

# XMD - Molecular Dynamics Program

---

Jon Rifkin, jon.rifkin@uconn.edu

v2.5.30 20 Jan 2002

XMD is a program for performing molecular dynamics simulations.

## Contents

### 1 Features

XMD is a computer program for performing molecular dynamics simulations. It is designed for the simulation of metals and ceramics. It has the following features.

#### Computer System

- C source code is available for easy porting to a variety of computers.
- Can be compiled using POSIX thread functions to take advantage of multi-CPU computers (see section on ?? (Parallel Processing)).

#### Molecular Dynamics

- Constant temperature using velocity re-scaling algorithm (CLAMP command).
- Constant pressure using either Andersen's algorithm or a simple volume "rescaling" algorithm suitable for lattice constant versus temperature calculations. (PRESSURE command).
- Efficient system relaxation (local minima) algorithm (QUENCH command).
- Constrain selected atoms to remain fixed in space (FIX command).
- Constrain selected atoms to remain in a line or plane (CONSTRAIN command).
- Apply an unique external force to individual atoms (EXTFORCE command).
- Tether selected atoms to their initial positions with springs of various spring constants (EXTSPRING command).
- Apply velocity damping coefficients to selected atoms (DAMP command).
- Uses pair potential, Embedded Atom Method potentials (EAM), Tersoff's Silicon-Carbide potential or Stillinger-Weber Si potential.

#### Program Input

- All program actions are controlled by commands read from one or more text files.
- A macro processor lets you enter strings from the command line or define strings within the command file (MACRO command) (see section on ?? (macros)).
- A built-in calculator with variables that can be used with all number input (CALC command) (see section on the ?? (Built-in Calculator)).
- Input file instruct the reading of another input file (READ command).
- Flexible creation (FILL command) and editing of atomic structures.
- Select subsets of atoms using a variety of patterns (SELECT, SET and TAG commands).

- Perform actions on selected atoms, such as moving them (MOVE), inserting defects (SCREW, WAVE), setting atom type (TYPE), setting atom mass (MASS), rotating selected atoms (ROTATE), etc.
- Some commands (MOVE, DAMP, EXTFORCE and EXTSPRING) allow input values to be a function of atomic coordinates.

### Program Output

- Save the complete simulation state for later continuation (WRITE STATE command).
- Accumulate system energies (ESAVE and WRITE ENERGY commands), atomic stress (SSAVE and WRITE STRESS commands), atom coordinates (WRITE PARTICLE command) and other information to text files during simulation.
- Coordinates can be stored in special compact format (RCV or COR commands) for later use with companion programs such as WMOVIEC which produces movies.
- Save atomic coordinates in a Protein Data Bank file for use with display software such as MSI's Cerius2 (WRITE PDB command), or standard XYZ format (WRITE XYZ command).
- Built-in plotting routines for plotting atom configuration in color on a Postscript device (PLOT command).

## 2 Introduction

XMD is a program which performs Computer Molecular Dynamics (CMD) simulations. It can use Embedded Atom Method (EAM) potentials, pair potentials, Tersoff's silicon-carbon potential [1], or Stillinger-Weber's Silicon potential [2]. With XMD you can generate specific lattice structures. With these lattice structures you can perform calculations using both static (energies or forces) and dynamic (molecular dynamics). XMD reads a command file which describes a an initial atomic system, the simulation to be performed, and the output desired. This command file is a normal text file (a file that can be read by a normal text editor). Normal text files are also used for recording energies during the course of a simulation. Special format files (files than cannot be used with text editors) are used for storing the particle coordinates (RCV files) and the complete CMD state (STA files).

## 3 File Formats

A variety of file formats can be used with XMD.

### Input File

XMD's main input is a text file. This file contains commands (described below) which control the simulation. This file is required.

### Output File

The output file is also a text file. It contains the messages generated by the commands in the input file. An output file is always generated.

### Energy / Stress / Box / Trajectory

These four types of files are also text files. They are optional and contain information generated during a dynamics simulation. They are controlled by the commands ESAVE, SSAVE, BSAVE and TSAVE (see below).

## **COR / RCV**

There are special format files (not Text files) that contain atom positions as a function of time. They are written by the `WRITE COR` and `WRITE RCV` commands. They can be read by XMD or by other support programs such as MSD, WMOVIEC, STRAIN, etc (see section on auxillary program below).

## **XYZ / PDB**

Like the previous pair of files, these contain atoms positions. They are files that can be read by a variety of third party programs. They are created by the `WRITE XYZ` and `WRITE PDB` commands.

## **STATE**

This file is a machine file which contain information needed to continue a simulation exactly. It is written and read with the `WRITE STATE` and `STATE` commands.

# **4 Usage**

The methods used to invoke XMD depend on the specific computer environment. At present XMD has been ported to UNIX and the IBM PC under DOS. Under these environments the command used is

```
xmd infile [ $1 $2 $3 .. $9 ] > outfile
```

The parameters `infile` and `outfile` are to be replaced by the names of your input and output file. The `>` is the "IO redirection" symbol under both DOS and UNIX, and it causes the output file to be written to `outfile`. If the `>` symbol and the `outfile` are omitted, the output is printed on the screen. Under DOS, the user must wait for this command to finish before another can be run. Under UNIX, one can enter either

```
nohup XMD infile > outfile &
```

on the IBM Risc 6000 using the AIX operating system, or

```
xmd infile > outfile &
```

under Linux and other Unix's. These commands will run XMD in "background" mode. That is, while the program is running, you will be able to run other commands and programs as well. Both `nohup` and `&` are special to UNIX.

You can include up to 9 optional parameters after the input file name denoted as `$1`, `$2`, .. `$9` in the example above. These strings will be substituted for the strings `$1`, `$2`, `$3`, .. `$9` in the input file. This allows you to leave some input file parameters unspecified until you run it. For example, if you would like to run a simulation at a variety of temperatures, you can put the string " `$1` " where XMD expects to see a temperature value. Then when you run the program, you can enter

```
xmd infile 100 > outfile
```

for one run, then

```
xmd infile 200 > outfile
```

for another, and run the simulation at 100 and 200 degrees Kelvin. For more information, see the sections on `??` (command line macros) and on the `??` (MACRO command).

By default, XMD will search your current directory for a file named `config.xmd`. This file can contain normal commands that you would like to be run every time you run XMD in that directory. If XMD cannot find the file in the current directory, it will then search the directory where the program itself is stored, and read that copy of `config.xmd`. Then, whether it finds `config.xmd` or not, XMD will read `infile`.

## 5 Theory

### 5.1 Molecular Dynamics

In the technique of molecular dynamics the motion for every atom is calculated. This algorithm proceeds as follows.

1. (1) The initial positions and velocities of every atom are specified.
2. (2) Using the interatomic potentials, the forces on these atoms are calculated.
3. (3) Using these forces, the atomic positions and velocities are advanced through a small time interval (called a time step). These new positions and velocities become new input to step (2), and when steps 2 and 3 are repeated, each repetition constitutes an additional time step.

Typically a molecular dynamics simulation will comprise thousands of such time steps, each time step corresponding to fraction of a picosecond ( $1/10^{12}$  seconds). This algorithm is essentially an integration of the Newtonian equations of motion over time to yield the particle positions and velocities.

### 5.2 Boundary Conditions

A molecular dynamics simulation is limited in the number of atoms one can simulate, typically XMD simulations are between 5,000 and 100,000 atoms. With such a small number of atoms, it is not possible to simulate bulk material unless one uses **repeating boundary conditions**. With repeating boundary conditions, one typically conducts a simulation within a box. Atoms that pass out one wall of the box pass back in to the box through the opposite wall. Thus at no time is an atom outside the box. This way there is no free surface, and the system simulates the bulk. While in theory, any parallelepiped (that is, a cell with triclinic symmetry) can be used as the repeating cell, at this time XMD is restricted to using a rectangular box. The repeating boundary conditions can be turned off in any direction independently of the others, in this way one can simulate an infinitely repeating solid (the bulk), an infinitely repeating sheet, an infinitely repeating "wire" or a finite fragment of material. There is more information on this under the ?? (BOX command).

### 5.3 Adiabatic and Isothermal Simulations

In thermodynamics, systems are divided into two kinds, adiabatic and isothermal. Adiabatic systems are thermally isolated from the external world, and no heat will flow in or out. The result is that the total energy (the sum of potential and kinetic energies) will always remain constant. Isothermal systems are systems which maintain a constant temperature through contact with a heat bath. Both kinds of systems can be simulated with molecular dynamics. See the section on ?? (Temperature Control) for more information.

### 5.4 Time Step Size

As explained in the above section on molecular dynamics, the forces and velocities at one step in time are used to calculate the resulting positions at the next step. The time difference between these two adjacent steps is called the time step size. The time step size must be specified by the user. Ideally one would like to use the largest time step size possible, for this way one would simulate the greatest time possible. However the molecular dynamics integration algorithm becomes unstable at large time step sizes. This can be understood by considering a single particle in a one dimensional harmonic well. Assume that this particle is to one side of the center and that its initial velocity is zero. This particle would experience a force toward

the center of the well. The product of the force with the time step size squared would yield the displacement of the particle. A small time step would advance the particle closer to the well center. A too large time step would overshoot the center of the well, and could even place the particle on the opposite side higher than its starting position. This would immediately cause a problem. The total energy of the particle will have increased, and subsequent steps at this large time step size would only increase the energy more. This clearly violates the conservation of energy.

In practice this violation of energy conservation is exactly the clue used to determine that a time step is too large. One wants to find the largest time step that will maintain the conservation of energy. For hints on how to implement a search for the optimum time step see the section ?? (Techniques).

## 5.5 Metropolis Algorithm

The Metropolis algorithm is a type of Monte Carlo technique used for two purposes (1) to calculate averages of a system at finite temperatures (thermodynamic averages) and (2) to simulate annealing of a system of particles. This algorithm varies the atomic positions at random and the properties of interest for each configuration are calculated. In a simple Monte Carlo (not Metropolis) scheme for calculating thermodynamic averages random atomic configurations are sampled and each contribution to the average is weighted by the factor  $\exp(-E/kT)$  where  $E$  is the system potential energy,  $k$  is Boltzmann's constant, and  $T$  is the temperature. However purely random atomic configurations will consist mostly of high energy (and therefore improbable) states, and the resulting weighting factor will be so small as to render insignificant the contribution to the average. Thus the bulk of the computational time will be spent calculating useless numbers. The alternative is to use the Metropolis method for sampling configuration space. This method samples phase space via a random walk. Steps in this random walk are only taken if the energy remains low enough to contribute a significant amount to the average. Steps in this random walk which raise the energy will be rejected unless a randomly generated number uniform on the interval  $[0..1]$  is less than the factor  $\exp(-dE/kT)$  (where  $dE$  is the energy of the new state minus the energy of the original). Steps which lower the energy are always accepted. Averages are taken for every configuration (if a new configuration is rejected then the old configuration re- contributes to the average). This selective random walk results in the averages taken being weighted by the factor  $\exp(-E/kT)$ . The controlling parameters are the temperature and the jump size. The jump size determines how far the atoms move with each random walk.

## 5.6 Constant Pressure Algorithms

Sometimes one wishes to simulate system which can change its volume in response to the combination of applied external pressure and the system's own internal stress. XMD has two different methods for accomplishing this. The first, Andersen's method [3], is designed reproduce the thermodynamics properties of an isobaric adiabatic (NPE) ensemble. Andersen's method is described in more detail below.

The second method is " pressure clamp " method (see the PRESSURE CLAMP command). This method was developed for XMD. It automatically rescales the system volume by the amount needed to balance an applied pressure. It is designed for determining the equilibrium volume as a function of applied temperature and pressure. Most typically, it is used to determine the thermal expansion of a simulated material.

### 5.6.1 Andersen's Constant Pressure Method

This is a method which maintains an external pressure by imposing an artificial force on every atom. This has the advantage of not creating a localized interface between the simulation and the external pressure source. It is implemented with the PRESSURE command. When using Andersen's constant pressure, you must assign a mass to the external system. This mass determines how quickly the volume of the system responds to the pressure. Without such a mass the system volume changes would be instantaneous - clearly

a non-physical situation. The exact value of the mass is probably unimportant. A good initial guess would be to set the external system mass to the total internal mass (the mass of all the atoms).

Andersen's constant pressure method is not suitable for determining thermal expansion. One reason is that it takes much longer to reach equilibrium volume than the "pressure clamp" method. The second reason is that this method introduces a fictitious pressure that is proportional to the system kinetic energy. The value of the external pressure is given by the simple gas law,

$$P = nKT/V$$

where

$N$  is the total number of atoms,  $k_B$  is Boltzmann's constant,  $T$  the temperature and  $V$  the average system volume.

### 5.6.2 The "Pressure Clamp" method

The "Pressure Clamp" works analogously to the temperature clamp. The volume of the system box is automatically re-sized to maintain a constant pressure. This works as follows. The user specifies a pressure that she would like to maintain, the default is zero. In addition, the user specifies a Bulk Modulus for the system. This can be the exact Bulk Modulus or an approximation. Then, for each molecular dynamics step, the system stress is calculated. By combining the system stress and Bulk Modulus, a change in the box size can be estimated which produces the desired pressure,

$$(dL_i/L_i)/C_{step} = B / (3 * S_i)$$

where  $L_i$  is the box size in the  $i$ 'th direction,  $dL_i$  is the change in that box size,  $S_i$  is the stress in the  $i$ 'th direction (the Voigt stress),  $B$  is the Bulk Modulus, and  $C_{step}$  is a parameter which controls how quickly the algorithm converges on a suitable box size. By default  $C_{step}$  is 33, but this is under user control.

### 5.6.3 Stress Calculation and the Virial Term

XMD by default does not include the virial term

$$S_{ij} = mass * V_i * V_j$$

in the stress calculation. This is a controversial issue, but I justify this from the fact that a perfect harmonic solid will not thermally expand, but the virial term predicts it will.

The virial term can be included in the stress calculation. See the STRESS THERMAL command for details.

### 5.6.4 Box Geometry

When using either of these two constant pressure algorithms, one has the option of letting each box direction ( $x, y, z$ ) change independently, or forcing the system to change uniformly. When changing uniformly, the ratio of  $x$ ,  $y$  and  $z$  box sizes remains the same, and consequently a system which starts out as a cubic lattice will remain a cubic lattice, and a tetragonal lattice will remain tetragonal and maintain its initial  $c/a$  ratio.

