

# University of Edinburgh

## School of Informatics

### Simulation of Radiation Damage to Metals

#### 4th Year Project Report Computer Science and Physics

Iain Bethune

March 1, 2005

**Abstract:** This project gives an overview of molecular dynamics as a tool for studying the physics of condensed matter systems such as metallic crystals under irradiation. It describes in detail the use of MDCASK as a program for doing molecular dynamics and the modification of the code to use arbitrary lattice structures and a Finnis-Sinclair empirical potential for titanium. The scaling behaviour of the code on serial and parallel computer systems is also investigated and discussed. Simulations of radiation damage to titanium crystals are then performed on `lomond`, a parallel supercomputer. The results of these simulations show that the number of defects produced in a crystal after irradiation and the size of the damaged volume in the crystal increase with the energy of the irradiating particles. It is also found that the damaged volume decreases with temperature. These results are shown to be in agreement with other published data.



## Acknowledgements

I would like to thank Prof. Graeme Ackland for his help and support in supervising this project. I would also like to thank Dr. Lorna Smith at EPCC for her help in getting MDCASK running correctly in a parallel environment.



# Contents

<b>1</b>	<b>Background</b>	<b>1</b>
1.1	Molecular Dynamics . . . . .	1
1.2	Interatomic Potentials . . . . .	2
1.3	Radiation Damage . . . . .	3
1.4	Parallel Supercomputing . . . . .	3
<b>2</b>	<b>Objectives</b>	<b>5</b>
2.1	Modifications to MDCASK . . . . .	5
2.2	Scaling behaviour . . . . .	5
2.3	Simulation and analysis . . . . .	6
<b>3</b>	<b>MDCASK</b>	<b>7</b>
3.1	Overview . . . . .	7
3.2	Using MDCASK . . . . .	10
3.2.1	Input . . . . .	10
3.2.2	Output . . . . .	13
3.3	MDCASK in parallel . . . . .	14
3.4	Modifications . . . . .	15
3.4.1	MDCASK . . . . .	15
3.4.2	Analysis . . . . .	17
3.4.3	Visualisation . . . . .	19
<b>4</b>	<b>Results</b>	<b>21</b>
4.1	Scaling Behaviour . . . . .	21
4.1.1	Serial . . . . .	21
4.1.2	Parallel . . . . .	22
4.2	Titanium . . . . .	24
4.3	Radiation Damage Experiments . . . . .	25
4.3.1	Temperature . . . . .	27
4.3.2	Energy . . . . .	28
<b>5</b>	<b>Conclusion</b>	<b>31</b>
5.0.3	Discussion . . . . .	31
5.0.4	Further Work . . . . .	33
	<b>Bibliography</b>	<b>35</b>

<b>A</b>	<b>Input Files</b>	<b>37</b>
A.1	mold.in . . . . .	37
A.2	eamdata . . . . .	37
A.3	latt.in . . . . .	38
<b>B</b>	<b>Output Files</b>	<b>39</b>
B.1	mdyn.con . . . . .	39
B.2	mdyn.ene . . . . .	39
<b>C</b>	<b>Data</b>	<b>41</b>
C.1	Serial Performance Test . . . . .	41
<b>D</b>	<b>Fitting the Biersack and Ziegler potential</b>	<b>43</b>

# 1. Background

## 1.1 Molecular Dynamics

There are many problems in condensed matter physics that are too complicated to be solved analytically. For example, the evolution of a system of  $N$  particles is governed by  $3N$  coupled differential equations of the form

$$\frac{d^2x}{dt^2} = F_x/m$$

describing their motion in 3 dimensions. Unfortunately, the force on each particle depends on the position of the rest of the particles, making this very difficult to solve, especially as a system of macroscopic size would typically involve around  $10^{23}$  particles. For bulk quantities such as temperature, pressure, and heat capacities thermodynamics can be of use, but there are cases where one would like to know finer detail than this. The formation of surface structures, faults in crystals, and radiation damage are a few of these cases. Or it may be that an experiment needs to be carried out under conditions that are not practical in laboratory conditions, such as extremes of temperature or pressure. This is where molecular dynamics can be used. [1]

Molecular dynamics is essentially a computer simulation technique for many-body physics problems. The interaction and evolution of a set of particles is calculated by solving the equations of motion. To generate the forces on the particles, and therefore the equations of motion, the interactions are described by a many-body potential function  $U$  such that

$$F_x = -\frac{\partial U}{\partial x}$$

To make the problem tractable this function is a short-ranged approximation to the real potential felt by the particles which is infinite-ranged, so that the force on a particle depends only on a finite number of neighbouring particles within this range.  $U$  can be derived from theory or by fitting parameters based on experimental observation (see section 1.2). The potential function, an initial arrangement of particles (e.g. a crystal structure), and details such as the mass of the particles, fully describes the system as far as molecular dynamics is concerned.

The molecular dynamics simulation engine then evolves the system according to Newtonian dynamics. This is done by integrating the equations of motion over a short time step, calculating the new positions, and repeating. The finite integration time step is an obvious source of approximation, and must therefore

be kept very short, typically of the order of femtoseconds or less, to stop the trajectories deviating far from their true values. Due to this small time step and the computational intensity of the calculations it requires large amounts of computing resource to run the system for much longer than several picoseconds. This makes it difficult to do measurements of systems that take macroscopic time to come to equilibrium, but is perfectly long enough to study defect formation, for example.

Molecular dynamics is a curious blend of theory and experiment. Certainly, the system on which we take measurements is not the real system, and is described by theory. However, the fact that molecular dynamics can be used to take measurements to test the predictions of theory suggests that it should be considered as a form of experiment. For this project, it is taken that molecular dynamics is a reliable form of experimentation, and that the results obtained (provided the model is good enough) are a good reflection of reality.

## 1.2 Interatomic Potentials

Most of the information about the system to be simulated is contained in the interatomic potential function, and as such this function should describe the system as accurately as possible. One of the simplest potentials is the Lennard-Jones potential, given by

$$U_i = \sum_j V(r_{ij})$$

where

$$V(r) = 4\epsilon \left[ \left( \frac{\sigma}{r} \right)^{12} - \left( \frac{\sigma}{r} \right)^6 \right]$$

$i$  is the index of the current particle, and  $j$  sums over all the other particles in the system (or within a cut-off range). The Lennard-Jones potential is good for describing systems of interacting neutral particles, such as gases, which are bound by the weak Van Der Waals dipole attraction. However, for studying crystals (or any system where strong bonding such as covalent or ionic can occur) this potential is not appropriate, and several others have been developed including the Embedded Atom Method (EAM) [2] and the Finnis-Sinclair (F-S) potential [3], both of which share the same form:

$$U_i = \sum_j V(r_{ij}) + F \left( \sum_j \phi(r_{ij}) \right)$$

In the case of the F-S potential, the embedding function  $F$  is square root. In this project a F-S potential for titanium will be used, as parameterised by G.J. Ackland in 1992 [4]

## 1.3 Radiation Damage

Radiation damage experiments study the effects that high energy particle bombardment has on a metal at the atomic level. This is of particular interest for the design of nuclear reactors, where the structural components will be subject to many radiation damage events during the lifetime of the reactor, and the resulting damage may cause significant changes in the metal over time [5]. A radiation damage event begins when one atom in the crystal lattice (the Primary Knock-on Atom, or PKA) gains a large amount of energy, perhaps by absorbing a stray neutron from a nuclear reactor core, and moves off rapidly, colliding with neighbouring atoms. A series of collisions occur, causing many atoms around the initial impact site to be displaced from their original positions. Eventually this cascade of collisions comes to a halt, with most of the atoms returning to a lattice site. However, some defects are left in the lattice, either vacancies - sites without an atom on them - or interstitials - atoms between two sites. Example pictures and videos of some cascades made during the project can be found online at <http://www.pyramid-productions.net/project/>.

There are three main types of cascade that may occur. Low energy cascades, where the PKA has energy of 100 eV or so typically leave scattered individual point defects in the lattice. These correspond to low energy neutron absorption events. Cascades where the PKA energy is of the order of 10 keV correspond to the energies of ‘fast’ neutrons, typically produced by nuclear fission reactions. As shown by Voskoboinikov *et al* [6] these cascades produce more defects, around 10 - 100 vacancy-interstitial pairs (Frenkel pairs), and clustering of defects into groups of 10 or more is observed. In a yet higher energy regime (several MeV), typical of particle energies in nuclear fusion reactions, it is thought that the atomic collisions will cause atoms to start sub-cascades, as well as the main cascade started by the PKA. See section 5.0.4 for details.

