Appendix A

The Penicillin MD code

The original "penicillin" program was designed to perform Molecular Dynamics simulations on clusters of atoms and ions in vacuo and produce a range of simulation data as output. Details of some of the simulation methodology can be found in Section 7.1.

The need to process many megabytes of raw output data from MD simulations resulted in the development of a large number of computer programs to perform complex operations and allow visual inspection of the data. In this section some notes on the use of "penicillin" are presented and the support programs are documented in detail.

During the course of this work the main penicillin code was overhauled to make it function correctly for large clusters and several bugs were corrected by the author. Two optional parallel algorithms were included and several programs were added to the suite of support codes.

A.1 Penicillin

A.1.1 Running penicillin

The operation of penicillin is controlled by the penicillin input file and a restart file. Since the operation of penicillin is not very well documented and even inconsistent at some points, it is best to configure single runs at first and try restarting simulations in test cases before attempting this in a full size simulation.

Creating new input files can be done with the penicillin program. The original author of penicillin attempted to make the input files both human friendly and machine readable. In practice this means that one has to be very careful changing things in input files generated by penicillin. An example of the input file used in section 7.8.1 is shown below.

```
                              penicillin

                      Molecular dynamics program.

                        Version 4.1.3  27/2/99

                     Written by Adrian Dornford Smith
                                    and
                               Volodya L. Bulatov

                  Parallel version (>4.1) By Gerdjan Busker


penicillin input file.

Simulation name is uo2

Boundary conditions = None
Simulation temperature = 1000.00 Kelvin
Time step = 1.000e-15 seconds
Number of energy rescaled cycles =  100000
Number of energy unscaled cycles = 1000000
Simulation status = start
Energy consevation tolerance = 1.000000e-03f eV
Step size status = variable
```

```
Comments are :-

#Instruction 1 = Change temperature by +3.00 K. Frequency = 1.00e-13 seconds
during rescaled.
#Heating to ambient temperature, then allow reequilibration


Potentials = :-

Type-1 Type-2     Name                Parameters

   O      O
   2      2     Buckingham      A = 9547.96  rho = 0.21916  C = 32.0
   U4     O
   1      2     Buckingham      A = 2484.2866  rho = 0.3410  C = 41.9847
   U4     U4
   1      1     Buckingham      A = 18600.00000  rho = 0.27468  C = 32.64000


Averages are :-

Output instructions are :-

Output 1 = Insight frame. Frequency = 5.00e-14 seconds. File = arc

Output 2 = Over write penicillin restart file. Frequency = 5.00e-14 seconds.
File = res

Output 3 = Over write penicillin restart file. Frequency = 5.00e-14 seconds.
File = res.bak

Output 4 = Displacement time in s. "  " Root mean square displacements. "  "
Temperature in K of the whole system. Frequency = 1.00e-14 seconds. File =
rmsd

Output 5 = Displacement time in s. "  " Root mean square displacements. Atom
type = 1 ( O ). "  " Temperature in K of all atoms of type = 1 ( type O ).
Frequency = 1.00e-14 seconds. File = O\_\_rmsd

Output 6 = Displacement time in s. "  " Root mean square displacements. Atom
type = 2 ( U4 ). "  " Temperature in K of all atoms of type = 2 ( type U4 ).
Frequency = 1.00e-14 seconds. File = U\_rmsd


      Atom definitions are :-

Atom type = 2   Mass = 16.000   Charge = -2.000   Label = O
Atom type = 1   Mass = 238.000  Charge = 4.000    Label = U4


Atom positions = :-

               Co-ordinates in Angstroms.
```

```
Type       x          y          z      Label

1 0 0 0 U4
1 -2.73367 -2.73367 0 U4
1 -2.73367 0 -2.73367 U4
1 0 -2.73367 -2.73367 U4
1 -2.73367 0 2.73367 U4
1 0 -2.73367 2.73367 U4
1 -2.73367 2.73367 0 U4
1 0 2.73367 -2.73367 U4
<etc>
2 -1.36684 -1.36684 1.36684 O
2 -1.36684 1.36684 -1.36684 O
2 1.36684 -1.36684 -1.36684 O
2 -1.36684 -1.36684 -1.36684 O
2 -1.36684 1.36684 1.36684 O
<etc>
```

The most important output from penicillin is in the form of ".arc" frames, collected in one large file. Additional information such as temperature, time, RMSD displacements and such can be written to files during runs although it is often more convenient to extract this data from the ".arc" files after the simulation has been completed. Penicillin keeps its state in the result file which holds all the potentials, times, number of iterations as well as the ion positions and the necessary time derivatives.