## 6 Implementation

In order to effectively use XMD it is helpful to understand the data structures and algorithms that the program uses.

## 6.1 Coordinates

Perhaps the most obvious data structure is the particle coordinates. Several versions of these coordinates are stored. (1) Current Step. The current value of the coordinates used to integrate the equations of motion for every step. These are coordinates are the most commonly used. When the commands WRITE RCV or WRITE STATE or WRITE PARTICLE are given, these coordinates are written. (2) Reference Step. When the command REFSTEP is given the current values of the coordinates (from item 1 above) are saved. These can later be restored as the current coordinates or used to calculate the particle displacements. (3) Neighbor list. This is only for internal use. It is a copy of the coordinates at the time of the last neighbor list update (see neighbor list below). It is kept to see how far the current coordinates have moved since the last update and thereby determine when a new neighbor list needs to be made.

## 6.2 Velocity, Force, and Higher Derivatives

The Gear algorithm used to integrate Newton's equations of motion uses up to the 5'th time derivative of the particle positions. The 1'st and 2'nd derivatives are just the velocity and the acceleration. The program can write out the velocities and accelerations the WRITE VELOCITY and WRITE ACCELERATION commands.

## 6.3 Particle Selection

For various operations on the particles (such as moving them or duplicating them) it is necessary to specify a subset of particles. This is done with the SELECT command. This command uses an array of flags, one flag for each particle. The flags in the array are set by calling SELECT one or more times. Then a command such as WRITE SEL COR can be used to write only those coordinates whose flags have been set.

Related to this array is the SET array. The SET command can be used to include any particle in any of up to 12 sets. First the SELECT the particles, then use the command " SET ADD 4 "for instance. At a later time you can select all particles from set 4 by " SELECT SET 4 " . The SET and SELECT SET commands are very useful for manipulating separate structures in a simulation.

A single atom can belong to more than one set. For example, you may have one set contain all the atoms in the box whose x value is less than half the box. Then you could make a second set of atoms whose y values are less than half the box. Likewise you can make a third set according to the atom's z values. Then some atoms will belong to three sets - those atoms whose x, y and z coordinates are all within the lower half box.

A related array is the TAG array, one byte of data for each atom. Thus each atom can have a single tag value whose range is 0 to 255. TAGs are not as flexible as SETs, because each atom can have only one TAG value, while an atom can belong to many SETs. However, there are 256 TAG values (and only 12 SETs). So using TAGs you can group atoms into 256 separate groups, as long as each atom belongs to only one group.

Each atom has a default TAG value of 0.

The SELECTION, SET and TAG values are lost whenever a new set of particles is read, the commands that do this are PARTICLE, COR, RCV and STATE. Sometimes it is desirable to read in coordinates with the COR or RCV command but preserve the SELECTION, SET and TAG values from a previous set of particles. For instance, in a simulation you might be curious about the current position of a set of particle which started inside a spherical region at step 0. If these particle positions are stored in a COR file at steps 0 and 2000, you can do the following,

```
cor input.cor 0
select ellipse 5 5 5 2 2 2
select keep on
```

```
cor input.cor 2000
write sel particle
```

This reads the particle positions from file `input.cor` at step 0, and selects those particle within an ellipsoid centered at (5,5,5) with axes of length (2,2,2). Since the axes lengths are equal, this is a sphere. The *select keep on* command says that SELECT, SET and TAG information will not be forgotten when the COR command is used. The next command reads the positions at step 2000, and lastly these positions are written to the output file.

## 6.4 Neighbor List

When calculating the energy and forces over all the atoms, the neighbors for each atom must be known. This calculation is done separately and stored for future use in a set of arrays called the neighbor list. This list is calculated at the beginning of each run, and it includes all the atom pairs that fall within the potential cutoffs plus an extra 10 %. This extra insures that the neighbor list will be valid until the atoms move enough to warrant a recalculation of the neighbor list. When two particles move a total of more than 10 %, then it is time to recalculate. This value of 10 % can be changed by the command NRANGE. It specifies the neighbor list cutoff range as a factor of the potential cutoff range. The default value is 1.1 (which yields a difference of 10 %).

## 6.5 External Forces

The term external force means a force that is applied by an external agent, in other words a force which is not due to the atoms themselves. For some simulations it is necessary to apply an external force to some or all of the particles. This is sometimes the case in crack simulations. Each atom can experience a unique external force, the only limitation being the convenience (or lack) of setting the forces. The command EXTFORCE assigns a single value of the external force to the selected particles.

## 6.6 Velocity Damping

In some simulations it is desirable to dampen the atomic motion, perhaps to absorb lattice vibrations generated by a moving dislocation, for example. This can be done with the DAMP command. This command assigns a damping term to the selected particles. The force is altered by the addition of the damping term,

$$-F_{\text{damp}} * v$$

where  $F_{\text{damp}}$  is the vector force due to the damping term, and  $v$  is the vector velocity.

## 6.7 Temperature Control

In molecular dynamics it is usually necessary to control the temperature of the simulation. There are two types of control, temperature initialization and temperature maintenance. Typically at the beginning of a simulation the position of every particle is specified but the velocities are not known, only the desired temperature is known. The command ITEMP will assign random velocities to all the particles with the appropriate Maxwell-Boltzmann distribution for the specified temperature. The command also has the option of assigning velocities in only one or two of the three x,y,z directions (leaving the other velocities unchanged). This can be employed to simulate a two or one dimensional solid, for if the initial coordinates are all confined to a plane or line, and all the velocities normal to the plane or line are zero, then the particles will remain confined to the required geometry. The velocities default to zero at the beginning of

a run, and if a run has already been in progress, you can initialize the velocities to zero explicitly (using the command `ITEMP 0`). Temperature maintenance is used to obtain a simulation at a constant temperature. Often one will simulate a phenomena which liberates heat, such as a phase transformation or work done by an external force, or perhaps one will simulate a system which absorbs heat. In order to maintain an approximate constant temperature a "temperature clamp" is applied. This is an algorithm which scales the instantaneous velocities in order to bring the temperature to its desired value. Scaling is accomplished by multiplying each particle velocity component by the same factor. Typically you do not want to reach your desired temperature in one step, because the temperature change will be too abrupt. Early simulations using just such a scheme produced marked noise in the high frequency end of the phonon spectrum. So instead the velocities are scaled by the 33<sup>rd</sup> (default value) root of this "one step" factor. This has the approximate effect of reaching the desired temperature after 33 steps. This number along with the desired temperature is specified by the `CLAMP` command. If no clamp is specified (or if a negative clamp is specified) then no temperature maintenance is done, and a adiabatic simulation results. Such a simulation is useful for testing the stability of the simulation (and hence time step size, see the `THEORY` section above). If the total energy reported by the program for an adiabatic simulation is not constant, then most likely the time step size is too large. See the section on `TECHNIQUES` below for more information on adjusting the time step size. Note that the temperature can also be affected by the `VELOCITY` command.

## 6.8 Quenching

In some simulations it is desirable to "quench" the atomic motions during a dynamics simulation. The purpose of quenching is to rapidly relax the system to the current local minimum potential energy. Quenching acts to alter the atomic motion but differs from the algorithm used by the `CLAMP` command. Whereas `CLAMP` scales velocities up or down in order to maintain a specific temperature, quenching sets the atomic velocities to zero whenever certain conditions are met. XMD currently implements two styles of quenching. In the first method, every atoms' velocity is zeroed whenever the total potential energy rises relative to the preceding time step. In the second method, an individual atom's velocity is zeroed whenever its energy alone rises relative to the preceding step. A rise in energy for a particular atom is determined by calculated for each atom. Whenever this quantity is negative for an atom, its velocity is set to zero. Quenching is controlled with the `QUENCH` command.

## 6.9 Fixing Particle Positions for Molecular Dynamics

For molecular dynamics simulations the box dimensions for a repeating box is always fixed, there is currently no provision for dynamic simulations with a varying box. However individual particles may be fixed using the `FIX` command. For example to study the structure of transformation interface you may want to hold BCC and FCC structures fixed at opposite ends of the box (to insure that neither BCC or FCC disappears entirely from the box) and allow the atoms in between move freely in order to study the resulting arrangement. The information about which particles are fixed are stored in an array of flags, one for each particle. The command `FIX [ON|OFF]` will set or unset the flags of the currently selected particles. Initially all flags are unset. The particles which are fixed are totally equivalent to other particles as far as the force and energy calculations are concerned, but their positions are not updated, and their velocities and higher time derivatives are set to zero. When particles are fixed, they cannot contribute kinetic energy to the system. However, when the kinetic energy per atom is reported (using the `ESAVE` or the `WRITE EKIN` commands) it is normalized by the number of both fixed and free particles. On the other hand, when the temperature is calculated (whose value is reported by the `WRITE TEMP` command), it is the temperature of the free particles only (that is, the total system kinetic energy divided by the number of free particles).

## 6.10 Repeating Boundary Conditions

By default repeating boundary conditions are in effect - this can be altered with the SURFACE command. The purpose of the repeating boundary is to avoid the free surface inherent in a finite system of particles. In the simulation the repeating boundaries form a rectangular box. The coordinates of the box range from 0 to some value specified with the BOX command. Repeating boundaries have primarily two effects on a simulation: (1) particles close to opposite walls are considered to be neighbors (and hence interact via the potential) with particles close to far wall and (2) particles which travel through one wall are translated by the length of the box so they come back in the opposite wall. This translation is called "wrapping". Wrapping takes place whenever particles are defined or displaced, and whenever repeating boundary conditions are created using the SURFACE command. Because the repeating boundary conditions can only be in the form of a rectangular box, their use enforces a orthonormal, tetragonal or cubic symmetry depending on the box symmetry.

## 6.11 Constraints

It is possible to apply a variety of constraints to particles using the CONSTRAIN command. The first form of the command restricts particles to motion along a line, or within a plane. This constraint can be applied individually to particles, each particle can have its own line or plane. The second form of the constraint command imposes a spherical wall around the particles. Any particles which attempt to pass through the wall are reflected back by a harmonic spring. This can be used to impose a pressure on a simulation - the resulting pressure can be monitored by using the WRITE or SSAVE commands to display the system stress. Care must be taken with the choice of spring constant, if the choice is too large then the motion integration may become unstable. It is recommended that before doing a production run, the user first do a short run with no temperature clamp to insure that the total energy is conserved (see discussion of under TIME STEP SIZE, page 3).

## 6.12 Run Identification

Typically in research one will conduct a series of runs. To aid in the subsequent identification of a computer run, the user can assign a run number (with the ?? (RUN command)) that will be stored in the STATE and RCV files produced by XMD. Furthermore every step is identified by a number, starting with 0 for the first step (i.e. the particle coordinates before the first integration step). In addition to the above output files, the step is also saved in the ESAVE, BSAVE and SSAVE files. Additional identification can be provided by a user specified title, entered with the TITLE command. The title is composed of 8 ASCII strings each up to 80 characters long, it is stored in the STATE, RCV, and COR files and is written out to the plot output. The current value of the run, step or title can be displayed into the standard output via the WRITE RUN, WRITE STEP, or WRITE TITLE command.

## 6.13 RCV File Format

The RCV file format was originally designed for sending molecular dynamics results (particle coordinates for various time steps) from remote super computers to local workstations using normal phone lines. Some computers would freely translate non-ASCII characters from one value to another, so the RCV file format avoids these characters. The resulting format uses standard ASCII text for miscellaneous data (i.e. temperature, box size, time step, etc.) but the file format uses a type of encoding for particle coordinates. Although better schemes could be implemented today, the RCV format persists because of the many utility program written that require it. The RCV file consists of one or more equivalent sections, each containing coordinate data and other data for a single simulation step. Besides coordinate data each section contains a 100 point

histogram of the coordinate RDF (radial distribution function). This was originally included in the RCV file because the RDF was used regularly and calculating the RDF was too slow on PC class workstations. The RDF is calculated as the number of atom pairs whose radii fall between 0 and twice the "unit cell distance". This unit cell distance can be an arbitrary number but is intended to be the unit cell length of a BCC lattice of the same density as the system being simulated. The assumption about the identity of the "unit cell length" is important only for some existing RCV analysis programs. The RDF function is encoded using the same encoding scheme as the particle encoding.

### 6.13.1 Coordinate and RDF Encoding Scheme

All the particle coordinates are mapped onto a range of integers from 0 to 9999. For dimensions with repeating boundaries the integer 0 corresponds to a coordinate of 0, and the integer 9999 corresponds to a coordinate equal to the box size. For dimensions with free surfaces, 0 corresponds to the smallest coordinate (which could be less than zero) and 9999 corresponds to the largest. Each resulting integer is coded as a pair of ASCII characters. Each ASCII character serves as a single digit in a base 100 representation and ranges from 0 to 99. The first digit (call it  $i$ ) is the low order digit, the second (call it  $j$ ) is the high order digit, so that the resulting integer is  $100*j+i$ . The ASCII characters 27 to 126 are mapped into the digits 0 to 99.

When coding dimensions with free surfaces the box size becomes the distance between the smallest and greatest coordinate. In addition a second value is stored in ASCII format: the translation. This number is subtracted from 0 or the box size to obtain the smallest or greatest coordinate respectively. The x, y and z coordinates of each particle are stored sequentially, up to 13 particles (or 39 coordinates or 78 characters) per line. The encoding for the RDF is very similar, the main difference being that (1) only 100 points are encoded (the points of the RDF) and (2) the integers 0 to 9999 are mapped onto the interval from 0 to MAXTABLE, where MAXTABLE is the maximum value of the RDF.

## 6.14 COR File Format

The COR file format is a replacement for the RCV file format. Its design was not restricted by the requirement to send files over normal phone lines. The primary differences between the COR and RCV formats are

- the COR file stores the atom types along with the coordinates,
- the particle coordinates are stored as two byte integers (i.e. integers between 0 and 65355).
- the RDF is not stored in the COR file and
- the COR file uses non-ascii characters, consequently it is difficult (or impossible) to edit with a typical text editor. Other information stored in a COR file
- Number of particles,
- Boundary conditions,
- Run and Step Number,
- Title,
- Energies - Kinetic, Potential, Total and "Bath".