## 1.4 Parallel Supercomputing

Molecular dynamics is very demanding of computing resources. To take measurements, we might like to simulate systems of several million particles over several thousand time steps. As shown in section 4.1.1 this would take weeks on a current personal computer.

There are two main ways to make a computer system do more work - firstly, make it work faster, and secondly, add more processors. Given that the speed of a single processor is dependent on improving designs and manufacturing processes, a lot of work has been done to allow programs to run using several processors at one time. This works best for problems which are “embarrassingly parallel” [7], i.e.

doubling the number of processors doubles the rate of work. We see in section 4.1.2 how well the MDCASK molecular dynamics code will parallelise.

To aid the development of parallel computer codes, much work has gone into the development of Grid technology. The idea is to make a set of computing resources - processors, disks, memory appear to the user as a single system, without them having to worry about how these components are linked together. Grids may run on special purpose supercomputers, or may be made up of a loosely coupled network of desktop PCs. Combined with libraries such as MPI [8] to handle the passing of data between processor nodes in the system, the task of the user programmer is much simpler. Clearly, designing for such a system would also allow the code to be more portable, as it will be able to run on any system architecture to which the message passing library has been ported.

The computer resource initially available for this project was 500 Allocation Units (1 AU  $\approx$  0.7 processor-hours) on **lomond**, a Sun Fire E15k machine with 52 0.9 GHz UltraSparc III processors and 52GB RAM. It runs the Sun ONE Grid Engine [9] allowing users to submit jobs on a 4 processor front-end which added to a queueing system for later execution on the remaining 48 processor back-end.

## 2. Objectives

### 2.1 Modifications to MDCASK

The software package used in this project is MDCASK [10], maintained by the Lawrence Livermore National Laboratory. It is used as a benchmarking tool as well as a molecular dynamics tool, and is available with data to simulate copper crystals using an EAM potential. One of the objectives of the project is to carry out molecular dynamics simulation of titanium, and in order to do this, two main modifications to MDCASK are necessary.

Firstly, since copper adopts a face-centred cubic structure and titanium adopts a hexagonal close-packed structure, the code must be modified to support this. In fact the code should allow an arbitrary crystal structure to be used, provided it can be described using an orthorhombic basis.

Secondly, the potential must be altered. Since the code currently uses an EAM potential, which has the same functional form as the F-S potential (see section 1.2), this can be done by replacing the functions  $V$ ,  $\phi$ , and  $F$  with the ones parameterised for Ti.

### 2.2 Scaling behaviour

As MDCASK will be used for simulating systems with large numbers of atoms (up to 1 million) it is important to maximise the performance of the code. The effect of system size and the use of link cells will be investigated to see how it affects the computation time of a serial version of MDCASK.

True parallelisation of a problem is very difficult to achieve, as there is almost always some communication between nodes working on different parts of the problem. In the case of molecular dynamics, each processor may need to pass data about particles on to another processor. Often all the processors need to communicate with the controlling processor, for example, to calculate the total energy of the system each processor must send the energy of its own particles, which are then summed together by the controlling processor and reported. This introduces a communication overhead, which may mean that simulating bigger and bigger systems can't be achieved by simply adding more processors to the grid.

This project will also investigate the way in which the system performance scales

with the number of processors used.

### 2.3 Simulation and analysis

Having modified the software, experiments will be carried out to confirm that the simulation model is an accurate representation of titanium, thus checking that the modifications have been implemented correctly.

MDCASK will then be used to carry out radiation damage simulations on titanium crystals, investigating the creation and spread of defects under varying conditions of temperature and impact energy. In order to do this, some method of identifying the defects must be implemented, along with software for performing the analysis. Visualisation of defect production as images or video will also be implemented.

# 3. MDCASK

## 3.1 Overview

At a high level MDCASK divides into two main sections - initialisation of the system, and the dynamic simulation engine.

MDCASK begins by reading in the parameters of the simulation from file. This includes the data for the interatomic potential, which is stored as a lookup table, the number of steps to run the simulation for, the dimensions of the atomic lattice to be simulated, and many more parameters, which are detailed in section 3.2. At this point it generates the entire crystal by taking a basis of a few atoms and replicating them in space to construct a lattice with the specified dimensions. In the original code, this generated a face-centred cubic lattice, which is adopted by copper, aluminium, silver, and gold, among others. The lattice routine can also leave part of the simulation space empty, allowing simulation of a crystal surface. At this point, other properties of the atoms may be set. It is possible to have several layers of static atoms, which are held fixed during the simulation. This could be used to melt half the crystal, for example, before releasing the static atoms and allowing the system to evolve at a solid-liquid phase boundary. It is also possible to define a region of atoms as temperature control atoms, that is, during the dynamic simulation, their velocities are rescaled so that the system will come to equilibrium at a particular temperature.

Before the simulation begins, each atom is given a random velocity in order to break symmetry and stop all the atoms evolving in exactly the same way. At this point, the velocities are all scaled so that the system has the temperature specified by the user. The temperature of the system is given by the equipartition of energy formula

$$T = \sum_i \frac{\frac{1}{2}m_i v_i^2}{\frac{3}{2}nk_B}$$

where  $m_i$  and  $v_i$  are the mass and velocity of particle  $i$  respectively.  $n$  is the total number of atoms in the system and  $k_B$  is the Boltzmann constant.

Thus the user's desired temperature is recovered by scaling all the velocities by  $\sqrt{\frac{T_{\text{required}}}{T}}$ . This same velocity scaling routine is used during the dynamic simulation to add or remove energy from the system via the temperature control atoms.

At this point the atoms are divided up into link cells. The link cell method is a way of dividing the simulation space up into smaller volumes, the number of which is specified by the user. When calculating the energy and force on an atom, we need to know all the ‘neighbour’ atoms which are within the range of the potential function. If we set the link cell size to be larger than the range of the potential, we can guarantee that all the neighbour atoms must lie within the current atom’s link cell or the 26 surrounding link cells, thus making the neighbour search much faster than searching over all the atoms in the system. The link cells also form part of the parallelisation process discussed in section 3.3.

For most simulations we will be using periodic boundary conditions (PBC) which means that each particle with position  $\mathbf{r}$  in the simulation actually represents a set of particles with positions  $\mathbf{r} + l\mathbf{a} + m\mathbf{b} + n\mathbf{c}$  where  $l, m, n$  are integers and  $\mathbf{a}, \mathbf{b}, \mathbf{c}$  are the lattice vectors. This means that there is effectively no edge of the system - atoms at one edge of the simulation space simply interact with the images of particles at the opposite side. When searching for neighbour particles we also make use of the minimum image criterion. Based on the assumption that the simulation ‘box’ has sides larger than twice the range of the potential the minimum image criterion is that each atom can only interact with one of these many image particles at a time, and it must be the closest one.

The next step is to calculate the initial energies and forces in the system. This involves finding all the neighbours of each atom and the distances to them, and applying these values to the functions  $V$ ,  $\phi$ , and  $F$  of the potential in section 1.2. These functions are stored as lookup tables which are read in at the start of the program, in order not to have to repeatedly calculate the functions for every particle. By summing all the contributions from the neighbours, the total energy and force is calculated.

**NB** In the code these functions are given different names

$$V \equiv \text{phi}, \phi \equiv \text{rho}, F \equiv \text{frho}$$

This entire setup process can be skipped, however, as MDCASK allows the option of reading in the entire state of the system directly from file. This effectively allows the user to continue previous simulation runs.

At this point the dynamic simulation begins and repeats for a fixed number of time steps.

