/out	y displacement from start of simulation (mxatms)
zz0	in/out	z displacement from start of simulation (mxatms)
weight	input	atomic masses (mxatms)
stress	input	stress tensor (9)

8.1.172.5 Comments

Atomic coordinates, velocities, forces and charges are passed into this subroutine for users who may wish to use such information for the evaluation of additional properties in the statistics files. See section 4.2.7 for a description of the calculated quantities printed.

8.1.173 STRIP**8.1.173.1 Header records**

```
subroutine strip(string,length)
```

```
c*****
c
c    DL_POLY routine to strip blanks from start of a string
c    maximum length is 255 characters
c
c    copyright daresbury laboratory 1993
c    author   t.forester       july 1993
c
c*****
```

8.1.173.2 Function

Strips blanks from the start of an alphanumeric string.

8.1.173.3 Dependencies

- none

8.1.173.4 Arguments**character:**

string	in/out	alphanumeric string

integers:

length	input	length of string (max value = 255)

8.1.173.5 Comments

8.1.174 STRUCOPT

8.1.174.1 Header records

```

      subroutine strucopt
      x  (loptim,idnode,mxnode,imcon,natms,ngroup,ntfree,
      x  lstfre,lstgtp,lstrgd,numgsit,cell,gvxx,gvyy,gvzz,
      x  gcmx,gcmz,gcmz,omx,omy,omz,vxx,vyy,vzz,fx,fy,
      x  fzz,xxx,yyy,zzz,xxt,yyt,zzt,q0,q1,q2,q3,rotinx,
      x  rotiny,rotinz)

C*****
C
C      dl_poly routine for zero Kelvin temperature optimization
C      if veloc.Force < 0 then velocity is set to zero in
C      preparation for integration of equations of motion
C
C      parallel replicated data version : block data
C
C      copyright daresbury laboratory 1994
C      author t.forester      march 1994
C      amended t.forester     dec 1994 : block data
C
C*****

```

8.1.174.2 Function

Adjusts atomic and rigid body velocities for a “steepest descent” like structure optimisation, prior to a call to a standard integration routine.

8.1.174.3 Dependencies

- IMAGES

8.1.174.4 Arguments

logical:

loptim	input	flag for structure optimization

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
imcon	input	key for periodic boundary conditions
natms	input	number of atoms in MD cell
ngroup	input	number of rigid body units in system
ntfree	input	number of sites in system not in rigid body units

integer arrays:

lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgadm)
numgsit	input	number of sites in a rigid unit type (mxungp)
real arrays:		
cell	input	MD cell vectors (9)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
gcmx	input	x coordinate of rigid unit c.o.m. (mxgrp)
gcmx	input	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	input	z coordinate of rigid unit c.o.m. (mxgrp)
omx	in/out	x component of angular velocity (mxgrp)
omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
vxx	output	x component of velocity (mxatms)
vyy	output	y component of velocity (mxatms)
vzz	output	z component of velocity (mxatms)
fx	input	x component of force (mxatms)
fy	input	y component of force (mxatms)
fz	input	z component of force (mxatms)
xxx	input	x coordinates (mxatms)
yyy	input	y coordinates (mxatms)
zzz	input	z coordinates (mxatms)
xxt	work	(mxatms)
yxt	work	(mxatms)
zzt	work	(mxatms)
q0	input	first component of quaternion vector (mxgrp)
q1	input	second component of quaternion vector (mxgrp)
q2	input	third component of quaternion vector (mxgrp)
q3	input	fourth component of quaternion vector (mxgrp)
rotinx	input	rotational inertia about x local body axis (mxungp)
rotiny	input	rotational inertia about y local body axis (mxungp)
rotinz	input	rotational inertia about z local body axis (mxungp)

8.1.174.5 Comments

The algorithm calculates

$$a = \underline{v}_i \cdot \underline{f}_i$$

for each site. If a is less than zero (i.e the velocity and force are in opposite directions) the velocity ($\underline{v}$) is set to 0. Otherwise the velocity is set to the component of the velocity in the

direction of the force. Similar adjustments are made to the rigid body c.o.m. translational velocity and angular velocity (based on the torque).

8.1.175 SUTTCHEN**8.1.175.1 Header records**

```

      subroutine suttchen
      x  (iatm,ik,ntpvdw,engmet,virmet,rmet,dlrpot,ilist,ltype,lstvdw,
      x  ltpvdw,rsqdf,xdf,ydf,zdf,fx,fy,fz,vv,ggg,stress,rho)
c
c*****
c
c      dl_poly subroutine for calculating metal forces atom
c      using sutton chen potential and verlet neighbour list
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1995
c      author    - w. smith          june 1995
c
c*****
c

```

8.1.175.2 Function

Evaluates energy and forces arising from Sutton Chen potentials.

8.1.175.3 Dependencies

- none

8.1.175.4 Arguments**integers:**

iatm	input	index of the central site
ik	input	number of neighbours around the central site
ntpvdw	input	number of Van der Waals potentials

reals:

engmet	output	Sutton Chen potential energy
virmet	output	Sutton Chen virial
rmet	input	radial cutoff for Sutton Chen potential
dlrpot	input	distance increment for short-range potential tabulation

integer arrays:

ilist	input	index of neighbours around central site (mxxdf)
ltype	input	index of atomic types (mxatms)
lstvdw	input	pointer from potential index to parameter table (mxvdw)
ltpvdw	input	key for non-bonded potentials (mxvdw)

real arrays:

rsqdf	input	square of interatomic distances (mxxdf)
xdf	input	x component of interatomic vector (mxxdf)
ydf	input	y component of interatomic vector (mxxdf)
zdf	input	z component of interatomic vector (mxxdf)
fx	in/out	x component of force (mxatms)
fy	in/out	y component of force (mxatms)
fz	in/out	z component of force (mxatms)
v	input	interpolation table for non-bonded potentials (mxgrid,mxvdw)
g	input	interpolation table for non-bonded force (mxgrid,mxvdw)
stress	in/out	stress tensor (9)
rho	input	Sutton Chen particle density (mxatms)

8.1.175.5 Comments

The subroutine requires a prior call to SCDENS to evaluate the Sutton Chen particle density.

8.1.176 SYMINV**8.1.176.1 Header records**

```

      subroutine syminv(nnn,aaa,bbb)
c
c*****
c
c      routine to invert a real symmetric matrix (reference:
c      computing methods v.ii, i.s. berezin and n.p.zhidkov,
c      pergamon press 1965). note that the matrices are
c      packed in the minimum storage mode, (i.e. a(k)=a(i,j),
c      where i>j and k=(i*(i-1))/2+j ).
c      the matrices aaa and bbb may be equivalenced though
c      this will destroy the contents of the original
c      array.
c
c      general version for all real symmetric matrices
c
c      copyright daresbury laboratory 1993
c      author w.smith (added to dl_poly july 1993)
c
c*****
c

```

8.1.176.2 Function

Finds the inverse of a real symmetric matrix.

8.1.176.3 Dependencies

- none

8.1.176.4 Arguments

integers:

nnn input dimension of matrix

real arrays:

aaa input matrix in triangular form (*)

bbb output inverse matrix in triangular form (*)

8.1.176.5 Comments

The program makes use of minimum storage requirements for the matrices. The array AAA has $NNN(NNN+1)/2$ entries defining the lower triangle of the real symmetric matrix. The

inverse matrix $\mathbf{B}^{-1}$ is given in the same form.

8.1.177 SYSBOOK**8.1.177.1 Header records**

```

      subroutine sysbook
x  (loglnk,lneut,lshmov,lcnb,idnode,imcon,mxnode,natms,
x  nneut,ngroup,nscons,ntangl,ntbond,ntcons,ntdihd,ntinv,
x  ntpmls,ntpmf,nspmf,ntfree,ntteth,ntshl,degfre,degrot,
x  keybnd,keyang,lashap,lexsit,lexatm,lishap,listang,
x  listbnd,listcon,listdih,listinv,listin,listme,listot,
x  listyp,lstang,lstbnd,lstcon,lstdih,lstinv,lstfre,
x  lstfrz,lstgtp,lstgst,lstrgd,lstcsit,nexatm,nexsit,
x  numang,lstme,lstbod,lstout,lstshl,numbonds,numcon,
x  numdih,numinv,numgsit,indpmf,lstpmf,numpmf,npmf,ind,
x  lstpmt,listpm,numgrp,nummols,numsit,numteth,numshl,
x  itest,index,kscons,msite,mconst,lsttet,lsttet,
x  listshl,neulst,lstneu,buffer,cell,gcmx,gcmz,gcmz,
x  gmass,gxx,gyy,gzz,prmdih,q0,q1,q2,q3,rotinx,rotiny,
x  rotinz,txx,tyy,tzz,weight,xxt,xxx,yyt,yyy,zzt,zzz,
x  accum,gaxs,rotmin)

c
c*****
c
c      dl_poly subroutine  defining global bookkeeping
c      arrays
c
c      copyright - daresbury laboratory 1997
c      author    - w. smith          july 1997
c
c*****
c

```

8.1.177.2 Function

SYSBOOK defines the DL_POLY_2 global bookkeeping arrays.

8.1.177.3 Dependencies

ERROR	EXCLUDE	EXCLUDE_ATOM
EXCLUDE_LINK	EXCLUDENEU	INTLIST
PASSQUAT	QUATBOOK	

8.1.177.4 Arguments**logicals:**

loglnk	input	flag for link cell algorithm
lneut	input	flag for neutral group electrostatics
lshmov	output	flag for shared bond constraints
lcnb	output	flag for rigid bodies connected by bond constraints

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
mxnode	input	total number of processors
natms	input	number of atoms in MD cell
nneut	output	number of charge groups in system
ngrp	input	number of rigid body units in system
nscons	output	number of constraints on current node
ntangl	output	total number of valence angles in system
ntbond	output	total number of bonds in system
ntcons	output	total number of constraints in system
ntdihd	output	total number of dihedral angles in the system
ntinv	output	total number of inversion angles in the system
ntpmls	input	number of types of molecules in the system
ntpmf	output	total number of PMF constraints in system
nspmf	output	number of PMF constraints on the current processor
ntfree	output	number of sites in system not in rigid body units
ntteth	output	total number of tethered atoms in system
ntshl	output	total number of core-shell units in system

reals:

degfre	output	number of thermodynamic degrees of freedom
degrot	output	rotational degrees of freedom

integer arrays:

keybnd	input	chemical bond potential key (mxtbnd)
keyang	input	angle potential key (mxtang)
lashap	output	see RDSHAKE_1 (mxproc)
lexsit	output	“excluded site” list for pair forces (mxsite,mxexcl)
lexatm	output	“excluded atom” list (msatms,mxexcl)
lishap	output	see RDSHAKE_1 (mxlshp)
listang	input	key and indices for valence angle interactions (mxangl,4)
listbnd	input	key and indices for chemical bond interactions (mxbond,3)
listcon	input	indices for constraint bonds (mxcons,3)
listdih	input	key and indices for dihedral interactions (mxdihd,5)
listinv	input	key and indices for inversion interactions (mxinv,5)
listin	work	(mxatms)
listme	input	see RDSHAKE_1 (mxatms)
listot	input	see RDSHAKE_1 (mxatms)

listyp	input	rigid unit type (mxungp)
lstang	input	Valence angle potential site list (mxtang,3)
lstbnd	input	Chemical bond potential site list (mxtbnd,2)
lstcon	input	constraint bond site list (mxtcon,2)
lst dih	input	dihedral angle potential site list (mxt dih,4)
lstinv	input	inversion angle potential site list (mxtinv,4)
lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
lstfrz	output	key for frozen atoms (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstgst	input	site indices for rigid bodies (mxungp,mxngp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgadm)
lstcsit	output	indices of rigid unit sites in constraints (2*mxcons)
nexatm	output	number of excluded atoms (msatms)
nexsit	output	number of excluded sites (mxsite)
numang	input	number of valence angles in a molecule (mxtmls)
lstme	output	indice list for MERGE1 (mxatms)
lstbod	output	pointer from atom index to rigid unit index (mxatms)
lstout	work	(mxatms)
lstshl	input	site indices for core-shell units (mxtshl,2)
numbonds	input	number of chemical bonds in a molecule (mxtmls)
numcon	input	number of constraint bonds in a molecule (mxtmls)
numdih	input	number of dihedral interactions in a molecule (mxtmls)
numinv	input	number of inversion interactions in a molecule (mxtmls)
numgsit	input	number of sites in a rigid unit type (mxungp)
indpmf	input	indices of sites in PMF constraints (mxspmf)
lstpmf	output	indices of atoms in PMF constraints on this processor (mxspmf,mspmf)
numpmf	input	number of PMF constraints in each molecular type (mxtmls)
npmf	input	number of sites in PMF units (2)
ind	work	(mxgrp,3)
lstpmt	output	flag of all atoms in system in PMF constraints (mxpmf)
listpm	output	flag of all atoms in PMF constraints on this processor (mxpmf)
numgrp	input	number of rigid units in a molecule (mxtmls)
nummols	input	number of molecules of each type (mxtmls)
numsit	input	number of sites on a molecule (mxtmls)
numteth	input	number of tethered sites in each molecular type (mxtmls)
numtyp	output	number of each atom type (mxsvdw)
numshl	input	number of core-shell units in each molecular type (mxtmls)
itest	work	(mxtmls)
index	work	(mxtmls)
kscons	work	(mxproc)
msite	work	(mxtmls)

mconst	work	(mxtmls)
lsttet	input	indices of tethered sites (mxteth)
listtet	output	indices of tethered atoms on current processor (msteth,2)
listshl	output	indices for core-shell interactions on current processor (mxshl,3)
neulst	output	indice of first site in charge group (mxneut)
lstneu	output	index of group to which site belongs (mxatms)
real arrays:		
buffer	work	(mxbuff)
cell	input	MD cell vectors (9)
gcmx	output	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	output	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	output	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	output	rigid unit mass (mxungp)
gxx	output	x component of site in local body frame (mxungp,mxngp)
gyy	output	y component of site in local body frame (mxungp,mxngp)
gzz	output	z component of site in local body frame (mxungp,mxngp)
prmdih	input	dihedral angle potential parameters (mxtdih,mxpdih)
q0	output	first component of quaternion vector (mxgrp)
q1	output	second component of quaternion vector (mxgrp)
q2	output	third component of quaternion vector (mxgrp)
q3	output	fourth component of quaternion vector (mxgrp)
rotinx	output	rotational inertia about x local body axis (mxungp)
rotiny	output	rotational inertia about y local body axis (mxungp)
rotinz	output	rotational inertia about z local body axis (mxungp)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
weight	output	atomic mass (mxatms)
xxt	work	(mxatms)
xxx	output	x coordinates (mxatms)
yyt	work	(mxatms)
yyy	output	y coordinates (mxatms)
zzt	work	(mxatms)
zzz	output	z coordinates (mxatms)
accum	work	(mxungp)
gaxs	work	(mxungp,9)
rotmin	work	(mxungp)

8.1.177.5 Comments

8.1.178 SYSDEF**8.1.178.1 Header records**

```

      subroutine sysdef
      x  (lneut,lmetal,lmsq,molnam,sitnam,unqatm,idnode,keyfce,keyfld,
      x  natms,ngroup,ntpatm,ntpmls,ntpvdw,ntptbp,ntpfbp,nshels,alpha,
      x  dlrpot,drewd,engunit,prmpmf,rcut,rwdw,rcuttb,rcutfb,indpmf,
      x  keyang,keybnd,keydih,keyinv,keytet,lfzsit,listyp,lstang,lstbnd,
      x  lstcon,lstdih,lstinvl,lstgst,lstvdw,lsttbp,lstfbp,lstshl,
      x  ltpsit,lsttet,ltpvdw,ltpfbp,ltpfbp,npmf,nugrp,numang,numbonds,
      x  numcon,numdih,numinv,numgrp,numgsit,nummols,numsit,numpmf,
      x  numteth,numshl,chgsl,erc,fer,pmfwght,ggg,prmang,prmbnd,prmcon,
      x  prmdih,prminv,prmfld,prmtet,prmvdw,prmtbp,prmfbp,prmsl,vvv,
      x  wgsit,rcut3b,rcut4b,buffer)

c
c*****
c
c      dl_poly subroutine for reading in the molecular specifications
c      of the system to be simulated
c      version for rigid unit data and neutral groups
c
c      copyright - daresbury laboratory 1992
c      author      - w. smith may 1992.
c      amended     - w.smith march 1994
c      amended     - t.forester april 1994
c      amended     - w.smith dec 1994 - get rec etc
c
c*****
c

```

8.1.178.2 Function

Reads in molecular specifications of the system from the FIELD file.

8.1.178.3 Dependencies

ERFCGEN	ERROR	FORGEN	FORTAB
GETREC	LOWCASE	METGEN	STRIP

8.1.178.4 Arguments

logicals:

lneut	output	flag for neutral group electrostatics

lmetal	output	flag for metal (Sutton Chen) potentials
lnsq	input	flag for “all-pairs” algorithm
characters:		
molnam	output	molecule names [a40] (mxtmls)
sitnam	output	site names [a40] (mxsite)
unqatm	output	unique atom type names [a8] (mxsite)
integers:		
idnode	input	identity of current processor
keyfce	input	key for non-bonded potential
keyfld	output	key for external field
natms	output	total number of atoms in the MD cell
ngrp	output	total number of rigid body units in the MD cell
ntpatm	output	number of unique types of atoms
ntpmls	output	number of unique types of molecule
ntpvdw	output	number of non-bonded potentials
ntptbp	output	number of three body potentials
ntpfbp	output	number of four body potentials
nshels	output	number of core-shell units
reals:		
alpha	input	Ewald sum convergence variable
dlrpot	output	distance increment for short-range potential tabulation
drewd	output	distance increment for Ewald sum tabulation
engunit	output	unit of energy relative to internal units
prmpmf	output	PMF constraint distance
rcut	output	non-bonded potential cut-off
rvdw	output	Van der Waals potential cut-off radius
rcuttb	output	three body potential cut-off
rcutfb	output	four body potential cut-off
integer arrays:		
indpmf	output	indices of sites in PMF constraints (mxspmf)
keyang	output	angle potential key (mxtang)
keybnd	output	bond potential key (mxtbnd)
keydih	output	dihedral potential key (mxt dih)
keyinv	output	inversion potential key (mxtinv)
keytet	output	tethering potential key (mxt dih)
lfzsit	output	flag for frozen site (mxsite)
listyp	output	rigid unit type (mxungp)
lstang	output	Valence angle potential site list (mxtang ,3)
lstbnd	output	Chemical bond potential site list (mxtbnd ,2)
lstcon	output	constraint bond site list (mxtcon ,2)
lstdih	output	dihedral angle potential site list (mxt dih ,4)
lstinv	output	inversion angle potential site list (mxtinv ,4)
lstgst	output	site indices for rigid bodies (mxungp , mxngp)
lstvdw	output	pointer from potential index to parameter table (mxvdw)

lsttbp	output	pointer from three body potential to parameter table (mxtbp)
lstfbp	output	pointer from four body potential to parameter table (mxfbp)
lstshl	output	site indices for core-shell units (mxtshl,2)
ltpsit	output	site type index (mxsite)
lsttet	output	site indices for tethered atoms (mxteth)
ltpvdw	input	key for non-bonded potentials (mxvdw)
ltptbp	output	key for three body potentials (mxtbp)
ltpfbp	output	key for four body potentials (mxfbp)
npmf	output	number of sites in PMF units (2)
nugrp	output	neutral group index (mxsite)
numang	output	number of valence angles in a molecule (mxtmls)
numbonds	output	number of chemical bonds in a molecule (mxtmls)
numcon	output	number of constraint bonds in a molecule (mxtmls)
numdih	output	number of dihedral angles in a molecule (mxtmls)
numinv	output	number of inversion angles in a molecule (mxtmls)
numgrp	output	number of rigid units in a molecule (mxtmls)
numgsit	output	number of sites in a rigid unit type (mxungp)
nummols	output	number of molecules of a given type (mxtmls)
numsit	output	number of sites in a molecule (mxtmls)
numpmf	output	number of PMF constraints in each molecular type (mxtmls)
numteth	output	number of tethered sites in each molecular type (mxtmls)
numshl	output	number of core-shell units in each molecular type (mxtmls)
real arrays:		
chgsit	output	site charges (mxsite)
erc	output	interpolation table for EWALD2 (mxgrid)
fer	output	derivative of erc (see EWALD2) (mxgrid)
pmfwght	output	weighting for PMF sites (mxspmf)
ggg	output	interpolation table for non-bonded force (mxgrid,mxvdw)
prmang	output	valence angle potential parameters (mxtang,mxpang)
prmbnd	output	chemical bond potential parameters (mxtbnd,mxpbnd)
prmcon	output	constraint bond lengths (mxtcon)
prmdih	output	dihedral angle potential parameters (mxt dih,mxpdih)
prminv	output	inversion angle potential parameters (mxtinv,mxpinv)
prmfld	output	external field potential parameters (mxfld)
prmtet	output	tethering potential parameters (mxteth,mxpbnd)
prmvdw	output	short-range potential parameters (mxvdw,mxpvdw)
prmtbp	output	three body potential parameters (mxtbp,mxpang)
prmfbp	output	four body potential parameters (mxfbp,mxpang)
prmshl	output	core-shell potential parameters (mxtshl)
vvv	output	interpolation table for non-bonded potentials (mxgrid,mxvdw)
wgtsit	output	site masses (mxsite)
rcut3b	output	three body potential cutoffs (mxtbp)
rcut4b	output	four body potential cutoffs (mxtbp)
buffer	work	(mxbuff)

8.1.178.5 Comments

See section 4.1.3 for a description of the FIELD file.

8.1.179 SYSDEF_RESPA**8.1.179.1** Header records

```

      subroutine sysdef_respa
      x  (lneut,lmetal,lnsq,molnam,sitnam,unqatm,idnode,keyfce,keyfld,
      x  natms,ngroup,ntpatm,ntpmls,ntpvdw,ntptbp,nshels,alpha,dlrpot,
      x  drewd,engunit,prmpmf,rcut,rvdw,rcuttb,indpmf,keyang,keybnd,
      x  keydih,keytet,lfzsit,lstang,lstbnd,lstcon,lstdih,
      x  lstvdw,lsttbp,lstshl,ltpsit,lsttet,ltpvdw,ltptbp,npmf,nugrp,
      x  numang,numbonds,numcon,numdih,numgrp,nummols,numsit,
      x  numpmf,numteth,numshl,chgsit,erc,fer,pmfwght,ggg,prmang,prmbnd,
      x  prmcon,prmdih,prmfld,prmtet,prmvdw,prmtbp,prmshl,vvv,wgtsit,
      x  rcut3b,buffer)

c
c*****
c
c      dl_poly subroutine for reading in the molecular specifications
c      of the system to be simulated
c      rigid group options removed ..
c
c      copyright - daresbury laboratory 1992
c      author      - w. smith may 1992.
c      amended     - w.smith march 1994
c      amended     - t.forester april 1994
c      amended     - w.smith dec 1994 - get rec etc
c      respa variant - t.forester june 1996
c
c*****
c

```

8.1.179.2 Function

Reads in molecular specifications of the system from the FIELD file.

8.1.179.3 Dependencies

ERFCGEN	ERROR	FORGEN	FORTAB
GETREC	LOWCASE	STRIP	

8.1.179.4 Arguments**logicals:**

lneut	output	flag for neutral group electrostatics

lmetal	output	flag for metal (Sutton Chen) potentials
lnsq	input	flag for “all-pairs” algorithm
characters:		
molnam	output	molecule names [a40] (mxtmls)
sitnam	output	site names [a40] (mxsite)
unqatm	output	unique atom type names [a8] (mxsite)
integers:		
idnode	input	identity of current processor
keyfce	input	key for non-bonded potential
keyfld	output	key for external field
natms	output	total number of atoms in the MD cell
ngrp	output	total number of rigid body units in the MD cell
ntpatm	output	number of unique types of atoms
ntpmls	output	number of unique types of molecule
ntpvdw	output	number of non-bonded potentials
ntptbp	output	number of three body potentials
nshels	output	number of core-shell units
reals:		
alpha	input	Ewald sum convergence variable
dlrpot	output	distance increment for short-range potential tabulation
drewd	output	distance increment for Ewald sum tabulation
engunit	output	unit of energy relative to internal units
prmpmf	output	PMF constraint distance
rcut	output	non-bonded potential cut-off
rvdw	output	Van der Waals potential cut-off radius
rcuttb	output	three body potential cut-off
integer arrays:		
indpmf	output	indices of sites in PMF constraints (mxspmf)
keyang	output	angle potential key (mxtang)
keybnd	output	bond potential key (mxtbnd)
keydih	output	dihedral potential key (mxt dih)
keytet	output	tethering potential key (mxt dih)
lfzsit	output	flag for frozen site (mxsite)
lstang	output	Valence angle potential site list (mxtang,3)
lstbnd	output	Chemical bond potential site list (mxtbnd,2)
lstcon	output	constraint bond site list (mxtcon,2)
lst dih	output	dihedral angle potential site list (mxt dih,4)
lstvdw	output	pointer from potential index to parameter table (mxvdw)
lsttbp	output	pointer from three body potential to parameter table (mxtbp)
lstshl	output	site indices for core-shell units (mxtshl,2)
ltps it	output	site type index (mxsite)
lsttet	output	site indices for tethered atoms (mxteth)
ltpvdw	input	key for non-bonded potentials (mxvdw)
ltptbp	output	key for three body potentials (mxtbp)

npmf	output	number of sites in PMF units (2)
nugrp	output	neutral group index (mxsite)
numang	output	number of valence angles in a molecule (mxtmls)
numbonds	output	number of chemical bonds in a molecule (mxtmls)
numcon	output	number of constraint bonds in a molecule (mxtmls)
numdih	output	number of dihedral angles in a molecule (mxtmls)
nummols	output	number of molecules of a given type (mxtmls)
numsit	output	number of sites in a molecule (mxtmls)
numpmf	output	number of PMF constraints in each molecular type (mxtmls)
numteth	output	number of tethered sites in each molecular type (mxtmls)
numshl	output	number of core-shell units in each molecular type (mxtmls)
real arrays:		
chgsit	output	site charges (mxsite)
erc	output	interpolation table for EWALD2 (mxgrid)
fer	output	derivative of erc (see EWALD2) (mxgrid)
pmfwght	output	weighting for PMF sites (mxspmf)
ggg	output	interpolation table for non-bonded force (mxgrid,mxvdw)
prmang	output	valence angle potential parameters (mxtang,mxpang)
prmbnd	output	chemical bond potential parameters (mxtbnd,mxpbnd)
prmcon	output	constraint bond lengths (mxtcon)
prmdih	output	dihedral angle potential parameters (mxt dih,mxpdih)
prmfld	output	external field potential parameters (mxfld)
prmtet	output	tethering potential parameters (mxteth,mxpbnd)
prmvdw	output	short-range potential parameters (mxvdw,mxpvdw)
prmtbp	output	three body potential parameters (mxtbp,mxpang)
prmsl	output	core-shell potential parameters (mxtshl)
vvv	output	interpolation table for non-bonded potentials (mxgrid,mxvdw)
wgtsit	output	site masses (mxsite)
rcut3b	output	three body potential cutoffs (mxtbp)
buffer	work	(mxbuff)

8.1.179.5 Comments

See section 4.1.3 for a description of the FIELD file. The RESPA algorithm is not compatible with metal potentials or rigid bodies.

8.1.180 SYSGEN

This description applies to the subroutine SYSGEN in DL_POLY_2 version 2.10 and earlier.

8.1.180.1 Header records

```

      subroutine sysgen
      x  (loglnk,lgoifr,lneut,lshmov,lzden,lcnb,lmetal,atmmam,cfgname,
      x  sitnam,idnode,imcon,keyens,keyfce,keyres,multt,mxnode,natms,
      x  nneut,ngroup,nscons,nstep,ntangl,ntbond,ntcons,ntdihd,ntinv,
      x  ntpmls,ntpmf,nspmf,ntpvdw,numacc,numrdf,ntpatm,ntfree,ntteth,
      x  ntshl,nzden,chip,chit,conint,degfre,degshl,degrot,delr,elrc,
      x  engunit,virlrc,rcut,rvdw,temp,tolnce,volm,keybnd,keyang,lashap,
      x  lexsit,lexatm,lfzsit,lishap,listang,listbnd,listcon,listdih,
      x  listinv,listin,listme,listot,listyp,lstang,lstbnd,lstcon,lstdih,
      x  lstinv,lstfre,lstfrz,lstgtp,lstgst,lstrgd,lstcsit,lstvdw,ltpsit,
      x  ltpvdw,ltype,nexatm,nexsit,numang,lstme,lstbod,lstout,lstshl,
      x  numbonds,numcon,numdih,numinv,numgsit,indpmf,lstpmf,numpmf,npmf,
      x  lstpmt,listpm,numgrp,nummols,numsit,numteth,numtyp,numshl,
      x  lsttet,lsttet,listshl,nugrp,neulst,lstneu,numfrz,buffer,cell,
      x  chge,chgsit,dens,dxt,dyt,dzt,fxx,fyy,fzz,gcmx,gcmz,gmass,
      x  gvxx,gvyy,gvzz,gxx,gyy,gzz,prmdih,prminv,prmvdw,q0,q1,q2,q3,
      x  ravval,rdf,rotinx,rotiny,rotinz,ssqval,stkval,stpval,sumval,
      x  txx,tyy,tzz,uxx,uyy,uzz,vxx,vyy,vzz,weight,wgtsit,xx0,xxt,xxx,
      x  yy0,yyt,yyy,zumval,zz0,zzt,zzz,omx,omy,omz,zdens,xxs,sys,zzs,
      x  elrcm,vlrcm,eta)

c
c*****
c
c      dl_poly subroutine for reading in the initial configuration
c      of the system to be simulated and defining global bookkeeping
c      arrays
c      includes frozen atom option - t.forester october 1993
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith          july 1992
c
c*****
c

```

8.1.180.2 Function

Reads in initial configuration from the CONFIG file and defines global bookkeeping arrays.