## 6.15 Atom Plots

XMD can produce graphical output of the simulation atoms. This is done with the PLOT commands. With these commands you can

- assign various symbols, sizes and colors to atoms,
- select the orientation of the plot,
- set the number of pages,
- plot 'tails' (lines which emanate from each atom to show their motion),
- plot bonds (lines connecting atoms within a certain distance from one another),
- set the output destination (typically a file or device) and format (typically Postscript).

## 6.16 REPEAT Command

Often XMD simulations involve repeating blocks of instructions. To make these easier to handle there is the REPEAT command. The REPEAT command is a simple loop command that looks like this,

```
repeat 10
  cmd 100
  write cor +femelt.cor
end
```

All commands between the REPEAT and the END are repeated 10 times in this example. You can even have REPEAT commands within other REPEAT commands.

## 6.17 Built-in Calculator

A useful feature of XMD is the built-in line calculator. It can be used in two ways. You can use expressions such as  $\sqrt{2}$  in place of a number for any XMD command. For example, you can have the command

```
fill cell sqrt(2)/2 0 0 0 sqrt(2)/2 0 0 0 1
```

and the value of  $\sqrt{2}$  will be used the FILL CELL command. Note that you cannot have any spaces in these expressions when used this way, otherwise the expression will be considered to be two (or more) separate values.

The calculator can also be used to set values. We could have written the above command as

```
fill cell x=sqrt(2)/2 0 0 0 x 0 0 0 1
```

This will set the variable X to  $\sqrt{2}/2$  and use this value for the first number. Then, it will use this value of X to set the 5'th number on the line.

You can also use the command CALC to perform calculations on variables (see the section on the ?? (CALC command)). This is especially useful when used with the REPEAT command, for example

```
calc TEMP=100
repeat 5
  itemp TEMP
  cmd 100
```

```

write cor +fetest.cor
calc TEMP = TEMP + 100
end

```

Here we are simulating a system at increasing temperatures, starting with 100 Kelvin and increasing to 500 Kelvin. Note that in the CALC command you do not need to remove spaces as you did in the first two examples. Also, variables are case insensitive, `temp` means the same as `TEMP`. Variables set at one place in the program, either by the CALC command or within other commands as in the first two examples, are available for the remainder of the program (except for the limited exception covered in the next paragraph).

Some commands offer an additional capability. The commands `MOVE`, `DAMP`, `EXTFORCE` and `EXTSPRING` interpret the variables `X`, `Y` and `Z` in a special manner (and as a consequence, if you used variables named `X`, `Y` or `Z` previously, they are not available for this command). These commands apply changes to each individual atom. You can use the special `X`, `Y` and `Z` variables to make these changes a function of the atom's position. See the ?? (`MOVE` command) for an example. See the `MOVE` command for an example.

## 6.18 Macros

XMD allows the use of "Macros". The first use is to allow options to be set from the command line. For example, if you run XMD on the input file `femelt.xml` with the following command

```
xmd femelt.xml third 1200
```

then the arguments `"third"` and `"1200"` can be used as variables `$1` and `$2` in the file `femelt.xml` as follows,

```

#
# Read initialize lattice and dynamics commands from another file
#
read femelt.pos
#
# Initialize temperature to "1200"
#
itemp $2
#
# Instruct XMD to save energies to file "third.e"
#
esave 10 $1.e
#
# Perform dynamics
#
cmd 1000

```

You can use up to 9 command line options labeled `$1` to `$9`.

You can also use the `MACRO` command to set macro with any name. For example,

```

...
macro $file femelt
...
esave 10 $file.e
bsave 10 $file.b
ssave 10 $file.s
cmd 1000

```

Note that there is no equals sign (=) in this command. Don't confuse it with the CALC command! The macro name is case insensitive, \$file and \$FILE are identical. Macro names can be any length and can consist of any combination of letter and numbers, and they can start with a number. You can even change the value of the command line option macros.

## 6.19 Using Color in Plots

When plotting you have the option of selecting colors for atoms. These colors are selected using one of two color models, the RGB model (red-green-blue) and the HSB model (hue-saturation-brightness). In order to obtain the desired color you must know how these models work.

### RGB red green blue

The values of red, green and blue can range between 0 and 1, and specify the amount of red, green or blue in the resulting color. If (red,green,blue) are (0,0,0), then the color is black, if (1,1,1) then the color is white, (1,0,0) is pure red, (0,1,0) pure green, et cetera.

### HSB hue saturation brightness

It may be useful to fix both saturation and brightness to one, and control the color exclusively by the hue. This will give you bright colors which are relatively easy to distinguish from one another. Here are descriptions of the HSB parameters. Hue is the shade of color, starting from red and varying like the color spectrum (red, orange, yellow, green, blue, violet, red). Note that the color spectrum is cyclic, and hue values of 0 and 1 are equivalent (and both red). Saturation ranges from 0 to 1 - at 1 you get the pure hue (whatever it is). At 0 you get a gray. Values of saturation between 0 and 1 mix a corresponding amount of gray with the current hue. The shade of gray is controlled by the brightness (this needs to be confirmed). Brightness also varies from 0 to 1. A brightness of 0 gives black regardless of the Hue and Saturation values. A brightness of 1 gives the most color if your saturation is 1, or gives a pure white if your saturation is 0.

## 6.20 Parallel Processing

XMD can be compiled to run on parallel computers which support POSIX threads. POSIX threads is a standard way of creating and controlling concurrent processes on a parallel machine. Only SMP computers are likely to support POSIX threads, these are "shared memory" systems that typically have up to 8 or 16 cpus. Massively parallel machines do not typically support this. At this time (Aug 1998) the SMP version of XMD has only been tested on a 2 CPU Pentium II 266 mhz running Linux version 2.0.32. It is expected that it will run fine under other versions of Linux for Intel. Other architectures are unknown.

Currently only the EAM potential can take advantage of parallel processing. Typical speed up on our 2 cpu machine has been 1.95, that is two cpus run 1.95 times faster than one cpu.

To compile the parallel version, you must edit the Makefile and uncomment the following lines,

```
#THREAD_LIB=-lpthread
#DEF=-DUSE_THREAD_LIB -D_REENTRANT $(INTEL_PATCH)
```

If you are using an Intel machine such as a Pentium, Pentium Pro or Pentium II you must also uncomment the line

```
#INTEL_PATCH=-DINTEL
```

This is needed to turn on short assembly language patch which compensates for a small bug in some Linux SMP kernels.

## 6.21 Reproducibility

Theoretically, if you repeat a molecular dynamics simulation, the results of the second should exactly equal the first. We call this property "reproducibility". In practice, however, this is not always true. This is a consequence of subtle interactions between XMD commands and computer round-off error. Before we describe these subtle interactions, let's talk briefly about computer round-off error.

### 6.21.1 Computer Round-off Error

In mathematics, the addition and multiplication exhibit the associative property,

$$a + (b + c) = (a + b) + c$$

On the computer, this property does not hold exactly. Consider the following numeric example of the above equation,

$$-1e50 + (1e50 + 1) = (-1e50 + 1e50) + 1$$

Although you can see by the laws of algebra that the two sides of this equation are equal, if you enter them into a typical calculator or computer program, the equation will become

$$0 = 1$$

The problem is that the computer combines  $1e50+1$  to get  $1e50$  because it does not have enough precision to differentiate between  $1e50+1$  and  $1e50$ . This is called Computer Round-off Error. This is an extreme example, but smaller versions occur all the time, where the result of a large number of sums depends on the order in which the sums were done. These kinds of sums occur throughout XMD in the force and energy calculations. If two otherwise identical runs perform these sums in differing order, then there will be small differences in atomic forces, which in turn will affect the values of velocities, positions and energies. Furthermore, it is a well known property of atomic ensemble trajectories that small differences between two trajectories tend to grow larger with time. So, after 1000 or more steps, these small differences can become large enough to observe.

### 6.21.2 Subtle XMD Command Interactions

We will describe here some specific cases in XMD simulations where round-off error can affect reproducibility.

In versions of XMD prior to 2.5.22, atoms that passed through the walls of a repeating box would not be wrapped back until a neighbor list was calculated, or until the atom coordinates were written to a file using WRITE RCV, WRITE COR, WRITE STATE or WRITE PARTICLE. Theoretically the simulation should not be affected as long as the atoms' neighbors are correctly identified. However, due to round-off error, results will differ when using wrapped and un-wrapped coordinates. Thus, for instance, if you ran a simulation for 10000 steps without writing a COR file, and then reproduced the same simulation but wrote a COR file every 1000 steps, the trajectory for the two simulations would start to diverge after the first 1000 steps, although it might not be detectable in the output for another 1000 or more additional steps. Version 2.5.22 of XMD wraps the particles after every CMD step, eliminating this source of irreproducibility. Prior to this change, one must be careful to reproduce all the WRITE COR, WRITE RCV, WRITE PARTICLE, etc statements between two runs to obtain identical results.

Another source of irreproducibility is the STATE command. In version of XMD prior to 2.5.22, if for instance you ran a simulation for 10000 steps and saved the state halfway (at step 5000) using the WRITE STATE

command, you would find upon restarting the simulation at step 5000 from the state file that the trajectory diverged from the original. The source of this discrepancy is the neighbor list, because upon restarting XMD with the state file at step 500 the neighbor list is calculated fresh, but in all likelihood the neighbor list at step 5000 in the original simulation was older. While the two neighbor lists give identical neighbors, they may tabulate neighbors in a different order. This difference in order can (and probably will) lead to round-off error as described above. This source of irreproducibility was addressed in XMD version 2.5.22, by recalculating the neighbor list in the original simulation whenever WRITE STATE is called. In this way the neighbor lists for the two runs will be in identical order at step 5000, and hence the trajectories will remain in sync.

However, a related source of irreproducibility still persists in XMD Version 2.5.22 and beyond, but this version does not exist in previous versions. If you compare two otherwise identical runs, but one calls WRITE STATE during the simulation while the other doesn't, then the WRITE STATE command will recalculate the neighbor list. At this point in the trajectory, the neighbor lists will differ, and the atomic trajectories will diverge. The cure for this is to only compare simulations which call WRITE STATE in the exact same sequence.

## 7 Command Summary

In this section is listed the commands to XMD. Some of these commands are shared by previous programs, namely CDCMD (an older version of XMD) and PLOTL (the PC lattice manipulation program used for creating and printing lattices). The command descriptions below adhere to the following syntax:

- The command lines are written in bold.
- Items in UPPERCASE must be entered exactly as shown (although the case entered is irrelevant).
- Items in lower case are to be substituted with the appropriate value.
- Items enclosed in square brackets [ ] are optional.
- Items separated by vertical bars | are choices.
- Choices enclosed by curly brackets { } are required, one must be entered.

```

# (comment)
* (comment)
BOX [SCALE] xbox ybox zbox
BSAVE nskip file
CALC {expression..}
CLAMP [SEL] { temp cstep | OFF }
CMD nstep
CONSTRAIN OFF
CONSTRAIN { LINE | PLANE } xdir ydir zdir xpnt ypnt zpnt
CONSTRAIN CAVITY ELLIPSOID xc yc zc xa ya za spring
CONSTRAIN CAVITY SPHERE xc yc zc radius spring
COR file [ [ run ] step ]
DAMP { OFF | ON [ [ formula [ formula .. ] ] damp ] }
DISP [SEL] { CLEAR | MOVE [scale] | READ n | REFCALC | SCALE scale }
DTIME dtime
DUP ndup xdisp ydisp zdisp
ECHO { ON | OFF }

```

```

ERASE file
ESAVE nskip file
EUNIT
EXTFORCE { CLEAR | [ formula [ formula .. ] ] fx fy fz }
EXTSPRING { CLEAR | [ formula [ formula .. ] ] kx ky kz }
FIX {ON | OFF }
FIX      { ON | OFF }
FILL     ALIGN          x y z
FILL     BOUNDARY BOX   x1 y1 z1 x2 y2 z2
FILL     BOUNDARY SPHERE radius xcenter ycenter zcenter
FILL     BOUNDARY CYLINDER r length xc yc zc xorient yorient zorient
FILL     CELL
        ax ay az
        bx by bz
        cx cy cz
FILL     GO
FILL     MARGIN         margin
FILL     ORIENT         ax ay az  bx by bz  cx cy cz
FILL     PARTICLE n
        type x y z
        ....
ITEMP [SEL] temp [ X | Y | Z ] [ X | Y | Z ] ...
LABEL nlabel
... (up to 8 lines)
MACRO name value
MACROF name format value
MASS mass
MC nstep dtemp
MOVE [ formula [ formula .. ] ] xdisp ydisp zdisp
NRANGE ratio
PARTICLE [ADD] np
.. type x y z
PLOT BOND { OFF | ON lo hi }
PLOT DEVICE { CANON | POSTSCRIPT }
PLOT DISP { ON | OFF | SCALE scale }
PLOT LAYER nlayer [ l1 [ l2 [ l3 .. ] ] ]
PLOT ORIENT { y/z z/y x/z z/x x/y y/x }
PLOT PAGE npage
PLOT SIZE xinch yinch
PLOT SYMBOL [FILL] [ RGB red green blue | HSB hue saturation brightness ]
{ NON | CIR | CRO | TRI | ITR | DIA | AST | SQU | NUM } radius
PLOT WRITE [SEL] [+]file
POSITION [ADD] np
.. type x y z
POSVEL [ADD] np
.. type x y z
POTENTIAL SET { EAM ntype | PAIR ntype | STILL | TERSOFF }
PRESSURE ANDERSEN xmass [ymass [zmass]]
PRESSURE CLAMP bulkmodulus [cstep] (bulk in units of MBAR)
PRESSURE EXTERNAL pressureX [ pressureY [ pressureZ ] ] (Units of MBAR)