Every time step, the new positions and velocities are calculated using a 4th-order Predictor-Corrector algorithm [11], which works as follows. The positions and their time derivatives up to 3rd order at time  $t + \delta t$  are computed based on a Taylor expansion around time  $t$  :

$$x(t + \delta t) = x(t) + \delta t \frac{dx(t)}{dt} + \frac{1}{2} \delta t^2 \frac{d^2 x(t)}{dt^2} + \frac{1}{6} \delta t^3 \frac{d^3 x(t)}{dt^3}$$

$$\frac{dx}{dt}(t + \delta t) = \frac{dx(t)}{dt} + \delta t \frac{d^2 x(t)}{dt^2} + \frac{1}{2} \delta t^2 \frac{d^3 x(t)}{dt^3}$$

$$\vdots$$

These new positions are then used to calculate the energies and forces on each particle in the system, in exactly the same way as was done initially. Given the force on each particle, an error signal is calculated which is the difference between the predicted acceleration and the actual acceleration:

$$correction = \frac{F}{m} - a_{predicted}$$

This error signal is then used to calculate the corrected values of position and their time derivatives by using a set of scaling factors calculated by Gear [11]

$$\begin{aligned} x &= x + correction * c_0, \\ \frac{dx}{dy} &= \frac{dx}{dy} + correction * c_1 \\ &\vdots \\ c_0 &= \frac{1}{6}, c_1 = \frac{5}{6}, c_2 = 1, c_3 = \frac{1}{3} \end{aligned}$$

**NB** For those who wish to read the code for this algorithm contained in the files `predic.f` and `correc.f` note that rather than storing the derivatives directly, MDCASK uses the following variables in order to reduce the number of arithmetic operations performed each step. It does however, make the code more difficult to read as it no longer corresponds exactly to the equations above.

$$x1t \equiv \frac{dx}{dt} \delta t, x2t \equiv \frac{1}{2} \frac{d^2 x}{dt^2} \delta t^2, x3t \equiv \frac{1}{6} \frac{d^3 x}{dt^3} \delta t^3$$

Once the corrected positions and velocities are calculated, the kinetic energy and temperature are calculated for that time step. Depending on the configuration of the system, MDCASK can now write data out to file, including the energy of the system and the positions of all the atoms. This entire cycle repeats until the simulation time period is finished.

## 3.2 Using MDCASK

### 3.2.1 Input

Input to MDCASK is given in 2 files - `mold.in` and `eamdata`, included as Appendices A.1 and A.2.

`mold.in` contains all the data for setting up the simulation. The list below details all the aspects of the simulation that can be modified using this file. Names in **typewriter** font refer to variable names in `mold.in` (see appendix A.1).

- Lattice parameters - sets the length of each of the sides of the (orthorhombic) lattice.
- PBC - set PBC on each direction. This should be set to true unless the simulation includes a surface.
- Remove Layers - it is possible to only have a partially filled simulation volume. If this is set to true, the lattice routine misses out the top `NSURFLUS` layers in the z-direction when generating the lattice.
- Static Layers - this allows an arbitrary number of layers of the lattice to be held static during the simulation.
- Temperature Control - if `TEMPCONT` is set to true, then the atoms in the lowest `NTCLAYERS` layers in the Z (ZTC) or X and Y (XYTC) directions are marked as temperature control atoms, and may have their velocities rescaled during the simulation run.
- Scale Velocity - every `NSCALE` steps, the temperature control atoms have their velocities adjusted so that the system will come to equilibrium at the temperature `TEMPREQ`. Figure 3.1 shows the result of starting a system of 4000 titanium atoms at 300K and allowing it to come to equilibrium with and without temperature control. Due to equipartition of energy, the uncontrolled system ends up with half its energy stored at potential energy and half as kinetic energy, thus it comes to equilibrium at 150K, compared with the controlled system which ends up at 300K. Note that the rescaling of velocities violates conservation of energy, so once the system has been brought to the desired state, temperature control should be turned off before measurements are made. The sharp peaks on the temperature controlled graph are the points at which the velocities are rescaled.
- Number of elements and mass - this sets up the number of different species in the simulation and their masses. To use several different species, interatomic potentials for the interaction between each of the different species must also be given.

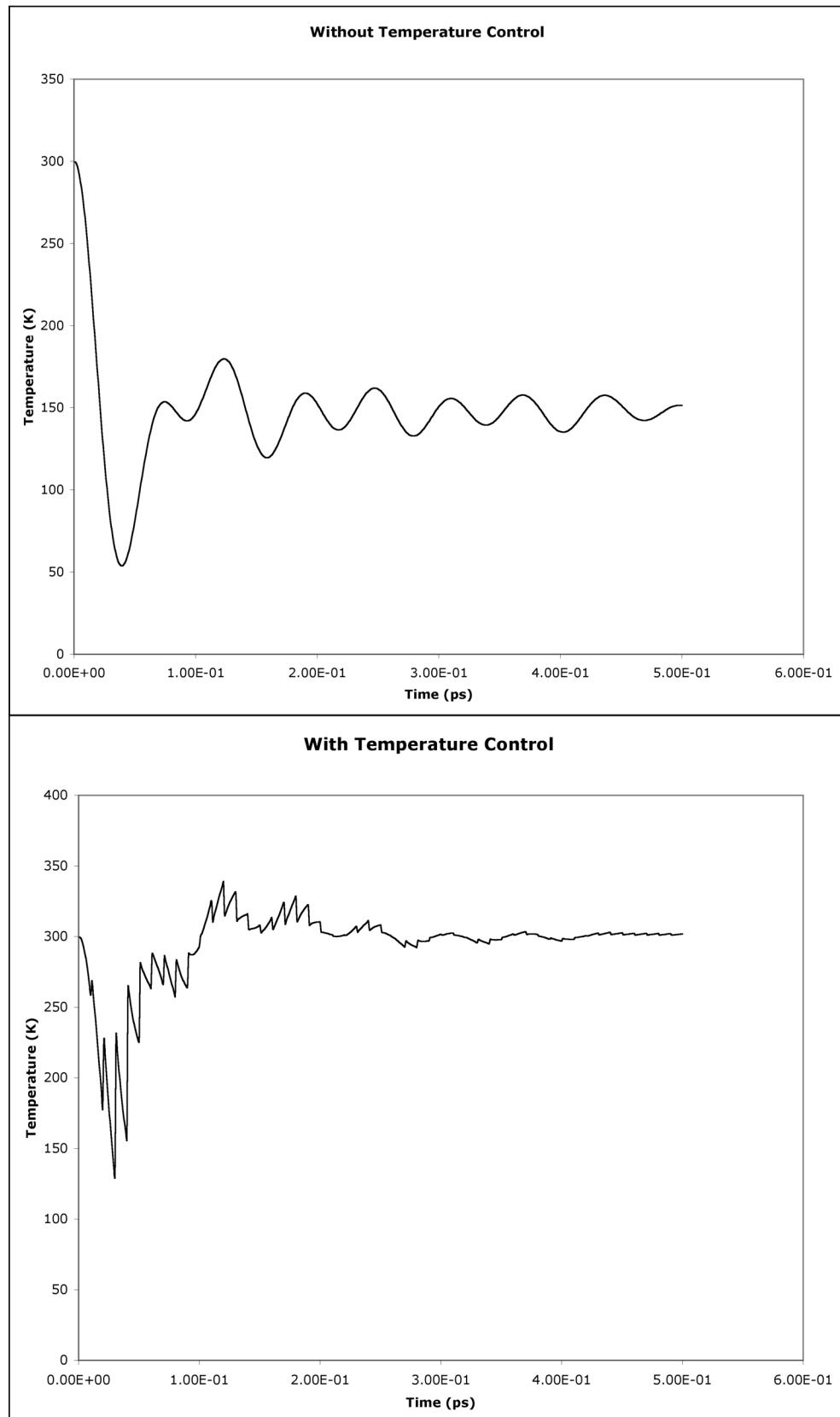


Figure 3.1: Temperature Control

- Timing - the initial timestep **DELTA** is used throughout the simulation run unless **CHNGDT** is true, in which case the routine **teacup** is used to determine the new time step based on the three parameters supplied here. **teacup** uses the longest possible timestep such that no atoms in the system move more than **DXMAX** during the time step. This shortens the timestep when there are fast-moving particles, making sure energy is conserved during collisions, but allows a longer timestep when there is little motion in the system, for example when reaching equilibrium.
- Temperature - the system is initially set to this temperature by rescaling the random initial velocities of the atoms
- Lattice Size - the matrix **B0** controls the size of the lattice. Setting the diagonal elements sets the x, y, and z dimensions respectively, in units of the lattice parameters.
- Simulation Length - the simulation is run for **NSTEPS** steps. Every **NWRCONF** steps all the simulation data is written out to a file. If **NIN** is true then the simulation state is read from file, rather than being generated by the initialisation routine.
- Recording positions - if **WCONF** is set then the positions of all the atoms are written to file every **NPRINT** steps, which allows the system to be drawn by a visualisation package. Only atoms with a potential energy greater than the cutoff are drawn.
- Random seed - this is used as a seed for a pseudorandom number generator, allowing reproducability of simulations.
- Primary Knock-on Atom (PKA) - When doing radiation damage experiments, one can include a single atom with high energy (as if, for example, it had absorbed a neutron). This atom can either be an atom already in the system (**NPKA** = -1) or it can be an extra atom (**NPKA** = -2). If (**NPKA** = 0) then instead of adding a single atom, the system is modified by adding a temperature distribution about the point specified. For a particle, the direction and energy of the PKA can be given, as well as its type (**IDNPKA**). The value of **INPKA** is the timestep at which the PKA is introduced to the system. This allows the simulation to reach equilibrium before being perturbed by the PKA.