8.1.180.3 Dependencies

DCELL	ERROR	EXCLUDE	EXCLUDE_ATOM
EXCLUDE_LINK	EXCLUDENEU	GAUSS	GDSUM
GETREC	GISUM	GSTATE	INTLIST
LRCMETAL	LRCORRECT	PASSQUAT	QUATBOOK
QUATQNCH	QUENCH	SHLQNCH	STRIP
VSCALEG	WARNING		

8.1.180.4 Arguments**logicals:**

loglnk	output	flag for link cell algorithm
lgofr	input	flag for calculation of radial distribution functions
lneut	input	flag for neutral group electrostatics
lshmov	output	flag for shared bond constraints
lzden	input	flag for calculating z density profile
lcnb	output	flag for rigid bodies connected by bond constraints
lmetal	input	flag for metal (Sutton Chen) potentials

characters:

atmnam	output	atom labels [a8], (mxatms)
cfgname	output	configuration name [a80]
sitnam	input	site labels [a8], (mxsite)

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
keyens	input	ensemble key
keyfce	input	key for non-bonded force field
keyres	input	key for restart of MD run
multt	input	multiple timestep interval
mxnode	input	total number of processors
natms	input	number of atoms in MD cell
nneut	output	number of charge groups in system
ngrp	input	number of rigid body units in system
nscons	output	number of constraints on current node
nstep	input	running counter of completed MD cycles
ntangl	output	total number of valence angles in system
ntbond	output	total number of bonds in system
ntcons	output	total number of constraints in system
ntdihd	output	total number of dihedral angles in the system
ntpmls	input	number of types of molecules in the system
ntpmf	output	total number of PMF constraints in system
nspmf	output	number of PMF constraints on the current processor

ntpvdw	input	number of types of non-bonded potentials
numacc	output	running counter of the number of steps used for statistics
numrdf	output	number of steps used for rdf statistics
ntpatm	output	number of unique atom types
ntfree	output	number of sites in system not in rigid body units
ntteth	output	total number of tethered atoms in system
ntshl	output	total number of core-shell units in system
nzden	output	number of configurations used for z density profiles
reals:		
chip	output	coupling parameter for barostat
chit	output	coupling parameter for thermostat
conint	output	time integral of chit and chip
degfre	output	number of thermodynamic degrees of freedom
degshl	output	core-shell psuedo degrees of freedom
degrot	output	rotational degrees of freedom
delr	input	border width for Verlet list
elrc	output	long range correction to energy
engunit	input	energy unit for input /output
virlrc	output	long range correction to virial
rcut	input	potential cut-off radius
rvdw	input	Van der Waals potential cut-off radius
temp	input	simulation temperature
tolnce	input	tolerance for SHAKE algorithm
volm	output	volume of MD cell
integer arrays:		
keybnd	input	chemical bond potential key (mxtbnd)
keyang	input	angle potential key (mxtang)
lashap	output	see RDSHAKE_1 (mxproc)
lexsit	output	“excluded site” list for pair forces (mxsite , mxexcl)
lexatm	output	“excluded atom” list (msatms , mxexcl)
lfzsit	input	flag for frozen site (mxsite)
lishap	output	see RDSHAKE_1 (mxlshp)
listang	input	key and indices for valence angle interactions (mxangl ,4)
listbnd	input	key and indices for chemical bond interactions (mxbond ,3)
listcon	input	indices for constraint bonds (mxcons ,3)
listdih	input	key and indices for dihedral interactions (mxdihd ,5)
listin	work	(mxatms)
listme	input	see RDSHAKE_1 (mxatms)
listot	input	see RDSHAKE_1 (mxatms)
listyp	input	rigid unit type (mxungp)
lstang	input	Valence angle potential site list (mxtang ,3)
lstbnd	input	Chemical bond potential site list (mxtbnd ,2)
lstcon	input	constraint bond site list (mxtcon ,2)
lstdih	input	dihedral angle potential site list (mxtdih ,4)

lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
lstfrz	output	key for frozen atoms (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstgst	input	site indices for rigid bodies (mxungp,mxngp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgadm)
lstcsit	output	indices of rigid unit sites in constraints (2*mxcons)
lstvdw	input	index number of non-bonded potential (mxvdw)
ltpsit	input	site type index (mxsite)
ltpvdw	input	key for non-bonded potentials (mxvdw)
ltype	input	list of atom types (mxatms)
nexatm	output	number of excluded atoms (msatms)
nexsit	output	number of excluded sites (mxsite)
numang	input	number of valence angles in a molecule (mxtmls)
lstme	output	indice list for MERGE1 (mxatms)
lstbod	output	pointer from atom index to rigid unit index (mxatms)
lstout	work	(mxatms)
lstshl	input	site indices for core-shell units (mxtshl,2)
numbonds	input	number of chemical bonds in a molecule (mxtmls)
numcon	input	number of constraint bonds in a molecule (mxtmls)
numdih	input	number of dihedral interactions in a molecule (mxtmls)
numgsit	input	number of sites in a rigid unit type (mxungp)
indpmf	input	indices of sites in PMF constraints (mxspmf)
lstpmf	output	indices of atoms in PMF constraints on this processor (mxspmf,mspmf)
numpmf	input	number of PMF constraints in each molecular type (mxtmls)
npmf	input	number of sites in PMF units (2)
lstpmt	output	flag of all atoms in system in PMF constraints (mxpmf)
listpm	output	flag of all atoms in PMF constraints on this processor (mxpmf)
numgrp	input	number of rigid units in a molecule (mxtmls)
nummols	input	number of molecules of each type (mxtmls)
numsit	input	number of sites on a molecule (mxtmls)
numteth	input	number of tethered sites in each molecular type (mxtmls)
numtyp	output	number of each atom type (mxsvdw)
numshl	input	number of core-shell units in each molecular type (mxtmls)
lstteth	input	indices of tethered sites (mxteth)
listteth	output	indices of tethered atoms on current processor (msteth,2)
listshl	output	indices for core-shell interactions on current processor (mxshl,3)
nugrp	input	neutral group index (mxsite)
neulst	output	indice of first site in charge group (mxneut)
lstneu	output	index of group to which site belongs (mxatms)
numfrz	output	number of frozen atoms of each type (mxsvdw)

real arrays:

buffer	work	(mxbuff)
cell	input	MD cell vectors (9)
chge	output	atomic charges (mxatms)
chgsit	input	site partial charge (mxsite)
dens	output	number density (mxsvdw)
dxt	work	(mxcons)
dyt	work	(mxcons)
dzt	work	(mxcons)
fxx	output	x component of force (mxatms)
fyy	output	y component of force (mxatms)
fzz	output	z component of force (mxatms)
gcmx	output	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	output	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	output	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	output	rigid unit mass (mxungp)
gvxx	output	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	output	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	output	z component of rigid unit c.o.m. velocity (mxgrp)
gxx	output	x component of site in local body frame (mxungp,mxngp)
gyy	output	y component of site in local body frame (mxungp,mxngp)
gzz	output	z component of site in local body frame (mxungp,mxngp)
prmdih	input	dihedral angle potential parameters (mxtdih,mxpdih)
prmvdw	input	parameters for non-bonded potentials (mxvdw,mxpvdw)
q0	output	first component of quaternion vector (mxgrp)
q1	output	second component of quaternion vector (mxgrp)
q2	output	third component of quaternion vector (mxgrp)
q3	output	fourth component of quaternion vector (mxgrp)
ravval	output	rolling averages statistics array (mxnstk)
rdf	output	histograms for rdfs (mxrdf,mxvdw)
rotinx	output	rotational inertia about x local body axis (mxungp)
rotiny	output	rotational inertia about y local body axis (mxungp)
rotinz	output	rotational inertia about z local body axis (mxungp)
ssqval	output	sum of squares statistics array (mxnstk)
stkval	output	stacked value statistics array, (mxstak,mxnstk)
stpval	output	step value statistics array, (mxnstk)
sumval	output	summed value statistics array, (mxnstk)
txx	work	(mxatms)
tyy	work	(mxatms)
tzz	work	(mxatms)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
vxx	output	x component of velocity (mxatms)
vyy	output	y component of velocity (mxatms)

vzz	output	z component of velocity (mxatms)
weight	output	atomic mass (mxatms)
wgtsit	input	site mass (mxsite)
xx0	output	increments in x coordinate from start of statistics period (mxatms)
xxt	work	(mxatms)
xxx	output	x coordinates (mxatms)
yy0	output	increments in y coordinate from start of statistics period (mxatms)
yyt	work	(mxatms)
yyy	output	y coordinates (mxatms)
zumval	output	statistics array, (mxnstk)
zz0	output	increments in z coordinate from start of statistics period (mxatms)
zzt	work	(mxatms)
zzz	output	z coordinates (mxatms)
omx	output	x component of angular velocity (mxgrp)
omy	output	y component of angular velocity (mxgrp)
omz	output	z component of angular velocity (mxgrp)
zdens	output	z density histograms (mxrdf,mxsvdw)
xxs	input	x component of atomic position at $t = 0$ (mxatms)
sys	input	y component of atomic position at $t = 0$ (mxatms)
zsz	input	z component of atomic position at $t = 0$ (mxatms)
elrcm	output	long range correction to Sutton Chen energy (2)
vlrcm	output	long range correction to Sutton Chen virial (2)
eta	output	cell scaling tensor (9)

8.1.180.5 Comments

See section 4.1.2 for description of CONFIG file.

8.1.181 SYSGEN

This description applies to the subroutine SYSGEN in DL_POLY_2 version 2.11 and after.

8.1.181.1 Header records

```

      subroutine sysgen
      x  (loglnk,lneut,atmnam,cfname,sitnam,idnode,imcon,
      x  keyens,keyfce,keyres,levcfg,multt,mxnode,ntpmls,delr,rcut,
      x  volm,lfzsit,lstfrz,ltpsit,ltype,nummols,numsit,nugrp,
      x  lstneu,buffer,cell,chge,chgsit,fx,fy,fz,vx,vy,vz,
      x  weight,wgtsit,xxx,yyy,zzz)

c
c*****
c
c      dl_poly subroutine for reading the configuration data file
c
c      copyright - daresbury laboratory 1997
c      author      - w. smith          july 1997
c
c*****
c

```

8.1.181.2 Function

Reads in initial configuration from the CONFIG file.

8.1.181.3 Dependencies

DBLSTR	DCELL	ERROR
GDSUM	GETREC	GSTATE
STRIP	WARNING	

8.1.181.4 Arguments**logicals:**

loglnk	output	flag for link cell algorithm
lneut	input	flag for neutral group electrostatics

characters:

atmnam	output	atom labels [a8], (mxatms)
cfname	output	configuration name [a80]
sitnam	input	site labels [a8], (mxsite)

integers:

<code>idnode</code>	input	identity of current processor
<code>imcon</code>	output	key for periodic boundary conditions
<code>keyens</code>	input	ensemble key
<code>keyfce</code>	input	key for non-bonded force field
<code>keyres</code>	input	key for restart of MD run
<code>levcfg</code>	output	configuration key
<code>multt</code>	input	multiple timestep interval
<code>mxnode</code>	input	total number of processors
<code>ntpmls</code>	input	number of types of molecules in the system
reals:		
<code>delr</code>	input	border width for Verlet list
<code>rcut</code>	input	potential cut-off radius
<code>volm</code>	output	volume of MD cell
integer arrays:		
<code>lfzsit</code>	input	flag for frozen site (<code>mxsite</code>)
<code>lstfrz</code>	output	key for frozen atoms (<code>mxatms</code>)
<code>ltpsit</code>	input	site type index (<code>mxsite</code>)
<code>ltype</code>	input	list of atom types (<code>mxatms</code>)
<code>nummols</code>	input	number of molecules of each type (<code>mxtmls</code>)
<code>numsit</code>	input	number of sites on a molecule (<code>mxtmls</code>)
<code>nugrp</code>	input	neutral group index (<code>mxsite</code>)
<code>lstneu</code>	output	index of group to which site belongs (<code>mxatms</code>)
real arrays:		
<code>buffer</code>	work	(<code>mxbuff</code>)
<code>cell</code>	output	MD cell vectors (9)
<code>chge</code>	output	atomic charges (<code>mxatms</code>)
<code>chgsit</code>	input	site partial charge (<code>mxsite</code>)
<code>fxx</code>	output	x component of force (<code>mxatms</code>)
<code>fyy</code>	output	y component of force (<code>mxatms</code>)
<code>fzz</code>	output	z component of force (<code>mxatms</code>)
<code>vxx</code>	output	x component of velocity (<code>mxatms</code>)
<code>vyy</code>	output	y component of velocity (<code>mxatms</code>)
<code>vzz</code>	output	z component of velocity (<code>mxatms</code>)
<code>weight</code>	output	atomic mass (<code>mxatms</code>)
<code>wgtsit</code>	input	site mass (<code>mxsite</code>)
<code>xxx</code>	output	x coordinates (<code>mxatms</code>)
<code>yyy</code>	output	y coordinates (<code>mxatms</code>)
<code>zzz</code>	output	z coordinates (<code>mxatms</code>)

8.1.181.5 Comments

See section 4.1.2 for description of CONFIG file.

8.1.182 SYSGEN_RESPA**8.1.182.1 Header records**

```

x (loglnk,lgofr,lneut,lshmov,lzden,atmnam,cfpname,sitnam,
x idnode,imcon,keyens,keyfce,keyres,multt,mxnode,natms,nneut,
x ngrp,nscons,nstep,ntangl,ntbond,ntcons,ntdihd,ntpmls,ntpmf,
x nspmf,ntpvdw,numacc,numrdf,ntpatm,ntteth,ntshl,nzden,chip,
x chit,conint,degfre,degshl,degrot,delr,elrc,engunit,virlrc,
x rcut,rvdw,temp,tolnce,volm,keybnd,keyang,lashap,lexsit,lexatm,
x lfzsit,lislap,listang,listbnd,listcon,listdih,listin,listme,
x listot,lstang,lstbnd,lstcon,lstdih,lstfre,lstfrz,numfrz,
x lstvdw,ltpsit,ltpvdw,ltype,nexatm,nexsit,numang,lstshl,
x numbonds,numcon,numdih,indpmf,lstpmf,numpmf,npmf,lstpmt,listpm,
x nummols,numsit,numteth,numtyp,numshl,lsttet,listtet,
x listshl,nugrp,neulst,lstneu,buffer,cell,chge,chgsit,dens,dxt,
x dyt,dzt,fxf,fyy,fzz,prmdih,prmvdw,ravval,rdf,ssqval,stkval,
x stpval,sumval,uxx,uyy,uzz,vxx,vyy,vzz,weight,wgtsit,xx0,xxt,
x xxx,yy0,yyt,yyy,zumval,zz0,zzt,zzz,zdens,xs,ys,zs,eta)
c
c*****
c
c      dl_poly subroutine for reading in the initial configuration
c      of the system to be simulated and defining global bookkeeping
c      arrays for RESPA scheme
c      t.forester modified from sysgen june 1996
c
c      copyright - daresbury laboratory 1996
c      sysgen author      - w. smith          july 1992
c
c*****
c

```

8.1.182.2 Function

Reads in initial configuration from the CONFIG file and defines global bookkeeping arrays.

8.1.182.3 Dependencies

DCELL	ERROR	EXCLUDE	EXCLUDE_ATOM
EXCLUDE_LINK	EXCLUDENEU	GAUSS	GDSUM
GETREC	GISUM	GSTATE	INTLIST
LRCORRECT	QUENCH	SHLQNCH	STRIP
VSCALE	WARNING		

8.1.182.4 Arguments**logicals:**

<code>loglnk</code>	output	flag for link cell algorithm
<code>lgofr</code>	input	flag for calculation of radial distribution functions
<code>lneut</code>	input	flag for neutral group electrostatics
<code>lshmov</code>	output	flag for shared bond constraints
<code>lzden</code>	input	flag for calculating z density profile

characters:

<code>atmnam</code>	output	atom labels [a8], (<code>mxatms</code>)
<code>cfgname</code>	output	configuration name [a80]
<code>sitnam</code>	input	site labels [a8], (<code>mxsite</code>)

integers:

<code>idnode</code>	input	identity of current processor
<code>imcon</code>	input	key for periodic boundary conditions
<code>keyens</code>	input	ensemble key
<code>keyfce</code>	input	key for non-bonded force field
<code>keyres</code>	input	key for restart of MD run
<code>multt</code>	input	multiple timestep interval
<code>mxnode</code>	input	total number of processors
<code>natms</code>	input	number of atoms in MD cell
<code>nneut</code>	output	number of charge groups in system
<code>ngrp</code>	input	number of rigid body units in system
<code>nscons</code>	output	number of constraints on current node
<code>nstep</code>	input	running counter of completed MD cycles
<code>ntangl</code>	output	total number of valence angles in system
<code>ntbond</code>	output	total number of bonds in system
<code>ntcons</code>	output	total number of constraints in system
<code>ntdihd</code>	output	total number of dihedral angles in the system
<code>ntpmls</code>	input	number of types of molecules in the system
<code>ntpmf</code>	output	total number of PMF constraints in system
<code>nspmf</code>	output	number of PMF constraints on the current processor
<code>ntpvdw</code>	input	number of types of non-bonded potentials
<code>numacc</code>	output	running counter of the number of steps used for statistics
<code>numrdf</code>	output	number of steps used for rdf statistics
<code>ntpatm</code>	output	number of unique atom types
<code>ntteth</code>	output	total number of tethered atoms in system
<code>ntshl</code>	output	total number of core-shell units in system
<code>nzden</code>	output	number of configurations used for z density profiles

reals:

<code>chip</code>	output	coupling parameter for barostat
<code>chit</code>	output	coupling parameter for thermostat
<code>conint</code>	output	time integral of <code>chit</code> and <code>chip</code>
<code>degfre</code>	output	number of thermodynamic degrees of freedom

degshl	output	core-shell psuedo degrees of freedom
degrot	output	rotational degrees of freedom
delr	input	border width for Verlet list
elrc	output	long range correction to energy
engunit	input	energy unit for input /output
virlrc	output	long range correction to virial
rcut	input	potential cut-off radius
rvdw	input	Van der Waals potential cut-off radius
temp	input	simulation temperature
tolnce	input	tolerance for SHAKE algorithm
volm	output	volume of MD cell
integer arrays:		
keybnd	input	chemical bond potential key (mxtbnd)
keyang	input	angle potential key (mxtang)
lashap	output	see RDSHAKE_1 (mxproc)
lexsit	output	"excluded site" list for pair forces (mxsite,mxexcl)
lexatm	output	"excluded atom" list (msatms,mxexcl)
lfzsit	input	flag for frozen site (mxsite)
lishap	output	see RDSHAKE_1 (mxlshp)
listang	input	key and indices for valence angle interactions (mxangl,4)
listbnd	input	key and indices for chemical bond interactions (mxbond,3)
listcon	input	indices for constraint bonds (mxcons,3)
listdih	input	key and indices for dihedral interactions (mxdihd,5)
listin	work	(mxatms)
listme	input	see RDSHAKE_1 (mxatms)
listot	input	see RDSHAKE_1 (mxatms)
lstang	input	Valence angle potential site list (mxtang,3)
lstbnd	input	Chemical bond potential site list (mxtbnd,2)
lstcon	input	constraint bond site list (mxtcon,2)
lstdih	input	dihedral angle potential site list (mxt dih,4)
lstfre	input	indices of non-rigid unit sites handled by current processor (mxatms)
numfrz	output	number frozen atoms of each type (mxsvdw)
lstfrz	output	key for frozen atoms (mxatms)
lstvdw	input	index number of non-bonded potential (mxvdw)
ltpsit	input	site type index (mxsite)
ltpvdw	input	key for non-bonded potentials (mxvdw)
ltype	input	list of atom types (mxatms)
nexatm	output	number of excluded atoms (msatms)
nexsit	output	number of excluded sites (mxsite)
numang	input	number of valence angles in a molecule (mxtmls)
lstshl	input	site indices for core-shell units (mxtshl,2)
numbonds	input	number of chemical bonds in a molecule (mxtmls)
numcon	input	number of constraint bonds in a molecule (mxtmls)

numdih	input	number of dihedral interactions in a molecule (mxtmls)
indpmf	input	indices of sites in PMF constraints (mxspmf)
lstpmf	output	indices of atoms in PMF constraints on this processor (mxspmf , mshpmf)
numpmf	input	number of PMF constraints in each molecular type (mxtmls)
npmf	input	number of sites in PMF units (2)
lstpmt	output	flag of all atoms in system in PMF constraints (mxpmf)
listpm	output	flag of all atoms in PMF constraints on this processor (mxpmf)
nummols	input	number of molecules of each type (mxtmls)
numsit	input	number of sites on a molecule (mxtmls)
numteth	input	number of tethered sites in each molecular type (mxtmls)
numtyp	output	number of each atom type (mxsvdw)
numshl	input	number of core-shell units in each molecular type (mxtmls)
lstteth	input	indices of tethered sites (mxsteth)
listteth	output	indices of tethered atoms on current processor (msteth ,2)
listshl	output	indices for core-shell interactions on current processor (mxshl ,3)
nugrp	input	neutral group index (mxsite)
neulst	output	indice of first site in charge group (mxneut)
lstneu	output	index of group to which site belongs (mxatms)
real arrays:		
buffer	work	(mxbuff)
cell	input	MD cell vectors (9)
chge	output	atomic charges (mxatms)
chgsit	input	site partial charge (mxsite)
dens	output	number density (mxsvdw)
dxt	work	(mxcons)
dzt	work	(mxcons)
dyt	work	(mxcons)
dzx	work	(mxcons)
fx	output	x component of force (mxatms)
fy	output	y component of force (mxatms)
fz	output	z component of force (mxatms)
prmdih	input	dihedral angle potential parameters (mxtdih , mxpdih)
prmvdw	input	parameters for non-bonded potentials (mxvdw , mxpvdw)
ravval	output	rolling averages statistics array (mxnstk)
rdf	output	histograms for rdfs (mxrdf , mxvdw)
rotinx	output	rotational inertia about x local body axis (mxungp)
rotiny	output	rotational inertia about y local body axis (mxungp)
rotinz	output	rotational inertia about z local body axis (mxungp)
ssqval	output	sum of squares statistics array (mxnstk)
stkval	output	stacked value statistics array, (mxstak , mxnstk)
stpval	output	step value statistics array, (mxnstk)
sumval	output	summed value statistics array, (mxnstk)
txx	work	(mxatms)
tyy	work	(mxatms)

tzz	work	(mxatms)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
vxx	output	x component of velocity (mxatms)
vyy	output	y component of velocity (mxatms)
vzz	output	z component of velocity (mxatms)
weight	output	atomic mass (mxatms)
wgtsit	input	site mass (mxsite)
xx0	output	increments in x coordinate from start of statistics period (mxatms)
xxt	work	(mxatms)
xxx	output	x coordinates (mxatms)
yy0	output	increments in y coordinate from start of statistics period (mxatms)
y yt	work	(mxatms)
yyy	output	y coordinates (mxatms)
zumval	output	statistics array, (mxnstk)
zz0	output	increments in z coordinate from start of statistics period (mxatms)
zzt	work	(mxatms)
zzz	output	z coordinates (mxatms)
zdens	output	z density histograms (mxrdf,mxsvdw)
xxs	input	x component of atomic position at $t = 0$ (mxatms)
yy s	input	y component of atomic position at $t = 0$ (mxatms)
z z s	input	z component of atomic position at $t = 0$ (mxatms)
eta	output	cell scaling tensor (9)

8.1.182.5 Comments

See section 4.1.2 for description of CONFIG file. This routine has reduced functionality commensurate with the r-RESPA integration scheme, in particular, the rigid body functionality is not present.

8.1.183 SYSINIT**8.1.183.1 Header records**

```

      subroutine sysinit
      x  (lgofr,lzden,lmatal,idnode,imcon,keyfce,keyres,mxnode,
      x  natms,nstep,numacc,numrdf,ntpatm,nzden,chip,chit,
      x  conint,elrc,engunit,virlrc,rvdw,volm,lstvdw,ltpvdw,lstfrz,
      x  ltype,numtyp,numfrz,buffer,cell,dens,prmvdw,ravval,rdf,
      x  ssqval,stkval,stpval,sumval,xx0,yy0,zz0,zumval,zdens,
      x  xxs,yyz,zzs,elrcm,vlrcm,eta)

c
c*****
c
c      dl_poly subroutine for reading the REVIVE file data and
c      defining the initial thermodynamic and structural accumulators.
c
c      copyright - daresbury laboratory 1997
c      author    - w. smith          july 1997
c
c*****
c

```

8.1.183.2 Function

Reads in REVIVE data file from a previous DL_POLY_2 run and sets the thermodynamic and correlation arrays at the start of the run.

8.1.183.3 Dependencies

ERROR	GDSUM	GISUM
LRCMETAL	LRCORRECT	

8.1.183.4 Arguments**logicals:**

lgofr	input	flag for calculation of radial distribution functions
lzden	input	flag for calculating z density profile
lmatal	input	flag for metal (Sutton Chen) potentials

integers:

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
keyfce	input	key for non-bonded force field
keyres	input	key for restart of MD run

mxnode	input	total number of processors
natms	input	number of atoms in MD cell
nstep	input	running counter of completed MD cycles
numacc	output	running counter of the number of steps used for statistics
numrdf	output	number of steps used for rdf statistics
ntpatm	output	number of unique atom types
nzden	output	number of configurations used for z density profiles
reals:		
chip	output	coupling parameter for barostat
chit	output	coupling parameter for thermostat
conint	output	time integral of chit and chip
elrc	output	long range correction to energy
engunit	input	energy unit for input /output
virlrc	output	long range correction to virial
rvdw	input	Van der Waals potential cut-off radius
volm	output	volume of MD cell
integer arrays:		
lstvdw	input	index number of non-bonded potential (mxvdw)
ltpvdw	input	key for non-bonded potentials (mxvdw)
lstfrz	output	key for frozen atoms (mxatms)
ltype	input	list of atom types (mxatms)
numtyp	output	number of each atom type (mxsvdw)
numfrz	output	number of frozen atoms of each type (mxsvdw)
real arrays:		
buffer	work	(mxbuff)
cell	input	MD cell vectors (9)
dens	output	number density (mxsvdw)
prmvdw	input	parameters for non-bonded potentials (mxvdw , mxpvdw)
ravval	output	rolling averages statistics array (mxnstk)
rdf	output	histograms for rdfs (mxrdf , mxvdw)
ssqval	output	sum of squares statistics array (mxnstk)
stkval	output	stacked value statistics array, (mxstak , mxnstk)
stpval	output	step value statistics array, (mxnstk)
sumval	output	summed value statistics array, (mxnstk)
xx0	output	increments in x coordinate from start of statistics period (mxatms)
yy0	output	increments in y coordinate from start of statistics period (mxatms)
zz0	output	increments in z coordinate from start of statistics period (mxatms)
zumval	output	statistics array, (mxnstk)
zdens	output	z density histograms (mxrdf , mxsvdw)
xxs	input	x component of atomic position at $t = 0$ (mxatms)
sys	input	y component of atomic position at $t = 0$ (mxatms)
zss	input	z component of atomic position at $t = 0$ (mxatms)
elrcm	output	long range correction to Sutton Chen energy (2)
vlrcm	output	long range correction to Sutton Chen virial (2)

eta output cell scaling tensor (9)

8.1.183.5 Comments

8.1.184 SYSTEMP**8.1.184.1 Header records**

```

      subroutine systemp
      x  (idnode,imcon,keyens,keyres,mxnode,natms,ngrp,nscons,
      x  ntcons,ntfree,ntshl,levcfg,degfre,degshl,degrot,temp,
      x  tolnce,lashap,lishap,listcon,listme,listot,lstfrz,lstgtp,
      x  lstrgd,lstme,numgsit,listshl,buffer,cell,dxt,dyt,dzt,
      x  fxx,fyy,fzz,gcmx,gcmx,gcmz,gmass,gvxx,gvyy,gvzz,
      x  gxx,gyy,gzz,q0,q1,q2,q3,rotinx,rotiny,rotinz,weight,
      x  uxx,uyy,uzz,vxx,vyy,vzz,txt,xxx,yyt,yyy,zzt,zzz,
      x  omx,omy,omz)

c
c*****
c
c      dl_poly subroutine for setting the initial system temperature
c
c      copyright - daresbury laboratory 1997
c      author      - w. smith          july 1997
c
c*****
c

```

8.1.184.2 Function

Initialises the system velocities and sets temperature.

8.1.184.3 Dependencies

ERROR	GAUSS	QUATQNCH
QUENCH	SHLQNCH	VSCALEG

8.1.184.4 Arguments**integers:**

idnode	input	identity of current processor
imcon	input	key for periodic boundary conditions
keyens	input	ensemble key
keyres	input	key for restart of MD run
mxnode	input	total number of processors
natms	input	number of atoms in MD cell
ngrp	input	number of rigid body units in system
nscons	output	number of constraints on current node

ntcons	output	total number of constraints in system
ntfree	output	number of sites in system not in rigid body units
ntteth	output	total number of tethered atoms in system
ntshl	output	total number of core-shell units in system
levcfg	input	configuration key
reals:		
degfre	output	number of thermodynamic degrees of freedom
degshl	output	core-shell psuedo degrees of freedom
degrot	output	rotational degrees of freedom
temp	input	simulation temperature
tolnce	input	tolerance for SHAKE algorithm
integer arrays:		
lashap	output	see RDSHAKE_1 (mxproc)
lishap	output	see RDSHAKE_1 (mxlshp)
listcon	input	indices for constraint bonds (mxcons ,3)
listme	input	see RDSHAKE_1 (mxatms)
listot	input	see RDSHAKE_1 (mxatms)
lstfrz	output	key for frozen atoms (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgadm)
lstme	output	indice list for MERGE1 (mxatms)
numgsit	input	number of sites in a rigid unit type (mxungp)
listshl	output	indices for core-shell interactions on current processor (mxshl ,3)
real arrays:		
buffer	work	(mxbuff)
cell	input	MD cell vectors (9)
dxt	work	(mxcons)
dyt	work	(mxcons)
dzt	work	(mxcons)
fx	output	x component of force (mxatms)
fy	output	y component of force (mxatms)
fz	output	z component of force (mxatms)
gcmx	output	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	output	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	output	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	output	rigid unit mass (mxungp)
gvxx	output	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	output	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	output	z component of rigid unit c.o.m. velocity (mxgrp)
gxx	output	x component of site in local body frame (mxungp , mxngp)
gyy	output	y component of site in local body frame (mxungp , mxngp)
gzz	output	z component of site in local body frame (mxungp , mxngp)
q0	output	first component of quaternion vector (mxgrp)

q1	output	second component of quaternion vector (mxgrp)
q2	output	third component of quaternion vector (mxgrp)
q3	output	fourth component of quaternion vector (mxgrp)
rotinx	output	rotational inertia about x local body axis (mxungp)
rotiny	output	rotational inertia about y local body axis (mxungp)
rotinz	output	rotational inertia about z local body axis (mxungp)
weight	output	atomic mass (mxatms)
uxx	work	(mxatms)
uyy	work	(mxatms)
uzz	work	(mxatms)
vxx	output	x component of velocity (mxatms)
vyy	output	y component of velocity (mxatms)
vzz	output	z component of velocity (mxatms)
txt	work	(mxatms)
xxx	output	x coordinates (mxatms)
yyt	work	(mxatms)
yyy	output	y coordinates (mxatms)
zzt	work	(mxatms)
zzz	output	z coordinates (mxatms)
omx	output	x component of angular velocity (mxgrp)
omy	output	y component of angular velocity (mxgrp)
omz	output	z component of angular velocity (mxgrp)

8.1.184.5 Comments

8.1.185 TETHFRC**8.1.185.1 Header records**

```

      subroutine tethfrc
      x  (idnode,mxnode,imcon,natms,nstep,ntteth,engtet,virtet,
      x  keytet,listtet,buffer,cell,fx,fy,fz,prmtet,xxx,yyy,
      x  zzz,xs,ys,zs,xdab,ydab,zdab,stress)

C*****
C
C      dl_poly routine to tether atoms to initial positions
C      includes stress tensor
C
C      replicated data version : block data
C
C      copyright daresbury laboratory 1994
C      author      t.forester feb 1994
C      amended     t.forester dec 1994 : block data
C
C*****

```

8.1.185.2 Function

Tethers atoms to a given position in the simulation cell with a user-specified tethering potential.

8.1.185.3 Dependencies

- ERROR
- GDSUM
- GSTATE
- IMAGES

8.1.185.4 Arguments**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors
imcon	input	key for periodic boundary conditions
natms	input	total number of atoms in the MD cell
nstep	input	MD step number
natms	input	total number of atoms in the MD cell
ntteth	input	total number of tethered atoms in system

reals:

engtet	output	tethering potential energy
virtet	output	tethering virial

integer arrays:

keytet	input	key for tether potential (mxttet)
listtet	input	index for potential and atomic coordinates (mxteth,2)

real arrays:

buffer	work	(mxbuff)
cell	input	MD cell vectors (9)
fxx	output	x component of force (mxatms)
fyy	output	y component of force (mxatms)
fzz	output	z component of force (mxatms)
prmtet	input	potential variables (mxteth,mxpbnd)
xxx	input	x coordinate (mxatms)
yyy	input	y coordinate (mxatms)
zzz	input	z coordinate (mxatms)
xxs	in/out	x component of atomic position at $t = 0$ (mxatms)
yyz	in/out	y component of atomic position at $t = 0$ (mxatms)
zzs	in/out	z component of atomic position at $t = 0$ (mxatms)
xdab	work	x component of a-b inter atomic vector (msbad)
ydab	work	y component of a-b inter atomic vector (msbad)
zdab	work	z component of a-b inter atomic vector (msbad)
stress	in/out	stress tensor (9)

8.1.185.5 Comments

8.1.186 THBFRC**8.1.186.1 Header records**

```

      subroutine thbfrc
      x  (idnode,mxnode,natms,imcon,rcuttb,engtbp,virtbp,
      x  latinx,ltype,lst,lct,link,lsttbp,ltptbp,prmtbp,cell,
      x  xxx,yyy,zzz,fx,fy,fz,rcut3b,stress)
c
c*****
c
c      dl_poly subroutine for calculating three body forces arising
c      from the angle between three atoms
c
c      Note: the subroutine converts coordinates to reduced units
c      to avoid a call to images.f. The link cell algorithm used
c      here necessitates a parallelepiped cell geometry
c
c      copyright - daresbury laboratory 1994
c      author    - w.smith march 1994
c
c*****
c

```

8.1.186.2 Function

Evaluates energy and forces from three body potentials of the valence angle type.

8.1.186.3 Dependencies

- DCELL
- ERROR
- GDSUM
- GSTATE
- INVERT

8.1.186.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	total number of atoms in the MD cell
imcon	input	key for periodic boundary conditions

reals:

rcuttb	output	three body potential cut-off
engtbp	output	three body potential energy
virtbp	output	three body potential virial

integer arrays:

latinx	work	link cell number (mxatms)
ltype	input	index of atomic types (mxatms)
lst	work	number of atoms in link cell (mxcell)
lct	work	head of chain in link cell (mxcell)
link	work	pointer to next atom in link cell (mxatms)
lsttbp	input	pointer from three body potential to parameter table (mxtbp)
ltptbp	input	key for three body potentials (mxtbp)
prmtbp	input	three body potential parameters (mxtbp , m xpang)
cell	input	MD cell vectors (9)
xxx	input	x coordinates (mxatms)
yyy	input	y coordinates (mxatms)
zzz	input	z coordinates (mxatms)
fxx	in/out	x component of force (mxatms)
fyy	in/out	y component of force (mxatms)
fzz	in/out	z component of force (mxatms)
rcut3b	input	three body potential cutoffs (mxtbp)
stress	in/out	stress tensor (9)

real arrays:**8.1.186.5 Comments**

Note this is **not** a general three body potential routine. It is restricted to valence angle potentials only. Uses the link-cell algorithm to determine the atomic triplets.