```

```

PRESSURE { ISOTROPIC | ORTHORHOMBIC }
PRESSURE OFF
PSTRAIN e dx dy dz nx ny nz
QUENCH nstep [nquench]
RCV file [ [run] step ]
READ file
REFSTEP [CLEAR | COPY | SWAP]
REMOVE
REPEAT nrepeat
... END
REPEAT { COR | RCV } filename [ step1 [ step2 ] ]
... END
ROTATE xaxis yaxis zaxis xcenter ycenter zcenter angle
RUN run
SCALE xscale [yscale [zscale]]
SCREW xburgers yburgers zburgers xorg yorg zorg [ xref yref zerf ]
SEED seed
SELECT [AND | OR | NOT | XOR ]
{ ALL
| BOX x1 y1 z1 x2 y2 z2
| EATOM lo hi
| ELLIPSE xc yc zc xr yr zr
| ILLIST nlist
i1 i2 i3 .....
| INDEX i1 [i2 [iskip]]
| NEAR n [ INDEX index | POINT x y z | SET set ]
| NEAR [ n ] RADIUS r1 r2 [ INDEX index | POINT x y z | SET set ]
| NONE
| PLANE xn yn zn x1 y1 z1 x2 y2 z2
| SET set
| TAG tag
| TYPE type
| VELOCITY [ ABS ] { X | Y | Z | MAG } lo hi
SELECT KEEP { ON | OFF }
SET { ADD | SUB | CLEAR } set
SIZE size
SSAVE nskip file
STATE
STEP step
STRESS THERMAL { ON | OFF }
SURFACE { ON | OFF } { X | Y | Z } [ X | Y | Z | ON | OFF ] ...
TAG tag
TYPE type
TYPELIST [ SEL ] ntype
TYPENAME t1 name1 [ t2 name2 ] [ t3 name3 ] ... [tn namen]
VELOCITY { LINEAR | ANGULAR } dx dy dz magnitude
VERBOSE { ON | OFF }
WAVE phase dx dy dz kx ky kz
WRITE [SEL] COR [+]fname
WRITE [SEL] PDB [+]fname

```

```

WRITE      [SEL] RCV  [+]fname
WRITE      [SEL] XMOL [+]fname
WRITE      [SEL] {ILIST | TYPELIST} [ [+]fname ]
WRITE      [SEL] [FILE [+]fname] { STRESS }
WRITE      [SEL] [FILE [+]fname] [AVG] [MIN] [MAX]
           { FORCE | PARTICLE | POSTVEL | VELOCITY }
WRITE      [SEL] [FILE [+]fname] { DISP | EXTFORCE | EXTSRING }
WRITE      [SEL] [FILE [+]fname] EATOM
WRITE      [SEL] [FILE [+]fname] { TEMP | EKIN | EPOT | ENERGY }
WRITE      [FILE [+]fname] RDF ntable rmin rmax [type1 type2]
WRITE      [FILE [+]fname] { BOX | RUN | STEP }
WRITE      STATE fname

```

## 8 Commands

### \ (backslash)

This is not really a command, but a special character which can be used to extend a command onto multiple lines. When the last character in a command is a " \ ", then the next line appended to the end of the current line. A command can span multiple lines, but it is limited to 1024 characters (typically between 12 and 20 lines).

### # (pound sign)

### \* (asterisk)

Used for commenting instruction files. Lines whose first non-blank character is an asterisk or a pound sign will be ignored.

### BOX [SCALE] xbox ybox zbox

Specifies the box size (in angstroms). The box size is relevant when (1) performing a calculation with repeating boundary conditions (see option SURFACE) and (2) when writing a RCV file (see RCV). If the option scale is chosen then any existing coordinates will be re-scaled (relative to the previous box size) to fit the new box.

### BSAVE nskip file

Causes CMD simulations to save the step number and the x, y and z box sizes every nskip steps in file. This is useful for monitoring the box size when using the PRESSURE command which allows the box size to change.

### CALC {expression..}

CALC is a rich command which provides a simple computational language. For instance if the command

```
CALC  x = 2^(1/3)
```

is given, then a variable x is created that is set equal to 2 to the 1/3 power. Later one could have a command

```
SCALE x
```

which would scale all the particles (and box) by  $2^{1/3}$ . This variable can also be written out with the statement

```
WRITE X
```

(see the WRITE string command below). Furthermore in all commands which require a number, an algebraic expression may be used (provided there are no embedded blanks). Thus you could also have the command

```
SCALE 2^(1/3)
```

The allowed operators are: + - \* / ^ ( ) =

There are also allowed functions which are: sin() cos() tan() exp() log() log10() acos() asin() atan() abs() sqrt() int() rand(). The functions sin(x), cos(x) and tan(x) expect angle x to be in radians. The functions asin(x), acos(x) and atan(x) return the angle in units of radians. The function exp(x) return  $e^x$ . The function log(x) returns log base e of x, log10(x) returns log base 10. The function abs(x) returns the absolute value of x. The function sqrt(x) returns the square root of x. The function int(x) returns the integer portion of x. The function rand(x) returns a random number from the uniform distribution between 0 and x.

There are two built in constants: pi and e.

There can be up to 128 variables set created in one program. See more on CALC under the section on ?? (implementation).

### **CLAMP [SEL] {temp [cstep] | OFF }**

Maintains the temperature for both MC and CMD simulations. For the MC temp is used as the Boltzmann temperature and cstep is ignored.

For the CMD the particles velocities are scaled by  $(T/\text{temp})(1/(2*\text{cstep}))$  at each CMD time step. Here T is the instantaneous system temperature. The application of this factor has the effect of forcing the particle velocities to a value appropriate for the temperature temp. The parameter cstep is used to control the rapidity at which the target temperature is approached. If temp is set to -1, then no temperature clamp is used (this would be an adiabatic system). See section on Temperature Control.

XMD can maintain up to 4 separate particle groups at different temperatures. When the SEL option is used, the temp and cstep or the OFF setting is applied to the selected particles, the CLAMP settings for all other particles remain unchanged. When the CLAMP command is given without the SEL option, then all particles are set the same.

### **CLAMP INFO**

This prints information about the up to 4 separate clamp settings.

### **CMD nstep**

Initiates a CMD simulation for nstep time steps. If ESAVE, BSAVE, SSAVE or TSAVE have been implemented then the corresponding data will be written to disk. Commands which directly affect the course of the CMD simulation are CLAMP, CONSTRAIN, DAMP, DTIME, EXTFORCE, EXTSRING, FIX, MASS, and SURFACE; as well as the particle types and coordinates as determined by PARTICLE, TYPE and STATE; and the interatomic potential as determined by POTSTATE and POTENTIAL commands.

### **CONSTRAIN OFF | {LINE | PLANE} xdir ydir zdir xpnt ypnt zpnt**

Applies (LINE or PLANE) or removes (OFF) a constraint to the selected particles. The LINE and PLANE constraints force a particle to remain on a line or plane, regardless of the forces it experiences. In practice this is done by ignoring any forces which are normal to the line or plane. The values xdir, ydir, zdir specify either the line direction or the plane normal. The values xpnt, ypnt, zpnt specify a coordinate within the line or plane (it can be any one of many coordinates) - this is necessary to locate the line or plane somewhere in space.

### **CONSTRAIN CAVITY ELLIPSOID spring xcenter ycenter zcenter xaxis yaxis zaxis**

## CONSTRAIN CAVITY SPHERE **xcenter ycenter zcenter radius**

This command places an ellipsoidal " cavity " in the simulation. The cavity walls reflect particles. When a particle passes into a wall it experiences a spring force pushing back out. The force is equal to

$$F = 1/2 * \text{spring} * dq^2$$

where  $dq$  is the normal distance from wall to the particle, and  $\text{spring}$  is the spring constant. Once the particle passes back out of the wall, it no longer feels the force. Only one cavity can be present in a simulation. All particles will be affected by the wall. ( $xcenter,ycenter,zcenter$ ) specifies the center of the cavity. For a spherical cavity, radius is the radius. For an ellipsoidal cavity, ( $xaxis,yaxis,zaxis$ ) are the  $x,y,z$  half-axis, analogous to spherical radii. They measure the distance from the ellipsoid center to the ellipsoid wall in the  $x,y,z$  directions. Please note the following things about the CONstrain CAVITY command.

- (1) Only one cavity can be present in a simulation.
- (2) The cavity is affected by the SCALE command. If the SCALE command scales non-uniformly (difference scale in  $x$ ,  $y$  and  $z$  directions) then the an initially spherical cavity will become an ellipsoid, the same as if it had been made with the CONstrain CAVITY ELLIPSOID command above.
- (3) The cavity does know about repeating boundary conditions, so it can overlap the box.
- (4) The ellipsoid cavity can only be orientated with its axis along the  $x$ ,  $y$  and  $z$  directions.
- (5) A spring constant that is too large will cause instability in the integration algorithm, the same kind of instability as having a time step too large. As a rough guide, make sure that spring satisfies the following relation,

$$k = 4 \pi^2 m / t^2$$

where  $k$  is the spring constant,  $m$  is the mass in grams of the lightest particle in the simulation, and  $t$  is the time step size in seconds.

## COR file [ **run** ] **step**]

Reads a COR step from file. If **step** is specified then the data for that step is read; otherwise the first step in the file is read. If **step** is not in the file then an error message is given and the program continues. If **run** is specified (which implies that **step** is also specified) then only the data that belongs to the **run** and **step** are read. If **step** is specified without **run**, and two COR steps have the same step number, then only the first is read. When both **run** and **step** are specified and no matching COR step is found, then an error message is given and the program continues.

The COR command differs from the RCV command mainly because it reads in particle types as well as coordinates.

## DAMP { **OFF** | **ON** [ **damp** ] }

Each atom has its own individual velocity damping coefficient (see section 6). If **ON** is specified along with **damp** then specified damping coefficient is applied to the selected atoms, and damping will be performed for each dynamics step. If **damp** is not specified, the atomic damping coefficients will not be changed, and damping will be performed with each dynamics step. If **OFF** is specified, then damping will not be performed, but the previous set of damping coefficients will be remembered, and can be applied by a subsequent DAMP ON command.

The expression for **damp** can contain the variables  $x$ ,  $y$  and  $z$ , which will equal the coordinates of each atom. Thus you can apply a damping term that is a function of individual atom's position. The optional formula expression can initialize one or more variables for subsequent use in the expressions **damp**. See the MOVE command for an example.

**DISP [SEL] { CLEAR | MOVE [scale] | READ n | REFCALC | SCALE scale }**

This command manipulates the displacements. Each particle can have a displacement associated with it. These displacements can be used to move the particles. Displacements can be written out using the WRITE DISP command (see the WRITE command). You cannot write out the displacements until you use the DISP command to create them.

Note however, that the results of the PLOT DISP command is not affected by this command. For the PLOT DISP command the displacements are calculated independently.

### **CLEAR**

CLEAR resets displacements to zero. If SEL is specified, only those selected particles have their displacements reset. Without the SEL option, all displacements are reset (and the memory required for displacements is released).

### **MOVE**

MOVE option adds the displacements to the current particles. The SEL option causes the selected particles to be moved.

### **READ**

READ n reads the value of displacements from the input. The displacements should follow on the next line, with enough values for each coordinate of all the particles (or the selected particles if SEL is set).

### **REFCALC**

REFCALC set the displacement to the difference between the current particles and the reference particles (current - reference). If SEL is set, only those displacements are affected.

### **SCALE**

SCALE scale multiplies all the displacements (or the selected displacements if SEL is set) by the amount scale

### **DTIME dtime**

This sets the value of the CMD time step in units of seconds. The optimum value of the time step is small enough to result in a stable integration of forces but large enough to provide an efficient use of computer time. See section on techniques for determining the optimum value of dtime.

### **DUP ndup xdisp ydisp zdisp**

The DUP command is used to create particles. Its need has been superceded by the newer FILL command.

DUP duplicates the selected particles (see SELECT below) ndup times and displaces each duplicate in the x, y and z directions by xdisp, ydisp and zdisp; relative to the previous duplicate. For example the command

```
DUP 3 1 0 0
```

will duplicate the selected particles three times. The first, second and third set are offset from the original particles by (1 0 0), (2 0 0), and (3 0 0) respectively. At the end of the command all previously selected particles and the newly created duplicates are selected. In the example at the end of this document the DUP command is used to create a BCC lattice.

### **ECHO { ON | OFF }**

By default every command read by XMD is echoed to the output. This echoing can be controlled with the ECHO command. ECHO OFF will stop command echoing. ECHO ON is the default.

## ERASE file

Erases the file name `file`. This is useful if you want to add simulations steps to the end of a file but you want to first clear the file as shown in this example,

```
erase aucrack.cor
repeat 10
  cmd 1000
  write cor +aucrack.cor
end
```

## ESAVE nskip file

Causes MC and CMD simulations to save the energy every `nskip` steps in file. For MC simulation two numbers are saved at each relevant step; the step number and the total potential energy (in that order). For the CMD case four numbers are saved: the step number, the total energy (the potential plus the kinetic energy), the potential energy, and the kinetic energy. Note that the potential energy includes the energies due to external forces, as specified with the `EXTFORCE` and `EXTSPRING` commands.

## EUNIT [ **ERG** | **EV** | **JOULE** | **K** | **uname uvalue** ]

Sets the units for energy, either ergs, electron volts, joules, Kelvin or a user specified name and unit. `uvalue` is the ratio of the unit `uname` to ergs. All energies printed by the program (either via the `ESAVE` command or the `WRITE ENERGY` command) will be in the units specified. The forces read or written by the `EXTFORCE` command or the `WRITE FORCE` command will be in the specified units per centimeter. The spring vectors read by the `EXTSPRING` command will be in specified units per centimeter squared. If no value is specified on the `EUNIT` command line then the current value of `uname` and `uvalue` will be printed. The program default value for `EUNIT` is Kelvin (or put another way, the default value is the inverse of Boltzman's constant).

## EXTFORCE { **CLEAR** | [ **formula** [ **formula ..** ] ] **fx fy fz** }

Places an external force on selected atoms equal to `fx`, `fy`, `fz`. If the `CLEAR` option is specified then forces on all atoms are reset to zero ("all the atoms" means not only the selected atoms). The unit for the forces is the energy unit specified by `EUNIT` (the default is ergs) divided by centimeters.

The `EXTFORCE` contributes to the potential (and total) energy written by the `ESAVE` command, but not to the energy written by the `WRITE ENERGY` command. The energy for an atom under an external force is

$$E = - F (r - r_0)$$

where `F` is the applied force, `r` is the atom position, and `r0` is the atom position at the time the `EXTFORCE` command was issued. Thus, for instance, this energy is zero if an atom is stationary. The external forces created with this command will be stored in the state file (see the `WRITE STATE` command).