The **eamdata** file holds the data for the potential functions.

In the header of the file the following data is stored (in order) : atomic number, atomic mass, lattice parameter (in Å). The next three parameters (**drar**, **drhoar**, and **acutal**) relate to the following data, which is a series of lookup table values for the three potential functions,  $\phi$  (**rho**),  $V$  (**phi**), and  $F$  (**frho**). Each function and its derivative are evaluated at 5000 points. The spacing between evaluation

points for  $\phi$  and  $V$  is `drar` and `acutal` gives the cut-off value for these functions, that is  $5000 * drar$ .  $F$  is evaluated at a spacing given by `drhoar`.

### 3.2.2 Output

Depending on the options set by the user in `mold.in` several files will be output by MDCASK.

The first, `mold.out`, simply contains a slightly extended version of what is printed to the terminal output during execution of the code. This is useful when running MDCASK as a batch job when the user doesn't have access to the terminal directly, and also provides extra information for identifying problems such as notifying when the timestep is altered.

`mdyn.con` is written if the user has turned on `WCONF`. It contains all the positions of the atoms in a format suitable for use in a visualisation tool. The file contains the following data in the header file:

- Number of steps into the simulation run
- Number of Primary Knock-on Atoms
- Total elapsed simulation time, temperature, pressure, total volume ( $\text{\AA}^3$ )
- Dimensions of the lattice in  $\text{\AA}$
- Coordinates of centre of kinetic energy
- Number of atoms

Then for each atom the following data is included:

- ID (which species it is)
- Coordinates in  $\text{\AA}$
- Potential energy
- Kinetic energy
- Pressure
- Shear
- Number of neighbours

`mdyn.ene` is written every time step in the simulation. It contains details of energy in the system. For each time step it has a line including the elapsed time in ps, the system temperature, total kinetic energy, total potential energy, total energy, total energy + initial potential energy, and pressure. This file is particularly useful for taking measurements and drawing graphs of physical quantities over time. In particular it can be used to quickly check if energy is conserved in a system.

`mdyn.rst` is the file containing all the simulation data. It is written after `NWRCONF` steps, and can be used to restart a simulation. To run a simulation starting from this data, the file must be renamed to `mdyn.inp` and the flag `NIN` must be set

to true in `mold.in.` `mdyn.rst` stores much more data per particle ( $\sim 330$  bytes) than `mdyn.con` ( $\sim 100$  bytes) due to the integration routine used. The Predictor-Corrector routine (see section 3.1) requires the positions, velocities, and 2nd and 3rd derivatives for each particle, whereas visualisation only requires the positions.

### 3.3 MDCASK in parallel

The parallel version of MDCASK is essentially the same as the serial version, except for the fact that the simulation is split over the different processing nodes in the system, using MPI message passing to coordinate the work. Once the simulation space has been divided up into link cells, each processor is assigned a block of link cells for it to process (its core link cells). Each processor also holds the positions of the atoms in the layer of link cells surrounding its core cells - these are known as the skin link cells, and are needed since an atom can be affected by neighbours in the next link cell (see section 3.1). At each time step, every processor calculates the forces on the atoms in its own core link cells, and updates their positions exactly as in the serial version. However, before calculation can proceed, all the processors need to update the states of the atoms in their skin cells, which would have been calculated by some other processor. This is where the bulk of the message passing is done, and in section 4.1.2 it is shown that this can cause a performance bottleneck.

If an atom crosses the boundary of the core link cells and moves into the skin cells then it must be migrated to the appropriate node for processing in subsequent timesteps. This is handled by the `migrate` routine. When atoms are sent to another node, they are added to that node's list of atoms, as well as being added to the correct link cell, so that they can be found during neighbour searches. After migrating atoms, the positions of atoms in the skin cells are updated by the routine `sowxyz`.

Another major use of message passing in MDCASK is to gather data for output. All the nodes in the system are equal in terms of the molecular dynamics computations that they perform, but node 0 also handles all the input and output. For example, every time step, the global temperature of the system needs to be calculated by receiving the local temperatures from each processing node. Furthermore, when the state of the system is written to file, the data for every single atom has to be passed to node 0 before it can be written. Thus message passing for I/O is also a potential bottleneck.

All MDCASK calls to MPI routines are wrapped by subroutines in `mpi.f`. This allow for encapsulating error handling code and also prevents the need for including the MPI header into every file that uses message passing. Despite the

wide range of communication modes provided by MPI, MDCASK only uses two, namely standard mode (`send` and `sreceive`) and immediate mode (`isend` and `ireceive`). Standard mode is typically used when sending small blocks of data or individual variables such as the temperature or the number of atoms on a node. Calls to standard mode send and receive block until the send or receive buffer is safe to use - in the case of send this is when the data has been copied into another buffer, and for receive this is when the buffer has been filled and is ready to use. Because of the blocking behaviour, sending large amounts of data using standard mode is inefficient as the calling node must wait until the buffer has finished copying. By contrast, immediate mode calls return instantly, but do not guarantee that the buffer is safe to use. MDCASK uses these calls for sending larger blocks of data such as arrays of particle positions, so that the calling process can continue working while the buffer is copied in the background. However, before the buffer data can be reused or overwritten, the program must perform an `MPI_WAIT` call to make sure that the data transfer has finished.

More complex operations such as summing values from all nodes (`collect.f`) and finding a global minimum (`minglob.f`) are achieved by having each node send a value to node 0, which receives them all, performs the operation, then uses an MPI broadcast call to send the result back to each node. However, this functionality could also have been achieved by using MPI's `reduce` routine, which allows many functions to be performed across several nodes, including `MPI_MIN` and `MPI_SUM`. These collective operations may give better performance than using send, receive and broadcast due to the possibility of optimization of such a complex operation, but this depends on the precise implementation of MPI for a particular architecture.

## 3.4 Modifications

### 3.4.1 MDCASK

In order to support the simulation of titanium, the modifications described in section 2.1 were made. A crystal structure can be described by a basis of some number of atoms which make up the unit cell, along with 3 lattice vectors. This basis is described in the file `latt.in` which has been added to the global header file `moldy.h`. The file takes the format of an integer which is the number of atoms in the basis, followed by the x, y, and z positions of the basis atoms expressed in fractional coordinates inside a unit cube with side length 2. See appendix A.3 for an example of this file for the HCP structure of titanium. The `fcc` routine was replaced with a modified version called `lattice` which rather than having an fcc basis hard-coded into it, reads the basis from file and then uses it to construct

the lattice in the same way as the `fcc` routine. Note that the number of atoms in the basis, and the size of lattice specified in `mold.in` must agree with the total number of atoms in the system, which is specified in `moldy.h`, and if changed will require recompiling the code. However, this routine now allows any crystal structure that can be described by a basis and an orthorhombic unit cell to be set up by MDCASK.

The other modification was to replace the `eamdata` file which contains lookup values for an EAM potential for copper with the functions required for titanium. For this a simple Java program was written which calculates the required values of  $\phi$  (`rho`),  $V$  (`phi`), and  $F$  (`frho`) based on the parameterisations obtained by G.J. Ackland [4] and writes them to a file in the format described in section 3.2.1. This potential was originally developed to study surface and defect formations and energies in titanium and as a result, the potential is fitted to describe circumstances in which the atoms are well spaced. In molecular dynamics this is not always the case, as the Primary Knock-on Atom enters the system with high energy, and may get very close to neighbouring atoms before recoiling. Ideally, a potential for simulating atomic collisions should have a very high energy when the atoms are overlapping (closer together than twice the atomic radius) in order to push them apart. The pairwise part of the given potential is only  $\sim 47\text{eV}$  when  $r = 0$  (two particles are in the same position). In practice, this means that a particle with more energy than this can pass directly through the site of another atom without being recoiled. For molecular dynamics, where particles can have energies of several 1000 eV, this potential breaks down and the simulation becomes unphysical.