The original version of penicillin was very easy to convert for use in a multi processor environment, due to its sensible core design. Unfortunately it is hard to modify simulation parameters during a simulation, partly because of the sensitivity of ionic systems to potential and ion position modification and partly because of the format of the input and state files. An important improvement that can be made to the simulation code would be to improve the input, state and output file structure and to allow transparent modifications to the simulation state at any time.

A.1.2 Multi processor penicillin

SMP - Symmetric Multi Processor

Since SMP shared memory systems based on the Intel MultiProcessor Specification (MPS) have recently become popular, it was a logical step to incorporate a shared memory parallel algorithm based on the MIMD (Multiple Instruction stream, Multiple Data stream) model into penicillin. Writing shared memory applications is a straightforward exercise and the choice of code libraries which facilitate the development of such applications are diverse. Since POSIX 1003.1c threads is an IEEE standard, a Linux implementation of POSIX threads (named Linuxthreads) was used to develop a parallel version of penicillin. On a two processor system the code performed 85% more time steps than on a single processor in the same system. Unfortunately not many manufacturers adhere to the published POSIX standards. Many implementations of UNIX (e.g. Silicon Graphics IRIX) run all the threads on one processor and do therefore not improve performance using this scheme.

Building pencillin for SMP hardware is described in detail in the source code package documentation (Version 4.1.0 and later). Running the code is no different from running the single processor version except that it will be nearly twice as fast on a system with two processors.

MPI - Message Passing Interface

The MPI (Message Passing Interface) is a relatively new standard which allows parallel algorithms where the data is exchanged between the processors by means of messages. These messages are exchanged transparently through shared memory or network connections. The advantage of MPI code is that the processors do not need to reside in the same machine or be of the same type. Furthermore, MPI is available for any computer system, from networked desktop computers to multi processor super computers.

A significant increase in performance is obtained as long as each time-step of the

simulation is much longer than the time it takes for the processors to exchange simulation data. In our algorithm the time it takes to perform a number of time steps is proportional the number of ions squared. The amount of data that needs to be passed between the processes on the different nodes only increases with n where the number of calculations increases with n^2. The introduction of parallelism is therefore very useful.

MPI support is available in penicillin Version 4.1.3 and later. Unfortunately the use of the code is more difficult than the single processor version. MPI systems need to be configured for the number and type of processors used and what method of communication will be used. Details can be found in the documentation of the MPI system used. MPI penicillin was tested with the "LAM"(Local Area Multicomputer) implementation of MPI available for free from `http://www.osc.edu/lam.html`. Using LAM, the code has been tested on a SGI Origin2000 with 6 processors and on a heterogenous networked cluster consisting of an SGI Indigo2 and three dual processor Intel systems. The performance scaled nearly linearly with the total amount of floating point processing power available.

A.2 Penicillin utilities

The penicillin utilities are all written in C++ and use the File_arc class by V. Bulatov. This class was developed with visualisation in mind, so support for ion colours and radii is included. Although File_arc contains a brief library of ion properties, it is usually better to provide a "atomproperties" file or an "ATOMPROPERTIES" environment variable pointing to such a file. The "atomproperties" contains lines with the following elements:

`<ion name> <ion charge> <mass> <radius> <red> <green> <blue>`

The ion charge is in units of electron charge, mass is in atomic mass units, the radius is in Ångstrom and the RGB colour indexes are between 0 and 1.

The programs are called by typing in the command name with a number of parameters. In this section the standard documentation convention is used where "<>" brackets

indicate mandatory parameters and square "[ ]" brackets indicate optional parameters.