The expressions for `fx`, `fy` and `fz` can contain the variables `x`, `y` and `z`, which will equal the coordinates of each atom moved. Thus you can apply an external force that is a function of an individual atom's position. The optional formula expression can initialize one or more variables for subsequent use in the expressions `fx`, `fy` and `fz`. See the `MOVE` command for an example.

## EXTSPRING { **CLEAR** | [ **formula** [ **formula ..** ] ] **kx ky kz** }

Places an external spring on selected atoms. If the `CLEAR` option is specified then all of the external springs are removed ("all the external springs" means not only those on selected atoms). The direction of the spring's force is parallel to the spring vector (`kx`, `ky`, `kz`). Also, only the atom's displacement parallel to the vector determines the magnitude of the force. The actual equation used is

where  $F$  is the force due to the spring,  $r$  is the atom position,  $r_0$  is the atom position at the time the EXTSPRING command was issued,  $k$  is the spring vector and is a unit vector parallel to  $(k_x, k_y, k_z)$ . The energy due to external springs is included in the potential (and total) energy written by the ESAVE command, but not by the WRITE ENERGY command. The units of the spring vector are the energy units specified by EUNIT (the default is ergs) divided by centimeters squared. The external springs created with this command will be stored in the state file (see the WRITE STATE command). The expressions for  $k_x$ ,  $k_y$  and  $k_z$  can contain the variables  $x$ ,  $y$  and  $z$ , which will equal the coordinates of each atom moved. Thus you can apply an external spring that is a function of an individual atom's position. The optional formula expression can initialize one or more variables for subsequent use in the expressions  $k_x$ ,  $k_y$  and  $k_z$ . See the MOVE command for an example.

### **FILL ...**

The FILL family of commands is used for creating lattices. It can create a regular lattice in any orientation and with any basis vectors and basis atoms. Please see the ?? (section on the FILL command).

### **FIX { ON | OFF }**

For CMD simulations FIX causes the selected particles to be either fixed (ON option) or free (OFF option). If FIX is not specified then the particles are free. See the section on ?? (fixed atoms) above.

### **ITEMP SEL temp [X|Y|Z] [X|Y|Z] ..**

Assigns random velocities appropriate to temp to all the particles. The assigned velocities are chosen at random but conform to a Boltzmann distribution. Normally random values are assigned to all components of the velocity,  $x$ ,  $y$  and  $z$ . However when performing a two or one dimensional simulation, you will want to assign velocities to specific components only. By specifying specific directions such as X, Y or Z you can assign values to those components only. The remaining components will be set to zero. As an example, if you are running a two dimensional simulation in the  $xy$  plane, the command

```
ITEMP 100 X Y
```

will assign velocities in the  $x$  and  $y$  directions, and set the  $z$  component to zero. If in addition the particles all have the same  $z$  coordinates, then the motion will be confined to the  $xy$  plane. See also the VELOCITY command.

### **LABEL nlabel**

This command is followed by  $nlabel$  lines of text,  $nlabel$  must not exceed 8. The text is used to add notes to simulations files, and is stored in COR, RCV and state files, as well as written out on plots.

### **MACRO name string**

This command assigns a string to the name. For example,

```
macro fname femelt
esave 10 $fname.e
bsave 10 $fname.b
ssave 10 $fname.s
```

tells XMD to replace "\$fname" with "femelt", so that the resulting file names will be femelt.e, femelt.b and femelt.s.

Macros are replaced upon input before any other processing is done. One consequence is that macro substitution is done before the commands are printed to output, so you will see the substituted macros in the output.

You can have blanks in the macro value, the macro value is the string comprised of the first non-blank character after the macro name until the end of the line. So you can have the following macro

```
macro sbox select box
  $sbox 0 0 0 5 10 10
  $sbox 5 0 0 10 10 10
```

This will expand to

```
macro sbox select box
select box 0 0 0 5 10 10
select box 5 0 0 10 10 10
```

The macro name can be any sequence of numbers and characters, and can even start with a number, such as

```
macro 1st out1
macro 2nd out2
```

Sometimes you need to tell XMD where the macro name ends, for example

```
macro fname cu_fit
select type 1
#
# We want to write COR file to "cu_fit1.cor"
#
# This will write to ".cor"
#
write cor $fname1.cor
#
# This will write to "cu_fit1.cor"
#
write cor $(fname)1.cor
#
# This will write to "cu_fit.cor"
#
select all
write cor $fname.cor
```

In the first `write cor` command, XMD thinks you want the macro named "fname1", but there is not such macro, so this macro name is removed and nothing is put in its place. The second `write cor` statement places "fname" in parenthesis, which indicates that this is the macro name. The third `write cor` statement does not need the parenthesis because XMD knows that macro names do not include periods (.), so the macro name must end there.

There are 9 special built-in macros, \$1 through \$dollar;9, these represent command lines parameters. See the section on the ?? (implementation of macros).

### **MACROF name format variable**

This command is similar to `MACRO`, but is assigns a `CALC` variable to a macro. For instance, if you want to write an individual `COR` file for specific steps,

```
repeat 5
  cmd 40
  macrof fname simul%03d nstep
  write cor $fname.cor
end
```

This would write the files simul040.cor, simul080.cor, simul120.cor, .. simul200.cor. The format parameter follows the standard of the C language printf() commands and can be either %d %i %u %x %f %e . Please see information for the C language printf() command. Under some Unix installations, you can learn more about printf with the command

```
man printf
```

### **MASS mass**

Assigns mass to all selected particles (units are atomic mass units).

### **MC nstep temp**

Perform a MC simulation for nstep steps and at temperature temp. The course of Monte Carlo simulations is affected by the commands CLAMP and SIZE. The command ESAVE affects the output from these simulations.

### **MOVE [ formula [formula .. ] ] xdisp ydisp zdisp**

Moves the selected particles by xdisp, ydisp and zdisp. The expressions for xdisp, ydisp and zdisp can contain the variables x, y and z, which will equal the coordinates of each atom moved. Thus you can apply a displacement that is a function of individual atoms. The optional formula expression can initialize one or more variables for subsequent use in the expressions xdisp, ydisp and zdisp. As an example, suppose we want to use the MOVE command to displace atoms away from the point (x0,y0,z0) by an amount  $\exp(-r/5)$ , where r is the distance from point (x0,y0,z0). We can use the following code to do this,

```
CALC  x0=5.0
CALC  y0=5.0
CALC  z0=5.0
MOVE  dx=x-x0 dy=y-y0 dz=z-z0 r=sqrt(dx^2 + dy^2 + dz^2) /
      dx*exp(-r/5) dy*exp(-r/5) dz*exp(-r/5)
```

The first three CALC statements initialize values for x0, y0 and z0, which will serve as the center for the atomic displacements. The MOVE command initializes four variables, dx, dy, dz and r, which are then used to specify the displacements. Note that the last three expressions are the actual displacements. Also note the use of the backslash character, " / ". When this character is the last character on a line, the command is continued on the next line. The backslash character may be used for any number of lines, as long as the resulting command does not exceed 1024 characters. Typically this is about 12 to 20 lines.

### **NRANGE ratio**

Set's the range needed by the neighbor search algorithm (see page 5). This neighbor range is longer than the interatomic potential range. The neighbor list includes all particle pairs whose separation is ratio times the maximum cutoff distance for the interatomic potentials (ionic and electron density for various species). By default ratio is 1.1. For a given potential the optimum value (in terms of computer time used) may vary.

### **PARTICLE [ADD] np**

See the ?? (POSITION command).

### **PLOT BOND { OFF | ON lo hi }**

Plots "bonds" between atoms. All atoms between lo and hi angstroms apart will be connected by lines.

### **PLOT DEVICE { CANON | POSTSCRIPT }**

Specifies the printer to be used for plot output. The default depends on the installation.

**PLOT DISP { ON | OFF | SCALE scale }**

Switches on/off displacement plotting. If REFSTEP has been specified previously in a program, then when the PLOT statement is given, atomic displacements will be plotted (if switched on). scale specifies a magnification scale for the displacement lines.

**PLOT LAYER nlayer [layer1 [layer2 ..... ]]**

Chooses the layers for particle plot. nlayer is the number of layers into which the box is divided (in the direction specified by PLOT ORIENT). If parameters layer1, layer2, etc. are specified, then only these layers are plotted. Otherwise all layers are plotted.

**PLOT ORIENT { X/Y | Y/X | Y/Z | Z/Y | X/Z | Z/X }**

Chooses the orientation of the plot. The default is y/x (which means y in the vertical direction and x in the horizontal). Orientation y/x means that the y direction is plotted vertically, and the x direction is plotted horizontally. The z direction is used to divide layers (see PLOT LAYER above).

**PLOT PAGE page**

Number of pages used for plot. Default is one.

**PLOT SYMBOL [FILL] [ RGB red green blue | HSB hue saturation brightness ] { AST | CIR | CRO | DIA**

Assigns a symbol to all of the selected particles. The FILL option causes the symbol to print as a solid block (valid for the circles, diamonds, triangles and squares, ignored otherwise). The RGB and HSB options assign a color to the atom (the default color is black). See the section ?? (Using Color in Plots) to learn about the meaning of the color parameters. radius specifies the distance from the center of the symbol to the furthest point. When no SYMBOL command is given then SYMBOL CIR 0.25 is assumed. A list of symbol options follows:

```
AST asterisk
CIR circle
CRO cross
DIA diamond
ITR inverted triangle (point downward)
NON no symbol
NUM index number for particle
SQU square
TRI triangle
```

The option NUM prints the number of the particle in place of a symbol. This is useful for finding the location of a specific particle. The NON option prints no symbol. This is useful when you want to plot bonds, but don't want to have a symbol.

**PLOT WRITE [SEL] [[+] file ]**

Places printer commands (either Canon LBP or PostScript commands depending on DEVICE command) describing the lattice plot in either the output file (specified on the command line) or in file. If a + is prefixed to the file name then the output is appended file. If the SEL option is specified then only the selected particles are plotted. Before PLOT is called symbols must first be assigned to the atoms (see the SYMBOL command). Also one may further embellish the plot by specifying DISP, PAGE or LAYER.

**POSITION [ADD] np**

Reads in a set of particle types and coordinates. With the ADD option, previous selections are preserved and the new particles are added to the selection list. Without the ADD option, all previous selections

are forgotten and all the new particles are selected. The parameter `np` is the number of particles for the program to read. For each particle enter its type and coordinates one a separate line after the `POSITION` statement as illustrated below (you can substitute `PARTICLE` for `POSITION`).

```
POSITION 4
1 1/4 1/4 1/4
2 1/4 3/4 3/4
2 3/4 1/4 3/4
2 3/4 3/4 1/4
```

### **POSVEL [ADD] np**

Reads in a set of particle types, coordinates and velocities. Coordinate units are angstroms, velocity units are cm/sec. With the `ADD` option, previous selections are preserved and the new particles are added to the selection list. Without the `ADD` option, all previous selections are forgotten and all the new particles are selected. The parameter `np` is the number of particles for the program to read. For each particle enter its type and coordinates one a separate line after the `POSVEL` statement as illustrated below.

```
POSVEL 4
1 1/4 1/4 1/4 0 0 0
2 1/4 3/4 3/4 0 0 0
2 3/4 1/4 3/4 20.0 0 0
2 3/4 3/4 1/4 -20.0 0 0
```

### **POTENTIAL SET { EAM ntype | PAIR ntype | STILL | TERSOFF }**

Sets the potential type. `EAM` selects the Embedded Atom Method, and `ntype` specifies the number of elements (or atom types) used in the simulation. Similarly for `PAIR`, it is used for pair potentials. Subsequent commands must specify the exact form of the potential functions. See the section on Interatomic Potentials.

The `STILL` option selects the Stillinger-Weber potential for Si, the `TERSOFF` option selects the Tersoff potential for Carbon/Silicon.

### **PSTRAIN e dx dy dz nx ny nz**

Performs an invariant plane strain on the selected particles. The new particle coordinates are given by

$$\mathbf{r}' = \mathbf{r} + \mathbf{e} \mathbf{d} (\mathbf{n} \cdot \mathbf{r})$$

where  $\mathbf{d}$  is the vector  $(dx,dy,dz)$  and  $\mathbf{n}$  is  $(nx,ny,nz)$ .  $\mathbf{r}$  is the original particle coordinate measured from the origin and  $\mathbf{r}'$  is the new particle coordinate.

### **PRESSURE ANDERSEN xmass [ ymass [ zmass ] ]**

### **PRESSURE CLAMP bulkmodulus [ cstep ]**

### **PRESSURE OFF**

### **PRESSURE EXTERNAL pressureX [ pressure Y [ pressure Z ] ]**

### **PRESSURE { ISOTROPIC | ORTHORHOMBIC }**

These commands control XMD's constant pressure algorithm (see section ?? (Constant Pressure)). By default the constant pressure algorithm is off, consequently the box size if fixed.

The command `PRESSURE ANDERSEN` turns on Andersen's constant pressure algorithm. With this algorithm a one or more masses must be specified. If only `xmass` is specified, then the volume mass is the same in all three directions. If only `xmass` and `ymass` is specified, then `zmass` is taken equal to `ymass`.

The command **PRESSURE CLAMP** turns on the pressure "clamp". It requires an estimate of the system bulk modulus in units of Mbars. An optional value `cstep` controls how quickly the box size fluctuates in response to internal stress.

The **PRESSURE OFF** command switches off both the **PRESSURE ANDERSEN** and **PRESSURE CLAMP** algorithms (note that only one algorithm can be used at one time).

The **PRESSURE EXTERNAL** command sets the external pressure of the system. The external pressure will have no effect until the **PRESSURE ANDERSEN** or **PRESSURE CLAMP** algorithms are in use. By default the external pressure is zero. You can also apply different pressures in the X, Y and Z directions when using **PRESSURE CLAMP** (you will get an error using **PRESSURE ANDERSEN** on your first **CMD** or **QUENCH** command). If only one pressure (`pressureX`) is specified, then all pressures are equal. If two pressures are specified (`pressureX pressureY`) then `pressureZ` is set equal to `pressureY`. You can specify three pressures to get independent X, Y and Z pressures.