For this reason, the potential was modified by fitting a Biersack and Ziegler [12] screened electrostatic potential to model the strong repulsion at short range where interactions with other neighbours become insignificant compared to the massive repulsion caused by the nearest atom. The form of the pairwise potential becomes

$$\begin{aligned} V(r) &= V_{old}(r)H(r - r_2) \\ &+ H(r_2 - r)H(r - r_1)\exp(B_0 + B_1r + B_2r^2 + B_3r^3) \\ &+ H(r_1 - r)\frac{Q_i Q_j}{r}\xi(r/r_s) \end{aligned}$$

where  $Q_i$  and  $Q_j$  are the nuclear charges,  $r_s = 0.4683766/(Q_i^{2/3} + Q_j^{2/3})$ ,

$$\xi(x) = 0.1818e^{-3.2x} + 0.5099e^{-0.9423x} + 0.2802e^{-0.4029x} + 0.02817e^{-0.2016x}$$

and  $H$  is the Heaviside step function

$$H(x < 0) = 0, H(x \geq 0) = 1$$

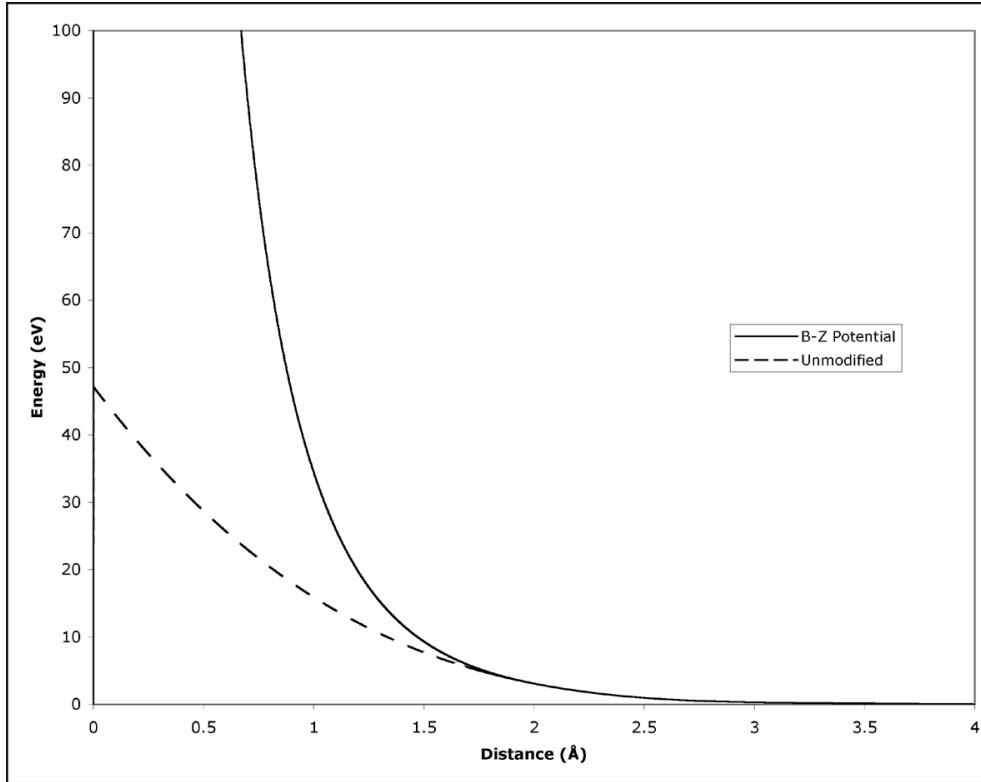


Figure 3.2: Biersack and Ziegler modified potential

and  $V_{old}$  is the original potential.

This potential diverges to infinity as the inter-particle separation becomes small, thus providing a suitable repulsion to cope with very high energy particles in the radiation damage simulation. Values of  $r_1 = 1.0\text{\AA}$  and  $r_2 = 2.0\text{\AA}$  were chosen so that the highly repulsive screened coulomb potential is only active at very short range, and that for typical lattice separations ( $> 2.0\text{\AA}$ ), the fitted potential for Ti is used. In the range  $1.0 < r < 2.0$  an exponential curve is used to join the two potentials. The parameters  $B_0 \dots B_3$  were chosen to ensure the potential and its first derivative are continuous at  $r_1$  and  $r_2$ . As shown in appendix D, this gives  $B_0 = 7.112596228$ ,  $B_1 = -4.393990755$ ,  $B_2 = 0.9407536255$ ,  $B_3 = -0.1197498592$ . A comparison of the unmodified potential with the Biersack and Ziegler form is shown in figure 3.2.

### 3.4.2 Analysis

The amount of data generated by a molecular dynamics simulation can be vast. The positions of every particle are written to disk, but this is not actually very useful when it comes to visualising a radiation damage simulation. It is necessary

to have some method of locating the defects in the crystal, so that only they are drawn. This also has the added benefit of reducing disk usage, as typically the defects make up only a small fraction of the whole crystal.

There are two types of defects that we would like to visualise: vacancies, which are lattice sites which have no atoms on them, and interstitial atoms, which occur when there is an extra atom around a particular lattice site. To identify these defects, an analysis algorithm was designed which first makes a list of all the lattice sites in the system. This can either be from a perfect lattice configuration, or it can use the initial configuration of the simulation. Then, for every snapshot of the system saved in the `mdyn.con` file, it makes a count of the number of atoms which are closest to each lattice site. For each site which has either 0 (vacancy) or greater than 1 nearest atom (interstitial), the coordinates of that site are written to file. The number of defects in each snapshot of the system is also stored, as well as ‘atoms’ at each of the 8 corners of the simulation box, so that the edges of the simulation can be seen when the defects are visualised. All the snapshots from a particular simulation run are stored in one file.

The analysis routine was implemented in Java, and initially used a brute force method whereby for every atom, the distance to each lattice site was calculated, and the site of minimum distance was found. Assuming that the number of atoms  $n$  is equal to the number of lattice sites, then this implementation has a runtime which is  $O(n^2)$ , and quickly becomes too slow for large systems of atoms. A second implementation relied on the fact that because the atoms are arranged in a regular lattice, the nearest neighbour of an atom must lie in the region of one unit cell width either side of it. To use this property, the lattice sites were first sorted by x-coordinate (an  $O(nlg(n))$ ) operation using a mergesort algorithm provided by Java’s `Arrays` class [13]). Then for each atom, its position in the list of sites was looked up by a binary search ( $O(lg(n))$ ), and then a number of sites either side of this position had their distances to the atom calculated, and the minimum was found. The number of sites to search was calculated by working as follows: For a system of  $40 * 40 * 40$  unit cells, each with a 4 atom basis, there are  $40 * 40 * 4 = 6400$  atoms in each plane of unit cells. Thus searching 6400 sites either side of the atom in question is guaranteed to locate the closest site. Although this method searches  $1/40$  as many atoms as the brute force method, it still scales as  $O(n^2)$ .

A better algorithm, which was the one eventually used, is to divide up the lattice sites into cubic link cells, each corresponding to one unit cell of the lattice. Then for each atom, the algorithm uses the atom’s coordinates to index into the list of link cells, then searches for the closest site in the 27 surrounding link cells. This algorithm offers  $O(n)$  performance, as the number of sites searched is constant, no matter how big the simulation is. It also has the useful property of making Periodic Boundary Conditions easy to implement - if an atom is located in a cell

at the edge of the simulation box, then the cells on the opposite edge are also examined to find the closest site.

### 3.4.3 Visualisation

A viewer written by Andrew Jones exists for viewing the results of the molecular dynamics simulations. Given the positions of a number of particles, it will draw them in a view that can be rotated, translated and scaled. The analysis software described above uses the same format for storing positions, so it is easy to extract one frame from a file containing several snapshots of the system to be viewed using this software. An example use would be examining a cluster of defects, where it would be necessary to look at the structure from several viewpoints.

To study how the defects develop during a radiation damage cascade, a program to generate frames for a video was also written. It reads a file containing several frames worth of position data, and renders them all from the same viewpoint using the drawing routines from Andrew Jones' software. The resulting pictures can then be assembled into a video, using software such as QTSuperImageSequencer [14] or similar. Figure 3.3 shows snapshots from a 1000eV cascade in titanium at various times after impact which were produced by running the analysis described above. Vacant sites are drawn in blue and interstitials in red. More examples of these pictures and videos can be seen online at <http://www.pyramid-productions.net/project/>.