A.2.1 ARC Files

ARC files are used by MSI/Biosym software to describe ionic configurations. Because the visualisation codes in the MSI software are useful to view molecular dynamics sequences, the ARC file was chosen as the output format for Penicillin simulations. However, the ARC format is not well enough specified and not flexible enough for molecular dynamics simulations. An additional problem is that each saved ARC frame is not at all a full representation of the simulation state.

A.2.2 arc2dist

Synopsis :

`arc2dist <file.arc> [options]`

`Options:`	`-a<start dist>`	Start of interval [m]
	`-b<end dist>`	End of interval [m]
	`-c`	Report concentration, not population
	`-i<include file>`	Include only atoms with number in `<include file>`
	`-n<atom name>`	Only use atoms with this name
	`-p`	Plot dist. using gnuplot
	`-g<script>`	Run a gnufit/gnuplot script
	`-f`	Do a fit with gnufit
	`-s<no steps>`	use `no steps` in distribution interval
	`-ts<time ps>`	Start time
	`-td<time ps>`	Diff time (if `-ts` xor `-te` given)
	`-te<time ps>`	End time
	`-x,-y,-z`	Don't use direction x,y and/or z

Description :

This program classifies the ranges of the ions and reports the population or the concentrations of ions for each class middle. Concentrations are corrected for class width. The program can call "gnuplot" directly and plot the resulting distribution on the terminal. Motion in x, y and z directions can be suppressed although the calculation of concentrations is not corrected for when an axis direction is eliminated. By default this program uses the first frame of `<file.arc>` for the starting positions and the last frame for the final positions, but this behaviour can be modified with the `-ts` and `-te` options.

Example :

The following example prints the radial displacement distribution of oxygen ions (named O in the simulation file "T500.arc"). Only the bulk ions are taken into account, by means of a ".bulk" file which was created using "arc2surf" (See A.2.3). The displacements are calculated between 100 ps and 1 ns and the resulting distribution ranges from 0 to 10 Å in 20 steps:

```
arc2dist T500.arc -iT500.bulk -n0 -a0 -b10e-10 -n20 -ts100 -te1000
```

By adding the `-p` option the resulting table is plotted directly to the terminal: `arc2dist T500.arc -iT500.bulk -n0 -a0 -b10e-10 -n20 -ts100 -te1000 -p`

A.2.3 arc2surf

Synopsis :

`arc2surf`	`<file.arc> [options]`	
`Options:`	`-a<atom name>`	
	`-c<cutoff dist Å>`	
	`-f<frame no.>`	
	`-n<no. of neighbours>`	Criterion for surface/bulk
	`-pb`	Print bulk atoms
	`-pd`	Print distances for each ion
	`-pi`	Print distribution of neighbours
	`-ps`	Print surface atoms
	`-t<time ps>`	

Description :

This program counts the number of neighbours each atom has within a cutoff distance (default 3.87 Å). The `-p` options determine what the output will be. For `-pb` and `-ps` the bulk or surface atom number are printed depending on the number of neighbours each atom has. An atom is bulk if its number of neighbours is greater than or equal to the number supplied with the `-n` option (default 6). By default the atom positions are taken from the first frame of `<file.arc>` but this can be changed with the `-f` or `-t` options.

Example :

This example creates two files, one of which contains the numbers of the bulk atoms and one the numbers of the surface atoms. The criterion for bulk ions is that they have 35 or more neighbours within a 4.83 Å radius. Surface ions are all ions that fail this criterion.

```
arc2surf T500.arc -c4.83 -n35 -pb > T500.bulk
arc2surf T500.arc -c4.83 -n35 -ps > T500.surf
```

A.2.4 arcdiet

Synopsis :

`arcdiet <file.arc> <sample interval> [options]`

`-f<first frame>`	Use frame `<first frame>` as the starting point for sampling
`-i<include file>`	Include only atoms with number in `<include file>`

Description :

"arcdiet" takes periodic frame samples from a .arc file and prints the resulting .arc data to STDOUT. The typical use of "arcdiet" is to reduce the .arc file size to allow quicker processing by e.g. "arc2dist"(See section A.2.2).