The **PRESSURE { ISOTROPIC | ORTHORHOMBIC }** command controls the symmetry of the repeating box. With **PRESSURE ISOTROPIC** (the default) the box contracts or expands uniformly, so a cubic lattice will remain a cubic lattice, and a tetragonal lattice will maintain its  $a/c$  ratio, etc. **PRESSURE ORTHORHOMBIC** allows the x, y and z directions to change size independently, which can result in an initialized cubic lattice losing its cubic symmetry, and a tetragonal lattice losing its  $a/c$  ratio, and even its  $a/b$  ratio.

### **QUENCH nstep [nquench]**

Similar to the **CMD** command. Performs **CMD** for `nstep` number of steps, but applies one of two quench algorithms to the resulting velocities, as a way to relax atomic configurations. If `nquench` is 1 (the default), then whenever the potential energy rises relative to the preceding time step, all the velocities are set to zero. If `nquench` is 2, then when any individual particle has  $f \cdot v < 0$  where  $f$  and  $v$  are the force and velocity for that particle, then the velocity for only that particle (not the entire system) is set to zero. This is equivalent to setting velocity to zero whenever the energy of an individual atom increases.

### **RCV [file [ [run] step]]**

Reads an **RCV** step. If `file` is specified then that file is read; otherwise the lines following the **RCV** command are read. If `step` is specified then the data for that step is read; otherwise the first step in the file is read. If `step` is not in the file then an error message is given and the program continues. If `run` is specified (which implies that `step` is also specified) then only the data that belongs to the `run` and `step` are read. If `step` is specified without `run`, and two **RCV** steps have the same step number, then only the first is read. When both `run` and `step` are specified and no matching **RCV** step is found, then an error message is given and the program continues. The **RCV** command reads in particle coordinates, as well as a Radial Distribution Function (RDF) table. The **RCV** file does not store atom types, so the **RCV** command sets the atom types all to 1.

### **READ file**

Opens the file named `file` and reads the commands as if they were in the current command file. At the end of the file control returns to calling file.

### **REFSTEP [CLEAR | COPY | SWAP]**

When used without options the current particles, displacements, types and RDF are copied to the reference step. The **CLEAR** option erases all of the information previously stored in the reference step. The **SWAP** option switches the reference and current step. The **COPY** option copies the **refstep** state to the current state, while retaining the contents of the reference step.

### **REMOVE**

Removes the selected particles from the list of current particles and reference particles.

**REPEAT** { nrepeat | { COR | RCV } file [ step1 [ step2 ] ] }

The first form of this command repeats nrepeat times each command between a REPEAT statement and an END statement, for example

```
REPEAT 10
  CMD 100
  WRITE COR +crack.cor
END
```

The second form reads a COR or RCV file, and for each step in the file executes the commands between the REPEAT and END statements. Note that this will change the current particle state, and the particle state after this command runs will be the last step in the COR file. The name of the file is file, and the beginning step1 and ending step2 step number may be specified. This is the step number assigned when the COR file was written. For example the COR file may have the steps 100, 200, 300, etc. The step numbers do not indicate the sequence of steps. For example, a value of 5 means step 5, not the fifth step - it could even be the first step. If the ending step is omitted then all steps are read to the end of the file. If the beginning step is omitted, then the command starts with the first step in the file.

**ROTATE** xaxis yaxis zaxis xcenter ycenter zcenter angle

This command rotates the selected particles about the specified axis by the given angle (in degrees). The axis direction is given by the vector xaxis yaxis zaxis (the magnitude of the vector is ignored). The coordinates xcenter ycenter zcenter locate the axis in space by specifying a point which the axis must pass through. The rotation follows the "right hand rule", meaning that when the thumb of one's right hand points in the direction of the axis the remaining fingers, when curled, point in the direction of a positive rotation.

**RUN** run

Sets the run number to run. Used by PLOT, STATE and WRITE RCV (see section on ?? (run identification)).

**SCALE** xscale [yscale [zscale]]

Scales the box and particles by xscale, yscale and zscale respectively in the x, y and z directions. If zscale is omitted it is set to the yscale, similarly if yscale is omitted it is set to xscale.

**SCREW** xburgers yburgers zburgers xorg yorg zorg [ xref yref zref ]

Apply displacements to selected particles to form a screw dislocation in a previously perfect lattice. xburgers, yburgers, zburgers, are the components of the burgers vector. xorg, yorg, zorg, specify a point through which the screw line passes. The optional xref, yref, zref, specify a reference direction non-parallel to the burgers vector (which is also the direction of the screw dislocation line). This reference together with the screw dislocation line defines the reference plane. All atoms within this reference plane undergo zero displacement. If no reference direction is specified, an arbitrary one is used. The displacement of each atom is given by

$$d_i = b * \sin(a_i)$$

where  $d_i$  is the atom displacement,  $a_i$  is the angle made with the reference plane by the line joining the atom  $i$  and the origin point, and  $b$  is the burgers vector.

**SEED** seed

Set the random number generator to seed. The random number generator is used by ITEMP as well as for random moves by MC command.

## SELECT

The full form of the SELECT command with all possible options is

```
SELECT [AND | OR | NOT | XOR ]
{ ALL
| BOX x1 y1 z1 x2 y2 z2
| EATOM lo hi
| ELLIPSE xc yc zc xr yr zr
| ILIST nlist i1 i2 i3 ...
| INDEX i1 [i2 [iskip]]
| LAYER {x|y|z} nlayer l1 [l2 [l3 [..] ] ]
| NEAR n [ INDEX index | POINT x y z | SET set ]
| NEAR [ n ] RADIUS r1 r2 [ INDEX index | POINT x y z | SET set ]
| PLANE xn yn zn x1 y1 z1 x2 y2 z2
| SET set
| TAG tag
| TYPE type
| VELOCITY [ ABS ] { X | Y | Z | MAG } lo hi }
```

This command selects a subset of particles according to the given criteria. AND selects only those particles that have been both previously selected or fit the current criteria. OR selects particles that belong to either group. NOT selects those particles which do not fit the given criteria. XOR selects those particles have been previously selected or which fit the current criteria, but not both.

### ALL

selects all particles.

### BOX x1 y1 z1 x2 y2 z2

selects all particles within a box whose opposite corners are (x1,y1,z1) and (x2,y2,z2).

### EATOM lo hi

selects all atoms whose potential energy falls between lo and hi, where the units are those specified by the EUNIT command (default is ergs). Important: This command uses the energies which last calculated for the atoms when the commands WRITE ENERGY, WRITE EATOM, CMD, QUENCH or MC where executed. This command will NOT re-calculate the energies.

### ELLIPSE xc yc zc xr yr zr

selects those particles within an ellipse centered on (xc,yc,zc) and whose axes are (xr,yr,zr).

### ILIST nlist / i1 i2 i3 ...

selects all particles whose indices are listed. nlist is the number of indices in the list. The list of indices is placed on the subsequent lines.

### INDEX i1 [i2 [iskip]]

selects particles numbered from i1 to i2. If i2 is omitted then only particle i1 is selected. If iskip is included then only every iskip'th particle starting at i1 and not exceeding i2 is selected.

### LAYER {x|y|z} nlayer l1 [l2 [l3 [..] ] ]

divides the simulation into nlayer layers in the x, y, or z direction and selects those atoms in layer(s) l1, l2, ... If there are repeating boundary conditions then it divides the box into nlayers, otherwise it divides the space between the maximum and minimum particles in the selected direction.

### NEAR n [ INDEX index | POINT x y z | SET set ]

selects n nearest particles to the a atom or point. If no options are specified, then it selects the n nearest atoms to the first of the currently selected atoms (the 'first' atom is the one with the lowest index). With the SET option, it selects the n nearest atoms to the first atom in the set (again, the 'first' atom is the one with the lowest index). If INDEX option is used, then the it

selects the n nearest atoms to the particle with that index. In these three cases, the atom itself is not selected, only the neighbors are selected. If the POINT option is used, it selects the n nearest atoms to the coordinate x,y,z.

**NEAR [ n ] RADIUS r1 r2 [ INDEX index | POINT x y z | SET set ]**

similar to the previous, this selects the nearest atoms to the 'first' atom from either the currently selected atoms or the given SET, or the neighbors to the atom with the given INDEX, or the POINT at x,y,z. However, it only selects that atoms between r1,r2 angstroms from the original atom or point. If the optional n is given, no more than the nearest n atoms between r1,r2 angstroms are selected.

**PLANE xn yn zn x1 y1 z1 x2 y2 z2**

selects particles between two parallel planes. The planes are normal to the vector (xn,yn,zn). The two planes pass through the points (x1,y1,z1) and (x2,y2,z2) respectively.

**SET set**

selects all particles in specified set (set value ranges from 1 to 8).

**TAG tag**

selects all particles with specified tag value (values range from 0 to 255).

**TYPE type**

chooses all particles of type type.

**VELOCITY [ ABS ] { X | Y | Z | MAG } lo hi**

chooses particles with velocities between lo and hi. One can use the X, Y or Z components of the velocity, in which case one can also use the ABS option to query the absolute value of the velocity. The MAG option compares the magnitude of the velocity, in which case the ABS option is irrelevant. The units for X, Y, Z and MAG are centimeters per second.

**SELECT KEEP { ON | OFF }**

Determines if SELECT, SET and TAG info is forgotten (OFF) or remembered (ON) when coordinates are read with the COR or RCV commands. The default is OFF. For further discussion see section ?? (Particle Selection).

**SET { ADD | SUB | CLEAR } set**

Either adds or subtracts (ADD or SUB) selected particles from set number set (which can range between 1 and 12). The CLEAR option resets the specified set to be empty.

**SIZE size**

Sets the size of a jump used by MC. Units are angstroms.

**SSAVE nskip file**

Causes the average internal scaled stress per particle (not the true stress) to be written to file every nskip steps. The data file is a text file with 7 columns. The first column is the step number, the following six columns are the Voight scaled stresses per atom in units of ergs. To obtain the true Voight stresses from the scale stresses, you must divide by the volume per atom. The program does not do this automatically, because when there are free surfaces and/or lattice voids the volume per atom is not well defined. SSAVE works when using EAM and PAIR potentials, but does not currently work for the Tersoff potential.

**STATE file**

Reads current state of CMD and MC simulation in the file file (where the file was created with the WRITE STATE command).

**STEP step**

Sets the step number to step. Used by PLOT, STATE, ESAVE and WRITE RCV (see WRITE below, page 23).

**STRESS THERMAL { ON | OFF }**

Most (all?) thermodynamics texts include the a term in the atomic velocities (the 'virial') when calculating the internal stress. By default XMD does not include this term, but includes only the internal stress which arises directly from the force. If you wish to include the virial term, then use the command

```
STRESS THERMAL ON
```

This is a controversial issue, I think it can be resolved by considering the internal stress and thermal expansion of a perfectly harmonic solid. Such a solid DOES NOT expand with increasing temperature, but it would expand if the virial term were correct. Anyway, there's probably a paper in this somewhere :)

**SURFACE { ON | OFF } { X | Y | Z } [X|Y|Z|ON|OFF] ...**

Determines whether a free surface or repeating boundary condition is used with the MC and CMD simulations (see page 7). SURFACE ON X Y Z creates a free surface in the x, y and z directions. SURFACE OFF X Y ON Z creates a free surface in the z direction and repeating boundary conditions in the x and y directions. By default repeating boundary conditions exist in all three directions. When there is a free surface the box dimension is ignored for that dimension.

**TAG tag**

Sets tag value for selected atoms. Every atom can have a tag value between 0 and 255. The default value is 0. Tag values are used similar to sets, but where a single atom can belong to any combination of sets, a single atom can only have one tag value. Thus if atoms are grouped by their tag values, each atom can belong to only one group. You can select atoms according to their tag value with the SELECT command.

**TYPE type**

Sets type value for selected atoms. The properties of each atom type is controlled by the potential model.

**TYPELIST [SEL] ntype**

Reads ntype number of types from input lines following command. These types are assigned existing particles (or selected particles). ntype can be either less or more than the number of particles (or selected particles). If less, then only the first ntype particles are changes. If more, then the extra types read in are ignored.

**TYPENAME t1 name1 [ t2 name2 ] [ t3 name3 ] ... [tn namen]**

Associates the type number t1 (or t2, etc.) with the element name name1 (or name2, etc.). This is typically used to assign element abbreviations (such as FE, NA) to be used in place of type numbers for certain output formats (currently WRITE XMOL and WRITE PDB).

**VELOCITY { LINEAR | ANGULAR} dx dy dz magnitude**

Sets the total linear or angular velocity for the selected particles. For the linear case, (dx, dy, dz) determines the direction of the velocity, and magnitude is the magnitude of the velocity (in centimeters per second).

For the angular case, (dx,dy,dz) is the direction of the rotational axis, and magnitude is the angular velocity (in radians per second). The axis is positioned through the center of mass of the selected

particles. The direction of positive angular velocity follows the right hand rule: if the thumb of the right hand is pointed along the axis, then the curled fingers of that hand indicate the direction of positive rotation.

## VERBOSE { ON | OFF }

Turns output comments on or off. At present, only the WRITE ATOM command produces comments.

## WAVE phase dx dy dz kx ky kz

Displaces the selected particles by a wave. The new position of each particle is

$$\mathbf{r}' = \mathbf{r} + \mathbf{d} \sin(\mathbf{k} \cdot \mathbf{r})$$

where  $\mathbf{d}$  is the wave vector (dx,dy,dz) and  $\mathbf{k}$  is (kx,ky,kz).  $\mathbf{r}$  is the original particle coordinate measured from the origin and  $\mathbf{r}'$  is the new particle coordinate.

## 9 Fill Command

The FILL command is used to create lattices. Essentially, you specify a region which will contain the lattice, and you specify the lattice geometry and orientation. The program then generates atomic coordinates that match your specifications.

The lattice is generated according to FILL PARTICLE and FILL CELL commands that you provide. The FILL PARTICLE command specifies a set of atoms that are placed repeatedly throughout the region. The FILL CELL command specifies the three vectors that describe how the set of atoms are placed in the region.