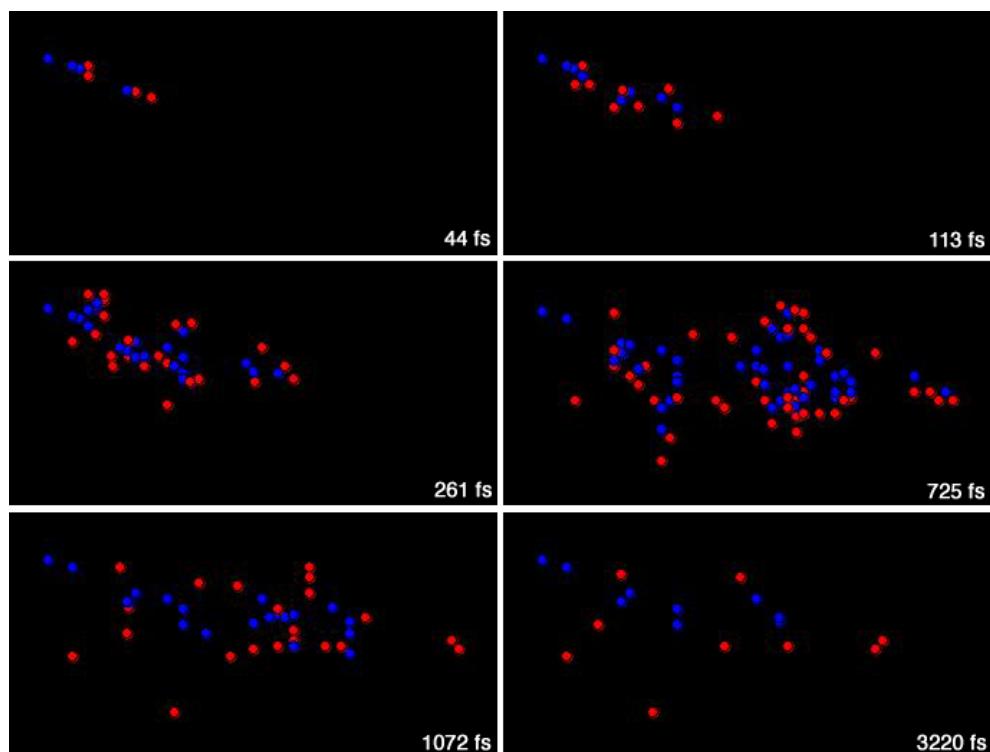


Figure 3.3: 1000eV Radiation Damage Cascade

# 4. Results

## 4.1 Scaling Behaviour

### 4.1.1 Serial

Tests were carried out with the serial version of MDCASK to see how the time taken to simulate a system varied with the size of the system. From the code, it is clear that the routines in the simulation loop - `predic`, `linvlc` and `correc` - all have a running time which is  $O(n)$ , where  $n$  is the number of atoms in the simulation. The routines `rhoeam` and `kraeam` which do the computation of the pairwise potential functions are clearly  $O(n^2)$  as they contain nested loops over two sets of atoms, both of which have at most  $n$  members. In fact the runtime is substantially better than this in practice. One of the loops is bounded by `nnbrs` which is the maximum number of neighbours that an atom has, and is typically much smaller than  $n$ . However, `nnbrs` must be several times the number of atoms per link cell i.e.

$$\text{nnbrs} > k * \frac{n}{\text{nlc}}$$

where `nlc` is the number of link cells and  $k \approx 30$  is some constant.

The size of the link cells is determined by the range of the potential function - each link cell must be at least as big as this range, so that all the neighbours of an atom (those that are within range of the potential) can be found within the  $3 * 3 * 3$  block of link cells surrounding that atom. In practice, this means that for a given system size, we can scale the number of link cells so that there are always (roughly) the same number of atoms per link cell, i.e.  $\frac{n}{\text{nlc}}$  is approximately constant. If this condition is met, the performance of MDCASK scales linearly with  $n$ , as the inner loop over all the neighbours of an atom is bounded by a constant.

Experiments to demonstrate this were carried on a computer with an Apple G4 1.25 GHz processor and 1GB RAM. The results of these are shown below (data in appendix C.1)

Figure 4.1 clearly shows that the time taken by the simulation is linearly proportional to the number of atoms, provided that the number of atoms per link cell is kept constant. The system was made up of titanium atoms, simulated for 100 time steps. The number of atoms per link cell was approximately 8. Note that

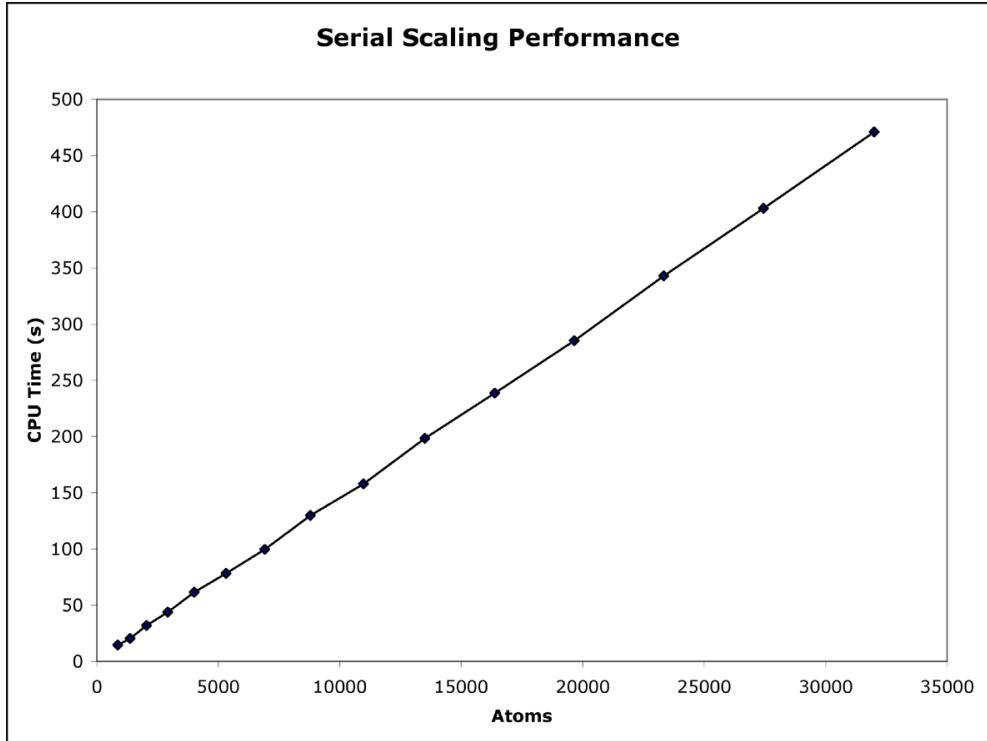


Figure 4.1: Serial Scaling Performance

this size of link cell is too small for the simulation to be physically accurate, but is fine for the purpose of timing experiments.

Figure 4.2 shows that maximising the number of link cells corresponds to an increase in computation speed. This confirms the relation above that increasing `n1c` decreases the number of neighbours, which therefore decreases the computation time. However, this gain is limited by the condition that the size of a link cell must be at least the range of the potential for the simulation to be physically correct so it is not possible to keep dividing the simulation space into more and more link cells. The system used for these tests was 32000 titanium atoms, which were simulated for 10 time steps.

#### 4.1.2 Parallel

The length and size of simulations that can be performed on a single processor computer is quite limited. This is a problem when simulating radiation damage because the higher the energy of the PKA, the larger the size of simulation space must be to absorb it and the longer the defects take to reach a final state. However, due to some of the issues inherent in the parallelisation of MDCASK described in section 3.3, it is not simply a case of using as many processing nodes