8.1.187 TIMCHK**8.1.187.1 Header records**

```

      subroutine timchk(ktim,time)
c
c*****
c
c      dlpoly timing routine.
c      time elapsed in seconds
c      makes use of intrinsic function 'etime' for non MPI version
c
c      copyright daresbury laboratory sept 1994
c      author t.forester sept 1994
c
c*****
c

```

8.1.187.2 Function

Gives the elapsed cpu time in seconds.

8.1.187.3 Dependencies

- system clock
- GSYNC
- MYNODE

8.1.187.4 Arguments

integers:

ktim	input	printing key

reals:

time	output	elapsed cpu time

8.1.187.5 Comments

Setting argument `ktim > 0` will result in a line being printed in the OUTPUT file stating the elapsed cpu time.

C preprocessing is used to select the appropriate system clock.

8.1.188 TRAJECT**8.1.188.1 Header records**

```

      subroutine traject
      x  (ltraj,cfname,atmnam,idnode,imcon,istray,keytrj,natms,
      x  nstraj,nstep,cell,chge,weight,xxx,yyy,zzz,vxx,vyy,vzz,
      x  fxx,fyy,fzz)
c
c*****
c
c      dl_poly subroutine for writing history file at selected
c      intervals in simulation
c
c      copyright - daresbury laboratory 1992
c      author    - w. smith dec 1992.
c
c*****
c

```

8.1.188.2 Function

Writes out the HISTORY file.

8.1.188.3 Dependencies

- none

8.1.188.4 Arguments**logicals:**

ltraj	input	flag for writing of HISTORY file

characters:

cfname	input	configuration name [a80]
atmnam	input	atom names [a8], (mxatms)

integers:

idnode	input	identity of current processor
imcon	input	periodic boundary key
istray	input	interval (timesteps) for writing file
keytrj	input	trajectory file information key
natms	input	number of atoms in MD cell
nstraj	input	start timestep of HISTORY file
nstep	input	current timestep

real arrays:

cell	input	cell dimensions (9)
chge	input	atomic charges (mxatms)

weight	input	atomic masses (mxatms)
xxx	input	x coordinates (mxatms)
yyy	input	y coordinates (mxatms)
zzz	input	z coordinates (mxatms)
vxx	input	x component of velocity (mxatms)
vyy	input	y component of velocity (mxatms)
vzz	input	z component of velocity (mxatms)
fxx	input	x component of force (mxatms)
fyy	input	y component of force (mxatms)
fzz	input	z component of force (mxatms)

8.1.188.5 Comments

The default version writes a formatted HISTORY file. The subroutine `TRAJECT_U` writes an unformatted HISTORY. See section 4.2.1 for a description of the formats.

8.1.189 VERTEST**8.1.189.1 Header records**

```

      subroutine vertest
      x    (newlst,idnode,mxnode,natms,delr,tstep,vxx,vyy,vzz)
c
c*****
c
c    DL_POLY subroutine to test for updating of Verlet list
c    replicated data version
c
c    copyright daresbury laboratory 1993
c    author -      t. forester may 1993
c
c*****
c

```

8.1.189.2 Function

Tests if the Verlet list needs to be (re)constructed. The list is constructed at the beginning of a job and reconstructed whenever any two particles have moved a distance greater than ($\text{delr}/2$) since the most recent construction of the list.

8.1.189.3 Dependencies

- ERROR
- GISUM

8.1.189.4 Arguments**logicals:**

newlst	output	flag for construction of new Verlet list

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	number of atoms in the MD cell

reals:

delr	input	border width of neighbourhood shell
tstep	input	timestep

real arrays:

vxx	input	x component of velocity (mxatms)
vyy	input	y component of velocity (mxatms)
vzz	input	z component of velocity (mxatms)

8.1.189.5 Comments

This routine must be called each timestep as it integrates the velocity to find atomic displacements. The accumulated displacements are held in local arrays **xsum**, **ysum** and **zsum** each of which is of dimension (**msatms**).

8.1.190 VERTEST1**8.1.190.1 Header records**

```

      subroutine vertest1
      x  (newlst,idnode,mxnode,imcon,natms,delr,cell,
      x  xxx,yyy,zzz,xdf,ydf,zdf)

C*****
C
C    DL_POLY subroutine to test for updating of Verlet list
C    replicated data version
C
C    based on atomic displacements
C
C    copyright daresbury laboratory 1993
C    author -      t. forester may 1993
C
C    $Author: trf $
C    $Date: 1996/08/19 10:30:10 $
C    $Revision: 1.1 $
C    $State: Exp $
C
C*****

```

8.1.190.2 Function

Tests if the Verlet list needs to be (re)constructed. The list is constructed at the beginning of a job and reconstructed whenever any two particles have moved a distance greater than ($\text{delr}/2$) since the most recent construction of the list.

8.1.190.3 Dependencies

- ERROR
- GISUM
- IMAGES

8.1.190.4 Arguments**logicals:**

newlst	output	flag for construction of new Verlet list

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use

<code>imcon</code>	input	periodic image convention key
<code>natms</code>	input	number of atoms in the MD cell
reals:		
<code>delr</code>	input	border width of neighbourhood shell
real arrays:		
<code>cell</code>	input	MD cell vectors
<code>xxx</code>	input	x coordinates (<code>mxatms</code>)
<code>yyy</code>	input	y coordinates (<code>mxatms</code>)
<code>zzz</code>	input	z coordinates (<code>mxatms</code>)

8.1.190.5 Comments

The subroutine VERTEST, which integrates velocities, is an alternative to this one.

8.1.191 VSCALE**8.1.191.1 Header records**

```

      subroutine vscale
      x (idnode,mxnode,imcon,natms,degfre,temp,lstfrz,
buffer,cell,weight,vxx,vyy,vzz,xxx,yyy,zzz)

c
c*****
c
c      dl_poly subroutine for scaling the velocity arrays to the
c      desired temperature
c      Only for systems without rigid body units
c
c      zeroes angular momentum in non-periodic system.
c
c      parallel replicated data version : block data
c
c      copyright daresbury laboratory 1992.
c      author - w.smith july 1992
c      amended - t.forester oct 1993
c      amended - t.forester dec 1994 : block data
c
c*****
c

```

8.1.191.2 Function

Scales velocities to give desired temperature. Also zeroes angular momentum in non-periodic systems.

8.1.191.3 Dependencies

- GDSUM
- INVERT
- MERGE

8.1.191.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors
imcon	input	periodic boundary key

natms	input	number of atoms in MD cell
reals:		
degfre	input	degrees of freedom
temp	input	desired temperature
integer arrays:		
lstfrz	input	key for frozen atoms (mxatms)
real arrays:		
buffer	work	(mxbuff)
cell	input	MD cell vectors
weight	input	atomic masses (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
xxx	input	x coordinate (mxatms)
yyy	input	y coordinate (mxatms)
zzz	input	z coordinate (mxatms)

8.1.191.5 Comments

8.1.192 VSCALEG**8.1.192.1 Header records**

```

      subroutine vscaleg
      x  (idnode,mxnode,imcon,natms,ngroup,sigma,lstfrz,lstgtp,lstrgd,
      x  numgsit,lstme,buffer,cell,gcmx,gcmz,gcmz,gmass,
      x  gvxx,gvyy,gvzz,gxx,gyy,gzz,q0,q1,q2,q3,omx,omy,omz,weight,
      x  vxx,vyy,vzz,xxx,yyy,zzz,txt,yty,zty,rotinx,rotiny,rotinz)
c
c*****
c
c      dl_poly subroutine for scaling the velocity arrays to the
c      desired temperature
c
c      zeroes angular momentum in non-periodic system.
c
c      parallel replicated data version : block data
c
c      copyright daresbury laboratory 1992.
c      author - w.smith july 1992
c      amended - t.forester oct 1993
c      amended - t.forester dec 1994 : block data
c
c*****
c

```

8.1.192.2 Function

Scales velocities to give desired temperature. Also zeroes angular momentum in non-periodic systems and calculates rigid unit c.o.m and rotational velocities from atomic contributions.

8.1.192.3 Dependencies

- GDSUM
- INVERT
- QUATQNCH

8.1.192.4 Arguments

integers:

idnode	input	identity of current processor
mxnode	input	number of processing nodes in use

imcon	input	periodic boundary key
natms	input	number of atoms in MD cell
ngrp	input	number of rigid body units in system
reals:		
sigma	input	required kinetic energy
integer arrays:		
lstfrz	input	key for frozen atoms (mxatms)
lstgtp	input	pointer from rigid unit index to rigid unit type (mxgrp)
lstrgd	input	indices of rigid unit sites handles by current processor (mxgadm)
numgsit	input	number of sites in a rigid unit type (mxungp)
lstme	input	indice list for MERGE1 (mxatms)
real arrays:		
buffer	work	(mxbuff)
weight	input	atomic masses (mxatms)
cell	input	MD cell vectors (9)
gcmx	input	x coordinate of rigid unit c.o.m. (mxgrp)
gcmy	input	y coordinate of rigid unit c.o.m. (mxgrp)
gcmz	input	z coordinate of rigid unit c.o.m. (mxgrp)
gmass	input	rigid unit mass (mxungp)
gvxx	in/out	x component of rigid unit c.o.m. velocity (mxgrp)
gvyy	in/out	y component of rigid unit c.o.m. velocity (mxgrp)
gvzz	in/out	z component of rigid unit c.o.m. velocity (mxgrp)
gxx	input	x component of site in local body frame (mxungp,mxngp)
gyy	input	y component of site in local body frame (mxungp,mxngp)
gzz	input	z component of site in local body frame (mxungp,mxngp)
q0	input	first component of quaternion vector (mxgrp)
q1	input	second component of quaternion vector (mxgrp)
q2	input	third component of quaternion vector (mxgrp)
q3	input	fourth component of quaternion vector (mxgrp)
omx	in/out	x component of angular velocity (mxgrp)
omy	in/out	y component of angular velocity (mxgrp)
omz	in/out	z component of angular velocity (mxgrp)
weight	input	atomic masses (mxatms)
vxx	in/out	x component of velocity (mxatms)
vyy	in/out	y component of velocity (mxatms)
vzz	in/out	z component of velocity (mxatms)
xxx	input	x coordinate (mxatms)
yyy	input	y coordinate (mxatms)
zzz	input	z coordinate (mxatms)
xxt	work	(mxatms)
yyt	work	(mxatms)
zzt	work	(mxatms)
rotinx	input	rotational inertia about x local body axis (mxungp)

rotiny	input	rotational inertia about y local body axis (mxungp)
rotinz	input	rotational inertia about z local body axis (mxungp)

8.1.192.5 Comments

8.1.193 WARNING**8.1.193.1 Header records**

```

      subroutine warning(idnode,kode,a,b,c)
c
c*****
c
c      dl_poly subroutine for printing warning messages and returning
c      control back to the main program
c
c      copyright - daresbury laboratory 1994
c      author    - t. forester    april 1994
c
c*****
c

```

8.1.193.2 Function

Prints warning messages.

8.1.193.3 Dependencies

- none

8.1.193.4 Arguments**integers:**

idnode	input	identity of current processor
kode	input	message number

reals:

a	input	first data item
b	input	second data item
c	input	third data item

8.1.193.5 Comments

8.1.194 XSCALE**8.1.194.1 Header records**

```

      subroutine xscale
      x  (idnode,mxnode,natms,keyens,imcon,tstep,
      x  cell,xxx,yyy,zzz,weight,eta,buffer)
c
c*****
c
c      dl_poly routine to scale positions with change in box shape
c
c      parallel replicated data version
c
c      copyright daresbury laboratory 1995
c      author t.forester      october 1995
c
c*****
c

```

8.1.194.2 Function

Adjusts coordinates for changes to box shape and size.

8.1.194.3 Dependencies

- IMAGES
- MERGE

8.1.194.4 Arguments**integers:**

idnode	input	identity of current processor
mxnode	input	total number of processors in use
natms	input	total number of atoms in the MD cell
keyens	input	key for integration algorithm (ensemble)
imcon	input	key for periodic boundary conditions

reals:

tstep	input	simulation time-step, Δt

real arrays:

cell	input	MD cell vectors (9)
xxx	in/out	x coordinates (mxatms)
yyy	in/out	y coordinates (mxatms)
zzz	in/out	z coordinates (mxatms)
weight	input	atomic masses (mxatms)

eta	input	cell scaling tensor (9)
buffer	work	(mxbuff)

8.1.194.5 Comments

The scaling algorithm used depends upon whether a Berendsen or Hoover barostat is in use. The subroutine is intended for reference coordinates not updated by the integration routine, such as the tethering atom reference positions.

8.1.195 ZDEN0**8.1.195.1 Header records**

```

      subroutine zden0
      x  (idnode,natms,mxnode,nzden,zlen,ltype,zzz,zdens)
c
c*****
c
c      dl_poly subroutine for accumulating statistic for density profile
c      zlen = length of cell in z direction
c
c      double precision accumulators
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1994
c      author    - t. forester    march 1994
c
c*****
c

```

8.1.195.2 Function

Accumulates Z density profile statistics.

8.1.195.3 Dependencies

- none

8.1.195.4 Arguments**integers:**

idnode	input	identity of current processor
natms	input	total number of atoms in the MD cell
mxnode	input	total number of processors in use
nzden	input	number of configurations used for z density profiles

reals:

zlen	input	height of cell in Z direction

integer arrays:

ltype	input	index of atomic types (mxatms)

real arrays:

zzz	input	z coordinates (mxatms)
zdens	output	z density histograms (mxrdf,mxsvdw)

8.1.195.5 Comments

8.1.196 ZDEN1**8.1.196.1 Header records**

```

      subroutine zden1
      x  (lpgr,cfgrname,unqatm,idnode,mxnode,ntpatm,nzden,
      x  volm,zlen,zdens,buffer)
c
c*****
c
c      dl_poly subroutine for calculating Z density profile
c      from accumulated data.
c      double precision version
c
c      parallel replicated data version
c
c      copyright - daresbury laboratory 1994
c      author    - t. forester    march 1994
c
c*****
c

```

8.1.196.2 Function

Calculates and prints Z density profiles.

8.1.196.3 Dependencies

- GDSUM

8.1.196.4 Arguments**logicals:**

lpgr	input	flag for printing functions

characters:

cfgrname	input	simulation header [a80]
unqatm	input	atom type labels [a8] (mxsvdw)

integers:

idnode	input	identity of current processor
mxnode	input	total number of processors in use
ntpatm	input	number of unique types of atoms
nzden	input	number of configurations used for z density profiles

reals:

volm	input	volume of MD cell
zlen	input	height of cell in Z direction

real arrays:

zdens	output	z density histograms (mxrdf,mxsvdw)
buffer	work	(mxbuff)

8.1.196.5 Comments

8.1.197 Communications Routines

In the parallel implementation DL_POLY_2 makes use of a number of communications (message passing) routines, for which there are several sources. DL_POLY_2 is therefore provided in a form which allows users to select the ones most appropriate to their hardware. This is done at compile time, by selecting the appropriate keywords with the `makefile` (see section 3.2.1). The options available include both message passing harnesses and native routines specific to a given parallel machine. The options available are as follows:

1. MPI (*Message Passing Interface*). This message passing harness is our recommended choice, since it is simple to implement and is also the emerging standard. However, it is not yet universally available, nor is it always the most efficient option. It looks the best bet for long term portability.
2. PVM (*Parallel Virtual Machine*). This option was removed from DL_POLY_2 with the release of version 2.13.
3. Intel native message passing routines. DL_POLY_2 was developed on an Intel iPSC/860, for which the message passing utilities were part of the FORTRAN. There was no parallel harness as such, everything sat nicely on the *Direct Connect* hardware. We strongly recommend the use of the Intel native communications routines if DL_POLY_2 is run on Intel hardware.
4. Cray SHMEM. Though MPI message passing is available for the Cray T3D, our experience shows that there is a lot to be gained by exploiting the memory-to-memory data passing of the T3D architecture, particularly in regard to latency. DL_POLY_2 therefore makes use of these in conjunction with MPI.

The specific routines exploited by DL_POLY_2 are briefly outlined below. The descriptions are far from comprehensive and the user is recommended to consult the proprietary documentation in all cases. The MPI and Intel options are mutually exclusive and the SHMEM option requires the user to be running MPI.

It should be noted that when running DL_POLY_2 on a single processor machine, these routines are not required. In which case, the code should be compiled using the SERIAL option. See section 3.2.1 for details of compilation.

8.1.197.1 MPI Routines

The MPI routines required by DL_POLY_2 are as follows:

- MPI_ALLREDUCE
- MPI_BARRIER
- MPI_COMM_RANK
- MPI_COMM_SIZE

- `MPI_FINALIZE`
- `MPI_INIT`
- `MPI_RECV`
- `MPI_SEND`

Note that the MPI routines require the user to specify data types and possibly operator types (see below). These have special forms, please refer to standard MPI documentation for more detail.

MPI_ALLREDUCE

`MPI_ALLREDUCE` is used by `DL_POLY_2` to determine the global sum of an array or the global maximum. It is called with the following arguments:

```
MPI_ALLREDUCE(aaa, bbb, nnn, d_type, o_type, c_handle, ierr)
```

where

- *aaa* - send buffer
- *bbb* - receive array
- *nnn* - data count
- *d_type* - data type (real, integer etc.)
- *o_type* - operation type (add, maximum etc.)
- *c_handle* - communicator handle
- *ierr* - error flag

Note these operations are not restricted to hypercube architectures.

MPI_BARRIER

`MPI_BARRIER` establishes a synchronisation point in a parallel code. It is called with the following arguments:

```
MPI_BARRIER(c_handle, ierr)
```

where

- *c_handle* - communicator handle
- *ierr* - error flag

MPI_COMM_RANK

MPI_COMM_RANK is used by DL_POLY_2 to determine the identity of the node running a process. It is called with the following arguments:

MPI_COMM_RANK(*c_handle*, *mynode*, *ierr*)

where

- *c_handle* - communicator handle
- *mynode* - identity of node
- *ierr* - error flag

MPI_COMM_SIZE

MPI_COMM_SIZE is used by DL_POLY_2 to determine the number of nodes the program is running on. It is called with the following arguments:

MPI_COMM_SIZE(*c_handle*, *numnodes*, *ierr*)

where

- *c_handle* - communicator handle
- *numnodes* - identity of node
- *ierr* - error flag

MPI_FINALIZE

MPI_FINALIZE closes down all MPI activities. It is called with the following arguments:

MPI_FINALIZE(*ierr*)

where

- *ierr* - error flag

MPI_INIT

MPI_INIT initialises MPI. It is called with the following arguments:

MPI_INIT(*ierr*)

where

- *ierr* - error flag

MPI_RECV

MPI_RECV receives incoming messages (blocked receive) . It is called with the following arguments:

MPI_RECV(*aaa*, *nnn*, *d_type*, *idnode*, *m_tag*, *c_handle*, *status*, *ierr*)

where

- *aaa* - receive buffer
- *nnn* - data count
- *d_type* - data type (integer, real etc.)
- *idnode* - identity of sending node
- *m_tag* - message tag (identifier)
- *c_handle* - communicator handle
- *status* - status buffer
- *ierr* - error flag

MPI_SEND

MPI_SEND send outgoing messages (blocked send). It is called with the following arguments:

MPI_SEND(*aaa*, *nnn*, *d_type*, *idnode*, *m_tag*, *c_handle*, *ierr*)

where

- *aaa* - outgoing buffer
- *nnn* - data count
- *d_type* - data type (integer, real etc.)
- *idnode* - identity of receiving node
- *m_tag* - message tag (identifier)
- *c_handle* - communicator handle
- *ierr* - error flag

8.1.197.2 Intel Routines

The Intel routines required are as follows:

- CSEND
- CRECV
- GSYNC
- GDSUM
- GIMAX
- GISUM
- IRECV
- MYNODE
- NODEDIM
- NUMNODES

CSEND

CSEND is a subroutine that sends *synchronous* messages (usually arrays) from one processing node to another. It is called with the following arguments:

CSEND(id,array,len,node,pid)

where

- id - is the identifier for the message
- array - is the array (real or integer) to be sent
- len - is the length of the array in *bytes*
- node - is the identity of the *receiving* node
- pid - is the process identity on the *receiving* node

The message identifier is an integer. In DL_POLY_2 this integer is generally unique to a particular message. It is essential that this uniqueness is maintained wherever possible, especially in the global sum routines GDSUM, GISUM and GSTATE.

The process identity pid is a redundant argument. In DL_POLY_2 it is always zero. DL_POLY_2 conceptually does not recognise alternative processes running on a single processing node.

The other arguments are self explanatory.

A *synchronous* message is one which requires numerical processing to wait for the message to be sent before processing continues.

CRECV

CRECV is a subroutine that receives *synchronous* messages (usually arrays) from another node. It is called with the following arguments:

CRECV(**id**,**array**,**len**)

where

- **id** - is the identifier for the message
- **array** - is the array (real or integer) to be sent
- **len** - is the length of the array in *bytes*

The message identifier is an integer. In DL_POLY_2 this integer is generally unique to a particular message. It is essential that this uniqueness is maintained wherever possible, especially in the global sum routines GDSUM, GISUM and GSTATE.

The other arguments are self explanatory. Note that the *sending* node is not specified. A replacement routine may require this information.

A *synchronous* message is one which requires numerical processing to wait for the message to arrive before processing continues. This has the advantage over the *asynchronous* routine IRECV (below) in that the integrity of the **array** is guaranteed. It is slightly slower however and has the hidden cost of a memory buffer.

GSYNC

GSYNC is a subroutine which suspends processing on each node until all nodes have reached the same point in the code, at which point all processes become synchronised. It is sometimes called a BARRIER operation.

GSYNC is called *without* arguments.

GDSUM

GDSUM is a subroutine that calculates the global real*8 sum of the elements of an array across processors. The result is an array with equal corresponding elements on each processor. It is called with the following arguments:

GDSUM(**aaa**,**nnn**,**bbb**)

where

- **aaa** - is the real*8 array to be summed (dimension **nnn**)
- **nnn** - is the number of elements in the array
- **bbb** - is a real*8 working array (dimension **nnn**)

Note that DL_POLY_2 provides a FORTRAN copy of this routine based on CSEND and IRECV.

GIMAX

GIMAX is a subroutine that determines the global max of the elements of an array across processors. The result is an array with maximum of corresponding elements on each processor. It is called with the following arguments:

GIMAX(**aaa**,**nnn**,**bbb**)

where

- **aaa** - is the integer array to be summed (dimension **nnn**)
- **nnn** - is the number of elements in the array
- **bbb** - is an integer working array (dimension **nnn**)

Note that DL_POLY_2 provides a FORTRAN copy of this routine based on CSEND and IRECV.

GISUM

GISUM is a subroutine that calculates the global integer sum of the elements of an array across processors. The result is an array with equal corresponding elements on each processor. It is called with the following arguments:

GISUM(**aaa**,**nnn**,**bbb**)

where

- **aaa** - is the integer array to be summed (dimension **nnn**)
- **nnn** - is the number of elements in the array
- **bbb** - is an integer working array (dimension **nnn**)

Note that DL_POLY_2 provides a FORTRAN copy of this routine based on CSEND and IRECV.

IRECV

IRECV is a subroutine that receives *asynchronous* messages (usually arrays) from another node. It is called with the following arguments:

IRECV(**id**,**array**,**len**)

where

- **id** - is the identifier for the message
- **array** - is the array (real or integer) to be sent

- **len** - is the length of the array in *bytes*

The message identifier is an integer. In DL_POLY_2 this integer is generally unique to a particular message. It is essential that this uniqueness is maintained wherever possible, especially in the global sum routines GDSUM, GISUM and GSTATE.

The other arguments are self explanatory. Note that the *sending* node is not specified. A replacement routine may require this information.

An *asynchronous* message is one which does not require numerical processing to wait for the message to arrive before processing continues. This has the advantage over the *synchronous* routine CRECV (above) in terms of speed, but the integrity of the **array** is not guaranteed and requires the user take responsibility for the consequences of that. It has no hidden memory cost.

MYNODE

MYNODE is a function that returns the unique node number a process is running on. i.e. the integer identifier for the calling node. The returned value must be one of the numbers running from 0 to $N_{nodes} - 1$ where N_{nodes} is the total number of nodes executing the program. The answer must be unique for each processing node. MYNODE is called without arguments.

NODEDIM

NODEDIM is a function that returns the dimension of the Intel hypercube the program is running on. i.e. the integer D in the formula $N_{nodes} = 2^D$, where N_{nodes} is the number of nodes. NODEDIM is called without arguments.

This routine is required only on hypercube parallel machines and even then only for the global summation routines GDSUM, GISUM and GSTATE. Implementations of DL_POLY_2 on non-hypercubes will obviously not require this routine, but will require alternative global summation strategies.

NUMNODES

NUMNODES is a function that returns the total number of processing nodes the program is running on. i.e. the integer N_{nodes} . NUMNODES is called without arguments.

8.1.197.3 SHMEM Routines

These routines are required for the Cray T3D version of DL_POLY_2 :

- SHMEM_PUT
- SHMEM_UDCFLUSH

It is important to note that these are used in conjunction with MPI routines in the T3D version of DL_POLY_2 .

SHMEM_PUT

SHMEM_PUT is a routine that copies a block of data from the memory of one processor into the memory of another. It is called with the following arguments;

SHMEM_PUT(**bbb**, **aaa**, **nnn**, **node**)

where

- **bbb** - is the name of the receiving array
- **aaa** - is the name of the sending array
- **nnn** - is the number of data items
- **node** - is the identifier of the target processor

SHMEM_UDCFLUSH

SHMEM_UDCFLUSH is a routine that ensures that data in the cache has been fully transferred to the receiving array before it can be overwritten. Its calling sequence is;

SHMEM_UDCFLUSH()

there are no arguments.

Bibliography

- [1] Smith, W., and Forester, T., 1996, *J. Molec. Graphics*, **14**, 136.
- [2] Smith, W., 1987, *Molecular Graphics*, **5**, 71.
- [3] van Gunsteren, W. F., and Berendsen, H. J. C. 1987, *Groningen Molecular Simulation (GROMOS) Library Manual*. BIOMOS, Nijenborgh, 9747 Ag Groningen, The Netherlands. Standard GROMOS reference.
- [4] Weiner, S. J., Kollman, P. A., Nguyen, D. T., and Case, D. A., 1986, *J. Comp. Chem.*, **7**, 230.
- [5] Brünger, A. T. 1992, *X-PLOR: A System for X-Ray Crystallography and NMR*. New Haven and London: Yale University Press.
- [6] Sutton, A. P., and Chen, J., 1990, *Philos. Mag. Lett.*, **61**, 139.
- [7] Mayo, S., Olafson, B., and Goddard, W., 1990, *J. Phys. Chem.*, **94**, 8897.
- [8] Smith, W., and Forester, T. R., 1994, *Comput. Phys. Commun.*, **79**, 52.
- [9] Smith, W., and Forester, T. R., 1994, *Comput. Phys. Commun.*, **79**, 63.
- [10] Allen, M. P., and Tildesley, D. J. 1989, *Computer Simulation of Liquids*. Oxford: Clarendon Press.
- [11] Ryckaert, J. P., Ciccotti, G., and Berendsen, H. J. C., 1977, *J. Comput. Phys.*, **23**, 327.
- [12] Fincham, D., 1992, *Molecular Simulation*, **8**, 165.
- [13] Forester, T., and Smith, W., 1995, *In preparation*.
- [14] Berendsen, H. J. C., Postma, J. P. M., van Gunsteren, W., DiNola, A., and Haak, J. R., 1984, *J. Chem. Phys.*, **81**, 3684.
- [15] Evans, D. J., and Morriss, G. P., 1984, *Computer Physics Reports*, **1**, 297.
- [16] Hoover, W. G., 1985, *Phys. Rev.*, **A31**, 1695.
- [17] Tuckerman, M., Berne, B., and Rossi, A., 1990, *J. Chem. Phys.*, **94**, 1465.

- [18] Stuart, S., Zhou, R., and Berne, B., 1996, *J. Chem. Phys.*, **105**, 1426.
- [19] Procacci, P., and Marchi, M., 1996, *J. Chem. Phys.*, **104**, 3003.
- [20] Smith, W., 1998, *Daresbury Laboratory*.
- [21] Melchionna, S., and Cozzini, S., 1998, *University of Rome*.
- [22] Brode, S., and Ahlrichs, R., 1986, *Comput. Phys. Commun.*, **42**, 41.
- [23] Hockney, R. W., and Eastwood, J. W. 1981, *Computer Simulation Using Particles*. McGraw-Hill International.
- [24] Vessal, B., 1994, *J. Non-Cryst. Solids*, **177**, 103.
- [25] Smith, W., Greaves, G. N., and Gillan, M. J., 1995, *J. Chem. Phys.*, **103**, 3091.
- [26] Smith, W., 1993, *CCP5 Information Quarterly*, **39**, 14.
- [27] Clarke, J. H. R., Smith, W., and Woodcock, L. V., 1986, *J. Chem. Phys.*, **84**, 2290.
- [28] Finnis, M. W., and Sinclair, J. E., 1984, *Philos. Mag. A*, **50**, 45.
- [29] Eastwood, J. W., Hockney, R. W., and Lawrence, D. N., 1980, *Comput. Phys. Commun.*, **19**, 215.
- [30] Smith, W., and Fincham, D., 1993, *Molecular Simulation*, **10**, 67.
- [31] Essmann, U., Perera, L., Berkowitz, M. L., Darden, T., Lee, H., and Pedersen, L. G., 1995, *J. Chem. Phys.*, **103**, 8577.
- [32] Hautman, J., and Klein, M. L., 1992, *Molec. Phys.*, **75**, 379.
- [33] Neumann, M., 1985, *J. Chem. Phys.*, **82**, 5663.
- [34] Fincham, D., and Mitchell, P. J., 1993, *J. Phys. Condens. Matter*, **5**, 1031.
- [35] Lindan, P. J. D., and Gillan, M. J., 1993, *J. Phys. Condens. Matter*, **5**, 1019.
- [36] McCammon, J. A., and Harvey, S. C. 1987, *Dynamics of Proteins and Nucleic Acids*. Cambridge: University Press.
- [37] Brown, D., and Clarke, J. H. R., 1984, *Molec. Phys.*, **51**, 1243.
- [38] Melchionna, S., Ciccotti, G., and Holian, B. L., 1993, *Molec. Phys.*, **78**, 533.
- [39] Tildesley, D. J., Streett, W. B., and Saville, G., 1978, *Molec. Phys.*, **35**, 639.
- [40] Tildesley, D. J., and Streett, W. B. Multiple time step methods and an improved potential function for molecular dynamics simulations of molecular liquids. In Lykos, P., editor, *Computer Modelling of Matter*. ACS Symposium Series No. 86, 1978.

- [41] Forester, T., and Smith, W., 1994, *Molecular Simulation*, **13**, 195.
- [42] Smith, W., 1991, *Comput. Phys. Commun.*, **62**, 229.
- [43] Smith, W., 1993, *Theoretica. Chim. Acta.*, **84**, 385.
- [44] Smith, W., 1992, *Comput. Phys. Commun.*, **67**, 392.
- [45] Vessal, B., Amini, M., Leslie, M., and Catlow, C. R. A., 1990, *Molecular Simulation*, **5**, 1.
- [46] Lewis, G. V., and Catlow, C. R. A., 1985, *Solid State Phys. C*, **18**, 1149.
- [47] Bush, T. S., Gale, J. D., Catlow, C. R. A., and Battle, D., 1994, *J. Mater. Chem.*, **4**, 831.
- [48] Vashishta, A., and Rahman, A., 1978, *Phys. Rev. Letters*, **40**, 1337.
- [49] Rahman, A., 1964, *Phys. Rev.*, **136**, A405.
- [50] Flyvbjerg, H., and Petersen, H. G., 1989, *J. Chem. Phys.*, **91**, 461.
- [51] Abramowitz, M., and Stegun, I. A. 1970, *Handbook of Mathematical Functions*. Dover Publications Inc, New York.