### 9.1 FILL Example 1: Filling a Sphere

Here is an example where we create a sphere fill with a B2 lattice.

```
fill boundary sphere 4.0 2.0 2.0 2.0
fill particle 2
  1 0.0 0.0 0.0
  2 0.5 0.5 0.5
fill align 2 2 2
fill cell
  1 0 0
  0 1 0
  0 0 1
fill orient 1 1 0 1 -1 0 0 0 1
fill go
```

The sphere is centered on point (2,2,2), and has a radius of 4 angstroms, as specified by the FILL BOUNDARY command. The FILL PARTICLE command gives the coordinates and types for the atoms of the repeating cell. There are two atoms for the repeating cell of a B2 lattice (a B2 lattice has the atom positions of a bcc lattice, but with two atom species). The FILL CELL command says that the cell is repeating by the three displacements (1,0,0), (0,1,0) and (0,0,1). The FILL ORIENT command says that the particles and displacements should be re-oriented so that the lattice directions that were previously the (100), (010) and (001) are now (110), (1-10) and (001). The FILL ALIGN command says that the origin of the repeating cell (where according to the FILL PARTICLE command the first atom is positioned) will lie on the point (2,2,2) in the lattice. In this example therefore, an atom of type 1 will be positioned at the center of the filled sphere. The FILL GO command tells XMD that we've entered all the FILL information and that it is time

to create the particles. Default values are used for whatever information has not been entered. Sometimes more than one FILL GO command is used. If any information is specified earlier commands becomes the default for later commands.

## 9.2 FILL Example 2: Creating an Diamond Cubic Lattice

In this example, we fill an entire simulation box with a diamond cubic lattice. The diamond cubic lattice is made by duplicating a pair of atoms using the repeating cell vectors for an FCC lattice. Since we not used a FILL BOUNDARY command, FILL BOUNDARY BOX is assumed with the system box as the region to fill.

```
box 8 8 8
fill particle 2
1 0 0 0
2 1/4 1/4 1/4
fill cell
  0 1/2 1/2
  1/2 0 1/2
  1/2 1/2 0
fill go
```

## 9.3 FILL Example 3: Twinning Plane

```
#
# Twinning plane in Diamond SiC lattice
#
#
# Box commensurate with [11-2] [111] [1-10] orientation
#
box 6*sqrt(6)/2 4*sqrt(3) 10*sqrt(2)/2
#
# Align particles so
# (1) box boundaries fall midway between adjacent planes
# (2) two twinned grains are properly aligned
#
fill align sqrt(6)/12 2*sqrt(3)-sqrt(3)/8 sqrt(2)/8
#
# Basis atoms for Diamond SiC
#
fill particle 2
1 0 0 0
2 1/4 1/4 1/4
#
# Cell vectors for fcc/diamond primitive lattice
#
fill cell
0 1/2 1/2
1/2 0 1/2
1/2 1/2 0
#
# Fill bottom y half of box
#
fill orient 1 1 -2 1 1 1 1 -1 0
```

```

fill boundary box 0 0 0 6 2.75 2
fill go
#
# Fill top y half of box
#
fill orient -1 -1 2 1 1 1 -1 1 0
fill boundary box 0 2.75 0 6 5.50 2
fill go

```

## 9.4 FILL commands

### FILL ALIGN *x y z*

You can use this command to insure that one of the generated atoms is positioned at  $(x,y,z)$ . This command makes the atom type positioned at  $(0,0,0)$  in the FILL PARTICLE command placed at  $(x,y,z)$  in the generated coordinates. If  $(x,y,z)$  is outside the filled region, then if the lattice was extended outside of the region the atom would have been at  $(x,y,z)$ . Note that if no atom is at  $(0,0,0)$  in the FILL PARTICLE command, then no atom will appear at  $(x,y,z)$ .

### FILL BOUNDARY BOX *x1 y1 z1 x2 y2 z2*

Makes the fill region a rectangular box with points  $(x1,y1,z1)$  and  $(x2,y2,z2)$  as the opposite corners. This box is aligned with the coordinate axis.

If no boundary is specified, then FILL BOUNDARY BOX is assumed, with the box begin the system box (specified by the BOX command). If neither FILL BOUNDARY BOX or BOX has been specified before FILL GO is run, then an error occurs.

### FILL BOUNDARY SPHERE *radius xcenter ycenter zcenter*

Makes the fill region a sphere centered on the point  $(xcenter,ycenter,zcenter)$  with a radius *radius*. Values are in angstroms.

### FILL BOUNDARY CYLINDER *r length xc yc zc xorient yorient zorient*

Makes the fill region a cylinder with radius *r*, length *length*, centered at point  $(xc,yc,zc)$  and with its axis oriented along the direction  $(xorient,yorient,zorient)$ . Values are in angstroms.

### FILL CELL

This chooses the repeating cell vectors. By default, these are (100), (010), (001). Three lines follow this command, one for each vector. On each line are the three components for that vector.

### FILL GO

Generate particles from values specified. The values specified become the defaults for the next FILL GO command.

### FILL MARGIN *margin*

Sometimes when using FILL to generate particles in two or more adjacent regions, particles in neighboring regions may lie too close to one another. This can cause a problem when calculating energies or force, for some potential models fail when atoms are too close. To prevent this, one can choose a *margin*. After the boundary is filled, the particles are then contracted toward the boundary center so that the minimum distance from any particle to the boundary walls is *margin* (units of angstroms). The default margin is zero.

### FILL ORIENT *ax ay az bx by bz cx cy cz*

This command chooses the lattice orientation. One assumes that the atom coordinates and cell vectors normally give a lattice in a [100] [010] [001] orientation. The vectors  $(ax,ay,az)$ ,  $(bx,by,bz)$  and

( $cx, cy, cz$ ) **must** be three mutually orthogonal vectors of any length. If they are not orthogonal an error message is written.

### **FILL PARTICLE npart**

This is similar to the PARTICLE command above. It specifies a set of atoms that will be repeated to fill the region. `npart` is the number of atoms. Following this command should be `npart` lines, one line for each atom, and on each line is the atom type (a number between 1 and 255) and the x, y and z coordinates for that atom.

## **10 Write Command**

The WRITE command is used to write all information from XMD (with the exception of the ESAVE, BSAVE, SSAVE and TSAVE commands). There are two modifiers that can be used with some write options: SEL and FILE.

The SEL option causes only those values that belong to selected atoms to be written out (see SELECT above).

The FILE `[+]filename` option directs the output to the specified file. The option `+` causes the output to be appended to the end of the specified file. Otherwise, the output overwrites any previously existing file.

A detailed description of each parameter follows.

### **WRITE [FILE `[+]filename`] BOX**

Writes the corresponding parameter. Note that each of the above parameters are constants that are set either by the user or by default. All of these values are described above under their command forms.

### **WRITE [SEL] COR `[+]filename`**

Writes the current step information in COR format to the file `filename`. If the `[+]` is included, the information is appended to the file end. Otherwise the file is over written.

### **WRITE [FILE `[+]filename`] CSTEP**

Writes the value of `cstep` from the CLAMP command.

### **WRITE [SEL] [FILE `[+]filename`] DISP**

Writes the displacement of each particle or selected particle relative to the reference step. Note that you must first create the displacements using the DISP command (see section on the ?? (DISP command)).

### **WRITE [FILE `[+]filename`] DTIME**

Writes the value of `dtime` from DTIME command.

### **WRITE [FILE `[+]filename`] [SEL] EATOM**

Writes the indices and energies for each atom or selected atom.

### **WRITE [FILE `[+]filename`] EKIN**

Writes the average kinetic energy per atom. This value of kinetic energy is produced either by the ITEMP command or the CMD command, whichever was executed last. Units are in the units specified by EUNIT command.

**WRITE [FILE [+]filename] ENERGY**

Writes the average potential energy per atom (the average taken is over all the atoms, but not over time steps). Units are in the units specified by EUNIT command. This energy is the internal energy. It includes only the energy due to the interatomic potential, and not externally applied forces (see the EXTFORCE and EXTSPRING commands).

**WRITE [FILE [+]filename] EPOT**

Same as WRITE ENERGY above.

**WRITE [FILE [+]filename] ETOT**

Writes the sum of the EKIN and EPOT.

**WRITE [FILE [+]filename] [SEL] STRESS**

Writes the product of the internal stress and the atomic volume averaged over either all the atoms or the selected atoms. The units are (erg cm). The actual stress per atom can be obtained by dividing the printed value by the atomic volume. Since XMD cannot always know the atomic volume (due to the presence of free surfaces and/or void in the lattice) XMD does not perform the division automatically. XMD writes 6 products, one for each of the Voight stresses.

**WRITE [FILE [+]filename] [SEL] EXTFORCE**

Writes the atom type and the x, y and z components of the external force on the atom for every atom or selected atom. Units are the current energy units (determined by EUNIT command) divided by centimeters. See also the following command, WRITE FORCE.

**WRITE [ FILE [+]filename] [SEL] EXTSPRING**

Writes the atom type and the x, y and z components of the external spring on the atom for every atom or selected atom. Units are the current energy units (determined by EUNIT command) divided by centimeters squared.

**WRITE [ FILE [+]filename] [SEL] FORCE**

Writes the atom type and the x, y and z components of force on the atom for every atom or selected atom. Units are the current energy units (determined by EUNIT command) divided by centimeters. This force is the internal force, and does not include the external force (which is set with the EXTFORCE command. Use

**WRITE EXTFORCE to write the external force).**

**WRITE ILIST [+]file**

Writes a list of atom indices corresponding to the currently selected atoms. This list can be used by the WMOVIE program (see section on Companion Utilities, page 29), and with a little modification can be read in by the SELECT command.

**WRITE [ FILE [+]filename ] NP**

Writes number of particles

**WRITE [ FILE [+]filename ] [SEL] PARTICLE**

Writes the type and coordinates of each particle or selected particle. The units for a coordinate is angstroms.

**WRITE [ FILE [+]filename ] [SEL] POSVEL**

Writes the type, coordinates and velocity of each particle or selected particle. Coordinate units are angstroms, velocity units are cm/sec.

**WRITE [ FILE [+]filename ] STEP**

Writes out the current value of STEP.

**WRITE [SEL] RCV [+]filename**

Writes the current step information in RCV format to the file filename. If the [+] is included, the information is appended to the file end. Otherwise the file is over written. Note: This command is obsolete, use the WRITE COR command instead.

**WRITE [ FILE [+]filename ] RDF nbin rmin rmax [type1 type2]**

Writes a table of the Radial Distribution Function. This table is a histogram of the number of atom pairs between a given range of separations. nbin is the number of bins in the histogram, and rmin and rmax are the minimum and maximum value of separation included in the table (units are in angstroms). The output from this command is the header line of the format.

If the optional type fields type1 and type2 are supplied, then only the RDF between these two types of atoms is written.

```
RDF nbin rmin rmax
```

which echoes back the input parameters followed by nbin lines. Each line has two number on it, the first is the value of the separation at the center of the bin, and the second is the number of atom pairs in that bin.

**WRITE [ FILE [+]filename ] RUN**

Writes out the current value of RUN.

**WRITE STATE file**

Writes all information needed to recreate the current CMD step. This file can be read by the STATE command.

**WRITE [ FILE [+]filename ] TEMP**

Writes out the current value of the temperature, as set by either the ITEMP or CMD command.

**WRITE [ FILE [+]filename ] TEMPC**

Writes out the value of tempc from the CLAMP command.

**WRITE [SEL] TYPELIST [+]file**

Writes out a list of atom types of all atoms or selected atoms. Can be used by WMOVIE program (see section on Companion Utilities, page 29).

**WRITE [ FILE [+]filename ] [SEL] VELOCITY**

Writes out the [selected] velocities in units of cm/sec.

**WRITE [ FILE [+]filename ] string**

Any string which is not an option specified above is interpreted to be a variable that has been set by the CALC command. The value of this variable is then written out. If a variable by this name does not exist, it will be created and set to zero. You can also put in a valid CALC expression such as "2\*cos(pi/4)". It will be evaluated and written.

**WRITE [SEL] { XMOL | PDB } [+]file**

Writes the current particle coordinates (or the selected coordinates if SEL is specified) to file in XMOL or PDB format. The + sign indicates that the information should be appended to file. XMOL is the program for viewing atoms under X-Windows and is freely available on the Internet. Writing with

the XMOL option creates a text file than can be read in by the XMOL program. PDB stands for Protein Data Bank. This is a data format use by many chemistry based program. XMD writes a non-standard PDB format file that has be tested sucessfully with the commercial chemistry modeling program Cerius2, but may not be compatible with other programs. Both XMOL and PDB use standard element abbreviations to label the atom. For instance, FE for iron, NA for sodium. Many potential files will configure XMD to use the correct type name. However, by default XMD will not know the correct element names and so will write the type numbers instaed. You can set the correct element names using the TYPENAME command.

## 11 Interatomic Potentials

At this point in time, the POTENTIAL command is incompletely documented. The recommended way to implement a potential is to obtain an existing potential file that will make the necessary calls to the POTENTIAL command. Contact the maintainer of XMD for more information (Jon Rifkin, jon.rifkin@uconn.edu).

### 11.1 Using Provided Potentials

### 11.2 Using Tersoff's Carbon-Silicon Potential

```
# Set constants for Tersoff's C-Si potential
#   (11 Nov 1996)
#
potential set tersoff
dtime 0.8e-15
calc DTIME=0.8e-15
calc C=1
calc Si=2
calc MassC=12.01
calc MassSi=28.086
calc A0=4.32
eunit erg
echo off
#
# Data about this potential
#
# Si A0 at OK = 5.432      ang
# Si B  at OK = 0.979e12  erg/cm^3
#
# Experimental values
# Si A0 at OK = 5.451
# Si B  at OC = 0.979e12  erg/cm^3
#
echo on
```

### 11.3 Using Stillinger-Webers Silicon Potential

```
# Set constants for Stillinger-Weber Si potential
#   (12 Feb 1998)
#
potential set still
calc DTIME=0.5e-15
```

```

dttime DTIME
calc Si=1
calc MassSi=28.086
calc AO=5.428
eunit erg

```

## 11.4 Creating EAM Potentials

In practice most EAM potentials, from whatever source, are stored as text tables. This document describes how to adapt a text table for use with XMD. The only program this requires is a text editor. Once you have converted the text table to an XMD readable format, you should store the potential in its own file. This way it can be easily accessed from any XMD input (using the READ command).