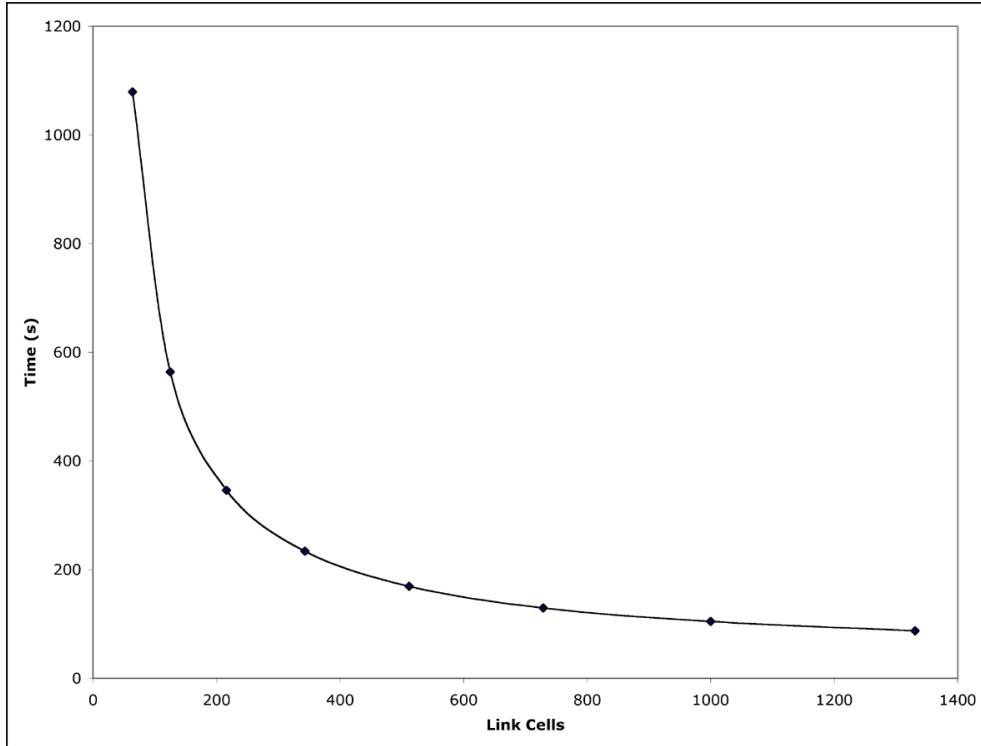


Figure 4.2: Varying Number of Link Cells

as possible.

This is shown in figure 4.3. For these results, a system of 256000 atoms was simulated for 100 time steps, using several different processor configurations. The number of link cells was  $40 * 40 * 40$ , except for the 12, 18, and 24 processor configurations, where  $42 * 42 * 42$ ,  $42 * 42 * 42$  and  $48 * 48 * 48$  link cells were used instead because the number of link cells in each dimension must be equal. For example, in the  $2 * 3 * 4 = 24$  processor configuration, 3 does not divide 40, and 48 is the next common multiple.

The solid line on the graph shows the actual time taken to perform the simulation for a given number of processors. The dotted line is the reciprocal of the time taken (rescaled to fit on the same axis), or the rate of computation. For a small number of processors, the rate of work scales linearly with the number of processors, as there is relatively little time spent passing messages between nodes, compared to the amount of time spent in calculation. For this simulation, the optimum number of processing nodes appears to be around 24. The rate of work here is slightly exaggerated, since a higher number of link cells are used, so the speed of calculation increases anyway (see section 4.1.1) but even so, the 32 processor configuration is still slower than using 16 processors.

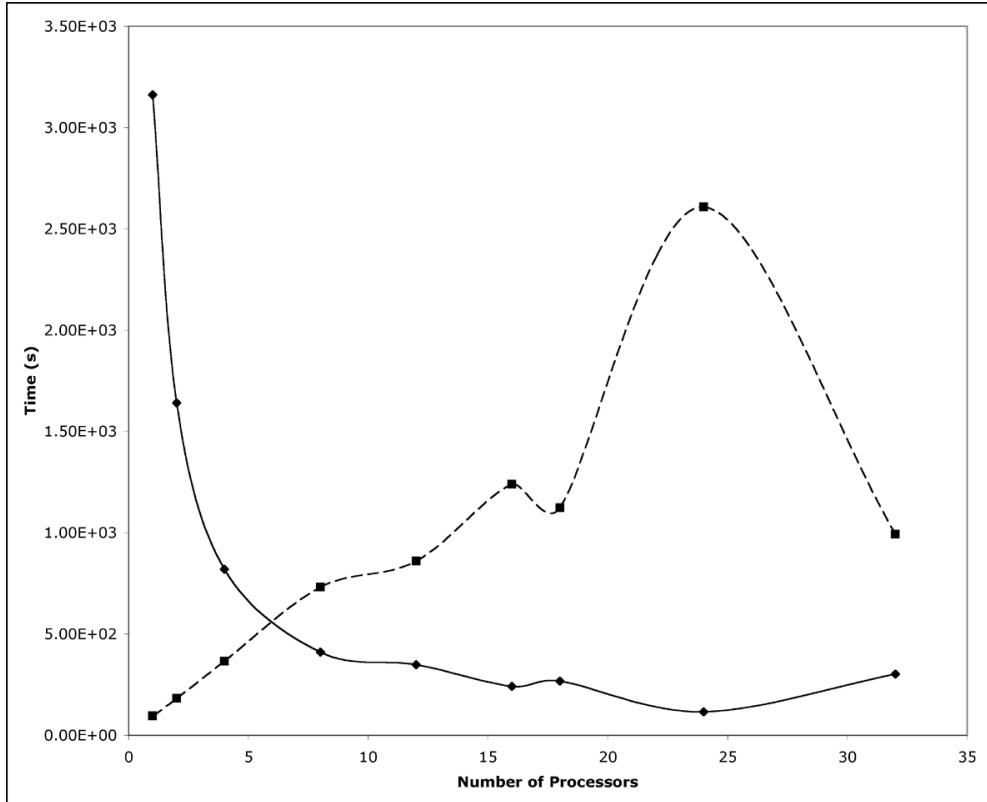


Figure 4.3: Scaling with many Processors

Each node in the 32 processor configuration has 2000 core link cells which it performs computation on and 1168 skin cells which must be updated by message passing each time step. This gives a ratio of skin cells to core cells of 0.58. By way of comparison, in the 16 processor configuration, this ratio is 0.45, and for 4 processors, it is only 0.21. Most of the computations of this size done in this project were run in an 8 processor configuration, which gives a reasonably efficient use of processor time, while still being fast enough that simulation runs can be completed in a matter of hours rather than days. As an aside, using only 8 processors is also an advantage because the queueing system in place on `lomond` (Section 1.4) always has slots available for 8 processor jobs, while anything larger may have to wait until evenings or weekends to be scheduled.

## 4.2 Titanium

To check that the Finnis-Sinclair potential for Titanium was implemented correctly, two checks were made. Firstly, when MDCASK generates the initial lattice configuration it calculates the total potential energy of the system. Dividing this

by the number of atoms in the system gives the crystal energy, or binding energy, which is negative, indicating that the crystal lattice is a bound state. This means that without energy being input to the system, atoms will stay in the crystal structure as it is a state of lower energy than being a free particle  $E = 0$ . A crystal with energy  $-E$  eV/atom would require  $E$  eV of energy per atom to free the atoms from the crystal structure (i.e. vapourising the sample). For titanium in an HCP lattice with the nearest neighbour separation of 2.9665 Å, the crystal energy is  $-4.85283$  eV/atom, compared with the value of  $-4.8527522$  eV/atom calculated by MDCASK. This is correct to 5 significant figures. The difference in values could arise from the fact that the potential is stored in a lookup table, so that values will be an approximation to the actual potential function.

Secondly, a series of simulations were run to examine how the energy of the crystal varied with temperature. A system of 4000 titanium atoms was set up at various temperatures and allowed to come to equilibrium. The total energy of the system and the equilibrium temperature were then measured. Figure 4.4 shows the relationship between temperature and energy. The constant volume heat capacity  $C_v$ , is given by  $\frac{\partial E}{\partial T}_v$ , the gradient of the graph. Converted to SI units, this is  $520.2 \text{ J kg}^{-1} \text{ K}^{-1}$ . The accepted value at  $0^\circ\text{C}$  is  $523 \text{ J kg}^{-1} \text{ K}^{-1}$ .

These two results confirm that the potential is implemented correctly, and are close enough to the true values to be confident that it is a good description of titanium under reasonable conditions.

## 4.3 Radiation Damage Experiments

Due to computational and disk space requirements on `lomond`, it was difficult to simulate systems larger than 256000 atoms. The largest energy cascade that can be contained within this size of simulation is approximately 1 keV. So this study was limited to the low energy regime described in section 1.3, and no evidence of defect clusters was found in the final configurations generated.