Appendix A

The DL_POLY_2 Makefile

```
# Master makefile for DL_POLY_2.0
# Author: W. Smith November 1999
# With DPP tool by J. Geronowicz
# Original by T. Forester 1995
#
#   $Author: wl $
#   $Date: 2001/06/12 13:17:25 $
#   $Revision: 1.14 $
#   $State: Exp $

# Define default settings

#=====

FFTW_LIBRARY = ""
BINROOT = ../execute
CC = cc
DPP = ./dpp
EX = DLPOLY.X
EXE = $(BINROOT)/$(EX)
FC=undefined
SHELL=/bin/sh
STRESS=STRESS
TYPE=3pt

# Define object files

#=====

OBJ_ALL = angfrc.o bndfrc.o cfgscan.o corshl.o coul0.o coul4.o \
```

```

coul2.o coul3.o conscan.o dblstr.o dcell.o diffsn0.o \
diffsn1.o dlpoly.o duni.o error.o ewald1.o ewald3.o \
exclude.o exclude_atom.o fldscan.o exclude_link.o \
exitcomms.o extnflld.o fbpfrfc.o fcap.o forces.o freeze.o gauss.o \
gdsum.o getrec.o gimax.o gisum.o gstate.o images.o initcomms.o \
intlist.o intstr.o invert.o invfrfc.o jacobi.o lowercase.o lrcmetal.o \
lrcorrect.o machine.o merge.o merge1.o merge4.o multiple.o \
multiple_nsq.o npt_b1.o npt_b3.o parset.o npt_h1.o npt_h3.o nve_1.o \
nvt_b1.o nvt_e1.o nvt_h1.o parlst_nsq.o parlink.o parlst.o passcon.o \
passpmf.o pmf_1.o pmf_shake.o primlst.o quench.o rdf0.o rdf1.o \
rdshake_1.o result.o revive.o scdens.o shellsort.o shlfrfc.o \
shlmerge.o shlqnch.o shmove.o simdef.o splice.o static.o strip.o \
strucopt.o sysdef.o sysgen.o systemp.o sysbook.o sysinit.o \
tethfrfc.o thbfrfc.o timchk.o traject.o verest.o vscaleg.o \
warning.o xscale.o zden0.o zden1.o

OBJ_SPME = bspcoe.o bspgen.o cpy_rtc.o ele_prd.o ewald_spme.o \
  scl_csum.o set_block.o spl_cexp.o spme_for.o dlpfft3.o

OBJ_HKE = hkgen.o hkewald1.o hkewald2.o hkewald3.o hkewald4.o

OBJ_NEU = coul0neu.o coul2neu.o coul3neu.o excludeneu.o forcesneu.o \
  multipleneu.o neutlst.o parneulst.o prneulst.o \
  parlinkneu.o rdf0neu.o

OBJ_RIG = nptq_b1.o nptq_b2.o nptq_b3.o nptq_b4.o nptq_h1.o nptq_h2.o \
  nptq_h3.o nptq_h4.o nveq_1.o nveq_2.o nvtq_b1.o nvtq_b2.o \
  nvtq_h1.o nvtq_h2.o passquat.o qshake.o quatbook.o quatqnch.o

OBJ_RRR = denloc.o dihfrc.o erfcgen.o ewald2.o ewald4.o forgen.o \
  fortab.o metgen.o srfrce.o srfrceneu.o suttchen.o

OBJ_4PT = denloc_4pt.o dihfrc_4pt.o erfcgen.o ewald2_4pt.o ewald4_4pt.o \
  forgen.o fortab.o metgen.o srfrce_4pt.o srfrceneu_4pt.o \
  suttchen_4pt.o

OBJ_RSQ = denloc_rsq.o dihfrc_rsq.o erfcgen_rsq.o ewald2_rsq.o ewald4_rsq.o \
  forgen_rsq.o fortab_rsq.o metgen_rsq.o srfrce_rsq.o srfrceneu_rsq.o \
  suttchen_rsq.o

OBJ_EXT = crecv.o csend.o gsync.o mynode.o nodedim.o numnodes.o

TIMER = etime.o

```

```
# Define targets
```

```
#=====
```

```
all:
```

```
@echo "Error - please specify a target machine!"
```

```
@echo "Permissible targets for this Makefile are:"
```

```
@echo "                                "
```

```
@echo "aix                                (serial)"
```

```
@echo "alpha-linux                        (serial)"
```

```
@echo "alpha-linux-fftw                    (serial)"
```

```
@echo "beowulf-absoft                      (parallel)"
```

```
@echo "cray-t3e                            (parallel)"
```

```
@echo "cray-t3e-serial                    (serial)"
```

```
@echo "cray-t3e-totalview                 (parallel)"
```

```
@echo "dec-alpha                          (serial)"
```

```
@echo "dec-alpha-ev6                     (serial)"
```

```
@echo "exemplar                          (serial)"
```

```
@echo "hitachi-sr2201                    (parallel)"
```

```
@echo "hp-c240                           (serial)"
```

```
@echo "pentium-absoft                    (serial)"
```

```
@echo "pentium-portland                  (serial)"
```

```
@echo "pentium-portland-fftw             (serial)"
```

```
@echo "rs6                                (serial)"
```

```
@echo "rs6k-pwr3                         (serial)"
```

```
@echo "sg10k                              (serial)"
```

```
@echo "sg10k-fftw                        (serial)"
```

```
@echo "sg2-r5k                           (serial)"
```

```
@echo "sg2k                              (serial)"
```

```
@echo "sg2k-i6.5                        (serial)"
```

```
@echo "sg2k-shmem                        (parallel)"
```

```
@echo "sg2k-shmem-debug                 (parallel)"
```

```
@echo "sg8k-mpi                         (parallel)"
```

```
@echo "sg8k-mpi-f                       (parallel)"
```

```
@echo "sp2-mpi                          (parallel)"
```

```
@echo "sp2-mpi-debug                   (parallel)"
```

```
@echo "sp2-mpi-fftw                    (parallel)"
```

```
@echo "sun-ultra                        (serial)"
```

```
@echo "                                "
```

```
@echo "Please examine Makefile for details"
```

```
# system specific targets follow :
```

```
#===== IBM SP2 version =====
```

```
sp2-mpi: dpp
```

```

cp /usr/lpp/ppe.poe/include/mpif.h mpif.h
$(MAKE) FC=mpxlf FFLAGS="-c -NS2048 -qarch=pwr2 -qnosave" \
CPFLAGS="-D$(STRESS) -DMPI -DESSL -D'POINTER=integer'" intlist.o
$(MAKE) LD="mpxlf -o" LDFLAGS="-lesslp2" FC=mpxlf \
FFLAGS="-c -O3 -NS2048 -qarch=pwr2 -qnosave" \
CPFLAGS="-D$(STRESS) -DMPI -DESSL -D'POINTER=integer'" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#===== IBM SP2 debug version =====
sp2-mpi-debug: dpp
cp /usr/lpp/ppe.poe/include/mpif.h mpif.h
$(MAKE) LD="mpxlf -o" LDFLAGS="-lesslp2" FC=mpxlf \
FFLAGS="-g -c -C -NS2048 -qarch=pwr2 -qnosave" \
CPFLAGS="-D$(STRESS) -DMPI -DESSL -D'POINTER=integer'" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#===== IBM SP2 FFTW version =====
sp2-mpi-fftw: dpp
cp /usr/lpp/ppe.poe/include/mpif.h mpif.h
$(MAKE) FC=mpxlf FFLAGS="-c -NS2048 -qarch=pwr2 -qnosave" \
CPFLAGS="-D$(STRESS) -DMPI -D'POINTER=integer' -DFFTW"
$(MAKE) LD="mpxlf -o" LDFLAGS="-lesslp2 -lfftw -lrfftw -L$(FFTW_LIBRARY)" \
FC=mpxlf FFLAGS="-c -O3 -NS2048 -qarch=pwr2 -qnosave" \
CPFLAGS="-D$(STRESS) -DMPI -D'POINTER=integer' -DFFTW -DESSL" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#===== Cray t3e (Manchester) =====
cray-t3e: dpp
cp /opt/ctl/mpt/mpt/include/mpif.h mpif.h
$(MAKE) FC="f90" FFLAGS="-c -dp -lmpi" \
CPFLAGS="-DCRAY -DCRAYT3D -D'POINTER=integer' -DSHMEM -D$(STRESS)" \
lowcase.o
$(MAKE) LD="f90 -o" FC="f90" FFLAGS="-c -dp -lmpi" LDFLAGS="-lmpi" \
CPFLAGS="-DCRAY -DCRAYT3D -D'POINTER=integer' \
-DSHMEM -D$(STRESS)" TIMER="" EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#===== Cray t3e (Manchester) with totalview flags =====
cray-t3e-totalview: dpp
cp /opt/ctl/mpt/mpt/include/mpif.h mpif.h
$(MAKE) FC="f90" FFLAGS="-c -dp -lmpi -g -X8" \
CPFLAGS="-DCRAY -DCRAYT3D -D'POINTER=integer' -DSHMEM -D$(STRESS)" \
lowcase.o
$(MAKE) LD="f90 -o" FC="f90" FFLAGS="-c -dp -lmpi -g -X8" \
LDFLAGS="-lmpi -g -X8" \

```

```
CPFLAGS="-DCRAY -DCRAYT3D -D'POINTER=integer' \
-D$HMEM -D$(STRESS)" TIMER="" EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
#===== Cray t3e (serial) =====
cray-t3e-serial: dpp
$(MAKE) LD="f90 -o" LDFLAGS="" FC=f90 \
FFLAGS="-c -dp -O3,aggress,unroll2,nojump" TIMER="" \
CPFLAGS="-D$(STRESS) -DSERIAL -P -D'POINTER=integer'" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
#===== DEC Alpha =====
dec-alpha: dpp
$(MAKE) LD="f90 -o" FC=f90 FFLAGS="-c -fast" \
LDFLAGS="-math_library fast -assume noaccuracy_sensitive" \
CPFLAGS="-D$(STRESS) -DSERIAL -D'POINTER=integer*8'" \
TIMER="" EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
#===== DEC Alpha ev6 =====
dec-alpha-ev6: dpp
$(MAKE) LD="f90 -o" FC=f90 FFLAGS="-c -arch ev6 -fast" \
LDFLAGS="-arch ev6 -math_library fast -assume noaccuracy_sensitive" \
CPFLAGS="-D$(STRESS) -DSERIAL -D'POINTER=integer*8'" \
TIMER="" EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
#===== Alpha Linux =====
alpha-linux: dpp
$(MAKE) FC=fort FFLAGS="-c -O -fast" \
CPFLAGS="-D$(STRESS) -DSERIAL -D'POINTER=integer*8'" qshake.o
$(MAKE) FC=fort LD="fort -o" FFLAGS="-c -O -fast" \
CPFLAGS="-D$(STRESS) -DSERIAL -D'POINTER=integer*8'" \
LDFLAGS="" EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
#===== Alpha Linux (plus SPME via FFTW) =====
alpha-linux-fftw: dpp
$(MAKE) FC=fort LD="fort -o" FFLAGS="-c -O -fast " \
CPFLAGS="-D$(STRESS) -DSERIAL -D'POINTER=integer*8' -DFFTW" \
LDFLAGS="-L$(FFTW_LIBRARY) -lfftw -lfftw " EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
#===== Convex/HP exemplar (serial) =====
exemplar: dpp
$(MAKE) LD="f90 -o" LDFLAGS="" FC=f90 FFLAGS="-c +ppu +O2 +DA2.0" \
CPFLAGS="-D$(STRESS) -DSERIAL" -D'POINTER=integer' \
TIMER="" EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```



```
#===== HP PA 9000 / C240 (serial) =====
```

```
hp-c240: dpp
```

```
$(MAKE) LD="f90 -o" LDFLAGS= FC=f90 FFLAGS="-c +ppu +O2 +DA2.0" \
CPFLAGS="-D$(STRESS) -DSERIAL -D'POINTER=integer'" \
TIMER="" EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
#===== IBM (AIX) Workstation version =====
```

```
aix: dpp
```

```
$(MAKE) FC=mpxlf FFLAGS="-c -NS2048 -qarch=pwr2 -qnosave" \
CPFLAGS="-D$(STRESS) -DSERIAL -D'POINTER=integer'" intlist.o
$(MAKE) LD="xlf -o" LDFLAGS="" FC=xlf FFLAGS="-c -O3 -NS2048 -qarch=pwr2 -qnosave" \
CPFLAGS="-D$(STRESS) -DSERIAL -D'POINTER=integer'" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
#===== RS/6000 P2SC (serial) =====
```

```
rs6k: dpp
```

```
$(MAKE) LD="xlf -o" LDFLAGS= FC=xlf \
FFLAGS="-c -O -qarch=pwr2 -qtune=pwr2" TIMER="" \
CPFLAGS="-D$(STRESS) -DSERIAL -D'POINTER=integer'" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
#===== RS/6000 Power3 (serial) =====
```

```
rs6k-pwr3: dpp
```

```
$(MAKE) LD="xlf -o" LDFLAGS="-L/usr/local/lib -lmass -lessl" \
FC=xlf FFLAGS="-c -O -qnosave -qarch=pwr3" TIMER="" \
CPFLAGS="-D$(STRESS) -DSERIAL" -DESSL -D'POINTER=integer' \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
#===== PentiumII (absoft) (serial) =====
```

```
pentium-absoft: dpp
```

```
$(MAKE) LD="/usr/bin/f90 -o" LDFLAGS="-lfio" TIMER="" \
FC=/usr/bin/f90 FFLAGS="-c -YEXT_NAMES=LCS -B108 -B100 -O" \
CPFLAGS="-D$(STRESS) -DSERIAL -P -D'POINTER=integer'" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
#===== PentiumII (portland) (serial) (with FFTW) =====
```

```
pentium-portland-fftw: dpp
```

```
$(MAKE) LD="/usr/local/pgi/linux86/bin/pgf90 -o" \
LDFLAGS="-L/usr/local/lib -lfftw -lfftw -L/usr/pgi/linux86/lib" \
FC=/usr/local/pgi/linux86/bin/pgf90 \
FFLAGS="-c -fast -Malign -Msecond_underscore" \
CPFLAGS="-D$(STRESS) -DSERIAL -P -D'pointer=integer' -DFFTW" \
TIMER="" EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```

#===== PentiumII (portland) (serial) =====
pentium-portland: dpp
$(MAKE) LD="/usr/local/pgi/linux86/bin/pgf90 -o" LDFLAGS="" \
FC="/usr/local/pgi/linux86/bin/pgf90 FFLAGS="-c -O -Malign" \
CPFLAGS="-D$(STRESS) -DSERIAL -P -D'POINTER=integer'" \
TIMER="" EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#===== Hitachi SR2201 Multiprocessor =====
hitachi-sr2201: dpp
cp /usr/include/mpif.h mpif.h
$(MAKE) FC=xf90 \
FFLAGS="-c -W0, 'form(fixed),opt(o(3)),langlvl(save(0))' -s,TRACE" \
CPFLAGS="-D$(STRESS) -DMPI -D'POINTER=integer'" intlist.o
$(MAKE) LDFLAGS="" LDLIBS="-lfmpi -lmpi" LD="xf90 -o" FC=xf90 \
FFLAGS="-c -W0, 'form(fixed),opt(o(3)),langlvl(save(0))' -s,TRACE" \
CC=xcc TIMER="" CPFLAGS="-D$(STRESS) -DMPI
-D'POINTER=integer'" EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#===== Silicon Graphics 10000 Worskstation =====
sg10k: dpp
$(MAKE) LD="f90 -o" LDFLAGS="-lscs" \
FC=f90 FFLAGS="-c -O2 -OPT:Olimit=0 -lscs" TIMER="" \
CPFLAGS="-DSGICRAY -D$(STRESS) -DSERIAL -D'POINTER=integer'" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#===== Silicon Graphics 10000 Worskstation (SPME via FFTW) =====
sg10k-fftw: dpp
$(MAKE) LD="f90 -o" LDFLAGS="-lscs -lfftw -lfftw -L$(FFTW_LIBRARY)" \
FC=f90 FFLAGS="-c -O2 -OPT:Olimit=0 -lscs" TIMER="" \
CPFLAGS="-D$(STRESS) -DSERIAL -D'POINTER=integer' -DFFTW" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#===== Silicon Graphics 8000 (parallel) =====
sg8k-mpi: dpp
cp /usr/include/mpif.h mpif.h
cp /usr/include/mpif_parameters.h mpif_parameters.h
$(MAKE) LD="f90 -O3 -64 -o" FC=f90 LDFLAGS="-lmpi" TIMER="" \
FFLAGS="-c -O3 -64 " CPFLAGS="-D$(STRESS) -DMPI \
-DSGICRAY -D'POINTER=integer'" EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#===== Silicon Graphics 8000 (parallel) =====
sg8k-mpi-f: dpp
cp /usr/include/mpif.h mpif.h

```

```

cp /usr/include/mpif_parameters.h mpif_parameters.h
$(MAKE) LD="f90 -Ofast -64 -o" FC=f90 LDFLAGS="-lmpi" TIMER="" \
FFLAGS="-c -Ofast -64" CPFLAGS="-D$(STRESS) -DMPI \
-DSGICRAY -D'POINTER=integer'" EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#===== Silicon Graphics Origin 2000 (parallel/SHMEM) =====
sg2k-shmem: dpp
cp /usr/include/mpif.h mpif.h
cp /usr/include/mpif_parameters.h mpif_parameters.h
$(MAKE) LD="f90 -o" FC=f90 \
FFLAGS="-c -64 -mips4 -Ofast=ip27 -IPA -OPT:Olimit=0 " \
LDFLAGS="-64 -mips4 -Ofast=ip27 -IPA -OPT:Olimit=0 -lsma -lscs -lmpi -lblas" \
OBJ_EXT="gsync.o mynode.o nodedim.o numnodes.o" TIMER="" \
CPFLAGS="-D$(STRESS) -DSGISHMEM -DSGICRAY -D'POINTER=integer'" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#===== Silicon Graphics Origin 2000 (parallel/SHMEM) =====
sg2k-shmem-debug: dpp
cp /usr/include/mpif.h mpif.h
cp /usr/include/mpif_parameters.h mpif_parameters.h
$(MAKE) LD="f90 -o" FC=f90 \
FFLAGS="-c -64 -mips4 -g " \
LDFLAGS="-64 -mips4 -lsma -lscs -lmpi -lblas" \
OBJ_EXT="gsync.o mynode.o nodedim.o numnodes.o" TIMER="" \
CPFLAGS="-D$(STRESS) -DSGISHMEM -DSGICRAY -D'POINTER=integer'" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#===== Silicon Graphics Origin 2000 (serial) =====
sg2k: dpp
$(MAKE) LD="f90 -o" LDFLAGS="-n32 -mips4" FC=f90 \
FFLAGS="-c -O3 -G 0 -mips4 -r10000 -c -r8 -n32" \
TIMER="" \
CPFLAGS="-D$(STRESS) -DSERIAL -D'POINTER=integer'" \
CFLAGS="-c -O3 -n32 -mips4 -r10000" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#=== Silicon Graphics Origin 2000 (serial, Irix 6.5) =====
sg2k-i6.5: dpp
$(MAKE) LD="f90 -o" LDFLAGS="-n32 -mips4 -IPA" \
FC=f90 FFLAGS="-c -n32 -mips4 -Ofast=ip27 -LN0:fusion=2" \
TIMER="" \
CPFLAGS="-D$(STRESS) -DSERIAL -D'POINTER=integer'" \
CFLAGS="-c -O3 -n32 -mips4 -r10000" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

```

```
#=== Silicon Graphics O2 R5k (serial) =====
sg2-r5k: dpp
$(MAKE) LD="f90 -o" LDFLAGS="-n32 -mips4" \
FC=f90 FFLAGS="-c -O3 -G 0 -mips4 -r5000 -c -r8 -n32" \
TIMER="" \
CPFLAGS="-D$(STRESS) -DSERIAL -D'POINTER=integer'" \
CFLAGS="-c -O3 -n32 -mips4 -r5000" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
#===== Sun Ultra-2 (serial) =====
sun-ultra: dpp
$(MAKE) LD="/opt/SUNWspro/bin/f90 -o" LDFLAGS= \
FC=/opt/SUNWspro/bin/f90 \
FFLAGS="-c -fnonstd -xarch=v8plusa -xchip=ultra -O2 -libmil -dalign" \
TIMER="" \
CPFLAGS="-D$(STRESS) -DSERIAL -D'POINTER=integer'" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
#===== Daresbury Beowulf (Absoft compiler) =====
beowulf-absoft: dpp
cp /usr/local/lam-6.2/h/mpif.h mpif.h
$(MAKE) LD="/usr/bin/f90 -o" \
LDFLAGS="-L/home/kcm/mpich-absoft/mpich/lib/LINUX/ch_p4/ -lmpich" \
FC=/usr/bin/f90 FFLAGS="-c -O -B100" TIMER="" \
CPFLAGS="-D$(STRESS) -DMPI -P -D'POINTER=integer' \
-I/home/kcm/mpich-absoft/mpich/include" \
EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
#=====
# Interpolation tables options
```

```
# Default code. Force tables interpolation in r-space 3pt interpolation
3pt: check $(OBJ_ALL) $(OBJ_RRR) $(OBJ_NEU) $(OBJ_RIG) $(OBJ_EXT) $(TIMER) $(OBJ_SPME) \
$(OBJ_HKE)
$(LD) $(EXE) $(OBJ_ALL) $(OBJ_RRR) $(OBJ_NEU) $(OBJ_RIG) $(OBJ_EXT) $(TIMER) \
$(LDFLAGS) $(LDLIBS) $(OBJ_SPME) $(OBJ_HKE)
```

```
# Force tables interpolation in r-space, 4pt interpolation
4pt: check $(OBJ_ALL) $(OBJ_4PT) $(OBJ_NEU) $(OBJ_RIG) $(OBJ_EXT) $(TIMER) $(OBJ_SPME) \
$(OBJ_HKE)
$(LD) $(EXE) $(OBJ_ALL) $(OBJ_4PT) $(OBJ_NEU) $(OBJ_RIG) $(OBJ_EXT) $(TIMER) \
$(LDFLAGS) $(LDLIBS) $(OBJ_SPME) $(OBJ_HKE)
```

```

# Force tables interpolation in r-squared
rsq: check $(OBJ_ALL) $(OBJ_RSQ) $(OBJ_NEU) $(OBJ_RIG) $(OBJ_EXT) $(TIMER) $(OBJ_SPME) \
$(OBJ_HKE)
$(LD) $(EXE) $(OBJ_ALL) $(OBJ_RSQ) $(OBJ_NEU) $(OBJ_RIG) $(OBJ_EXT) $(TIMER) \
$(LDFLAGS) $(LDLIBS) $(OBJ_SPME) $(OBJ_HKE)

#=====
# Check that a machine has been specified
check:
@if test $(FC) = "undefined";\
then echo "You must specify a target machine!"; \
exit 99;\
fi

#=====
# Clean up the source directory
clean:
rm -f $(OBJ_ALL) $(OBJ_RRR) $(OBJ_EXT) $(OBJ_NEU) $(OBJ_HKE) \
$(OBJ_RIG) $(TIMER) $(OBJ_SPME) $(OBJ_4PT) $(OBJ_RSQ) \
*.tmp.f mpif.h

#=====
# Compile preprocessor code
dpp: dpp.c
$(CC) dpp.c -o dpp

#=====
# Declare dependencies : c preprocess all .f files
.f.o:
$(DPP) $(CPFLAGS) *.f > *.tmp.f
$(FC) $(FFLAGS) *.tmp.f
mv *.tmp.o *.o

.c.o:
$(CC) -c *.c

#=====
# Declare dependency on parameters file

$(OBJ_ALL): dl_params.inc
$(OBJ_RRR): dl_params.inc
$(OBJ_4PT): dl_params.inc
$(OBJ_RSQ): dl_params.inc
$(OBJ_NEU): dl_params.inc

```