Here is a portion of what you are trying to make; an XMD potential file.

```

#
# This is an example XMD potential file
#
eunit eV
potential set eam 2
#
potential pair 1 1 10 0.0 5.0
900.0 800.0 700.0 <---
600.0 500.0 400.0 <--- -- This is the original table
300.0 200.0 100.0 0.0 <---
#
potential dens 1 10 0.0 5.0
900.0 800.0 700.0
600.0 500.0 400.0
300.0 200.0 100.0 0.0
#
potential embed 1 20 0.0 1000
....

```

1. Lines beginning with the pound sign (#) are comments, and can be placed anywhere within the file.
2. You should tell XMD what energy units you are using for your potential table. The command is

```
EUNIT (unit)
```

where (unit) can be either eV, ERG, K, JOULE, or

```
EUNIT (name) (value)
```

where (name) is a name that you specify, and (value) is the number of ergs in one of your units.

You can specify this command more than once, which is useful if your tables do not all use the same energy units.

3. Next, you must tell XMD that you are using an EAM potential and the number of atom types in this potential. The command for this is for this is

POTENTIAL SET EAM n

where n is the number of types. Thus if you have an alloy with two component elements, you would use the command

POTENTIAL SET EAM 2

- Next, you must specify each EAM function with a different table. If for example you have 2 atom types, Ni and Al, then there will be 7 EAM functions: two electron density functions (one for each atom type), two embedding functions (again one for each type), and three pair functions (one for Ni-Ni interactions, one for Al-Al, and one cross potential for the Ni-Al interaction). In general there are  $(5n + n^2)/2$  EAM functions for n atom types.

The tables are entered with commands such as

	atom types	table size	range	
	-----	-----	-----	
POTENTIAL PAIR	1 1	2000	1.2	6.3
POTENTIAL DENS	1	2000	1.2	6.3
POTENTIAL EMBED	1	1000	0	20.0

These commands work in the following way.

- POTENTIAL - Identifies the command as a type of potential command.
- PAIR, DENS, EMBED - Identifies the table as either a pair potential, electron density or embedding function.
- (atom type(s)) - Specify to which atom type the function belongs. Note that **pair potential functions require two integers**, since in the EAM model the pair potential depends on both the atom types involved. The electron density and embedding function only depend on the atom type generating the electron density (the DENS function) or the atom type receiving the electron density (the EMBED function).
- (table size) - A integer that specifies the number of entries in the table which follow the command.
- (range) - Two numbers which specify the input range for the table. In the example above, both the pair and electron density functions accept input distances from 1 to 6.3 angstroms. For any distance shorter than the allowed range the EAM function will stop the program. For any distance longer than the range (the function cutoff) the function will return zero.

- Last comes the table. The number of values in the table must match the number specified above. The first value in the table is the function value at the start of the range - the last value corresponds to the end of the range.

Thus, **for a pair potential or electron density function the last number in the table should be zero**, since the function must go to zero at the cutoff. Similarly, **for the embedding function the first value in the table must be zero**, since it must be zero when the electron density is zero.

## 12 Techniques and Examples

### 12.1 Creating a B2 Lattice

Lattices are created with the FILL command (although they can also be created using the DUP command). The following code will create a B2 lattice with a unit cell length of 2.88, oriented with the [100], [010], [001] lattice directions in the (x,y,z) space directions.

```

#
# create a b2 lattice using atoms types 1 and 2
#
# make repeating box
box 4 4 4
#
# Use fill command to generate atom positions
#
fill particle 2
1 1/4 1/4 1/4
2 3/4 3/4 3/4
fill go
#
# scale lattice to 2.88 unit cell lengths
#
scale 2.88

```

This will create a lattice of 4 by 4 by 4 cubic unit cells, each with a type 1 atom at the corner and a type 2 in the center. The MOVE command centers the particles within the repeating boundaries. This is useful for minimizing the wrapping that occurs when particles pass through a box wall. By centering the particles in the box, position the outlying atoms as far from the boundaries as possible.

## 12.2 Creating an L12 Lattice

The following code will create a B2 lattice with a unit cell length of 2.88, oriented with the  $\langle 100 \rangle$ ,  $\langle 010 \rangle$ ,  $\langle 001 \rangle$  lattice directions in the (x,y,z) space directions. The L12 lattice is an FCC lattice with one atoms species on the corners and the other on the faces, for example Ni3Al. The following code will create a L12 lattice with a unit cell length of 4.183 oriented with the [100], [010], [001] lattice directions in the (x,y,z) space directions.

```

#
# CREATE A L12 LATTICE USING ATOM TYPES 1 AND 2
#
# make repeating box
#
box 4 4 4
#
# Generate atom positions
#
fill particle 4
1 1/4 1/4 1/4
2 1/4 3/4 3/4
2 3/4 1/4 3/4
2 3/4 3/4 1/4
fill go
#
# Scale lattcie to 4.183 unit cell length
#
scale 4.183

```

## 12.3 Creating a B2 Lattice in $\langle 110 \rangle$ $\langle 1-10 \rangle$ $\langle 001 \rangle$ Orientation

We will recreate the B2 lattice from a previous example, but this time in a different orientation. First, the box size must be changed to fit the repeating boundary conditions of the new orientation (if we had free surfaces in all directions, this wouldn't be necessary). We will choose the orientation [111], [1-10] and [11-2] for the x, y and z directions. In this orientation, the repeat distances for the B2 lattice are  $\sqrt{3}/2$ ,  $\sqrt{2}$  and  $\sqrt{6}$  assuming a unit cell length of one (the repeat distance is the minimum separation between two atoms in the specified directions). The repeating box chosen must be a multiple of these distances, and the distance for each box direction must match the corresponding orientation vector.

```
#
# create a b2 lattice using atoms types 1 and 2
#
# make repeating box
box 4*sqrt(3)/2 4*sqrt(2) 2*sqrt(6)
#
# Use fill command to generate atom positions
#
fill particle 2
1 1/4 1/4 1/4
2 3/4 3/4 3/4
fill orient 1 1 1 1 -1 0 1 1 -2
fill go
#
# scale lattice to 2.88 unit cell lengths
#
scale 2.88
```

## 12.4 Finding the Optimum Time Step

As explained above in the THEORY section under TIME STEP SIZE (see page 3), there is an optimum time step size when conducting a molecular dynamics simulation. Physically this time step size is about 0.0333 to 0.01 of the smallest vibrational period in the simulation. The smallest vibrational period depends most strongly on the potential used, and less strongly on the particular lattice structure and temperature. The optimum time step is found through trial and error, the testing done with an adiabatic simulation. Below we have the input for such a trial and error simulation.

```
#
# Trial and error to find optimum time step size
#
# Read potential for nial
read /cmd/nial.pot
#
# Make repeating box and lattice (in units of a0)
#
box 6 6 6
fill particle 2
1 1/4 1/4 1/4
2 3/4 3/4 3/4
fill go
#
# Scale up to units of angstroms (2.8712 unit cell)
#
```

```

scale 2.8712
#
# Save energies from every dynamics step in file "timestep.e'
#
esave 1 timestep.e
#
# Set particle masses (in atomic mass units)
#
select type 1
mass 58.71
select type 2
mass 26.982
#
# Set adiabtic simulation at starting temperature of 200K
#
clamp off
itemp 200
#
# Vary time step
#
# Set initial timestep size variable
calc stepsize = 4e-16
# Do 6 separate runs of 20 steps each
repeat 6
  # Set timestep size using dtime command
  dtime stepsize
  # 20 steps of dynamics (energies are written to tempstep.e file)
  cmd 20
  # Double the time step size
  calc stepsize = 2 * stepsize
end

```

The command `ESAVE` writes out four columns of numbers to the output file: the time step, the total energy, the potential energy and the kinetic energy. In practice the total system energy (the second column) will fluctuate, particularly at the first 10 or so steps of a simulation (before the higher time derivatives have had a chance to be initialized during the course of the integration). The user needs to judge for himself what amount of fluctuation is tolerable. In the past fluctuations of about 1 part per 5000 of the total system energy per twenty time steps have been allowed. It is useful to plot the value of the total energy versus time step to gauge when the system becomes unstable.

## 12.5 Initialize the Temperature

Initializing the energy can be done with a single command, the `ITEMP` command. However it is necessary to keep in mind how the redistribution of kinetic energy into the system can affect the simulation. Most often, a lattice is started with all of the atoms perfectly in place, which is the lowest possible energy position for that lattice. When kinetic energy is added to this perfect lattice, about half of it goes into the potential energy needed to place the atoms off their perfect positions. After a short period of time (say 200 time steps) the temperature of the lattice will be half of the temperature when the `ITEMP` command was first issued.

## 12.6 Finding the Equilibrium Lattice Constant

Often one wants to simulate a specific lattice structure, but is unsure of the exact lattice constant, or of the temperature dependence of lattice constant. Of course, in the case of a tetragonal or orthorhombic lattice, there are 2 or 3 lattice constants. It is important use the correct lattice constant when doing simulations, otherwise the system will be under stress. Since materials expand with increasing temperature, the lattice constant must be determined for every simulation temperature. So how do you find the equilibrium lattice constant(s) as a function of temperature? The easiest way is to use the PRESSURE CLAMP command.

This command allows the simulation box to adjust its size in accordance with an applied pressure (typically zero pressure) and the lattice motion. By writing out the box size using the BSAVE command, you can monitor the box size as a function of time step, and determine the average box size(s). Here is an example where we calculate the box size for Copper at 600 degrees Kelvin.

```
#
# Finding the equilibrium lattice constant at 600K for Copper
#
# Read potential for nial
read /cmd/cu.pot
#
# Make repeating box and lattice (in units of a0)
#
box 6 6 6
fill particle 4
1 1/4 1/4 1/4
1 1/4 3/4 3/4
1 3/4 1/4 3/4
1 3/4 3/4 1/4
fill go
#
# Scale up to units of angstroms (3.61)
#
scale 3.61
#
# Save box size from every dynamics step in file "constant.b"
#
bsave 10 constant.b
#
# Set particle masses (in atomic mass units)
#
select type 1
mass 63.55
#
# Set time step size
#
dtime 5e-15
#
# Set temperature clamp and starting temperature at 600K
#
clamp 600
itemp 600
#
# Set pressure clamp
#
pressure clamp 1.37
```

```
#  
# Equilibrate for 10000 steps  
#  
#  
cmd 10000
```

The file constant.b will contain the x, y and z box sizes every 10 steps. For this example the box size increases at the simulation start until its 600K equilibrium value is reached, and then it fluctuates around this value. One can average the x, y and z box sizes together to obtain the average box size (after equilibrium is reached). In our example the box size is 6 times the lattice constant (note in the example the box is made up of 6x6x6 unit cells).

If one is simulating a tetragonal or orthorhombic lattice one should use the `PRESSURE ORTHORHOMBIC` option, so that the x, y and z box sizes can vary independently. By default, the ratio between the x, y and z box sizes are preserved.

## 12.7 Calculating the Surface Energy

## 12.8 Simulating a Crack

# 13 Trouble Shooting a Molecular Dynamics Simulation

### Little or No Motion

Either the temperature clamp near or at 0K, or system was started with no initial velocities (`ITEMP` command) when in a stable state.

### Potential and / or Kinetic Energy Increases Too Much

This can be caused by a using a time step that is too large. If the energy increases at a physically unreasonable rate, then the cause is almost certainly a too large time step. Sometime the energy can increase slowly enough so that it appears to be due to physical causes. If unsure test the time step size (see the section

### Electron Density (xxx) out of range [xx, xx] for atom ? (type n).

This message specifies a particle number in place of the "?" in the heading above. This is caused by two particles being too close together when using some EAM potential models. Sometimes this happens because a simulation is started with a faulty initial configuration. This is certainly the case if this error occurs at the first time step. Sometimes a faulty initial configuration will not manifest until a few time steps, as when there is an unrealistic localized stress which forces two atoms close together after a few steps.

It is usefull to examine the nearest neighbors of the particle mentioned in the error messages. This can be done with the following code fragment in XMD,

```
select near 1 index ?  
write sel particle
```

where "?" is the number from the error message. This will write the type and position of the particle closest to the particle in the message, and may offer a clue as to the root problem.

The second way this might arise is if the time step is too large, which imparts too much energy to one or more particles and sometimes results in two atoms approaching too closely. If this is the case, you can rerun the simulation with `CLAMP OFF` and see if the total energy rises, a sign that the time step is too large.

## 14 Companion Utilities

There are several utility program which work with XMD, either by producing input for XMD (such as ATOMCELL) or by analyzing the results from XMD (SLIP, WMOVIE). A partial list follows.

### SLIP

Reads a COR or RCV file. Calculates the displacement of each atom relative to a reference lattice, and then for each atom, it calculates the average displacement of it and its neighbors, and subtracts this from the calculated displacement. The result is how the displacement of an atom differs from that of its neighbors. This is useful for highlighting slip planes.

### STRAIN

Reads an RCV file, compares two different sets of particle positions, and calculates the relative strain between the two. Also calculates the displacement, the inhomogeneous displacements and variations on these.

### MSD

Read an RCV file. Calculates the mean squared displacement at each step relative to some initial step.

### VMOVIEC

Views from DOS a movie file generated by WMOVIEC.

### WMOVIEC

Reads an RCV or COR file. Generates a movie file for latter viewing by VMOVIEC.

## 15 References

1. J. Tersoff, "Modeling solid-state chemistry: Interatomic potentials for multicomponent systems", Phys Rev B 39 (8), 5566 (15 Mar 1989).
2. F. H. Stillinger and T. A. Weber, "Computer Simulation of local order in condensed phases of silicon", Phys Rev B 31 (8) 5262 (14 Apr 1985).
3. H. C. Andersen, "Molecular dynamics simulations at constant pressure and/or temperature", J Chem Phys 72 (4) (15 Feb 1980).
4. N. Metropolis et al., "Equation of State Calculations by Fast Computing Machines", J Chem Phys, 21 (6) (June 1953).

## 16 Acknowledgements

XMD was written by Jon Rifkin [jon.rifkin@uconn.edu](mailto:jon.rifkin@uconn.edu) at the University of Connecticut. Many useful ideas were contributed by Philip C. Clapp, Charlotte Becquart, Elena Sevilla, Zeng Pei, Rongde Ge, Sha Xianwei, Wang Xianowei and others.

This software was been developed and tested under Linux using the Gnu C compiler.