Example :

```
arcdiet T500.arc 100 -f100 > T500-reduced.arc
```

Starting at frame 100, the file "T500.arc" is sampled every 100 frames and the resulting .arc file is written in "T500-reduced.arc". The resulting .arc file is approximately 100 times smaller.

A.2.5 arc2pov

Synopsis :

`arc2pov <file.arc> [options]`

`Options:`	`-f<n>`	Frame number n
	`-t<time>`	Frame time
	`-i<include file>`	Include only atoms with number in `<include file>`
	`-c`	Output coordinates only, no POV-Ray header

Description :

With cas2pov any saved ".arc" frame can be be converted to a POV-Ray file which can be

rendered with the POV-Ray ray tracing software to produce high quality graphics. Ionic colours and radii are taken from the File_ARC internal ion data library or from a personal library (See section A.2).

Example :

The following commands will generate a high quality rendering of the tenth frame of the "UO2.arc" output file. `arc2pov UO2.arc -f 10 > file.pov`

`povray -Ifile.pov -Ofile.tga`

A.2.6 arc2arc

Synopsis :

`arc2arc <file.arc> <frame no.>`

Description :

The specification of the ARC file has been stretched a bit during the development of penicillin. The current ARC files are therefore no longer readable by the original applications that are aware of this format (e.g. GULP, MSI software etc.). "arc2arc" creates an ARC frame that can be read by the original software. In retrospect, the choice of the ARC format was unfortunate, because the format can not hold any dynamic data such as time derivatives and masses.

A.2.7 cas2pen

Synopsis :

`cas2pen` `[-s <shapefile>] [-a] [-u] <.res file> ..`

`Options:`	`-s <shapefile>`	Use planes defined in `<shapefile>` to cut a cluster
	`-a`	Ignore previous -s
	`-u`	Unit cell only

Description: Ion positions are input from a CASCADE output file and are dumped in a format that is understood by "penicillin". A shape file is usually made to define stable crystal surfaces.

An ion at position (x, y, z) is considered to be inside a plane when $ax + by + cz <= d$ for all planes defined in the shape file.

A.2.8 casplot

Synopsis :

`casplot` `[-s <shape file>] <file.res>`

`Options:`	`-s <shape file>`	use `<shape file>` to cut the cluster

Description :

Casplot is an interactive utility to show the ion positions resulting from a CASCADE defect calculation. The utility is included in the "castools" package available from http://abulafia.mt.ic.ac.uk/busker and has been offered for inclusion in the Daresbury software archive. Some documentation remains to be finished before this can happen. Clusters can be cut using a shape file in the same manner as described in section A.2.7. Further options include the addition of sticks to create ball and stick models, the measurement of distances and angles, dumping the contents of the screen to a graphics file, zooming, rotating etc.

A particularly useful feature is the ability to view the relaxation of ions surrounding a defect in a perfect lattice. Casplot includes an option to create continuous graphics output, thus allowing the user to record actions on screen and create an animated sequence which can be published in HTML documents on the internet or reproduced on a computer screen.

The next release (Version 1.1) of castools will include a java applet for which balls and sticks geometry files can be created, allowing the presentation of interactive schematic diagrams on the internet (e.g. http://abulafia.mt.ic.ac.uk/busker/yttria/java.html). Version 1.1 will be released under the GNU General Public License (http://www.gnu.org).

The code was designed with portablity in mind and will run on many different systems. The ability to cut clusters was used in this study to create the initial clusters for molecular dynamics simulations from CASCADE perfect lattice simulations.

Example :

Figure A.1 shows the investigation of a simple substitutional Ru^{4+} ion in a uranium vacancy in UO_{2+x}. The sticks show the slight displacement of the nearest neighbour uranium and oxygen ions towards the defect.

A.2.9 mdviz

Synopsis :

```
mdviz     <file.arc> [options]
Options:  -i<include file>
```

Description :

MDviz plays back Penicillin MD simulations on the screen. Atoms can be excluded or included with the "include file" mechanism (See section A.2.3 for details on how to create include files). The MD playback sequence can be stopped and started with the space bar and the relative sizes of the ions can be increased and decreased with the "l" and "s"

keys respectively. As in "casplot" (Section A.2.8) the cluster of atoms can be moved and rotated with the mouse.