```
$(OBJ_RIG): dl_params.inc
$(OBJ_EXT): dl_params.inc
$(OBJ_SPME): dl_params.inc
$(OBJ_HKE): dl_params.inc
```

Appendix B

Periodic Boundary Conditions in DL_POLY

Introduction

DL_POLY_2 is designed to accommodate a number of different periodic boundary conditions, which are defined by the shape and size of the simulation cell. Briefly, these are as follows (which also indicates the IMCON flag defining the simulation cell type in the CONFIG File - see 4.1.2):

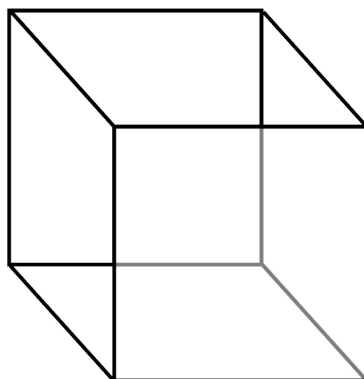
1. None e.g. isolated polymer in space. (IMCON=0).
2. Cubic periodic boundaries.(IMCON=1).
3. Orthorhombic periodic boundaries.(IMCON=2).
4. Parallelepiped periodic boundaries.(IMCON=3).
5. Truncated octahedral periodic boundaries. (IMCON=4).
6. Rhombic dodecahedral periodic boundaries. (IMCON=5).
7. Slab (X,Y periodic, Z nonperiodic). (IMCON=6).
8. Hexagonal prism periodic boundaries. (IMCON=7).

We shall now look at each of these in more detail. Note that in all cases the cell vectors and the positions of the atoms in the cell are to be specified in Angstroms ($\text{\AA}$).

No periodic boundary (IMCON=0)

Simulations requiring no periodic boundaries are best suited to *in vacuo* simulations, such as the conformational study of an isolated polymer molecule. This boundary condition is not recommended for studies in a solvent, since evaporation is likely to be a problem.

Note this boundary condition cannot be used with the Ewald summation method.

Cubic periodic boundaries (IMCON=1)

The cubic MD cell.

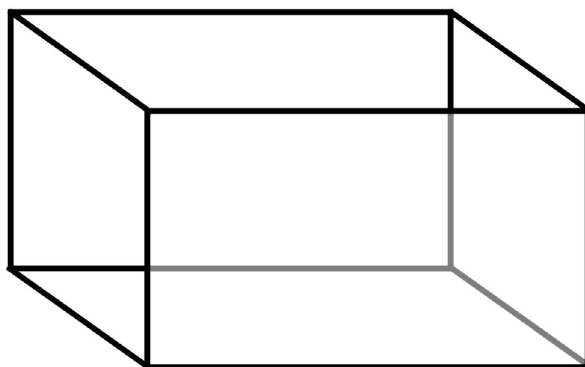
The cubic MD cell is perhaps the most commonly used in simulation and has the advantage of great simplicity. In DL_POLY_2 the cell is defined with the principle axes passing through the centres of the faces. Thus for a cube with sidelength D , the cell vectors appearing in the CONFIG file should be: $(D,0,0)$; $(0,D,0)$; $(0,0,D)$. Note the origin of the atomic coordinates is the centre of the cell.

The cubic boundary condition can be used with the Ewald summation method.

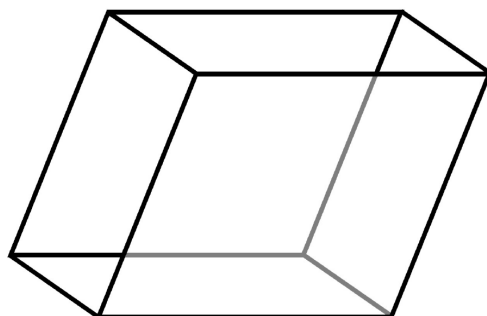
Orthorhombic periodic boundaries (IMCON=2)

The orthorhombic cell is also a common periodic boundary, which closely resembles the cubic cell in use. In DL_POLY_2 the cell is defined with principle axes passing through the centres of the faces. For an orthorhombic cell with sidelengths D (in X-direction), E (in Y-direction) and F (in Z-direction), the cell vectors appearing in the CONFIG file should be: $(D,0,0)$; $(0,E,0)$; $(0,0,F)$. Note the origin of the atomic coordinates is the centre of the cell.

The orthorhombic boundary condition can be used with the Ewald summation method.



The orthorhombic MD cell.

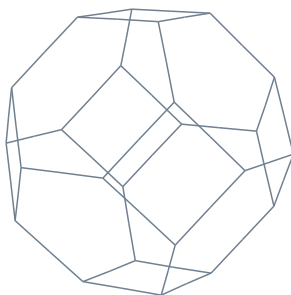
Parallelepiped periodic boundaries (IMCON=3)

The parallelepiped MD cell.

The parallelepiped (e.g. monoclinic or triclinic) cell is generally used in simulations of crystalline materials, where its shape and dimension is commensurate with the unit cell of the crystal. Thus for a unit cell specified by three principal vectors $\underline{a}$, $\underline{b}$, $\underline{c}$, the MD cell is defined in the DL_POLY_2 CONFIG file by the vectors (La_1, La_2, La_3) , (Mb_1, Mb_2, Mb_3) , (Nc_1, Mc_2, Nc_3) , in which L,M,N are integers, reflecting the multiplication of the unit cell in each principal direction. Note that the atomic coordinate origin is the centre of the MD cell.

The parallelepiped boundary condition can be used with the Ewald summation method.

Truncated octahedral boundaries (IMCON=4)



The truncated octahedral MD cell.

This is one of the more unusual MD cells available in DL_POLY, but it has the advantage of being more nearly spherical than most other MD cells. This means it can accommodate a larger spherical cutoff for a given number of atoms, which leads to greater efficiency. This can be very useful when simulating (for example) a large molecule in solution, where fewer solvent molecules are required for a given simulation cell width.

The principal axes of the truncated octahedron (see figure) pass through the centres of the square faces, and the width of the cell, measured from square face to square face along a principal axis defines the width D of the cell. From this, the cell vectors required in the DL_POLY_2 CONFIG file are simply: $(D,0,0)$, $(0,D,0)$, $(0,0,D)$. These are also the cell vectors defining the enscribing cube, which possesses twice the volume of the truncated octahedral cell. Once again, the atomic positions are defined with respect to the cell centre.

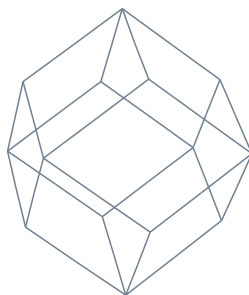
The truncated octahedron can be used with the Ewald summation method.

Rhombic dodecahedral boundaries (IMCON=5)

This is another unusual MD cell (see figure), but which possesses similar advantages to the truncated octahedron, but with a slightly greater efficiency in its use of the cell volume (the ratio is about 74% to 68%).

The principal axis in the X-direction of the rhombic dodecahedron passes through the centre of the cell and the centre of a rhombic face. The Y-axis does likewise, but is set at 90 degrees to the X-axis. The Z-axis completes the orthonormal set and passes through a vertex where four faces meet. If the width D of the cell is defined as the perpendicular distance between two opposite faces, the cell vectors required for the DL_POLY_2 CONFIG file are: $(D,0,0)$, $(0,D,0)$, $(0,0,\sqrt{2}D)$. These also define the enscribing orthorhombic cell, which has twice the MD cell volume. In DL_POLY_2 the centre of the cell is also the origin of the atomic coordinates.

The rhombic dodecahedron can be used with the Ewald summation method.



The rhombic dodecahedral MD cell.

Slab boundary conditions (IMCON=6)

Slab boundaries are periodic in the X- and Y-directions, but not in the Z-direction. They are particularly useful for simulating surfaces. The periodic cell in the XY plane can be any parallelogram. The origin of the X,Y atomic coordinates lies on an axis perpendicular to the centre of the parallelogram. The origin of the Z coordinate is where the user specifies it, but at or near the surface is recommended.

If the XY parallelogram is defined by vectors $\underline{A}$ and $\underline{B}$, the vectors required in the CONFIG file are: $(A_1, A_2, 0)$, $(B_1, B_2, 0)$, $(0, 0, D)$, where D is any real number (including zero). If D is nonzero, it will be used by DL_POLY to help determine a 'working volume' for the system. This is needed to help calculate RDFs etc. (The working value of D is in fact taken as one of: $3 \times \text{cutoff}$; or $2 \times \max \text{abs}(Z \text{ coordinate}) + \text{cutoff}$; or the user specified D, whichever is the larger.)

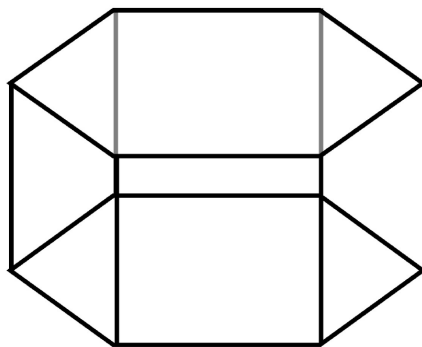
Note that the standard Ewald sum cannot be used with this boundary condition. DL_POLY_2 switches automatically to the Hautman-Klein-Ewald method instead [32].

The surface in a system with charges can also be modelled with DL_POLY_2 if periodicity is allowed in the Z-direction. In this case slabs of ions well-separated by vacuum zones in the Z-direction can be handled with IMCON=2 or 3.

Hexagonal prism boundaries (IMCON=7)

In this case the Z-axis lies along a line joining the centres of the hexagonal faces. The Y-axis is perpendicular to this and passes through the centre of one of the faces. The X-axis completes the orthonormal set and passes through the centre of an edge that is parallel to the Z-axis. (Note: It is important to get this convention right!) The origin of the atomic coordinates is the centre of the cell. If the length of one of the hexagon edges is D, the cell vectors required in the CONFIG file are: $(3D, 0, 0)$, $(0, \sqrt{3}D, 0)$, $(0, 0, H)$, where H is the prism height (the distance between hexagonal faces). The orthorhombic cell also defined by these vectors encloses the hexagonal prism and possesses twice the volume, but the height and the centre are the same.

The Ewald summation method may be used with this periodic boundary condition.



The hexagonal MD cell.

This MD cell is particularly suitable for simulating strands or fibres (i.e. systems with a pronounced anisotropy in the Z-direction), such as DNA strands in solution, or stretched polymer chains.

Appendix C

DL_POLY Error Messages and User Action

Introduction

In this appendix we document the error messages encoded in DL_POLY_2 and the recommended user action. The user response to these messages is different according to which version of the code is being run. Versions preceding DL_POLY_2 version 2.11 are significantly different from those appearing after. This is primarily because version 2.11 introduced FORTRAN 90 dynamic array allocation, which greatly simplifies the use of the code, but requires a different form of user intervention when array boundaries are violated. The correct response is described as the **standard user response** in the appropriate sections below, to which the user should refer before acting on the error encountered.

The reader should also be aware that some of the error messages listed below may be either disabled in, or absent from, the installed version of DL_POLY_2 . Disabled messages generally apply to older releases of the code, while missing messages apply to newer versions of the code and will not usually apply to previous releases. They are all included for completeness. Note that the wording of some of the messages may also have changed over time, usually to provide more specific information. The most recent wording appears below.

DL_POLY_2 Versions 2.10 and Earlier

Many of the DL_POLY_2 error messages refer to array bound checks and the user is advised to consider what other arrays may need to be altered when making the appropriate change. (If you are not familiar with DL_POLY_2 , the error messages will advise you as each array bound is violated, but you will not be told of all errors in one pass.) Users are warned that reckless increases in the array dimensions are likely to result in the code being too large for any given memory. A way to avoid many of these difficulties is to make use of the utility program PARSET (see 7.3.10), which can greatly reduce the work required.

The **standard user response** when an array dimension error occurs is to locate the relevant parameter in the DL_POLY_2 DL_PARAMS.INC file and increase the default number.

It should be remembered that any change in a parameter specified in DL_PARAMS.INC requires the *whole program* to be recompiled. However the standard makefile provided in the *build* directory does this automatically.

DL_POLY_2 Versions 2.11 and Later

Version 2.11 of DL_POLY_2 differs from previous versions in that it incorporates the functionality of the utility code PARSET within the code itself and uses FORTRAN 90 dynamic array allocation to set the array sizes at run time. This is a considerable advance on previous practice and means that a single executable may be compiled to cover all the likely uses of the code. It is not foolproof however. Sometimes an estimate of the required array sizes is difficult to obtain and the calculated value may be too small. For this reason DL_POLY_2 retains all the old array dimension checks and will terminate as before when an array bound error occurs.

When a dimension error occurs, the **standard user response** is to edit the DL_POLY_2 subroutine PARSET.F. Locate where the variable defining the array dimension is fixed and increase accordingly. To do this you should make use of the dimension information that DL_POLY_2 prints in the OUTPUT file prior to termination. If no information is supplied, simply doubling the size of the variable will usually do the trick. If the variable concerned is defined in one of the support subroutines CFGSCAN.F, FLDSCAN.F, CONSCAN.F you will need to insert a new line in PARSET.F to redefine it - after the relevant subroutine has been called! Finally the code must be recompiled, but in this case it will be necessary only to recompile PARSET.F and not the whole code.

The DL_POLY_2 Error Messages

Message 1: error - PVM_NODES unset

The code was C-preprocessed with the flag -DPVM set but the number of PVM nodes was not stated.

Action:

Delete the module INITCOMMS.o from the *source* directory and re-make the executable, this time including the directive PVM_NODES=*n* (where *n* is the number of nodes you require) with the *make* command.

Message 2: error - machine not a hypercube

The number of nodes on the parallel machine is not a power of 2.

Action:

Specify an appropriate number of processors for job execution. If you are using PVM see *Action:* for error message 1.

Message 3: error - unknown directive found in CONTROL file

This error most likely arises when a directive is misspelt.

Action:

Locate incorrect directive in CONTROL file and replace.

Message 4: error - unknown directive found in FIELD file

This error most likely arises when a directive is misspelt or is encountered in an incorrect location in the FIELD file, which can happen if too few or too many data records are included.

Action:

Locate the erroneous directive in the FIELD file and correct error.

Message 5: error - unknown energy unit requested

The DL_POLY_2 FIELD file permits a choice of units for input of energy parameters. These may be: electron volts (**ev**); kilocalories (**kcal**); kilojoules (**kj**); or the DL_POLY_2 internal units (10 J mol^{-1}) (**internal**). There is no default value. Failure to specify any of these correctly, or reference to other energy units, will result in this error message. See documentation of the FIELD file.

Action:

Correct energy keyword on **units** directive in FIELD file and resubmit.

Message 6: error - energy unit not specified

A **units** directive is mandatory in the FIELD file. This error indicates that DL_POLY_2 has failed to find the required record.

Action:

Add **units** directive to FIELD file and resubmit.

Message 7: error - energy unit respecified

DL_POLY_2 expects only one **units** directive in the FIELD file. This error results if it encounters another - implying an ambiguity in units.

Action:

Locate extra **units** directive in FIELD file and remove.

Message 8: error - time step not specified

DL_POLY_2 requires a **timestep** directive in the CONTROL file. This error results if none is encountered.

Action:

Insert **timestep** directive in CONTROL file with an appropriate numerical value.

Message 10: error - too many molecule types specified

DL_POLY_2 has a set limit on the number of kinds of molecules it will handle in any simulation (this is not the same as the number of molecules). If this permitted maximum is exceeded, the program terminates. The error arises when the **molecules** directive in the FIELD file specifies too large a number.

Action:

Standard user response. Fix parameter **mxtnls**.

Message 11: error - duplicate molecule directive in FIELD file

The number of different types of molecules in a simulation should only be specified once. If DL_POLY_2 encounters more than one **molecules** directive, it will terminate execution.

Action:

Locate the extra **molecule** directive in the FIELD file and remove.

Message 12: error - unknown molecule directive in FIELD file

Once DL_POLY_2 encounters the **molecules** directive in the FIELD file, it assumes the following records will supply data describing the intramolecular force field. It does not then expect to encounter directives not related to these data. This error message results if it encounters a unrelated directive. The most probable cause is incomplete specification of the data (e.g. when the **finish** directive has been omitted.)

Action:

Check the molecular data entries in the FIELD file and correct.

Message 13: error - molecule species not specified

This error arises when DL_POLY_2 encounters non-bonded force data in the FIELD file, *before* the molecular species have been specified. Under these circumstances it cannot assign the data correctly, and therefore terminates.

Action:

Make sure the molecular data appears before the non-bonded forces data in the FIELD file and resubmit.

Message 14: error - too many unique atom types specified

This error arises when DL_POLY_2 scans the FIELD file and discovers that there are too many different types of atoms in the system (i.e. the number of unique atom types exceeds the `mxsvdw` parameter).

Action:

Standard user response. Fix parameter `mxsvdw`.

Message 15: error - duplicate pair potential specified

In processing the FIELD file, DL_POLY_2 keeps a record of the specified short range pair potentials as they are read in. If it detects that a given pair potential has been specified before, no attempt at a resolution of the ambiguity is made and this error message results. See specification of FIELD file.

Action:

Locate the duplication in the FIELD file and rectify.

Message 16: error - strange exit from FIELD file processing

This should never happen! However one remote possibility is that there are more than 10,000 directives in the FIELD file! It simply means that DL_POLY_2 has ceased processing the FIELD data, but has not reached the end of the file or encountered a **close** directive. Probable cause: corruption of the DL_POLY_2 executable or of the FIELD file. We would be interested to hear of other reasons!

Action:

Recompile the program or recreate the FIELD file. If neither of these works, send the problem to us.

Message 17: error - strange exit from CONTROL file processing

See notes on message 16 above.

Message 18: error - duplicate 3-body potential specified

DL_POLY_2 has encountered a repeat specification of a 3-body potential in the FIELD file.

Action:

Locate the duplicate entry, remove and resubmit job.

Message 19: error - duplicate 4-body potential specified

A 4-body potential has been duplicated in the FIELD file.

Action:

Locate the duplicated 4-body potential and remove. Resubmit job.

Message 20: error - too many molecule sites specified

DL_POLY_2 has a fixed limit on the number of unique molecular sites in any given simulation. If this limit is exceeded, the program terminates.

Action:

Standard user response. Fix parameter `mxsite`.

Message 22: error - unsuitable radial increment in TABLE file

This arises when the tabulated potentials presented in the TABLE file have an increment that is greater than that used to define the other potentials in the simulation. Ideally the increment should be $r_{cut}/(mxgrid-4)$, where r_{cut} is the potential cutoff for the short range potentials and `mxgrid` is the parameter defining the length of the interpolation arrays. An increment less than this is permissible however.

Action:

The tables must be recalculated with an appropriate increment.

Message 23: error - incompatible FIELD and TABLE file potentials

This error arises when the specification of the short range potentials is different in the FIELD and TABLE files. This usually means that the order of specification of the potentials is different. When DL_POLY_2 finds a change in the order of specification, it assumes that the user has forgotten to enter one.

Action:

Check the FIELD and TABLE files. Make sure that you correctly specify the pair potentials in the FIELD file, indicating which ones are to be presented in the TABLE file. Then check the TABLE file to make sure all the tabulated potentials are present in the order the FIELD file indicates.

Message 24: error - end of file encountered in TABLE file

This means the TABLE file is incomplete in some way: either by having too few potentials included, or the number of data points is incorrect.

Action:

Examine the TABLE file contents and regenerate it if it appears to be incomplete. If it look intact, check that the number of data points specified is what DL_POLY_2 is expecting.

Message 25: error - wrong atom type found in CONFIG file

On reading the input file CONFIG, DL_POLY_2 performs a check to ensure that the atoms specified in the configuration provided are compatible with the corresponding FIELD file. This message results if they are not.

Action:

The possibility exists that one or both of the CONFIG or FIELD files has incorrectly specified the atoms in the system. The user must locate the ambiguity, using the data printed in the OUTPUT file as a guide, and make the appropriate alteration.

Message 30: error - too many chemical bonds specified

DL_POLY_2 sets a limit on the number of chemical bond potentials that can be specified in the FIELD file. Termination results if this number is exceeded. See FIELD file documentation. Do not confuse this error with that described by message 31 (below).

Action:

Standard user response. Fix parameter `mxtbnd`.

Message 31: error - too many chemical bonds in system

DL_POLY_2 sets a limit on the number of chemical bond potentials in the simulated system as a whole. (This number is a combination of the number of molecules and the number of bonds per molecule, divided by the number of processing nodes.) Termination results if this number is exceeded. Do not confuse this error with that described by message 30 (above).

Action:

Standard user response. Fix the parameter `mxbond`.

Message 32: error - integer array memory allocation failure

DL_POLY_2 has failed to allocate sufficient memory to accommodate one or more of the integer arrays in the code.

Action:

This may simply mean that your simulation is too large for the machine you are running on. Consider this before wasting time trying a fix. Try using more processing nodes if they are available. If this is not an option investigate the possibility of increasing the heap size for your application. Talk to your systems support people for advice on how to do this.

Message 33: error - real array memory allocation failure

DL_POLY_2 has failed to allocate sufficient memory to accommodate one or more of the real arrays in the code.

Action:

This may simply mean that your simulation is too large for the machine you are running on. Consider this before wasting time trying a fix. Try using more processing nodes if they are available. If this is not an option investigate the possibility of increasing the heap size for your application. Talk to your systems support people for advice on how to do this.

Message 34: error - character array memory allocation failure

DL_POLY_2 has failed to allocate sufficient memory to accommodate one or more of the character arrays in the code.

Action:

This may simply mean that your simulation is too large for the machine you are running on. Consider this before wasting time trying a fix. Try using more processing nodes if they are available. If this is not an option investigate the possibility of increasing the heap size for your application. Talk to your systems support people for advice on how to do this.

Message 35: error - logical array memory allocation failure

DL_POLY_2 has failed to allocate sufficient memory to accommodate one or more of the logical arrays in the code.

Action:

This may simply mean that your simulation is too large for the machine you are running on. Consider this before wasting time trying a fix. Try using more processing nodes if they are available. If this is not an option investigate the possibility of increasing the heap size for your application. Talk to your systems support people for advice on how to do this.

Message 40: error - too many bond constraints specified

DL_POLY_2 sets a limit on the number of bond constraints that can be specified in the FIELD file. Termination results if this number is exceeded. See FIELD file documentation. Do not confuse this error with that described by message 41 (below).

Action:

Standard user response. Fix the parameter `mxtcon`.

Message 41: error - too many bond constraints in system

DL_POLY_2 sets a limit on the number of bond constraints in the simulated system as a whole. (This number is a combination of the number of molecules and the number of per molecule, divided by the number of processing nodes.) Termination results if this number is exceeded. Do not confuse this error with that described by message 40 (above).

Action:

Standard user response. Fix the parameter **mxcons**.

Message 42: error - transfer buffer too small in merge1

The buffer used to transfer data between nodes in the MERGE1 subroutines has been dimensioned too small.

Action:

Standard user response. Fix the parameter **mxbuff**.

Message 45: error - too many atoms in CONFIG file

DL_POLY_2 limits the number of atoms in the system to be simulated and checks for the violation of this condition when it reads the CONFIG file. Termination will result if the condition is violated.

Action:

Standard user response. Fix the parameter **mxatms**. Consider the possibility that the wrong CONFIG file is being used (e.g similar system, but larger size.)

Message 46: ewlbuf array too small in ewald1

The **ewlbuf** array used to store structure factor data in subroutine EWALD1 has been dimensioned too small.

Action:

Standard user response. Fix the parameter **mxebuf**.

Message 47: error - transfer buffer too small in merge

The buffer used to transfer data between nodes in the MERGE subroutines has been dimensioned too small.

Action:

Standard user response. Fix the parameter **mxbuff**.

Message 48: error - transfer buffer too small in fortab

The buffer used to transfer data between nodes in the FORTAB subroutines has been dimensioned too small.

Action:

Standard user response. Fix the parameter **mxbuff**.

Message 49: error - frozen core-shell unit specified

The DL_POLY_2 option to freeze the location of an atom (i.e. hold it permanently in one position) is not permitted for core-shell units. This includes freezing the core or the shell independently.

Action:

Remove the frozen atom option from the FIELD file. Consider using a non-polarisable atom instead.

Message 50: error - too many bond angles specified

DL_POLY_2 limits the number of valence angle potentials that can be specified in the FIELD file and checks for the violation of this. Termination will result if the condition is violated. Do not confuse this error with that described by message 51 (below).

Action:

Standard user response. Fix the parameter `mxtang`.

Message 51: error - too many bond angles in system

DL_POLY_2 limits the number of valence angle potentials in the system to be simulated (actually, the number to be processed by each node) and checks for the violation of this. Termination will result if the condition is violated. Do not confuse this error with that described by message 50 (above).

Action:

Standard user response. Fix the parameter `mxangl`. Consider the possibility that the wrong CONFIG file is being used (e.g similar system, but larger size.)

Message 52: error - end of FIELD file encountered

This message results when DL_POLY_2 reaches the end of the FIELD file, without having read all the data it expects. Probable causes: missing data or incorrect specification of integers on the various directives.

Action:

Check FIELD file for missing or incorrect data and correct.

Message 53: error - end of CONTROL file encountered

This message results when DL_POLY_2 reaches the end of the CONTROL file, without having read all the data it expects. Probable cause: missing **finish** directive.

Action:

Check CONTROL file and correct.

Message 55: error - end of CONFIG file encountered

This error arises when DL_POLY_2 attempts to read more data from the CONFIG file than is actually present. The probable cause is an incorrect or absent CONFIG file, but it may be due to the FIELD file being incompatible in some way with the CONFIG file.

Action:

Check contents of CONFIG file. If you are convinced it is correct, check the FIELD file for inconsistencies.

Message 57: error - too many core-shell units specified

DL_POLY_2 has a restriction of the number of types of core-shell unit in the FIELD file and will terminate if too many are present. Do not confuse this error with that described by message 59 (below).

Action:

Standard user response. Fix the parameter `mxtshl`.

Message 59: error - too many core-shell units in system

DL_POLY_2 limits the number of core-shell units in the simulated system. Termination results if too many are encountered. Do not confuse this error with that described by message 57 (above).

Action:

Standard user response. Fix the parameter `mxshl`.

Message 60: error - too many dihedral angles specified

DL_POLY_2 will accept only a limited number of dihedral angles in the FIELD file and will terminate if too many are present. Do not confuse this error with that described by message 61 (below).

Action:

Standard user response. Fix the parameter `mxdih`.

Message 61: error - too many dihedral angles in system

The number of dihedral angles in the whole simulated system is limited by DL_POLY_2. Termination results if too many are encountered. Do not confuse this error with that described by message 60 (above).

Action:

Standard user response. Fix the parameter `mxdihd`.

Message 62: error - too many tethered atoms specified

DL_POLY_2 will accept only a limited number of tethered atoms in the FIELD file and will terminate if too many are present. Do not confuse this error with that described by message 63 (below).

Action:

Standard user response. Fix the parameter `mxteth`.

Message 63: error - too many tethered atoms in system

The number of tethered atoms in the simulated system is limited by DL_POLY_2 . Termination results if too many are encountered. Do not confuse this error with that described by message 62 (above).

Action:

Standard user response. Fix the parameter `msteth`.

Message 65: error - too many excluded pairs specified

This error can arise when DL_POLY_2 is identifying the atom pairs that cannot have a pair potential between them, by virtue of being chemically bonded for example (see subroutine EXCLUDE). Some of the working arrays used in this operation may be exceeded, resulting in termination of the program.

Action:

Standard user response. Fix the parameter `mxexcl`.

Message 66: error - incorrect boundary condition for HK ewald

The Hautman-Klein Ewald method can only be used with XY planar periodic boundary conditions (i.e. `imcon` = 6).

Action:

Either the periodic boundary condition, or the choice of calculation of the electrostatic forces must be changed.

Message 67: error - incorrect boundary condition in thbfr

Three body forces in DL_POLY_2 are only permissible with cubic , orthorhombic and parallelepiped periodic boundaries. Use of other boundary conditions results in this error.

Action:

If nonperiodic boundaries are required, the only option is to use a very large simulation cell,

with the required system at the centre surrounded by a vacuum. This is not very efficient however and use of a realistic periodic system is the best option.

Message 69: error - too many link cells required in thbfr

The calculation of three body forces in DL_POLY_2 is handled by the link cell algorithm. This error arises if the required number of link cells exceeds the permitted array dimension in the code.

Action:

Standard user response. Fix the parameter `mxcell`.

Message 70: error - constraint bond quench failure

When a simulation with bond constraints is started, DL_POLY_2 attempts to extract the kinetic energy of the constrained atom-atom bonds arising from the assignment of initial random velocities. If this procedure fails, the program will terminate. The likely cause is a badly generated initial configuration.

Action:

Some help may be gained from increasing the cycle limit, by following the standard user response to increase the control parameter `mxshak`. You may also consider reducing the tolerance of the SHAKE iteration, the directive `shake` in the CONTROL file. However it is probably better to take a good look at the starting conditions!

Message 71: error - too many metal potentials specified

The number of metal potentials that can be specified in the FIELD file is limited. This error results if too many are used.

Action:

Standard user response. Fix the parameter `mxvdw`. Note that this parameter must be *double* the number of required metal potentials. Recompile the program.

Message 73: error - too many inversion potentials specified

The number of inversion potentials specified in the FIELD file exceeds the permitted maximum.

Action:

Standard user response. Fix the parameter `mxtinv`.

Message 75: error - too many atoms in specified system

DL_POLY_2 places a limit on the number of atoms that can be simulated. Termination results if too many are specified.

Action:

Standard user response. Fix the parameter `mxatms`.

Message 77: error - too many inversion potentials in system

The simulation contains too many inversion potentials overall, causing termination of run.

Action:

Standard user response. Fix the parameter `mxinv`.

Message 79: error - incorrect boundary condition in fbpfrc

The 4-body force routine assumes a cubic or parallelepiped periodic boundary condition is in operation. The job will terminate if this is not adhered to.

Action:

You must reconfigure your simulation to an appropriate boundary condition.

Message 80: error - too many pair potentials specified

DL_POLY_2 places a limit on the number of pair potentials that can be specified in the FIELD file. Exceeding this number results in termination of the program execution.

Action:

Standard user response. Fix the parameters `mxsvdw` and `mxvdw`.

Message 81: error - unidentified atom in pair potential list

DL_POLY_2 checks all the pair potentials specified in the FIELD file and terminates the program if it can't identify any one of them from the atom types specified earlier in the file.

Action:

Correct the erroneous entry in the FIELD file and resubmit.

Message 82: error - calculated pair potential index too large

In checking the pair potentials specified in the FIELD file DL_POLY_2 calculates a unique integer index that henceforth identifies the potential within the program. If this index becomes too large, termination of the program results.

Action:

Standard user response. Fix the parameters `mxsvdw` and `mxvdw`.

Message 83: error - too many three body potentials specified

DL_POLY_2 has a limit on the number of three body potentials that can be defined in the FIELD file. This error results if too many are included.

Action:

Standard user response. Fix the parameter `mxtbp`.

Message 84: error - unidentified atom in 3-body potential list

DL_POLY_2 checks all the 3-body potentials specified in the FIELD file and terminates the program if it can't identify any one of them from the atom types specified earlier in the file.

Action:

Correct the erroneous entry in the FIELD file and resubmit.

Message 85: error - required velocities not in CONFIG file

If the user attempts to start up a DL_POLY_2 simulation with the **restart** or **restart scale** directives (see description of CONTROL file,) the program will expect the CONFIG file to contain atomic velocities as well as positions. Termination results if these are not present.

Action:

Either replace the CONFIG file with one containing the velocities, or if not available, remove the **restart** directive altogether and let DL_POLY_2 create the velocities for itself.

Message 86: error - calculated 3-body potential index too large

DL_POLY_2 has a permitted maximum for the calculated index for any three body potential in the system (i.e. as defined in the FIELD file). If there are m distinct types of atom in the system, the index can possibly range from 1 to $(m^2 * (m - 1))/2$. If the internally calculated index exceeds this number, this error report results.

Action:

Standard user response. Fix the parameter `mxtbp`.

Message 87: error - too many link cells required in fbpfrc

The FBPFRC subroutine uses link cells to compute the four body forces. This message indicates that the link cell arrays have insufficient size to work properly.

Action:

Standard user response. Fix the parameter `mxcell`.

Message 89: error - too many four body potentials specified

Too many four body potential have been defined in the FIELD file. Certain arrays must be increased in size to accommodate the data.

Action:

Standard user response. Fix the parameter `mxfbp`.

Message 90: error - system total electric charge nonzero

In DL_POLY_2 a check on the total system charge will result in an error if the net charge of the system is nonzero. (Note: In DL_POLY_2 this message has been disabled. The program merely prints a warning stating that the system is not electrically neutral but it does not terminate the program - watch out for this.)

Action:

Check the specified atomic charges and their populations. Make sure they add up to zero. If the system is required to have a net zero charge, you can enable the call to this error message in subroutine SYSDEF.

Message 91: error - unidentified atom in 4-body potential list

The specification of a four-body potential in the FIELD file has referenced an atom type that is unknown.

Action:

Locate the errant atom type in the four body potential definition in the FIELD file and correct. Make sure this atom type is specified by an `atoms` directive earlier in the file.

Message 93: error - cannot use shell model with rigid molecules

The dynamical shell model implemented in DL_POLY_2 is not designed to work with rigid molecules. This error results if these two options are simultaneously selected.

Action:

In some circumstances you may consider overriding this error message and continuing with your simulation. For example if your simulation does not require the polarisability to be a feature of the rigid species, but is confined to free atoms or flexible molecules in the same system. The appropriate error trap is found in subroutine SYSDEF.

Message 95: error - potential cutoff exceeds half cell width

In order for the minimum image convention to work correctly within DL_POLY_2, it is necessary to ensure that the cutoff applied to the pair potentials does not exceed half the perpendicular width of the simulation cell. (The perpendicular width is the shortest

distance between opposing cell faces.) Termination results if this is detected. In NVE simulations this can only happen at the start of a simulation, but in NPT, it may occur at any time.

Action:

Supply a cutoff that is less than half the cell width. If running constant pressure calculations, use a cutoff that will accommodate the fluctuations in the simulation cell. Study the fluctuations in the OUTPUT file to help you with this.

Message 97: error - cannot use shell model with neutral groups

The dynamical shell model was not designed to work with neutral groups. This error results if an attempt is made to combine both.

Action:

There is no general remedy for this error if you wish to combine both these capabilities. However if your simulation does not require the polarisability to be a feature of rigid species (comprising the charged groups), but is confined to free atoms or flexible molecules in the same system, you may consider overriding this error message and continuing with your simulation. The appropriate error trap is found in subroutine SYSDEF.

Message 99: error - cannot use shell model with constraints

The dynamical shell model was not designed to work in conjunction with constraint bonds. This error results if both are used in the same simulation.

Action: There is no general remedy if you wish to combine both these capabilities. However if your simulation does not require the polarisability to be a feature of the constrained species, but is confined to free atoms or flexible molecules, you may consider overriding this error message and continuing with your simulation. The appropriate error trap is in subroutine SYSDEF.

Message 100: error - forces working arrays too small

There are a number of arrays in DL_POLY_2 that function as workspace for the forces calculations. Their dimension is equal to the number of atoms in the simulation cell divided by the number of nodes. If these arrays are likely to be exceeded, DL_POLY_2 will terminate execution.

Action:

Standard user response. Fix the parameter `msatms`.

Message 101: error - calculated 4-body potential index too large

DL_POLY_2 has a permitted maximum for the calculated index for any four body potential in the system (i.e. as defined in the FIELD file). If there are m distinct types of atom in the system, the index can possibly range from 1 to $(m^2 * (m + 1) * (m + 2))/6$. If the internally calculated index exceeds this number, this error report results.

Action:

Standard user response. Fix the parameter **mxfbp**.

Message 102: error - parameter mxproc exceeded in shake arrays

The RD-SHAKE algorithm distributes data over all nodes of a parallel computer. Certain arrays in RD-SHAKE have a minimum dimension equal to the maximum number of nodes DL_POLY_2 is likely to encounter. If the actual number of nodes exceeds this, the program terminates.

Action:

Standard user response. Fix the parameter **mxproc**.

Message 103: error - parameter mxlshp exceeded in shake arrays

The RD-SHAKE algorithm requires that information about ‘shared’ atoms be passed between nodes. If there are too many atoms, the arrays holding the information will be exceeded and DL_POLY_2 will terminate execution.

Action:

Standard user response. Fix the parameter **mxlshp**.

Message 105: error - shake algorithm failed to converge

The RD-SHAKE algorithm for bond constraints is iterative. If the maximum number of permitted iterations is exceeded, the program terminates. Possible causes include: a bad starting configuration; too large a time step used; incorrect force field specification; too high a temperature; inconsistent constraints involving shared atoms etc.

Action:

Corrective action depends on the cause. It is unlikely that simply increasing the iteration number will cure the problem, but you can try: follow the standard user response to increase the control parameter **mxshak**. But the trouble is much more likely to be cured by careful consideration of the physical system being simulated. For example, is the system stressed in some way? Too far from equilibrium?

Message 106: error - neighbour list array too small in parlink

Construction of the Verlet neighbour list in subroutine `parlink` nonbonded (pair) force has exceeded the neighbour list array dimensions.

Action:

Standard user response. Fix the parameter `mxlist`.

Message 107: error - neighbour list array too small in parlinkneu

Construction of the Verlet neighbour list in subroutine `parlinkneu` nonbonded (pair) force has exceeded the neighbour list array dimensions.

Action:

Standard user response. Fix the parameter `mxlist`.

Message 108: error - neighbour list array too small in parneulst

Construction of the Verlet neighbour list in subroutine `parneulst` nonbonded (pair) force has exceeded the neighbour list array dimensions.

Action:

Standard user response. Fix the parameter `mxlist`.

Message 109: error - neighbour list array too small in parlst_nsq

Construction of the Verlet neighbour list in subroutine `parlst_nsq` nonbonded (pair) force has exceeded the neighbour list array dimensions.

Action:

Standard user response. Fix the parameter `mxlist`.

Message 110: error - neighbour list array too small in parlst

Construction of the Verlet neighbour list in subroutine `parlst` nonbonded (pair) force has exceeded the neighbour list array dimensions.

Action:

Standard user response. Fix the parameter `mxlist`.

Message 112: error - vertest array too small

This error results when the dimension of the `DL_POLY_2` `VERTEST` arrays, which are used in checking if the Verlet list needs updating, have been exceeded.

Action:

Standard user response. Fix the parameter `mslst`.

Message 120: error - invalid determinant in matrix inversion

DL_POLY_2 occasionally needs to calculate matrix inverses (usually the inverse of the matrix of cell vectors, which is of size 3×3). For safety's sake a check on the determinant is made, to prevent inadvertent use of a singular matrix.

Action:

Locate the incorrect matrix and fix it - e.g. are cell vectors correct?

Message 130: error - incorrect octahedral boundary condition

When calculating minimum images DL_POLY_2 checks that the periodic boundary of the simulation cell is compatible with the specified minimum image algorithm. Program termination results if an inconsistency is found. In this case the error refers to the truncated octahedral minimum image, which is inconsistent with the simulation cell. The most probable cause is the incorrect definition of the simulation cell vectors present in the input file CONFIG, these must equal the vectors of the enclosing cubic cell.

Action:

Check the specified simulation cell vectors and correct accordingly.

Message 135: error - incorrect hexagonal prism boundary condition

When calculating minimum images DL_POLY_2 checks that the periodic boundary of the simulation cell is compatible with the specified minimum image algorithm. Program termination results if an inconsistency is found. In this case the error refers to the hexagonal prism minimum image, which is inconsistent with the simulation cell. The most probable cause is the incorrect definition of the simulation cell vectors present in the input file CONFIG, these must equal the vectors of the enclosing orthorhombic cell.

Action:

Check the specified simulation cell vectors and correct accordingly.

Message 140: error - incorrect dodecahedral boundary condition

When calculating minimum images DL_POLY_2 checks that the periodic boundary of the simulation cell is compatible with the specified minimum image algorithm. Program termination results if an inconsistency is found. In this case the error refers to the rhombic dodecahedral minimum image, which is inconsistent with the simulation cell. The most probable cause is the incorrect definition of the simulation cell vectors present in the input file CONFIG, these must equal the vectors of the enclosing tetragonal simulation cell.

Action:

Check the specified simulation cell vectors and correct accordingly.

Message 145: error - no van der waals potentials defined

This error arises when there are no VDW potentials specified in the FIELD file but the user has not specified **no vdw** in the CONTROL file. In other words DL_POLY_2 expects the FIELD file to contain VDW potential specifications.

Action:

Edit the FIELD file to insert the required potentials or specify **no vdw** in the CONTROL file.

Message 150: error - unknown van der waals potential selected

DL_POLY_2 checks when constructing the interpolation tables for the short ranged potentials that the potential function requested is one which is of a form known to the program. If the requested potential form is unknown, termination of the program results. The most probable cause of this is the incorrect choice of the potential keyword in the FIELD file or one in the wrong columns (input is formatted).

Action:

Read the DL_POLY_2 documentation and find the potential keyword for the potential desired. Insert the correct index in the FIELD file definition and ensure it occurs in the correct columns (17-20). If the correct form is not available, look at the subroutine FORGEN (or its variant) and define the potential for yourself. It is easily done.

Message 151: error - unknown metal potential selected

The metal potentials available in DL_POLY_2 are confined to density dependent forms of the Sutton-Chen type. This error results if the user attempts to specify another.

Action:

Re-specify the potential as Sutton-Chen type if possible. Check the potential keyword appears in columns 17-20 of the FIELD file.

Message 153: error - metals not permitted with multiple timestep

The multiple timestep algorithm cannot be used in conjunction with metal potentials in DL_POLY_2 .

Action:

The simulation must be run without the multiple timestep option.

Message 160: error - unaccounted for atoms in exclude list

This error message means that DL_POLY_2 has been unable to find all the atoms described in the exclusion list within the simulation cell. This should never occur, if it does it means a serious bookkeeping error has occurred. The probable cause is corruption of the code somehow.

Action:

If you feel you can tackle it - good luck! Otherwise we recommend you get in touch with the program authors. Keep all relevant data files to help them find the problem.

Message 170: error - too many variables for statistic array

This error means the statistics arrays appearing in subroutine STATIC are too small. This can happen if the number of unique atom types is too large.

Action:

Standard user response. Fix the parameter `mxnstk`. `mxnstk` should be at least $(45 + \text{number of unique atom types})$.

Message 180: error - Ewald sum requested in non-periodic system

DL_POLY_2 can use either the Ewald method or direct summation to calculate the electrostatic potentials and forces in periodic (or pseudo-periodic) systems. For non-periodic systems only direct summation is possible. If the Ewald summation is requested (with the **ewald sum** or **ewald precision** directives in the CONTROL file) without periodic boundary conditions, termination of the program results.

Action:

Select periodic boundaries by setting the variable `imcon` > 0 in the CONFIG file (if possible) or use a different method to evaluate electrostatic interactions e.g. by using the **coul** directive in the CONTROL file.

Message 185: error - too many reciprocal space vectors

DL_POLY_2 places hard limit on the number of *k* vectors to be used in the Ewald sum and terminates if more than this is requested.

Action:

Either consider using fewer *k* vectors in the Ewald sum (and a larger cutoff in real space) or follow standard user response to reset the parameters `kmaxb`, `kmaxc`.

Message 186: error - transfer buffer array too small in sysgen

In the subroutine SYSGEN.F DL_POLY_2 requires dimension of the array `buffer` (defined by the parameter `mxbuff`) to be no less than the parameter `mxatms` or the product of pa-

parameters `mxnstk*mxstak`. If this is not the case it will be unable to restart the program correctly to continue a run. (Applies to parallel implementations only.)

Action:

Standard user response. Fix the parameter `mxbuff`.

Message 190: error - buffer array too small in splice

DL_POLY_2 uses a workspace array named `buffer` in several routines. Its declared size is a compromise of several rôles and may sometimes be too small (though in the supplied program, this should happen only very rarely). The point of failure is in the `SPLICE` routine, which is part of the RD-SHAKE algorithm.

Action:

Standard user response. Fix the parameter `mxbuff`.

Message 200: error - rdf buffer array too small in revive

This error indicates that the buffer array used to globally sum the rdf arrays in subroutine `REVIVE` is too small.

Action:

Standard user response. Fix the parameter `mxbuff`. Alternatively `mxrdf` can be set smaller.

Message 220: error - too many neutral groups in system

DL_POLY_2 has a fixed limit on the number of charged groups in a simulation. This error results if the number is exceeded.

Action:

Standard user response. Fix the parameter `mxneut`.

Message 230: error - neutral groups improperly arranged

In the DL_POLY_2 `FIELD` file the charged groups must be defined in consecutive order. This error results if this convention is not adhered to.

Action:

The arrangement of the data in the `FIELD` file must be sorted. All atoms in the same group must be arranged consecutively. Note that reordering the file in this way implies a rearrangement of the `CONFIG` file also.

Message 250: error - Ewald sum requested with neutral groups

DL_POLY_2 will not permit the use of neutral groups with the Ewald sum. This error results if the two are used together.

Action:

Either remove the **neut** directive from the FIELD file or use a different method to evaluate the electrostatic interactions.

Message 260: error - parameter mxexcl exceeded in excludeneu routine

An error has been detected in the construction of the excluded atoms list for neutral groups. This occurs when the parameter **mxexcl** is exceeded in the EXCLUDENEU routine.

Action:

Standard user response. Fix parameter **mxexcl**.

Message 300: error - incorrect boundary condition in parlink

The use of link cells in DL_POLY_2 implies the use of appropriate boundary conditions. This error results if the user specifies octahedral, dodecahedral or slab boundary conditions.

Action:

The simulation must be run with cubic, orthorhombic or parallelepiped boundary conditions.

Message 301: error - too many rigid body types

The maximum number of rigid body types permitted by DL_POLY_2 has been exceeded.

Action:

Standard user response. Fix the parameter **mxungp**.

Message 302: error - too many sites in rigid body

This error arises when DL_POLY_2 finds that the number of sites in a rigid body exceeds the dimensions of the appropriate storage arrays.

Action:

Standard user response. Fix the parameter **mxngp**.

Message 303: error - too many rigid bodies specified

The maximum number of rigid bodies in a simulation has been reached. Do not confuse this with message 304 below.

Action:

Standard user response. Fix the parameter `mxgrp`.

Message 304: error - too many rigid body sites in system

This error occurs when the total number of sites within all rigid bodies exceeds the permitted maximum. Do not confuse this with message 303 above.

Action:

Standard user response. Fix the parameter `mxgatms`.

Message 305: error - box size too small for link cells

The link cells algorithm in DL_POLY_2 cannot work with less than 27 link cells. Depending on the cell size and the chosen cut-off, DL_POLY_2 may decide that this minimum cannot be achieved and terminate.

Action:

If a smaller cut-off is acceptable use it. Otherwise do not use link cells. Consider running a larger system, where link cells will work.

Message 306: error - failed to find principal axis system

This error indicates that the routine QUATBOOK has failed to find the principal axis for a rigid unit.

Action:

This is an unlikely error. The code should correctly handle linear, planar and 3-dimensional rigid units. Check the definition of the rigid unit in the CONFIG file, if sensible report the error to the authors.

Message 310: error - quaternion setup failed

This error indicates that the routine QUATBOOK has failed to reproduce all the atomic positions in rigid units from the centre of mass and quaternion vectors it has calculated.

Action:

Check the contents of the CONFIG file. DL_POLY_2 builds its local body description of a rigid unit type from the *first* occurrence of such a unit in the CONFIG file. The error most likely occurs because subsequent occurrences were not sufficiently similar to this reference structure. If the problem persists increase the value of the variable `tol` in QUATBOOK and recompile. If problems still persist double the value of `dettest` in QUATBOOK and recompile. If you still encounter problems contact the authors.

Message 320: error - site in multiple rigid bodies

DL_POLY_2 has detected that a site is shared by two or more rigid bodies. There is no integration algorithm available in this version of the package to deal with this type of model.

Action:

The only course is to redefine the molecular model (e.g. introducing flexible bonds and angles in suitable places) to allow DL_POLY_2 to proceed.

Message 321: error - quaternion integrator failed

The quaternion algorithm has failed to converge. If the maximum number of permitted iterations is exceeded, the program terminates. Possible causes include: a bad starting configuration; too large a time step used; incorrect force field specification; too high a temperature; inconsistent constraints involving shared atoms etc.

Action:

Corrective action depends on the cause. Try reducing the timestep or running a zero kelvin structure optimization for a hundred timesteps or so. It is unlikely that simply increasing the iteration number will cure the problem, but you can try: follow the standard user response to increase the parameter `mxquat`. But the trouble is much more likely to be cured by careful consideration of the physical system being simulated. For example, is the system stressed in some way? Too far from equilibrium?

Message 330: error - mxewld parameter incorrect

DL_POLY_2 has two strategies for parallelization of the reciprocal space part of the Ewald sum. If EWALD1 is used the parameter `mxewld` should equal the parameter `msatms`. If EWALD1A is used this parameter should equal `mxatms`.

Action:

Standard user response. Set the parameter `mxewld` to the value appropriate for the version of EWALD1 you are using. Recompile the program.

Message 331: error - mxhke parameter incorrect

The parameter `mxhke`, which defines the dimension of some arrays used in the Hautman-Klein Ewald method, should equal the parameter `msatms`.

Action:

Standard user response. Set the parameter `mxhke` to the value required. Recompile the program.

Message 332: error - mxhko parameter too small

The parameter **mxhko** defines the maximum order for the Taylor expansion implicit in the Hautman-Klein Ewald method. DL_POLY_2 has a maximum of **mxhko** = 3, but it can be set to less in some implementations. If this error arises when the user requestes an order in excess of this parameter.

Action:

Standard user response. Set the parameter **mxhko** to a higher value (if it is <3) and recompile the program. Alternatively request a lower order in the CONTROL file through the **nhko** variable (see 4.1.1).

Message 340: error - invalid integration option requested

DL_POLY_2 has detected an incompatibility in the simulation instructions, namely that the requested integration algorithm is not compatible with the physical model. It *may* be possible to override this error trap, but it is up to the user to establish if this is sensible.

Action:

This is a non recoverable error, unless the user chooses to override the restriction.

Message 350: error - too few degrees of freedom

This error can arise if a small system is being simulated and the number of constraints applied is too large.

Action:

Simulate a larger system or reduce the number of constraints.

Message 360: error - frozen atom found in rigid body

DL_POLY_2 does not permit a site in a rigid body to be frozen i.e. fixed in one location in space.

Action:

Remove the ‘freeze’ condition from the site concerned. Consider using a very high site mass to achieve a similar effect.

Message 380: error - simulation temperature not specified

DL_POLY_2 has failed to find a **temp** directive in the CONTROL file.

Action:

Place a **temp** directive in the CONTROL file, with the required temperature specified.

Message 381: error - simulation timestep not specified

DL_POLY_2 has failed to find a **timestep** directive in the CONTROL file.

Action:

Place a **timestep** directive in the CONTROL file, with the required timestep specified.

Message 382: error - simulation cutoff not specified

DL_POLY_2 has failed to find a **cutoff** directive in the CONTROL file.

Action:

Place a **cutoff** directive in the CONTROL file, with the required forces cutoff specified.

Message 383: error - simulation forces option not specified

DL_POLY_2 has failed to find any directive specifying the electrostatic interactions options in the CONTROL file.

Action:

Ensure the CONTROL file contains at least one directive specifying the electrostatic potentials (e.g. **ewald**, **coul**, **no electrostatics** etc.)

Message 384: error - verlet strip width not specified

DL_POLY_2 has failed to find the **delr** directive in the CONTROL file.

Action:

Insert a **delr** directive in the CONTROL file, specifying the width of the verlet strip augmenting the forces cutoff.

Message 385: error - primary cutoff not specified

DL_POLY_2 has failed to find the **prim** directive in the CONTROL file. Necessary only if multiple timestep option required.

Action:

Insert a **prim** directive in the CONTROL file, specifying the primary cutoff radius in the multiple timestep algorithm.

Message 386: error - primary cutoff larger than rcut

The primary cutoff specified by the **prim** directive in the CONTROL file exceeds the value specified for the forces cutoff (directive **cut**). Applies only if the multiple timestep option is required.

Action:

Locate the **prim** directive in the CONTROL file, and alter the chosen cutoff. Alternatively, increase the real space cutoff specified with the **cut** directive. Take care to avoid error number 398.

Message 387: error - system pressure not specified

The target system pressure has not been specified in the CONTROL file. Applies to NPT simulations only.

Action:

Insert a **press** directive in the CONTROL file specifying the required system pressure.

Message 388: error - npt incompatible with multiple timestep

The use of NPT (constant pressure) and temperature is not compatible with the multiple timestep option.

Action:

Simulation must be run at fixed volume in this case. But note it may be possible to use NPT without the multiple timestep, in order to estimate the required system volume, then switch back to multiple timestep and NVT dynamics at the required volume.

Message 389: number of pimd beads not specified in field file

The user has failed to specify how many quantum beads is required in a Path Integral Molecular Dynamics simulation. Applies to PIMD version of DL_POLY_2 only.

Action:

The required number of beads must be specified in the FIELD file.

Message 390: error - npt ensemble requested in non-periodic system

A non-periodic system has no defined volume, hence the NPT algorithm cannot be applied.

Action:

Either simulate the system with a periodic boundary, or use another ensemble.

Message 391: incorrect number of pimd beads in config file

The CONFIG file must specify the position of all the beads in a PIMD simulation, not just the positions of the parent atoms, otherwise this error results.

Action:

The CONFIG file must be reconstructed to provide the required data.

Message 392: error - too many link cells requested

The number of link cells required for a given simulation exceeds the number allowed for by the DL_POLY_2 arrays.

Action:

Standard user response. Fix the parameter `mxcell`.

Message 394: error - minimum image arrays exceeded

The work arrays used in IMAGES have been exceeded.

Action: Standard user response. Fix the parameter `mxxdf`.

Message 396: error - interpolation array exceeded

DL_POLY_2 has sought to read past the end of an interpolation array. This should never happen!

Action:

Contact the authors.

Message 398: error - cutoff too small for rprim and delr

This error can arise when the multiple timestep option is used. It is essential that the primary cutoff (`rprim`) is less than the real space cutoff (`rcut`) by at least the Verlet shell width `delr` (preferably much larger!). DL_POLY_2 terminates the run if this condition is not satisfied.

Action:

Adjust `rcut`, `rprim` and `delr` to satisfy the DL_POLY_2 requirement. These are defined with the directives `cut`, `prim` and `delr` respectively.

Message 400: error - rvdw greater than cutoff

DL_POLY_2 requires the real space cutoff (`rcut`) to be larger than, or equal to, the van der Waals cutoff (`rvdw`) and terminates the run if this condition is not satisfied.

Action:

Adjust `rvdw` and `rcut` to satisfy the DL_POLY_2 requirement.

Message 402: error - van der waals cutoff unset

The user has not set a cutoff (`rvdw`) for the van der Waals potentials. The simulation cannot proceed without this being specified.

Action:

Supply a cutoff value for the van der Waals terms in the CONTROL file using the directive **rvdw**, and resubmit job.

Message 410: error - cell not consistent with image convention

The simulation cell vectors appearing in the CONFIG file are not consistent with the specified image convention.

Action:

Locate the variable **imcon** in the CONFIG file and correct to suit the cell vectors.

Message 412: error - mxxdf parameter too small for shake routine

In DL_POLY_2 the parameter **mxxdf** must be greater than or equal to the parameter **mxcons**. If it is not, this error is a possible result.

Action:

Standard user response. Fix the parameter **mxxdf**.

Message 414: error - conflicting ensemble options in CONTROL file

DL_POLY_2 has found more than one **ensemble** directive in the CONTROL file.

Action:

Locate extra **ensemble** directives in CONTROL file and remove.

Message 416: error - conflicting force options in CONTROL file

DL_POLY_2 has found incompatible directives in the CONTROL file specifying the electrostatic interactions options.

Action:

Locate the conflicting directives in the CONTROL file and correct.

Message 418: error - bond vector work arrays too small in bndfrc

The work arrays in BDNFRC have been exceeded.

Action:

Standard user response. Fix the parameter **msbad**.

Message 419: error - bond vector work arrays too small in angfrc

The work arrays in ANGFRC have been exceeded.

Action:

Standard user response. Fix the parameter `msbad`.

Message 420: error - bond vector work arrays too small in tethfrc

The work arrays in TETHFRC have been exceeded.

Action:

Standard user response. Fix the parameter `msbad`.

Message 421: error - bond vector work arrays too small in dihfrc

The work arrays in DIHFRC have been exceeded.

Action:

Standard user response. Fix the parameter `msbad`.

Message 422: error - all-pairs must use multiple timestep

In DL_POLY_2 the 'all pairs' option must be used in conjunction with the multiple timestep.

Action:

Activate the multiple timestep option in the CONTROL file and resubmit.

Message 423: error - bond vector work arrays too small in shlfrc

The dimensions of the interatomic distance vectors have been exceeded in subroutine SHLFRC.

Action:

Standard user response. Fix the parameter `msbad`. Set equal to the value of the parameter `mxshl`.

Message 424: error - electrostatics incorrect for all-pairs

When using the all pairs option in conjunction with electrostatic forces, the electrostatics must be handled with either the standard Coulomb sum, or with the distance dependent dielectric.

Action:

Rerun the simulation with the appropriate electrostatic option.

Message 425: error - transfer buffer array too small in shlmerge

The buffer used to transfer data between nodes in the subroutine SHLMERGE has been dimensioned too small.

Action:

Standard user response. Fix the parameter `mxbuff`.

Message 426: error - neutral groups not permitted with all-pairs

DL_POLY_2 will not permit simulations using both the neutral group and all pairs options together.

Action:

Switch off one of the conflicting options and rerun.

Message 427: error - bond vector work arrays too small in invfrc

The work arrays in subroutine INVFRM have been exceeded.

Action:

Standard user response. Fix the parameter `msbad`.

Message 430: error - integration routine not available

A request for a nonexistent ensemble has been made or a request with conflicting options that DL_POLY_2 cannot deal with (e.g. a Evans thermostat with rigid body equations of motion).

Action:

Examine the CONTROL and FIELD files and remove inappropriate specifications.

Message 432: error - intlist failed to assign constraints

If the required simulation has constraint bonds DL_POLY_2 attempts to apportion the molecules to processors so that, if possible, there are no shared atoms between processors. If this is not possible, one or more molecules may be split between processors. This message indicates that the code has failed to carry out either of these successfully.

Action:

The error may arise from a compiler error. Try recompiling INTLIST without the optimization flag turned on. If the problem persists it should be reported to the authors, (after checking the input data for inconsistencies).

Message 433: error - specify rcut before the Ewald sum precision

When specifying the desired precision for the Ewald sum in the CONTROL file, it is first necessary to specify the real space cutoff **rcut**.

Action:

Place the **cut** directive *before* the **ewald precision** directive in the CONTROL file and rerun.

Message 434: error - illegal entry into STRESS related routine

The calculation of the stress tensor in DL_POLY_2 requires additional code that must be included at compile time through the use of the STRESS keyword. If this is not done, and DL_POLY_2 is later required to calculate the stress tensor, this error will result.

Action:

The program must be recompiled with the STRESS keyword activated. This will ensure all the relevant code is in place. See section 3.2.1.

Message 436: error - unrecognised ensemble

An unknown ensemble option has been specified in the CONTROL file.

Action:

Locate **ensemble** directive in the CONTROL file and amend appropriately.

Message 438: error - PMF constraints failed to converge

The constraints in the potential of mean force algorithm have not converged in the permitted number of cycles. (The SHAKE algorithm for PMF constraints is iterative.) Possible causes include: a bad starting configuration; too large a time step used; incorrect force field specification; too high a temperature; inconsistent constraints involving shared atoms etc.

Action:

Corrective action depends on the cause. It is unlikely that simply increasing the iteration number will cure the problem, but you can try: follow standard user response to increase the parameter **mxshak**. But the trouble is much more likely to be cured by careful consideration of the physical system being simulated. For example, is the system stressed in some way? Too far from equilibrium?

Message 440: error - undefined angular potential

A form of angular potential has been requested which DL_POLY_2 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable

to DL_POLY_2 if this is possible. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines SYSDEF and ANGFRM will be required.

Message 442: error - undefined three body potential

A form of three body potential has been requested which DL_POLY_2 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_2 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines SYSDEF and THBFRM will be required.

Message 443: error - undefined four body potential

DL_POLY_2 has been requested to process a four-body potential it does not recognise.

Action:

Check the FIELD file and make sure the keyword is correctly defined. Make sure that subroutine FBPMFRM contains the code necessary to deal with the requested potential. Add the code required if necessary, by amending subroutines SYSDEF and FBPMFRM.

Message 444: error - undefined bond potential

DL_POLY_2 has been requested to process a bond potential it does not recognise.

Action:

Check the FIELD file and make sure the keyword is correctly defined. Make sure that subroutine BNDFRM contains the code necessary to deal with the requested potential. Add the code required if necessary, by amending subroutines SYSDEF and BNDFRM.

Message 446: error - undefined electrostatic key in dihfrc

The subroutine DIHFRC has been requested to process a form of electrostatic potential it does not recognise.

Action:

The error arises because the integer key **keyfrc** has an inappropriate value (which should not happen in the standard version of DL_POLY_2). Check that the FIELD file correctly specifies the potential. Make sure the version of DIHFRC does contain the potential you are specifying. Report the error to the authors if these checks are correct.

Message 448: error - undefined dihedral potential

A form of dihedral potential has been requested which DL_POLY_2 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_2 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines SYSDEF and DIHFRC (and its variants) will be required.

Message 449: error - undefined inversion potential

A form of inversion potential has been encountered which DL_POLY_2 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_2 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines SYSDEF and INVFRG will be required.

Message 450: error - undefined tethering potential

A form of tethering potential has been requested which DL_POLY_2 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_2 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines SYSDEF and TETHFRG will be required.

Message 451: error - three body potential cutoff undefined

The cutoff radius for a three body potential has not been defined in the FIELD file.

Action:

Locate the offending three body force potential in the FIELD file and add the required cutoff. Resubmit the job.

Message 452: error - undefined pair potential

A form of pair potential has been requested which DL_POLY_2 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable

to DL_POLY_2 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines SYSDEF and FORGEN will be required.

Message 453: error - four body potential cutoff undefined

The cutoff radius for a four-body potential has not been defined in the FIELD file.

Action:

Locate the offending four body force potential in the FIELD file and add the required cut-off. Resubmit the job.

Message 454: error - undefined external field

A form of external field potential has been requested which DL_POLY_2 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_2 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines SYSDEF and EXTNFLD will be required.

Message 456: error - core and shell in same rigid unit

It is not sensible to fix both the core and the shell of a polarisable atom in the same molecular unit. Consequently DL_POLY_2 will abandon the job if this is found to be the case.

Action:

Locate the offending core-shell unit (there may be more than one in your FIELD file) and release the shell (preferably) from the rigid body specification.

Message 458: error - too many PMF constraints - param. mspmf too small

The number of constraints in the potential of mean force is too large. The dimensions of the appropriate arrays in DL_POLY_2 must be increased.

Action:

Standard user response. Fix the parameter mspmf.

Message 460: error - too many PMF sites - parameter mxspmf too small

The number of sites defined in the potential of mean force is too large. The dimensions of the appropriate arrays in DL_POLY_2 must be increased.

Action:

Standard user response. Fix the parameter **mxspmf**.

Message 462: error - PMF UNIT record expected

A **pmf unit** directive was expected as the next record in the FIELD file but was not found.

Action:

Locate the **pmf** directive in the FIELD file and examine the following entries. Insert the missing **pmf unit** directive and resubmit.

Message 464: error - thermostat time constant must be > 0.d0

A zero or negative value for the thermostat time constant has been encountered in the CONTROL file.

Action:

Locate the **ensemble** directive in the CONTROL file and assign a positive value to the time constant.

Message 466: error - barostat time constant must be > 0.d0

A zero or negative value for the barostat time constant has been encountered in the CONTROL file.

Action:

Locate the **ensemble** directive in the CONTROL file and assign a positive value to the time constant.

Message 468: error - r0 too large for snm potential with current cutoff

The specified location (**r0**) of the potential minimum for a shifted n-m potential exceeds the specified potential cutoff. A potential with the desired minimum cannot be created.

Action:

To obtain a potential with the desired minimum it is necessary to increase the van der Waals cutoff. Locate the **rvdw** directive in the CONTROL file and reset to a magnitude greater than **r0**. Alternatively adjust the value of **r0** in the FIELD file. Check that the FIELD file is correctly formatted.

Message 470: error - n<m in definition of n-m potential

The specification of a n-m potential in the FIELD file implies that the exponent **m** is larger than exponent **n**. (Not all versions of DL_POLY_2 are affected by this.)

Action:

Locate the n-m potential in the FIELD file and reverse the order of the exponents. Resubmit the job.

Message 474: error - mxxdf too small in parlst subroutine

The parameter **mxxdf** defining working arrays in subroutine PARLST of DL_POLY_2 has been found to be too small.

Action:

Standard user response. Fix the parameter **mxxdf**.

Message 475: error - mxxdf too small in parlst_nsq subroutine

The parameter **mxxdf** defining working arrays in subroutine PARLST_NSQ DL_POLY_2 has been found to be too small.

Action:

Standard user response. Fix the parameter **mxxdf**.

Message 476: error - mxxdf too small in parneulst subroutine

The parameter **mxxdf** defining working arrays in subroutine PARNEULST is too small.

Action:

Standard user response. Fix the parameter **mxxdf**.

Message 477: error - mxxdf too small in prneulst subroutine

The parameter **mxxdf** defining working arrays in subroutine PRNEULST is too small.

Action:

Standard user response. Fix the parameter **mxxdf**.

Message 478: error - mxxdf too small in forcesneu subroutine

The parameter **mxxdf** defining working arrays in subroutine FORCESNEU is too small.

Action:

Standard user response. Fix the parameter **mxxdf**.

Message 479: error - mxxdf too small in multipleneu subroutine

The parameter **mxxdf** defining working arrays in subroutine MULTIPLNEU is too small.

Action:

Standard user response. Fix the parameter `mxxdf`.

Message 484: error - only one potential of mean force permitted

It is not permitted to define more than one potential of mean force in the FIELD file.

Action:

Check that the FIELD file contains only one PMF specification. If more than one is needed, DL_POLY_2 cannot handle it.

Message 486: error - HK real space screening function cutoff violation

DL_POLY_2 has detected an unacceptable degree of inaccuracy in the screening function near the radius of cutoff *in real space*, which implies the Hautman-Klein Ewald method will not be sufficiently accurate.

Action:

The user should respecify the HK control parameters given in the CONTROL file. Either the convergence parameter should be increased or the sum expanded to incorporate more images of the central cell. (Warning: increasing the convergence parameter may cause failure in the reciprocal space domain.) (See 4.1.1).

Message 487: error - HK recip space screening function cutoff violation

DL_POLY_2 has detected an unacceptable degree of inaccuracy in the screening function near the radius of cutoff *in reciprocal space*, which implies the Hautman-Klein Ewald method will not be sufficiently accurate.

Action:

The user should respecify the HK control parameters given in the CONTROL file. Either the convergence parameter should be reduced or more k-vectors used. (Warning: reducing the convergence parameter may cause failure in the real space domain.) (See 4.1.1).

Message 488: error - HK lattice control parameter set too large

The Hautman-Klein Ewald method in DL_POLY_2 permits the user to perform a real space sum over nearest-neighbour and next-nearest-neighbour cells (i.e. up to `nlatt=2`). If the user specifies a larger sum than this, this error will result.

Action:

The user should respecify the HK control parameters given in the CONTROL file and set `nlatt` to a maximum of 2. (See 4.1.1).

Message 490: error - PMF parameter mxpmf too small in passpmf

The bookkeeping arrays have been exceeded in PASSPMF

Action:

Standard user response. Fix the parameter `mxpmf`. Set equal to `mxatms`.

Message 492: error - parameter mxcons < number of PMF constraints

The parameter `mxcons` is too small for the number of PMF constraints in the system.

Action:

Standard user response. Fix the value of `mxcons`.

Message 494: error in csend: pvmfinit send

The PVM routine PVMFINITSEND has returned an error. It is invoked by the routine CSEND.

Action:

Check your system implementation of PVM.

Message 496: error in csend: pvmfpack

The PVM routine PVMFPACK has returned an error. It is invoked by the routine CSEND.

Action:

Check your system implementation of PVM.

Message 498: error in csend: pvmf send

The PVM routine PVMFSEND has returned an error. It is invoked by the routine CSEND.

Action:

Check your system implementation of PVM.

Message 500: error in crecv: pvmfrecv

The PVM routine PVMFRECV has returned an error. It is invoked by the routine CRECV.

Action:

Check your system implementation of PVM.

Message 502: error in crecv: pvmfunpack

The PVM routine PVMFUNPACK has returned an error. It is invoked by the routine CRECV.

Action:

Check your system implementation of PVM.

Message 504: error - cutoff too large for TABLE file

The requested cutoff exceeds the information in the TABLE file.

Action:

Reduce the value of the vdw cutoff (`rvdw`) in the CONTROL file or reconstruct the TABLE file.

Message 506: error - work arrays too small for quaternion integration

The working arrays associated with quaternions are too small for the size of system being simulated. They must be redimensioned.

Action:

Standard user response. Fix the parameter `msggrp`.

Message 508: error - rigid bodies not permitted with RESPA algorithm

The RESPA algorithm implemented in DL_POLY_2 is for atomic systems only. Rigid bodies or constraints cannot be treated.

Action:

There is no cure for this. The code simply does not have this capability. Consider writing it for yourself!

Message 510: error - structure optimiser not permitted with RESPA

The RESPA algorithm in DL_POLY_2 has not been implemented to work with the structure optimizer. You have asked for a forbidden operation.

Action:

There is no fix for this. In any case it does not make sense to use the RESPA algorithm for this purpose.

Message 513: error - SPME not available for given boundary conditions

The SPME algorithm in DL_POLY_2 does not work for aperiodic (IMCON=0) or slab (IMCON=6) boundary conditions.

Action:

If the system must have aperiodic or slab boundaries, nothing can be done. In the latter case however, it may be acceptable to represent the same system with slabs replicated in the z direction, thus permitting a periodic simulation.

Message 514: error - SPME routines have not been compiled in

The inclusion of the SPME algorithm in DL_POLY_2 is optional at the compile stage. If the executable does not contain the SPME routines, but the method is requested by the user, this error results.

Action:

DL_POLY_2 must be recompiled with the SPME flags set. Beware that your system has the necessary fast Fourier transform routines to permit this.

Appendix D

DL_POLY Arrays

<code>a(9)</code>	work array
<code>aaa(9)</code>	work array
<code>aaa(nnn)</code>	work array
<code>accum(mxungp)</code>	work array
<code>ahk(0:mxhko)</code>	coefficients for the HK-Ewald expansion
<code>amsd(mxsvdw)</code>	mean squared displacement array
<code>amt(2)</code>	work array
<code>atmnam(mxatms)</code>	names of atoms in the system
<code>b(9)</code>	work array
<code>bbb(10)</code>	work array
<code>bbb(nnn)</code>	work array
<code>bscx(kmaxd)</code>	work array, B-splines x coefficients
<code>bscy(kmaxe)</code>	work array, B-splines y coefficients
<code>bscz(kmaxf)</code>	work array, B-splines z coefficients
<code>bsdx(mxspme,mxspl)</code>	work array, B-spline x derivatives
<code>bsdy(mxspme,mxspl)</code>	work array B-spline y derivatives
<code>bsdz(mxspme,mxspl)</code>	work array B-spline z derivatives
<code>bspix(mxspme,mxspl)</code>	work array B-splines x
<code>bspy(mxspme,mxspl)</code>	work array B-splines y
<code>bspz(mxspme,mxspl)</code>	work array B-splines z
<code>buff(4)</code>	work array
<code>buffer(4)</code>	work array
<code>buffer(8)</code>	work array
<code>buffer(10)</code>	work array
<code>buffer(mxatms)</code>	transfer buffer in ewald routines
<code>buffer(mxbuff)</code>	main coordinate transfer buffer
<code>cell(9)</code>	simulation cell vectors (1-3)=A vector, (4-6)=A vector, (7-9)=C vector.
<code>cell0(9)</code>	work array
<code>celn(9)</code>	work array

celprp(10)	cell property array
chge(msatms)	atomic charge array
chgsit(mxsite)	atomic site charge array
ckc(msatms)	cosine of Fourier transform of atomic charge density
ckc(mxatms)	cosine of Fourier transform of atomic charge density
ckc(mxewld)	cosine of Fourier transform of atomic charge density
cks(msatms)	sine of Fourier tranform of atomic charge density
cks(mxatms)	sine of Fourier tranform of atomic charge density
cks(mxewld)	sine of Fourier tranform of atomic charge density
clm(msatms)	partial FT (cosine) of atomic charge density
clm(mxatms)	partial FT (cosine) of atomic charge density
clm(mxewld)	partial FT (cosine) of atomic charge density
crn(0:mxhko,0:mxhko)	matrix of binomial coefficients for HK-Ewald
csp(mxspl)	work space for B-spline calculation
dens(mxsvdw)	number density of unique atom types
dens0(mxsvdw)	work array
dhn(mxegr,0:mxhko)	derivatives of HK-Ewald convergence functions
dsq(mspmf)	square of PMF constraint distance
dxp(mspmf)	x component of bond vector during PMF SHAKE iteration
dxt(mxcons)	x component of bond vector during SHAKE iteration
dxs(mxcons)	x component of bond vectors (at start of SHAKE)
dyp(mspmf)	y component of bond vector during PMF SHAKE iteration
dyt(mxcons)	y component of bond vector during SHAKE iteration
dys(mxcons)	y component of bond vectors (at start of SHAKE)
dzp(mspmf)	z component of bond vector during PMF SHAKE iteration
dzt(mxcons)	z component of bond vector during SHAKE iteration
dzz(mxcons)	z component of bond vectors (at start of SHAKE)
elc(msatms,0:1)	cosine of $\exp(ik.r(x))$ terms for Ewald sum
elc(mxatms,0:1)	cosine of $\exp(ik.r(x))$ terms for Ewald sum
elc(mxewld,0:1)	cosine of $\exp(ik.r(x))$ terms for Ewald sum
els(msatms,0:1)	sine of $\exp(ik.r(x))$ terms for Ewald sum
els(mxatms,0:1)	sine of $\exp(ik.r(x))$ terms for Ewald sum
els(mxewld,0:1)	sine of $\exp(ik.r(x))$ terms for Ewald sum
emc(msatms,0:kmaxb)	cosine of $\exp(ik.r(y))$ terms for Ewald sum
emc(mxatms,0:kmaxb)	cosine of $\exp(ik.r(y))$ terms for Ewald sum
emc(mxewld,0:kmaxb)	cosine of $\exp(ik.r(y))$ terms for Ewald sum
ems(msatms,0:kmaxb)	sine of $\exp(ik.r(y))$ terms for Ewald sum
ems(mxatms,0:kmaxb)	sine of $\exp(ik.r(y))$ terms for Ewald sum
ems(mxewld,0:kmaxb)	sine of $\exp(ik.r(y))$ terms for Ewald sum
enc(msatms,0:kmaxc)	cosine of $\exp(ik.r(z))$ terms for Ewald sum
enc(mxatms,0:kmaxc)	cosine of $\exp(ik.r(z))$ terms for Ewald sum
enc(mxewld,0:kmaxc)	cosine of $\exp(ik.r(z))$ terms for Ewald sum
ens(msatms,0:kmaxc)	sine of $\exp(ik.r(z))$ terms for Ewald sum
ens(mxatms,0:kmaxc)	sine of $\exp(ik.r(z))$ terms for Ewald sum

<code>ilist(mxxdf)</code>	abbreviated Verlet neighbour list for specific atom
<code>ind(mxgrp,3)</code>	work array
<code>index(mxtmls)</code>	work array
<code>indpmf(mxspmf)</code>	indices of sites in PMF constraints
<code>itest(mxtmls)</code>	work array
<code>jfox(mxatms)</code>	work array
<code>jlist(mxxdf)</code>	Verlet neighbour list in group based cutoff scheme
<code>key1(kmaxd)</code>	reverse bit address arrays for FFT routine DLPFFT3
<code>key2(kmaxe)</code>	reverse bit address arrays for FFT routine DLPFFT3
<code>key3(kmaxf)</code>	reverse bit address arrays for FFT routine DLPFFT3
<code>keyang(mxtang)</code>	type key for all valence angle potentials
<code>keybnd(mxtbnd)</code>	type key for all bond potentials
<code>keydih(mxtdih)</code>	type key for all dihedral angle potentials
<code>keyinv(mxtinv)</code>	inversion potential key
<code>keytet(mxteth)</code>	type key for all tethering potentials
<code>kscons(0:mxproc-1)</code>	work array
<code>lashap(mxproc)</code>	'last atom' marker for RD-SHAKE inter-node comms
<code>latinx(mxatms)</code>	work array
<code>lct(mxcell)</code>	head of chain for linked list
<code>lentry(msatms)</code>	'last atom' marker for Verlet neighbour list
<code>lentry1(msatms)</code>	number of neighbours in first Verlet neighbour list
<code>lentry2(msatms)</code>	number of neighbours in second Verlet neighbour list
<code>lentry3(msatms)</code>	number of neighbours in third Verlet neighbour list
<code>lexatm(msatms,mxexcl)</code>	'excluded atom' list for pair force terms
<code>lexsit(mxsite,mxexcl)</code>	'excluded site' list for pair force terms
<code>lfzsit(mxsite)</code>	'frozen site' key
<code>link(mxatms)</code>	link list array used in link-cells
<code>lishap(mxlshp)</code>	'shared atoms' list for RD-SHAKE comms
<code>list(msatms,mxlist)</code>	Verlet neighbour list
<code>list1(msatms,mxlist1)</code>	the third Verlet neighbour list
<code>list2(msatms,mxlist1)</code>	the second Verlet neighbour list
<code>list3(msatms,mxlist1)</code>	the third Verlet neighbour list
<code>listang(mxangl,4)</code>	Valence angle potential bookkeeping list
<code>listbnd(mxbond,3)</code>	Bond potential bookkeeping list
<code>listcon(mxcons,3)</code>	Bond constraint bookkeeping list
<code>listdih(mxdihd,5)</code>	Dihedral angle potential bookkeeping list
<code>listin(mxatms)</code>	Incoming 'shared atoms' transfer buffer
<code>listinv(mxinv,5)</code>	bookkeeping array for inversion potential parameters
<code>listme(mxatms)</code>	Outgoing 'shared atoms' transfer buffer
<code>listot(mxatms)</code>	Global 'shared atoms' register
<code>lll(*)</code>	integer array for global integer summations
<code>listpm(mxpmf)</code>	indices for sites in PMF constraint
<code>listshl(mxshl,3)</code>	indices for core-shell interactions on current processor
<code>listtet(msteth,2)</code>	indices of tethered atoms on current processor

<code>listyp(mxungp)</code>	rigid body group type
<code>lst(mxcell)</code>	number of atoms in link cell
<code>lstang(mxtang,3)</code>	Valence angle potential site list
<code>lstbnd(mxtbnd,2)</code>	Bond potential site list
<code>lstbod(mxatms)</code>	pointer from site index to rigid unit index
<code>lstcon(mxtcon,2)</code>	Bond constraint site list
<code>lstcsit(2*mxcons)</code>	indices of rigid unit sites in constraints
<code>lst dih(mxt dih,4)</code>	Dihedral angle potential site list
<code>lstfbp(mxfbp)</code>	pointer from four body potential to parameter table
<code>lstfre(mxatms)</code>	atoms not involved in rigid bodies
<code>lstfrz(mxatms)</code>	frozen atom atomic key
<code>lstgot(mxatms)</code>	indices of atoms in rigid body units
<code>lstgst(mxungp,mxngp)</code>	indices of sites in a rigid body
<code>lstgtp(mxgrp)</code>	rigid body group type
<code>lstinv(mxtinv,4)</code>	inversion angle potential site list
<code>lstme(mxatms)</code>	sites integrated on current processor
<code>lstneu(mxatms)</code>	charge group index
<code>lstout(mxatms)</code>	work array
<code>lstpmf(mxspmf,mspmf)</code>	indices of atoms in PMF constraints
<code>lstpmt(mxatms)</code>	key for atoms in PMF constraints
<code>lstpmt(mxpmf)</code>	key for atoms in PMF constraints
<code>lstrgd(mxgadm)</code>	list of atoms in rigid body on current processor
<code>lstshl(mxtshl,2)</code>	indices of sites in shell-core units
<code>lsttbp(mxtbp)</code>	index to three body potential
<code>lsttet(mxteth)</code>	index to tethering potential
<code>lstvdw(mxvdw)</code>	pointer from potential index to parameter table
<code>ltpfbp(mxfbp)</code>	key for four body potential type
<code>ltpsit(mxsite)</code>	atomic type index for molecular sites
<code>ltpfbp(mxtbp)</code>	three body potential key
<code>ltpvdw(mxvdw)</code>	potential energy type index
<code>ltype(mxatms)</code>	atomic type index for all atoms in simulation
<code>mmm(*)</code>	integer array for global integer summations
<code>mconst(mxtmls)</code>	work array
<code>molnam(mxtmls)</code>	names of molecules in system
<code>m site(mxtmls)</code>	work array
<code>nauxfft(4)</code>	buffer required by Cray FFT routine
<code>neulst(mxneut)</code>	list of first atomic site in charge group
<code>nix(27)</code>	link cell offset in x direction
<code>nix(508)</code>	link cell offset in x direction
<code>niy(27)</code>	link cell offset in y direction
<code>niy(508)</code>	link cell offset in y direction
<code>niz(27)</code>	link cell offset in z direction
<code>niz(508)</code>	link cell offset in z direction
<code>nexatm(msatms)</code>	'last entries' for excluded atom list

nexsit(mxsite)	'last entries' for excluded site list
noxatm(msatms)	'last atom' counter for excluded atom list in use
npmf(2)	number of sites in PMF constraint units
nugrp(mxsite)	charge group pointer for sites
numang(mxtmls)	number of valence angles in each molecule
numbonds(mxtmls)	number of chemical bonds in each molecule
numcon(mxtmls)	number of bond constraints in each molecule
numdih(mxtmls)	number of dihedral potentials in given molecule
numfrz(mxsvdw)	number of frozen atoms of given type in simulation
numgrp(mxtmls)	number of rigid bodies in a given molecule
numgsit(mxungp)	number of sites in a given rigid body type
numinv(mxtmls)	number of inversion angles in a molecule
nummols(mxtmls)	population of each molecule type in simulation
numpmf(mxtmls)	number of pmf constraints in a given molecule
numshl(mxtmls)	number of core-shell units in a given molecule
numsit(mxtmls)	number of distinct sites in each molecule
numteth(mxtmls)	number of tethers in a given molecule
numtyp(mxsvdw)	number of atoms of given type in simulation
omg(9)	stress tensor contributions from reciprocal space part of Ewald sum
omx(mxgrp)	angular velocity about x principal axis
omx1(msgrp)	work array
omy(mxgrp)	angular velocity about y principal axis
omy1(msgrp)	work array
omz(mxgrp)	angular velocity about z principal axis
omz1(msgrp)	work array
opx(msgrp)	new angular velocity about x principal axis
opy(msgrp)	new angular velocity about y principal axis
opz(msgrp)	new angular velocity about z principal axis
oqx(msgrp)	angular velocity about x principal axis at half timestep
oqy(msgrp)	angular velocity about y principal axis at half timestep
oqz(msgrp)	angular velocity about z principal axis at half timestep
parpot(mxptbp)	three body potential parameters input array
parvdw(mxpvdw)	short range potential parameters input array
pmfnrm(2)	pmf normalization for weighting factors
pmfwght(mxspmf)	pmf site weighting
pp(2*mxhko)	polynomial array for HK-Ewald method
prmang(mxtang,mxpang)	valence angle potential parameters array
prmbnd(mxtbnd,mxpbnd)	chemical bond potential parameters array
prmcon(mxtcon)	bond constraint parameters (bondlengths)
prmdih(mxtdih,mxpdih)	dihedral angle potential parameters array
prmfbp(mxfbp,mxpbang)	four body potential parameters
prmfld(mxfld)	parameters for external field
prminv(mxtinv,mxpinv)	inversion angle potential parameters
prmschl(mxtshl)	spring constant fro core-shell units

prmtbp(mxtpb, mxpang)	three body potential parameters
prmtet(mxteht, mxpbnd)	tethering potential parameters
prmvdw(mxvdw, mxpvdw)	short range potential parameters array
q0(mxgrp)	first quaternion array
q1(mxgrp)	second quaternion array
q2(mxgrp)	third quaternion array
q3(mxgrp)	four quaternion array
qn0(msgrp)	new first quaternion array
qn1(msgrp)	new second quaternion array
qn2(msgrp)	new third quaternion array
qn3(msgrp)	new fourth quaternion array
qqc(kamxd, kmaxe, kmaxf)	SPME charge array
qqq(kamxd, kmaxe, kmaxf)	FT of qqc array (complex)
ravval(mxnstk)	rolling averages array
rcell(9)	reciprocal of simulation cell matrix
rcut3b(mxtpb)	cutoff for three body potentials
rcut4b(mxtpb)	four body potential cutoffs
rdf(mxrdf, mxvdw)	rdf histogram array (rdf accumulator)
redmass(mxcons)	effective reduced mass in the QSHAKE algorithm
rho(mxatms)	Sutton-Chen particle density
rot(9)	rotational matrix
rotl(3,3)	work array
roti(9)	moment of inertia tensor
rotinr(3,3)	work array
rotinv(9)	inverse of moment of inertia tensor
rotinx(mxungp, 2)	rotational inertia, and its inverse, about x principal axis
rotiny(mxungp, 2)	rotational inertia, and its inverse, about y principal axis
rotinz(mxungp, 2)	rotational inertia, and its inverse, about z principal axis
rotmin(mxungp)	work array
rsq(mxlist)	square of interatomic distance
rsqdf(mxxdf)	square of interatomic distance
sitnam(mxsite)	names of atoms in molecular sites
slm(msatms)	partial FT (sine) of atomic charge density
slm(mxatms)	partial FT (sine) of atomic charge density
slm(mxewld)	partial FT (sine) of atomic charge density
ssqval(mxnstk)	accumulators of squared system variables
sss(mxxdf)	the in-plane (XY) separation between atoms (for HKE)
stkval(mxstak, mxnstk)	history stack of system variables
stpval(mxnstk)	array of current system variables
stres1(9)	work array for stress tensor
stres2(9)	work array for stress tensor
stresl(9)	long range contribution to stress tensor
stresp(9)	time derivative of long range contribution to stress tensor
stress(9)	stress tensor

<code>summas(2)</code>	work array
<code>sumval(mxnstk)</code>	accumulators for system variables
<code>tqx(msgrp)</code>	torque about x principal axis
<code>tqy(msgrp)</code>	torque about y principal axis
<code>tqz(msgrp)</code>	torque about z principal axis
<code>txx(mxatms)</code>	x component of coordinate transfer buffer
<code>tyy(mxatms)</code>	y component of coordinate transfer buffer
<code>tzz(mxatms)</code>	z component of coordinate transfer buffer
<code>unqatm(mxsite)</code>	list of unique atom names in system
<code>uxx(mxatms)</code>	x component of velocity work array
<code>uyy(mxatms)</code>	y component of velocity work array
<code>uzz(mxatms)</code>	z component of velocity work array
<code>v(n,n)</code>	work array
<code>vlrcm(2)</code>	long range correction to virial for metal potentials
<code>vvv(mxgrid,mxvdw)</code>	short-range potential (energy) interpolation table
<code>vx1(msatms)</code>	work array
<code>vxx(mxatms)</code>	x component of atomic velocities
<code>vy1(msatms)</code>	work array
<code>vyy(mxatms)</code>	y component of atomic velocities
<code>vz1(msatms)</code>	work array
<code>vzz(mxatms)</code>	z component of atomic velocities
<code>weight(mxatms)</code>	atomic mass array
<code>wgtsit(mxsite)</code>	site atomic mass array
<code>ww1(kmaxd)</code>	Fourier complex x exponentials
<code>ww2(kmaxe)</code>	Fourier complex y exponentials
<code>ww3(kmaxf)</code>	Fourier complex z exponentials
<code>xa(2,mspmf)</code>	work array
<code>xdab(msbad)</code>	x component of atomic separation (work array)
<code>xdac(msbad)</code>	x component of atomic separation (work array)
<code>xdad(msbad)</code>	x component of atomic separation (work array)
<code>xdbc(msbad)</code>	x component of atomic separation (work array)
<code>xdcd(msbad)</code>	x component of atomic separation (work array)
<code>xdf(mxxdf)</code>	x component of interatomic distance vector
<code>xold(mslst)</code>	array of x displacements in Verlet list tester
<code>xx0(mxatms)</code>	x component of atomic displacement (for MSD)
<code>xx1(mxatms)</code>	x component of 1st derivative for Gear algorithm
<code>xx2(mxatms)</code>	x component of 2nd derivative for Gear algorithm
<code>xx3(mxatms)</code>	x component of 3d derivative for Gear algorithm
<code>xx4(mxatms)</code>	x component of 4th derivative for Gear algorithm
<code>xx5(mxatms)</code>	x component of 5th derivative for Gear algorithm
<code>xxs(mxatms)</code>	x position at step 0
<code>xxt(mxatms)</code>	x component of SHAKE increment vector
<code>xxx(mxatms)</code>	x component of atomic position vectors
<code>ya(2,mspmf)</code>	work array

y _{dab} (msbad)	y component of atomic separation (work array)
y _{dac} (msbad)	z component of atomic separation (work array)
y _{dad} (msbad)	z component of atomic separation (work array)
y _{dbc} (msbad)	y component of atomic separation (work array)
y _{dcb} (msbad)	y component of atomic separation (work array)
y _{df} (mxxdf)	y component of interatomic distance vector
y _{old} (mslst)	array of y displacements in Verlet list tester
yy ₀ (mxatms)	y component of atomic displacement (for MSD)
yy ₁ (mxatms)	y component of 1st derivative for Gear algorithm
yy ₂ (mxatms)	y component of 2nd derivative for Gear algorithm
yy ₃ (mxatms)	y component of 3d derivative for Gear algorithm
yy ₄ (mxatms)	y component of 4th derivative for Gear algorithm
yy ₅ (mxatms)	y component of 5th derivative for Gear algorithm
yy _s (mxatms)	y position at step 0
yy _t (mxatms)	y component of SHAKE increment vector
yy _y (mxatms)	y component of atomic position vectors
z _a (2,mspmf)	work array
z _{dab} (msbad)	z component of atomic separation (work array)
z _{dac} (msbad)	z component of atomic separation (work array)
z _{dad} (msbad)	z component of atomic separation (work array)
z _{dbc} (msbad)	z component of atomic separation (work array)
z _{dcb} (msbad)	z component of interatomic distance vector
z _{dens} (mxrdf, mxsvdw)	z density histograms
z _{df} (mxxdf)	z component of interatomic distance vector
z _{gc} (0:2*mxhko)	global sums of products of cosine of wavevector and power of atomic z coordinates (for HKE)
z _{gs} (0:2*mxhko)	global sums of products of sine of wavevector and power of atomic z coordinates (for HKE)
z _{np} (mxhke, 0:2*mxhko)	powers of atomic z coordinates (for HKE)
z _{old} (mslst)	array of z displacements in Verlet list tester
z _{umval} (mxnstk)	rolling average accumulators of system variables
zz ₀ (mxatms)	z component of atomic displacement (for MSD)
zz ₁ (mxatms)	z component of 1st derivative for Gear algorithm
zz ₂ (mxatms)	z component of 2nd derivative for Gear algorithm
zz ₃ (mxatms)	z component of 3d derivative for Gear algorithm
zz ₄ (mxatms)	z component of 4th derivative for Gear algorithm
zz ₅ (mxatms)	z component of 5th derivative for Gear algorithm
zz _d (mxxdf)	atomic z coordinate to power 2n-1 (for HKE)
zz _n (mxxdf)	atomic z coordinate to power 2n (for HKE)
zz _s (mxatms)	z position at step 0
zz _t (mxatms)	z component of SHAKE increment vector
zz _z (mxatms)	z component of atomic position vectors

Appendix E

DL_POLY Subroutine Dependencies

Calling : Called		Called : Calling
*****		*****
angfrfc: error		angfrfc : dlpoly
angfrfc: gdsum		*****
angfrfc: gstate		bndfrfc : dlpoly
angfrfc: images		*****
*****		bspcoe : ewald_spme
bndfrfc: error		*****
bndfrfc: gdsum		bspgen : ewald_spme
bndfrfc: gstate		*****
bndfrfc: images		ccfft3d : ewald_spme
*****		ccfft3d : simdef
bspcoe :		*****
*****		cfgscan : parset
bspgen :		*****
*****		conscan : parset
cfgscan: dcell		*****
cfgscan: gdsum		corshl : dlpoly
*****		*****
conscan: dcell		coul0 : forces
conscan: error		coul0 : multiple
conscan: getrec		coul0 : multiple_nsq
conscan: lowercase		coul0 : multiple_nsq
conscan: strip		*****
conscan: dblstr		coul0neu : forcesneu
conscan: intstr		coul0neu : multipleneu
*****		*****
corshl: gdsum		coul2 : forces