A.3 Future Development

The penicillin code works very well for medium scale simulations of ionic particles and has been modified to increase scalability in a cost effective way. In principle, very large clusters can be simulated for nanoseconds using off-the-shelf hardware.

In terms of flexibility, it would be beneficial to redesign the input and output files Penicillin currently uses. The input file format is error prone and difficult to read and the output format does not fully describe the state of the simulation. Modifications to the simulation parameters during the simulation are difficult and even continuing an old simulation can be difficult.

Ideally, the input file would only contain potentials parameters, a description of the atom or ion classes and simulation directives. The simulation data should only be kept in a file which contains frames describing ion position, time derivatives of the positions and ion masses. The output frames can be used to start a new simulation at any time and one should be able to add or remove ions from the simulation at will.

Due to the object oriented architecture of the penicillin utilities, it is very easy to change to an alternative file format. The Penicillin code itself will need a fair amount of restructuring to accomodate the changes suggested in this section, but the core algorithms can be left untouched.

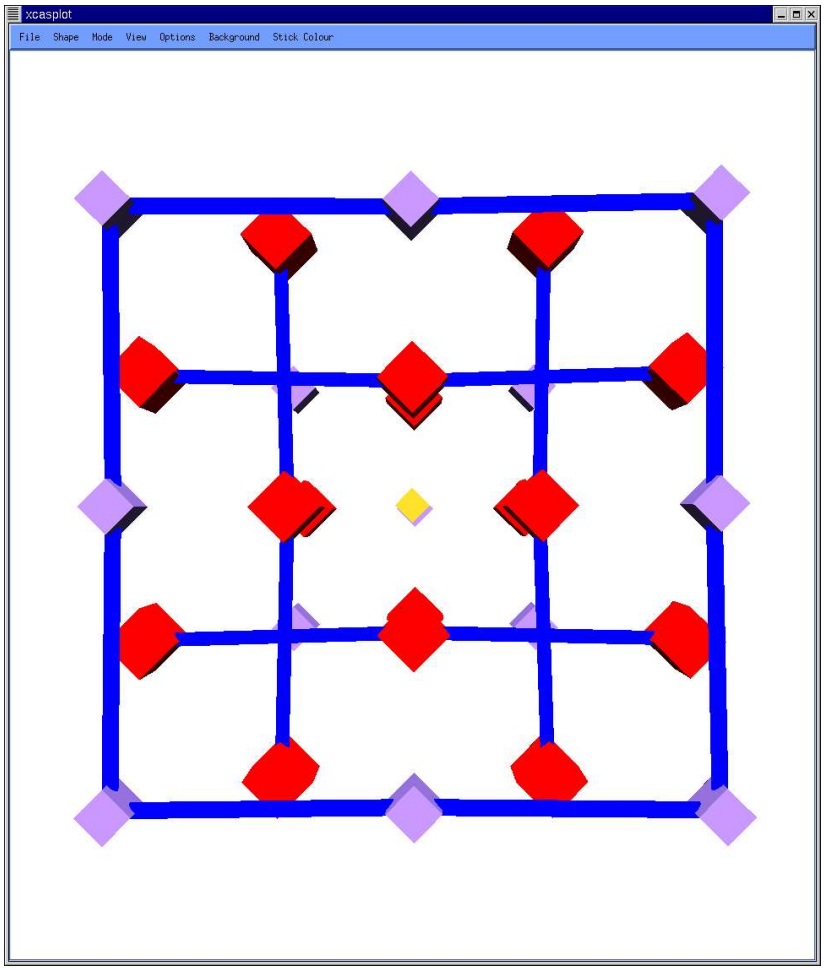

Figure A.1: A demonstration of some of the features of casplot. The simulated data comes from a CASCADE run in the study of ruthenium defects in UO_{2+x}. In this picture the atoms are represented as cubes because these are rendered more quickly by the computer. However, a wide choice of shapes such as tetrahedrons, isocahedrons, spheres and teapots are available. The Sticks can be attached to the atoms using a computer mouse.