```

***** | coul2 : multiple
coul0 : | coul2 : multiple_nsq
***** | coul2 : multiple_nsq
coul0neu : | *****
***** | coul2neu : forcesneu
coul2 : | coul2neu : multipleneu
***** | *****
coul2neu: | coul3 : forces
***** | coul3 : multiple
coul3 : | *****
***** | coul3neu : forcesneu
coul3neu : | coul3neu : multipleneu
***** | *****
coul4 : | coul4 : forces
***** | coul4 : multiple
cpy_rtc : | *****
***** | cpy_rtc : ewald_spme
***** | *****
crecv: mpi_recv | *****
***** | csend : gdsum
csend: mpi_send | csend : gimax
***** | csend : gisum
dblstr: intstr | csend : merge
***** | csend : merge.hcube
dcell : | csend : merge.systol
***** | csend : merge1
denloc : | csend : merge1.hcube
***** | csend : merge1.systol
diffsn0 : | csend : merge4
***** | csend : merge4.hcube
diffsn1: gdsum | csend : passcon
***** | csend : shlmerge
dihfrc: error | csend : shmove
dihfrc: gdsum | *****
dihfrc: gstate | dblstr : conscan
dihfrc: images | dblstr : fldscan
***** | dblstr : fortan
dihfrc_4pt: error | dblstr : simdef
dihfrc_4pt: gdsum | dblstr : sysdef
dihfrc_4pt: gstate | dblstr : sysgen
dihfrc_4pt: images | *****
***** | dcell : cfgscan
dihfrc_rsq: error | dcell : conscan
dihfrc_rsq: gdsum | dcell : dlpoly
dihfrc_rsq: gstate | dcell : ewald1

```

```

dihfrc_rsqu: images | dcell : ewald1a
***** | dcell : ewald_spme
dlpfft3 : | dcell : fbpfrc
***** | dcell : hkewald1
dlpoly: angfrc | dcell : parlink
dlpoly: bndfrc | dcell : parlinkneu
dlpoly: corshl | dcell : parset
dlpoly: dcell | dcell : simdef
dlpoly: dihfrc | dcell : static
dlpoly: error | dcell : sysgen
dlpoly: exitcomms | dcell : thbfrc
dlpoly: extnfld | *****
dlpoly: fbpfrc | dcft3 : ewald_spme
dlpoly: fcap | *****
dlpoly: forces | dcopy : merge
dlpoly: forcesneu | dcopy : merge.hcube
dlpoly: freeze | dcopy : merge.systol
dlpoly: gdsum | dcopy : merge4.hcube
dlpoly: gstate | dcopy : shmove
dlpoly: gstate | *****
dlpoly: gsync | denloc : scdens
dlpoly: initcomms | *****
dlpoly: invfrc | diffsn0 : static
dlpoly: lrcmetal | *****
dlpoly: machine | diffsn1 : static
dlpoly: multiple | *****
dlpoly: multiple_nsqu | dihfrc : dlpoly
dlpoly: multipleneu | *****
dlpoly: npt_b1 | dlpfft3 : ewald_spme
dlpoly: npt_b3 | *****
dlpoly: npt_h1 | duni : gauss
dlpoly: npt_h3 | *****
dlpoly: nptq_b1 | ele_prd : ewald_spme
dlpoly: nptq_b2 | *****
dlpoly: nptq_b3 | erfcgen : sysdef
dlpoly: nptq_b4 | *****
dlpoly: nptq_h1 | error : angfrc
dlpoly: nptq_h2 | error : bndfrc
dlpoly: nptq_h3 | error : conscan
dlpoly: nptq_h4 | error : dihfrc
dlpoly: nve_1 | error : dihfrc_4pt
dlpoly: nveq_1 | error : dihfrc_rsqu
dlpoly: nveq_2 | error : dlpoly
dlpoly: nvt_b1 | error : ewald1

```

dlpoly: nvt_e1		error : ewald1a
dlpoly: nvt_h1		error : ewald_spme
dlpoly: nvtq_b1		error : exclude
dlpoly: nvtq_b2		error : excludeneu
dlpoly: nvtq_h1		error : extnfld
dlpoly: nvtq_h2		error : fbpfrfc
dlpoly: parlink		error : fldscan
dlpoly: parlinkneu		error : forcesneu
dlpoly: parlst		error : forcesneu
dlpoly: parlst_nsq		error : forgen
dlpoly: parneulst		error : forgen_rsq
dlpoly: parset		error : fortatb
dlpoly: pmf_1		error : fortatb_rsq
dlpoly: quatqnch		error : hkewald1
dlpoly: result		error : hkewald2
dlpoly: revive		error : hkgen
dlpoly: shlfrc		error : images
dlpoly: shlqnch		error : intlist
dlpoly: shpalloc		error : invfrc
dlpoly: simdef		error : merge
dlpoly: static		error : merge.hcube
dlpoly: strucopt		error : merge.systol
dlpoly: sysbook		error : merge1
dlpoly: sysdef		error : merge1.hcube
dlpoly: sysgen		error : merge1.systol
dlpoly: sysinit		error : merge4
dlpoly: systemp		error : merge4.hcube
dlpoly: tethfrc		error : metgen
dlpoly: thbfrc		error : metgen_rsq
dlpoly: timchk		error : multiple_nsq
dlpoly: trajet		error : multipleneu
dlpoly: vertest		error : nodedim
dlpoly: vscaleg		error : npt_b3
dlpoly: xscale		error : npt_h3
dlpoly: zden0		error : nptq_b1
*****		error : nptq_b2
duni:		error : nptq_b3
*****		error : nptq_b4
ele_prd		error : nptq_h1
*****		error : nptq_h2
erfcgen :		error : nptq_h3
*****		error : nptq_h4
erfcgen_rsq :		error : nst_b0
*****		error : nst_h0

error: exitcomms		error : nveq_1
error: gsync		error : nveq_2
*****		error : nvtq_b1
ewald1: dcell		error : nvtq_b2
ewald1: error		error : nvtq_h1
ewald1: gdsum		error : nvtq_h2
ewald1: invert		error : parlink
*****		error : parlinkneu
ewald1a: dcell		error : parlinkneu
ewald1a: error		error : parlst
ewald1a: invert		error : parlst_nsq
*****		error : parneulst
ewald2 :		error : passcon
*****		error : passcon
ewald2_4pt :		error : passpmf
*****		error : passquat
ewald2neu :		error : pmf_shake
*****		error : prneulst
ewald2_rsqu :		error : quatbook
*****		error : quench
ewald3 :		error : rdshake_1
*****		error : revive
ewald3_neu :		error : shlfrfc
*****		error : shlmerge
ewald4 :		error : simdef
*****		error : splice
ewald4_4pt :		error : static
*****		error : sysbook
ewald4_rsqu :		error : sysdef
*****		error : sysgen
ewald_spme: bspcoe		error : sysinit
ewald_spme: bspgen		error : systemp
ewald_spme: ccfft3d		error : tethfrfc
ewald_spme: cpy_rtc		error : thbfrfc
ewald_spme: dcell		error : vertest
ewald_spme: dcft3		error : vertest1
ewald_spme: dlpfft3		*****
ewald_spme: ele_prd		ewald1 : forces
ewald_spme: error		ewald1 : multiple
ewald_spme: fftwnd_f77_one		*****
ewald_spme: invert		ewald2 : forces
ewald_spme: scl_csum		ewald2 : multiple
ewald_spme: set_block		*****
ewald_spme: spl_cexp		ewald3 : forces

```

ewald_spme: spme_for      | ewald3 : multiple
ewald_spme: zzfft3d      | *****
*****                  | ewald4 : multiple
*****                  | *****
exclude: error           | ewald_spme : forces
exclude: gimax           | ewald_spme : multiple
exclude: gstate          | *****
exclude: warning         | exclude : sysbook
*****                  | *****
exclude_atom :          | exclude_atom : sysbook
*****                  | *****
exclude_link:           | exclude_link : sysbook
*****                  | *****
excludeneu: error       | excludeneu : sysbook
excludeneu: gimax        | *****
excludeneu: gstate      | *****
*****                  | exit : exitcomms
*****                  | *****
exitcomms: mpi_finalize | exitcomms : dlpoly
exitcomms: exit          | exitcomms : error
*****                  | *****
extnflld: error         | extnflld : dlpoly
*****                  | *****
fbpfrfc: dcell          | fbpfrfc : dlpoly
fbpfrfc: error           | *****
fbpfrfc: gdsum           | fcap : dlpoly
fbpfrfc: gstate         | *****
fbpfrfc: invert         | fftw3d_f77_create_plan : simde
*****                  | fftwnd_f77_one : ewald_spme
*****                  | *****
fcap :                  | fldscan : parset
*****                  | *****
fldscan: error          | forces : dlpoly
fldscan: getrec         | *****
fldscan: lowercase      | forcesneu : dlpoly
fldscan: strip          | *****
fldscan: dblstr         | *****
fldscan: intstr         | forgen : sysdef
*****                  | *****
forces: coul0           | *****
forces: coul2           | *****
forces: coul3           | *****
forces: coul4           | *****
forces: ewald1          | *****
forces: ewald2          | *****
forces: ewald3          | *****
forces: ewald_spme      | *****
                      | *****
                      | gsum : vscale

```

forces: gdsum		gdsum : angfrc
forces: hkewald1		gdsum : bndfrc
forces: hkewald2		gdsum : cfgscan
forces: hkewald3		gdsum : corshl
forces: images		gdsum : diffsn1
forces: rdf0		gdsum : dihfrc
forces: scdens		gdsum : dihfrc_4pt
forces: srfrce		gdsum : dihfrc_rsq
forces: suttchen		gdsum : dlpoly
*****		gdsum : ewald1
forcesneu: error		gdsum : fbpfrc
forcesneu: coul0neu		gdsum : forces
forcesneu: coul2neu		gdsum : forcesneu
forcesneu: coul3neu		gdsum : fortab
forcesneu: error		gdsum : fortab_rsq
forcesneu: gdsum		gdsum : hkewald1
forcesneu: gimax		gdsum : invfrc
forcesneu: gstate		gdsum : multiple
forcesneu: images		gdsum : multiple_nsq
forcesneu: neutlst		gdsum : multipleneu
forcesneu: rdf0neu		gdsum : npt_b0
forcesneu: srfrceneu		gdsum : npt_b1
*****		gdsum : npt_b3
forgen: error		gdsum : npt_h0
*****		gdsum : npt_h1
forgen_rsq: error		gdsum : npt_h3
*****		gdsum : nptq_b1
fortab: dblstr		gdsum : nptq_b2
fortab: error		gdsum : nptq_b3
fortab: gdsum		gdsum : nptq_b4
fortab: getrec		gdsum : nptq_b4
fortab: intstr		gdsum : nptq_h1
fortab: strip		gdsum : nptq_h2
*****		gdsum : nptq_h3
fortab_rsq: error		gdsum : nptq_h4
fortab_rsq: gdsum		gdsum : nst_b0
fortab_rsq: getrec		gdsum : nst_h0
fortab_rsq: intstr		gdsum : nve_0
fortab_rsq: strip		gdsum : nve_1
*****		gdsum : nveq_1
freeze :		gdsum : nveq_2
*****		gdsum : nvt_b0
gauss: duni		gdsum : nvt_b1
*****		gdsum : nvt_e0

gdsum: csend		gdsum : nvt_e1
gdsum: mpi_allreduce		gdsum : nvt_h0
gdsum: msgwait		gdsum : nvt_h1
gdsum: mynode		gdsum : nvtq_b1
gdsum: shmem_barrier_all		gdsum : nvtq_b2
gdsum: shmem_put		gdsum : nvtq_h1
gdsum: shmem_put64		gdsum : nvtq_h2
gdsum: shmem_udcflush		gdsum : pmf_1
*****		gdsum : pmf_shake
getrec: gisum		gdsum : qshake
getrec: gstate		gdsum : quatbook
getrec: gsync		gdsum : rdf1
*****		gdsum : rdshake_1
gimax: csend		gdsum : revive
gimax: gsync		gdsum : scdens
gimax: mpi_allreduce		gdsum : shlfrfc
gimax: msgwait		gdsum : splice
gimax: mynode		gdsum : spme_for
*****		gdsum : sysgen
gisum: csend		gdsum : sysinit
gisum: gsync		gdsum : tethfrfc
gisum: mpi_allreduce		gdsum : thbfrfc
gisum: msgwait		gdsum : vscaleg
gisum: mynode		gdsum : zden1
*****		*****
gstate: gisum		getrec : conscan
*****		getrec : fldscan
gsync: mpi_barrier		getrec : fortatb
gsync: shmem_barrier_all		getrec : fortatb_rsqu
*****		getrec : simdef
hkewald1: dcell		getrec : sysdef
hkewald1: error		getrec : sysgen
hkewald1: gdsum		*****
hkewald1: invert		gimax : exclude
hkewald1: warning		gimax : excludeneu
*****		gimax : forcesneu
hkewald2: error		gimax : intlist
hkewald2: invert		gimax : merge1.hcube
*****		gimax : multipleneu
hkewald3 :		gimax : nptq_b1
*****		gimax : nptq_b2
hkewald4 :		gimax : nptq_b3
*****		gimax : nptq_b4
hkgen: error		gimax : nptq_h1

hkgen: warning		gimax : nptq_h2
*****		gimax : nptq_h3
images: error		gimax : nptq_h4
images: invert		gimax : nveq_1
*****		gimax : nveq_2
initcomms: mpi_init		gimax : nvtq_b1
*****		gimax : nvtq_b2
intlist: error		gimax : nvtq_h1
intlist: gimax		gimax : nvtq_h2
intlist: gstate		gimax : parlink
intlist: passcon		gimax : parlinkneu
intlist: passpmf		gimax : parlst
intlist: shellsort		gimax : parneulst
*****		gimax : prneulst
intstr:		*****
*****		gisum : getrec
invert :		gisum : gstate
*****		gisum : parlst_nsq
invfrfc: error		gisum : passcon
invfrfc: gdsum		gisum : passpmf
invfrfc: gstate		gisum : passquat
invfrfc: images		gisum : sysinit
*****		gisum : vertest
jacobi :		gisum : vertest1
*****		*****
kinstr :		gstate : angfrfc
*****		gstate : bndfrfc
lowcase :		gstate : dihfrfc
*****		gstate : dihfrfc_4pt
lrcmetal :		gstate : dihfrfc_rsq
*****		gstate : dlpoly
lrcorrect :		gstate : dlpoly
*****		gstate : exclude
machine: mynode		gstate : excludeneu
machine: numnodes		gstate : fbpfrc
*****		gstate : forcesneu
merge: csend		gstate : getrec
merge: dcopy		gstate : intlist
merge: error		gstate : invfrfc
merge: gsync		gstate : merge1
merge: mpi_irecv		gstate : merge1.hcube
merge: mpi_send		gstate : merge1.systol
merge: mpi_wait		gstate : multipleneu
merge: msgwait		gstate : nptq_b1

merge: scopy		gstate : nptq_b2
merge: shmem_barrier_all		gstate : nptq_b3
merge: shmem_put		gstate : nptq_b4
merge: shmem_put64		gstate : nptq_h1
merge: shmem_udcflush		gstate : nptq_h2
*****		gstate : nptq_h3
merge.hcube: csend		gstate : nptq_h4
merge.hcube: dcopy		gstate : nveq_1
merge.hcube: error		gstate : nveq_2
merge.hcube: gsync		gstate : nvtq_b1
merge.hcube: mpi_irecv		gstate : nvtq_b2
merge.hcube: mpi_send		gstate : nvtq_h1
merge.hcube: mpi_wait		gstate : nvtq_h2
merge.hcube: msgwait		gstate : parlink
merge.hcube: scopy		gstate : parlinkneu
merge.hcube: shmem_barrier_all		gstate : parlst
merge.hcube: shmem_put		gstate : parlst_nsq
merge.hcube: shmem_put64		gstate : parneulst
merge.hcube: shmem_udcflush		gstate : passcon
*****		gstate : passquat
merge.systol: csend		gstate : prneulst
merge.systol: dcopy		gstate : qshake
merge.systol: error		gstate : quatbook
merge.systol: gsync		gstate : quench
merge.systol: mpi_irecv		gstate : rdshake_1
merge.systol: mpi_send		gstate : sysgen
merge.systol: mpi_wait		gstate : tethfrc
merge.systol: msgwait		gstate : thbfrc
merge.systol: scopy		*****
merge.systol: shmem_barrier_all		gsync : dlpoly
merge.systol: shmem_put		gsync : error
merge.systol: shmem_put64		gsync : getrec
merge.systol: shmem_udcflush		gsync : gimax
*****		gsync : gisum
merge1: csend		gsync : merge
merge1: error		gsync : merge.hcube
merge1: gstate		gsync : merge.systol
merge1: gsync		gsync : merge1
merge1: mpi_irecv		gsync : merge1.systol
merge1: mpi_send		gsync : merge4
merge1: mpi_wait		gsync : merge4.hcube
merge1: msgwait		gsync : passcon
merge1: shmem_barrier_all		gsync : shlmerge
merge1: shmem_put		gsync : shmove

```

merge1: shmem_put64 | *****
merge1: shmem_udcflush | hkewald1 : forces
***** | hkewald1 : multiple
merge1.hcube: csend | *****
merge1.hcube: error | hkewald2 : forces
merge1.hcube: gimax | hkewald2 : multiple
merge1.hcube: gstate | *****
merge1.hcube: mpi_irecv | hkewald3 : forces
merge1.hcube: mpi_send | hkewald3 : multiple
merge1.hcube: mpi_wait | *****
merge1.hcube: msgwait | hkewald4 : multiple
merge1.hcube: shmem_barrier_all | *****
merge1.hcube: shmem_put | hkgen : sysdef
merge1.hcube: shmem_put64 | *****
merge1.hcube: shmem_udcflush | images : angfrc
***** | images : bndfrc
merge1.systol: csend | images : dihfrc
merge1.systol: error | images : dihfrc_4pt
merge1.systol: gstate | images : dihfrc_rsq
merge1.systol: gsync | images : forces
merge1.systol: mpi_irecv | images : forcesneu
merge1.systol: mpi_send | images : invfrc
merge1.systol: mpi_wait | images : multiple
merge1.systol: msgwait | images : multiple_nsq
merge1.systol: shmem_put | images : multipleneu
merge1.systol: shmem_udcflush | images : neutlst
***** | images : npt_b0
merge4: csend | images : npt_b1
merge4: error | images : npt_b3
merge4: gsync | images : npt_h0
merge4: mpi_irecv | images : npt_h1
merge4: mpi_send | images : npt_h3
merge4: mpi_wait | images : nptq_b1
merge4: msgwait | images : nptq_b2
merge4: shmem_barrier_all | images : nptq_b3
merge4: shmem_put | images : nptq_b4
merge4: shmem_put64 | images : nptq_h1
merge4: shmem_udcflush | images : nptq_h2
***** | images : nptq_h3
merge4.hcube: csend | images : nptq_h4
merge4.hcube: dcopy | images : nst_b0
merge4.hcube: error | images : nst_h0
merge4.hcube: gsync | images : nve_0
merge4.hcube: mpi_irecv | images : nve_1

```

```

merge4.hcube: mpi_send      | images : nveq_1
merge4.hcube: mpi_wait      | images : nveq_2
merge4.hcube: msgwait       | images : nvt_b0
merge4.hcube: scopy         | images : nvt_b1
merge4.hcube: shmem_barrier_all | images : nvt_e0
merge4.hcube: shmem_put     | images : nvt_e1
merge4.hcube: shmem_put64   | images : nvt_h0
merge4.hcube: shmem_udcflush | images : nvt_h1
*****                     | images : nvtq_b1
metgen: error               | images : nvtq_b2
*****                     | images : nvtq_h1
metgen_rsqr: error          | images : nvtq_h2
*****                     | images : parlink
multiple: coul0             | images : parlinkneu
multiple: coul2             | images : parlst
multiple: coul3             | images : parlst_nsqr
multiple: coul4             | images : parneulst
multiple: ewald1            | images : pmf_1
multiple: ewald2            | images : pmf_shake
multiple: ewald3            | images : primlst
multiple: ewald4            | images : prneulst
multiple: ewald_spme        | images : qshake
multiple: gdsum             | images : quatbook
multiple: hkewald1          | images : quatqnch
multiple: hkewald2          | images : quench
multiple: hkewald3          | images : rdshake_1
multiple: hkewald4          | images : scdens
multiple: images            | images : shlfrc
multiple: primlst           | images : strucopt
multiple: rdf0              | images : tethfrc
multiple: srfrce            | images : vertest1
*****                     | images : xscale
multiple_nsqr: coul0        | *****
multiple_nsqr: coul0        | initcomms : dlpolr
multiple_nsqr: coul2        | *****
multiple_nsqr: coul2        | intlist : sysbook
multiple_nsqr: error        | *****
multiple_nsqr: gdsum        | intstr : conscan
multiple_nsqr: images       | intstr : dblstr
multiple_nsqr: primlst      | intstr : fldscan
multiple_nsqr: rdf0         | intstr : fortab
multiple_nsqr: srfrce       | intstr : fortab_rsqr
*****                     | intstr : simdef
multipleneu: coul0neu       | intstr : sysdef

```

```

multipleneu: coul2neu      | intstr : sysgen
multipleneu: coul3neu      | *****
multipleneu: error         | invert  : vscale
multipleneu: gdsum         | invert  : vscaleg
multipleneu: gimax         | invert  : ewald1
multipleneu: gstate        | invert  : ewald1a
multipleneu: images        | invert  : ewald_spme
multipleneu: neutlst       | invert  : fbpfrc
multipleneu: prneulst      | invert  : hkewald1
multipleneu: rdf0neu       | invert  : hkewald2
multipleneu: srfrceneu     | invert  : images
*****                     | invert  : parlink
mynode: mpi_comm_rank      | invert  : parlinkneu
*****                     | invert  : quatbook
neutlst: images            | invert  : thbfrc
*****                     | *****
nodedim: mpi_comm_size     | invfrc  : dlpoly
*****                     | *****
nodedim: error             | jacobi  : quatbook
*****                     | *****
npt_b0: gdsum              | lowercase : conscan
npt_b0: images             | lowercase : fldscan
npt_b0: merge              | lowercase : simdef
*****                     | lowercase : sysdef
npt_b1: gdsum              | *****
npt_b1: images             | lrcmetal : dlpoly
npt_b1: merge              | lrcmetal : sysinit
npt_b1: rdshake_1          | *****
*****                     | lrcorrect : sysinit
npt_b3: error              | *****
npt_b3: gdsum              | machine  : dlpoly
npt_b3: images             | *****
npt_b3: merge              | merge    : npt_b0
npt_b3: rdshake_1          | merge    : npt_b1
*****                     | merge    : npt_b3
npt_h0: gdsum              | merge    : npt_h0
npt_h0: images             | merge    : npt_h1
npt_h0: merge              | merge    : npt_h3
*****                     | merge    : nptq_b1
npt_h1: gdsum              | merge    : nptq_b2
npt_h1: images             | merge    : nptq_b3
npt_h1: merge              | merge    : nptq_b4
npt_h1: rdshake_1          | merge    : nptq_h1
*****                     | merge    : nptq_h2

```

```

npt_h3: error | merge : nptq_h3
npt_h3: gdsum | merge : nptq_h4
npt_h3: images | merge : nst_b0
npt_h3: merge | merge : nst_h0
npt_h3: rdshake_1 | merge : nve_0
***** | merge : nve_1
nptq_b1: error | merge : nveq_1
nptq_b1: gdsum | merge : nveq_2
nptq_b1: gimax | merge : nvt_b0
nptq_b1: gstate | merge : nvt_b1
nptq_b1: images | merge : nvt_e0
nptq_b1: merge | merge : nvt_e1
nptq_b1: merge1 | merge : nvt_h0
nptq_b1: rdshake_1 | merge : nvt_h1
***** | merge : nvtq_b1
nptq_b2: error | merge : nvtq_b2
nptq_b2: gdsum | merge : nvtq_h1
nptq_b2: gimax | merge : nvtq_h2
nptq_b2: gstate | merge : parlink
nptq_b2: images | merge : parlinkneu
nptq_b2: merge | merge : pmf_1
nptq_b2: merge1 | merge : quatbook
nptq_b2: merge4 | merge : quatqnch
nptq_b2: qshake | merge : revive
***** | merge : vscale
nptq_b3: error | merge : xscale
nptq_b3: gdsum | *****
nptq_b3: gimax | merge1 : nptq_b1
nptq_b3: gstate | merge1 : nptq_b2
nptq_b3: images | merge1 : nptq_b3
nptq_b3: merge | merge1 : nptq_b4
nptq_b3: merge1 | merge1 : nptq_h1
nptq_b3: rdshake_1 | merge1 : nptq_h2
***** | merge1 : nptq_h3
nptq_b4: gdsum | merge1 : nptq_h4
nptq_b4: error | merge1 : nveq_1
nptq_b4: gdsum | merge1 : nveq_2
nptq_b4: gimax | merge1 : nvtq_b1
nptq_b4: gstate | merge1 : nvtq_b2
nptq_b4: images | merge1 : nvtq_h1
nptq_b4: merge | merge1 : nvtq_h2
nptq_b4: merge1 | merge1 : quatqnch
nptq_b4: merge4 | *****
nptq_b4: qshake | merge4 : nptq_b2

```

```

***** | merge4 : nptq_b4
nptq_h1: error | merge4 : nptq_h2
nptq_h1: gdsum | merge4 : nptq_h4
nptq_h1: gimax | merge4 : nveq_2
nptq_h1: gstate | merge4 : nvtq_b2
nptq_h1: images | merge4 : nvtq_h2
nptq_h1: merge | merge4 : quatbook
nptq_h1: merge1 | *****
nptq_h1: rdshake_1 | metgen : sysdef
***** | *****
nptq_h2: error | mpi_allreduce : gdsum
nptq_h2: gdsum | mpi_allreduce : gimax
nptq_h2: gimax | mpi_allreduce : gisum
nptq_h2: gstate | *****
nptq_h2: images | mpi_barrier : gsync
nptq_h2: merge | *****
nptq_h2: merge1 | mpi_comm_rank : mynode
nptq_h2: merge4 | *****
nptq_h2: qshake | mpi_comm_size : nodedim
***** | mpi_comm_size : numnodes
nptq_h3: error | *****
nptq_h3: gdsum | mpi_finalize : exitcomms
nptq_h3: gimax | *****
nptq_h3: gstate | mpi_init : initcomms
nptq_h3: images | *****
nptq_h3: merge | mpi_irecv : merge
nptq_h3: merge1 | mpi_irecv : merge.hcube
nptq_h3: rdshake_1 | mpi_irecv : merge.systol
***** | mpi_irecv : merge1
nptq_h4: error | mpi_irecv : merge1.hcube
nptq_h4: gdsum | mpi_irecv : merge1.systol
nptq_h4: gimax | mpi_irecv : merge4
nptq_h4: gstate | mpi_irecv : merge4.hcube
nptq_h4: images | mpi_irecv : passcon
nptq_h4: merge | mpi_irecv : shlmerge
nptq_h4: merge1 | mpi_irecv : shmove
nptq_h4: merge4 | *****
nptq_h4: qshake | mpi_recv : crecv
***** | *****
nst_b0: error | mpi_send : csend
nst_b0: gdsum | mpi_send : merge
nst_b0: images | mpi_send : merge.hcube
nst_b0: merge | mpi_send : merge.systol
***** | mpi_send : merge1

```

```

nst_h0: error | mpi_send : merge1.hcube
nst_h0: gdsum | mpi_send : merge1.systol
nst_h0: images | mpi_send : merge4
nst_h0: merge | mpi_send : merge4.hcube
***** | mpi_send : passcon
numnodes: mpi_comm_size | mpi_send : shlmerge
***** | mpi_send : shmove
nve_0: gdsum | *****
nve_0: images | mpi_wait : merge
nve_0: merge | mpi_wait : merge.hcube
***** | mpi_wait : merge.systol
nve_1: gdsum | mpi_wait : merge1
nve_1: images | mpi_wait : merge1.hcube
nve_1: merge | mpi_wait : merge1.systol
nve_1: rdshake_1 | mpi_wait : merge4
***** | mpi_wait : merge4.hcube
nveq_1: error | mpi_wait : passcon
nveq_1: gdsum | mpi_wait : shlmerge
nveq_1: gimax | mpi_wait : shmove
nveq_1: gstate | *****
nveq_1: images | msgwait : gdsum
nveq_1: merge | msgwait : gimax
nveq_1: merge1 | msgwait : gisum
nveq_1: rdshake_1 | msgwait : merge
***** | msgwait : merge.hcube
nveq_2: error | msgwait : merge.systol
nveq_2: gdsum | msgwait : merge1
nveq_2: gimax | msgwait : merge1.hcube
nveq_2: gstate | msgwait : merge1.systol
nveq_2: images | msgwait : merge4
nveq_2: merge | msgwait : merge4.hcube
nveq_2: merge1 | msgwait : passcon
nveq_2: merge4 | msgwait : shlmerge
nveq_2: qshake | msgwait : shmove
***** | *****
nvt_b0: gdsum | multiple : dlpoly
nvt_b0: images | *****
nvt_b0: merge | multiple_nsq : dlpoly
***** | *****
nvt_b1: gdsum | multipleneu : dlpoly
nvt_b1: images | *****
nvt_b1: merge | mynode : gdsum
nvt_b1: rdshake_1 | mynode : gimax
***** | mynode : gisum

```



```

nvt_e0: gdsum | mynode : machine
nvt_e0: images | mynode : timchk
nvt_e0: merge | *****
***** | neutlst : forcesneu
nvt_e1: gdsum | neutlst : multipleneu
nvt_e1: images | *****
nvt_e1: merge | npt_b1 : dlpoly
nvt_e1: rdshake_1 | *****
***** | npt_b3 : dlpoly
nvt_h0: gdsum | *****
nvt_h0: images | npt_h1 : dlpoly
nvt_h0: merge | *****
***** | npt_h3 : dlpoly
nvt_h1: gdsum | *****
nvt_h1: images | nptq_b1 : dlpoly
nvt_h1: merge | *****
nvt_h1: rdshake_1 | nptq_b2 : dlpoly
***** | *****
nvtq_b1: error | nptq_b3 : dlpoly
nvtq_b1: gdsum | *****
nvtq_b1: gimax | nptq_b4 : dlpoly
nvtq_b1: gstate | *****
nvtq_b1: images | nptq_h1 : dlpoly
nvtq_b1: merge | *****
nvtq_b1: merge1 | nptq_h2 : dlpoly
nvtq_b1: rdshake_1 | *****
***** | nptq_h3 : dlpoly
nvtq_b2: error | *****
nvtq_b2: gdsum | nptq_h4 : dlpoly
nvtq_b2: gimax | *****
nvtq_b2: gstate | numnodes : machine
nvtq_b2: images | *****
nvtq_b2: merge | nve_1 : dlpoly
nvtq_b2: merge1 | *****
nvtq_b2: merge4 | nveq_1 : dlpoly
nvtq_b2: qshake | *****
***** | nveq_2 : dlpoly
nvtq_h1: error | *****
nvtq_h1: gdsum | nvt_b1 : dlpoly
nvtq_h1: gimax | *****
nvtq_h1: gstate | nvt_e1 : dlpoly
nvtq_h1: images | *****
nvtq_h1: merge | nvt_h1 : dlpoly
nvtq_h1: merge1 | *****

```

```

nvtq_h1: rdshake_1 | nvtq_b1 : dlpoly
***** | *****
nvtq_h2: error | nvtq_b2 : dlpoly
nvtq_h2: gdsum | *****
nvtq_h2: gimax | nvtq_h1 : dlpoly
nvtq_h2: gstate | *****
nvtq_h2: images | nvtq_h2 : dlpoly
nvtq_h2: merge | *****
nvtq_h2: merge1 | parlink : dlpoly
nvtq_h2: merge4 | *****
nvtq_h2: qshake | parlinkneu : dlpoly
***** | *****
parlink: dcell | parlst : dlpoly
parlink: error | *****
parlink: gimax | parlst_nsq : dlpoly
parlink: gstate | *****
parlink: images | parneulst : dlpoly
parlink: invert | *****
parlink: merge | parset : dlpoly
***** | *****
parlinkneu: error | passcon : intlist
parlinkneu: dcell | *****
parlinkneu: error | passpmf : intlist
parlinkneu: gimax | *****
parlinkneu: gstate | passquat : sysbook
parlinkneu: images | *****
parlinkneu: invert | pmf_1 : dlpoly
parlinkneu: merge | *****
parlinkneu: shellsort | pmf_shake : pmf_1
***** | *****
parlst: error | prmlst : multiple
parlst: gimax | prmlst : multiple_nsq
parlst: gstate | *****
parlst: images | prneulst : multipleneu
***** | *****
parlst_nsq: error | qshake : nptq_b2
parlst_nsq: gisum | qshake : nptq_b4
parlst_nsq: gstate | qshake : nptq_h2
parlst_nsq: images | qshake : nptq_h4
***** | qshake : nveq_2
parneulst: error | qshake : nvtq_b2
parneulst: gimax | qshake : nvtq_h2
parneulst: gstate | *****
parneulst: images | quatbook : sysbook

```

```

***** | *****
parset: cfgscan | quatqnch : dlpoly
parset: conscan | quatqnch : systemp
parset: dcell | quatqnch : vscaleg
parset: fldscan | *****
***** | quench : systemp
***** | *****
passcon: csend | rdf0 : forces
passcon: error | rdf0 : multiple
passcon: error | rdf0 : multiple_nsq
passcon: gisum | *****
passcon: gstate | rdf0neu : forcesneu
passcon: gsync | rdf0neu : multipleneu
passcon: mpi_irecv | *****
passcon: mpi_send | rdf1 : result
passcon: mpi_wait | *****
passcon: msgwait | *****
***** | rdshake_1 : npt_b1
***** | rdshake_1 : npt_b3
passpmf: error | rdshake_1 : npt_h1
passpmf: gisum | rdshake_1 : npt_h3
***** | rdshake_1 : nptq_b1
***** | rdshake_1 : nptq_b3
passquat: error | rdshake_1 : nptq_h1
passquat: gisum | rdshake_1 : nptq_h3
passquat: gstate | rdshake_1 : nve_1
***** | rdshake_1 : nveq_1
***** | rdshake_1 : nvt_b1
pmf_1: gdsum | rdshake_1 : nvt_e1
pmf_1: images | rdshake_1 : nvt_h1
pmf_1: merge | rdshake_1 : nvtq_b1
pmf_1: pmf_shake | rdshake_1 : nvtq_h1
pmf_1: rdshake_1 | rdshake_1 : pmf_1
***** | *****
***** | result : dlpoly
***** | *****
primlst: images | revive : dlpoly
***** | revive : result
***** | *****
prneulst: error | scdens : forces
prneulst: gimax | *****
prneulst: gstate | scl_csum : ewald_spme
prneulst: images | *****
***** | *****
qshake: gdsum | scopy : merge
qshake: gstate | scopy : merge.hcube

```

```

qshake: images | scopy : merge.systol
qshake: shmove | scopy : merge4.hcube
qshake: splice | scopy : shmove
***** | *****
quatbook: error | set_block : ewald_spme
quatbook: gdsum | *****
quatbook: gstate | shellsort : intlist
quatbook: images | shellsort : parlinkneu
quatbook: invert | shellsort : quatbook
quatbook: jacobi | *****
quatbook: merge | shlfrc : dlpoly
quatbook: merge4 | *****
quatbook: shellsort | shlmerge : shlqnch
***** | *****
quatqnch: images | shlqnch : dlpoly
quatqnch: merge | shlqnch : systemp
quatqnch: merge1 | *****
***** | shmem_barrier_all : gdsum
quench: error | shmem_barrier_all : gsync
quench: gstate | shmem_barrier_all : merge
quench: images | shmem_barrier_all : merge.hcub
quench: shmove | shmem_barrier_all : merge.syst
quench: splice | shmem_barrier_all : merge1
***** | shmem_barrier_all : merge1.hcu
rdf0 : | shmem_barrier_all : merge4
***** | shmem_barrier_all : merge4.hcu
rdf0neu : | shmem_barrier_all : shlmerge
***** | shmem_barrier_all : shmove
rdf1: gdsum | *****
***** | shmem_put : gdsum
rdshake_1: error | shmem_put : merge
rdshake_1: gdsum | shmem_put : merge.hcube
rdshake_1: gstate | shmem_put : merge.systol
rdshake_1: images | shmem_put : merge1
rdshake_1: shmove | shmem_put : merge1.hcube
rdshake_1: splice | shmem_put : merge1.systol
***** | shmem_put : merge4
result: rdf1 | shmem_put : merge4.hcube
result: revive | shmem_put : shlmerge
result: timchk | shmem_put : shmove
result: zden1 | *****
***** | shmem_put64 : gdsum
revive: gdsum | shmem_put64 : merge
revive: error | shmem_put64 : merge.hcube

```

```

revive: merge | shmem_put64 : merge.systol
***** | shmem_put64 : merge1
scdens: denloc | shmem_put64 : merge1.hcube
scdens: gdsum | shmem_put64 : merge4
scdens: images | shmem_put64 : merge4.hcube
***** | shmem_put64 : shlmerge
scl_sum : | shmem_put64 : shmove
***** | *****
set_block : | shmem_udcflush : gdsum
***** | shmem_udcflush : merge
shellsort : | shmem_udcflush : merge.hcube
***** | shmem_udcflush : merge.systol
shlfrfc: error | shmem_udcflush : merge1
shlfrfc: gdsum | shmem_udcflush : merge1.hcube
shlfrfc: images | shmem_udcflush : merge1.systol
***** | shmem_udcflush : merge4
shlmerge: csend | shmem_udcflush : merge4.hcube
shlmerge: error | shmem_udcflush : shlmerge
shlmerge: gsync | shmem_udcflush : shmove
shlmerge: mpi_irecv | *****
shlmerge: mpi_send | shmove : qshake
shlmerge: mpi_wait | shmove : quench
shlmerge: msgwait | shmove : rdshake_1
shlmerge: shmem_barrier_all | *****
shlmerge: shmem_put | shpalloc : dlpoly
shlmerge: shmem_put64 | *****
shlmerge: shmem_udcflush | simdef : dlpoly
***** | *****
shlqnch: shlmerge | spl_cexp : ewald_spme
***** | *****
shmove: csend | splice : pmf_shake
shmove: dcopy | splice : qshake
shmove: gsync | splice : rdshake_1
shmove: mpi_irecv | splice : quench
shmove: mpi_send | *****
shmove: mpi_wait | spme_for : ewald_spme
shmove: msgwait | *****
shmove: scopy | srfrce : forces
shmove: shmem_barrier_all | srfrce : multiple
shmove: shmem_put | srfrce : multiple_nsq
shmove: shmem_put64 | *****
shmove: shmem_udcflush | srfrceneu : forcesneu
***** | srfrceneu : multipleneu
simdef: ccfft3d | *****

```

```

simdef: dblstr                | static: dlpoly
simdef: dcell                 | *****
simdef: error                 | strip : conscan
simdef: intstr                | strip : fldscan
simdef: fftw3d_f77_create_plan | strip : fortab
simdef: getrec                | strip : fortab_rsq
simdef: lowercase             | strip : simdef
simdef: strip                 | strip : sysdef
simdef: zzfft3d               | strip : sysgen
*****                        | *****
scl_cexp :                    | strucoat : dlpoly
*****                        | *****
splice: error                 | suttchen : forces
splice: gdsum                 | *****
*****                        | sysbook : dlpoly
spme_for: gdsum              | *****
*****                        | sysdef : dlpoly
srfrce :                      | *****
*****                        | sysgen : dlpoly
srfrce_4pt :                  | *****
*****                        | sysinit : dlpoly
srfrce_rsq :                  | *****
*****                        | systemp : dlpoly
srfrceneu :                   | *****
*****                        | tethfrc : dlpoly
static: dcell                 | *****
static: diffsn1               | thbfrc : dlpoly
static: diffsn0               | *****
static: error                 | timchk : dlpoly
*****                        | timchk : result
strip :                       | *****
*****                        | timef : timchk
strucoat: images              | *****
*****                        | trajet : dlpoly
suttchen :                    | *****
*****                        | vertest : dlpoly
suttchen_4pt :                | *****
*****                        | vscaleg : dlpoly
suttchen_rsq :                | vscaleg : systemp
*****                        | *****
sysbook: error                | warning : exclude
sysbook: exclude              | warning : hkewald1
sysbook: exclude_atom         | warning : hkgen
sysbook: exclude_link         | warning : sysdef

```

```

sysbook: excludeneu      | warning : sysgen
sysbook: intlist         | *****
sysbook: passquat        | xscale : dlpoly
sysbook: quatbook        | *****
*****                   | zden0 : dlpoly
sysdef: dblstr           | *****
sysdef: erfcgen          | zden1 : result
sysdef: error            | *****
sysdef: forgen           | zzfft3d : ewald_spme
sysdef: fortab           | zzfft3d : simdef
sysdef: getrec           | *****
sysdef: hkgen            |
sysdef: intstr           |
sysdef: lowercase        |
sysdef: metgen           |
sysdef: strip            |
sysdef: warning          |
*****                   |
sysgen: dblstr           |
sysgen: dcell            |
sysgen: error            |
sysgen: gdsum            |
sysgen: getrec           |
sysgen: gstate           |
sysgen: intstr           |
sysgen: strip            |
sysgen: warning          |
*****                   |
sysinit: error           |
sysinit: gdsum           |
sysinit: gisum           |
sysinit: lrcmetal        |
sysinit: lrcorrect       |
*****                   |
systemp: error           |
systemp: gauss           |
systemp: quatqnch        |
systemp: quench          |
systemp: shlqnch         |
systemp: vscaleg         |
*****                   |
tethfrc: error           |
tethfrc: gdsum           |
tethfrc: gstate          |

```

```

tethfrc: images
*****
thbfrc: dcell
thbfrc: error
thbfrc: gdsum
thbfrc: gstate
thbfrc: invert
*****
timchk: timef
timchk: mynode
*****
traject :
*****
traject_u :
*****
vertest: error
vertest: gisum
*****
vertest1: error
vertest1: gisum
vertest1: images
*****
vscale: gdsum
vscale: invert
vscale: merge
*****
vscaleg: gdsum
vscaleg: invert
vscaleg: quatqnc
*****
warning :
*****
xscale: images
xscale: merge
*****
zden0 :
*****
zden1: gdsum
*****

```


Index

- algorithm, 20, 66, 111
 - Brode-Ahlrichs, 33, 83–85
 - FIQA, 20, 21, 67, 77
 - multiple timestep, 80, 81, 83, 92, 93, 114, 116, 145, 191, 832, 839, 841, 843
 - QSHAKE, 20, 21, 67, 78, 80, 87
 - RD-SHAKE, 20, 21, 66, 67, 69, 86, 87, 829, 834
 - SHAKE, 20, 82, 86, 87, 316, 845
 - Verlet, 20, 33, 34, 48, 54, 66–70, 72, 73, 86
- AMBER, 17, 19, 32, 103
- angular restraints, 39
- barostat, 20, 78, 113, 849
 - Berendsen, 75, 80, 192
 - Hoover, 72
- boundary conditions, 19, 53, 92, 102, 191, 192, 806, 823, 825
 - cubic, 121, 823, 825, 835
 - hexagonal prism, 121
 - parallelpiped, 823, 825, 835
 - rhombic dodecahedron, 121
 - slab, 835
 - truncated octahedron, 121
- CCP5, 17, 26, 27
- charge groups, 124, 192, 828, 834
- COMMON blocks, 22, 23, 98
- constrains
 - bond, 316
- constraints
 - bond, 18, 20, 34, 67–69, 75, 76, 78, 86, 87, 92, 93, 126, 127, 144, 191, 192, 315, 420, 819, 824, 828, 829, 844
 - Gaussian, 57, 70, 191
 - PMF, 69, 127
- CVS, 22, 23
- direct Coulomb sum, 53, 54, 56
- distance dependant dielectric, 56, 63, 113, 118, 843
- distance restraints, 36
- DLPROTEIN, 28, 102
- ensemble, 20, 842, 844
 - Berendsen $N\sigma T$, 20, 21, 67, 113, 115, 117
 - Berendsen NPT, 20, 67, 115, 117
 - Berendsen NVT, 20, 67, 113, 115, 117
 - canonical, 70
 - Evans NVT, 20, 67, 113, 115, 117
 - Hoover $N\sigma T$, 20, 21, 67, 115
 - Hoover NPT, 20, 21, 67, 113, 115
 - Hoover NVT, 20, 67, 115
 - microcanonical, *see* ensemble,NVE
 - NVE, 70, 113, 115, 117
- error messages, 107, 812–854
- Ewald
 - Hautman Klein, 53, 61, 106, 113, 810
 - optimisation, 104, 105
 - SPME, 22, 53, 59, 97, 104, 114, 192, 324, 326, 394
 - summation, 53, 54, 57–59, 80, 82, 85, 104–106, 113, 115, 116, 191, 192, 833, 835, 837, 845
- force field, 18, 19, 32, 34, 41, 52, 53, 103, 815, 829
 - AMBER, 19, 32
 - DLPOLY, 18, 32–65
 - Dreiding, 19, 32, 50, 51, 192, 193

- GROMOS, 19, 32
- forcefield
 - Ceramics, 152
 - Dreiding, 152
- FORTRAN 77, 21, 23, 91, 97, 303
- FORTRAN 90, 21, 22
- FTP, 26–28
- Graphical User Interface, 26, 27, 102, 104, 120
- GROMOS, 17, 19, 32
- Java GUI, 19, 27, 104, 152
- licence, 17
- long range corrections
 - Sutton-Chen, 50
 - van der Waals, 48
- parallelisation, 20, 82
 - Ewald summation, 85
 - intramolecular terms, 83
 - Replicated Data, 20
 - Verlet neighbour list, 85
- polarisation, 64, 65, 192
 - dynamical shell model, 65, 827, 828
- potential
 - bond, 19, 33–36, 40, 42, 46, 51, 65, 83, 86, 102, 126, 144, 315, 316, 420, 818, 846
 - Coulombic, *see* potential,electrostatic
 - dihedral, 19, 32, 33, 39, 40, 42, 43, 82, 83, 130, 144, 315, 316, 420, 822, 847
 - electrostatic, 18, 24, 33, 36, 38, 53, 54, 80, 81, 112–114, 118, 144, 191, 192, 839, 843, 846
 - four-body, 19, 32–34, 46, 51, 52, 135, 136, 144, 315, 316, 420, 817, 825, 827, 829, 846, 848
 - improper dihedral, 19, 42, 83
 - intramolecular, 46, 52, 92
 - inversion, 19, 32, 43–45, 51, 52, 131, 315, 316, 420, 825, 847
 - metal, 19, 49, 824, 832
 - nonbonded, 18, 33, 34, 83, 85, 99, 102, 103, 114, 123, 126, 129, 132, 815
 - Sutton-Chen, *see* potential,metal
 - tabulated, 138, 817
 - tethered, 45, 46, 131, 144, 192, 315, 316, 420, 823, 847
 - three-body, 19, 32–34, 36, 46, 50, 51, 85, 102, 133, 135, 144, 316, 420, 816, 824, 826, 847
 - torsion, *see* potential,dihedral
 - valence angle, 18, 19, 32–34, 36, 37, 43, 50, 51, 82, 85, 92, 102, 127, 129, 144, 315, 316, 420, 821
 - van der Waals, 33, 36, 38, 81, 92, 98, 116, 130, 316, 841
- quaternions, 20, 67, 77, 114
- reaction field, 63, 64, 114
- respa, 21, 26, 81
- rigid body, 18, 20, 21, 46, 67, 76, 78, 79, 81, 87, 92, 191, 192, 315, 316, 420, 827, 835–838, 844, 848, 853
- rigid bond, *see* constraints,bond
- rigid ion, 21, 81
- rigid molecule, *see* rigid body
- stress tensor, 38, 41, 45, 46, 48, 49, 51, 52, 55, 56, 58, 64, 65, 69, 72, 845
- sub-directory, 300, 302
 - bench, 25
 - build, 25
 - data, 25
 - execute, 25, 813
 - java, 25
 - public, 25
 - respa, 25
 - sdk, 25
 - source, 25
 - utility, 25
- thermostat, 20, 53, 78, 81, 113, 844, 849
 - Berendsen, 75, 78, 80, 191, 192
 - Nosé-Hoover, 72, 73, 78, 80, 93, 191

units

DLPOLY, 24, 145

energy, 124

pressure, 24, 73, 114, 145

Verlet neighbour list, 59, 80, 83, 85, 92,
116, 830

WWW, 18, 27, 28

X-PLOR, 17