For each of the radiation damage experiments, a system of titanium atoms was set up in an HCP lattice at a particular temperature, and allowed to evolve towards equilibrium for some time before the PKA was given its initial impact energy. The direction of motion of the PKA was chosen such that it did not coincide with any of the primary planes in the HCP lattice. The amount of time allowed before the impact depended on the size of the system as larger systems take longer to reach equilibrium, but was typically 0.5 ps for 108000 atom systems and 0.75 ps for 256000 atom systems. After impact, the system undergoes a cascade of collisions (described in section 1.3) and eventually stops in some final state, which is used for taking measurements. The exact time at which the cascade can be

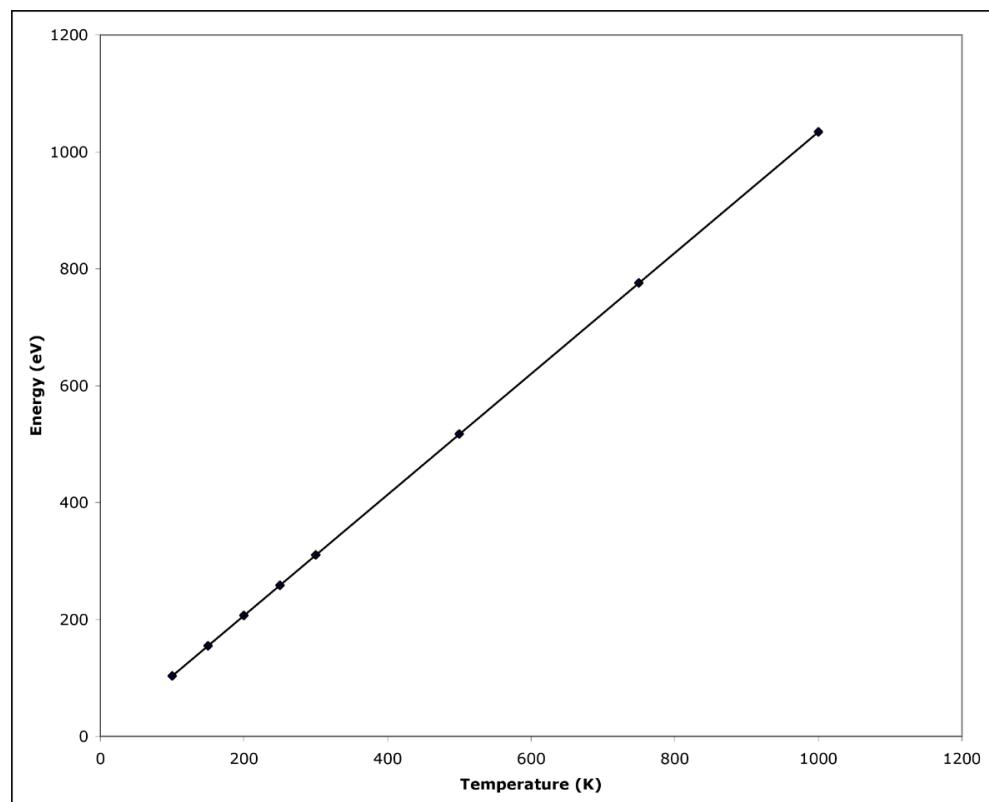


Figure 4.4: Energy and Temperature for titanium

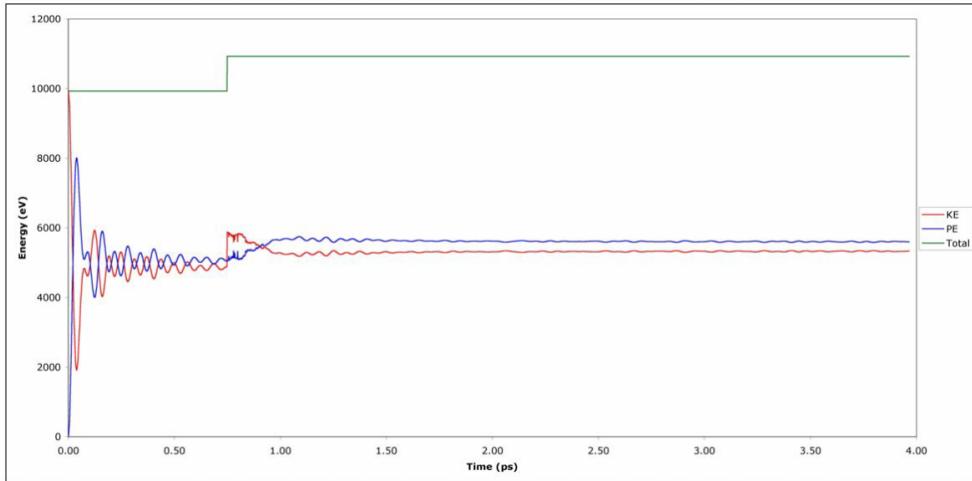


Figure 4.5: KE and PE against Time

said to have ended is not precise, as defects will continue to migrate in the lattice for a long time. However, for the larger system, the simulation was run for 3750 time steps after the PKA event, which corresponds to about 3.25 ps, depending on the variable time step. The smaller system was run for 2750 time steps after impact, which is about 2.5 ps of simulation time. In this approximation, the majority of the defects are static for the last 0.5 ps of simulation, indicating that this is a reasonable approximation to a ‘final state’ for the cascade.

The smaller system size was used for PKAs with energy  $E_{pka} < 500\text{eV}$  and the larger for  $500\text{eV} \leq E_{pka} < 1\text{keV}$ .

Figure 4.5 shows a typical graph of the potential and kinetic energies of the system against time. The system can clearly be seen to be reaching equilibrium as the oscillations of KE and PE damp down before the PKA energy is inserted and the KE increases, before the system eventually reaches equilibrium again. The total energy is also shown, indicating that (apart from the PKA energy input) energy is conserved throughout the simulation.

### 4.3.1 Temperature

The first series of simulations show how the defect production is affected by temperature. The system studied was a block of 108000 titanium atoms in an HCP lattice. The crystal was bombarded with a 500eV PKA and the number of vacancy-interstitial (Frenkel) pairs in the final configuration was measured. The average distance of the surviving defects from the point of PKA impact was also measured. 34 simulations were made at a range of temperatures from 50K to 600K, and the results are shown in figure 4.6.

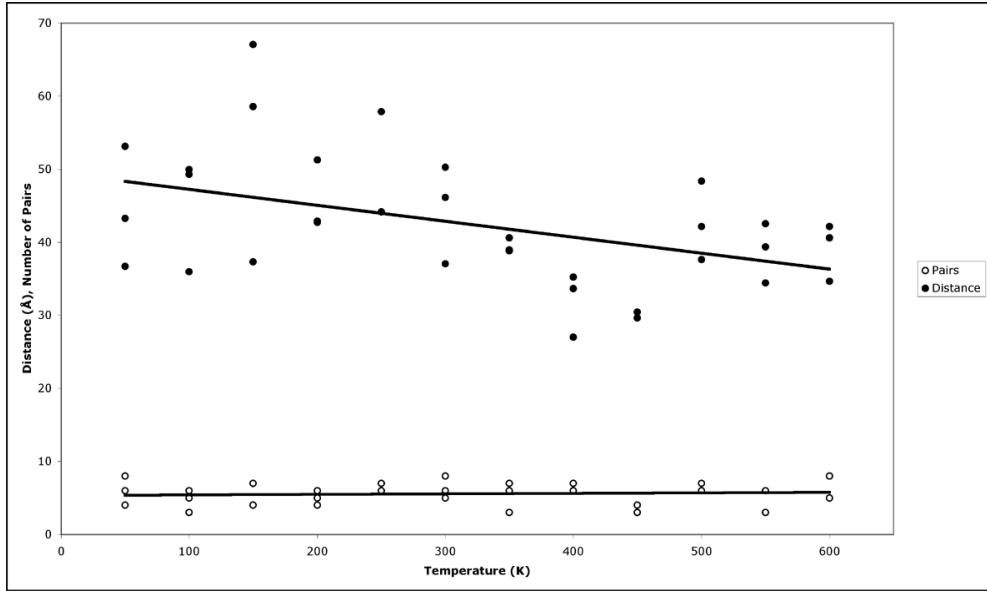


Figure 4.6: Surviving Frenkel Pairs and Average Defect Distance against Temperature

The constant trend of the number of surviving Frenkel pairs shows that the number of defects created by a radiation damage cascade does not depend on the temperature. As a perfect lattice is a minimal energy arrangement, each defect in the lattice must correspond to an increase in the lattice energy. With a fixed amount of energy input by the PKA, it is therefore expected that a fixed number of defects will be created. According to Gao *et al* [15], increased temperature causes fewer defects to be produced, due to recombination of interstitials and vacancies as the defects are mobile for longer at higher temperatures. However, this effect only causes a reducing of a few percent every hundred degrees, and thus is not noticeable for such small defect yields and temperature range.

The average distance of defects from the initial PKA position shows a decreasing trend with temperature. This confirms the result noted by Bacon *et al* [5] that “cascades tend to have a more compact from at higher temperature”.

### 4.3.2 Energy

The second series of simulations show how the final defect arrangement is affected by the energy of the PKA. The system studied was an HCP lattice of titanium atoms, with the size of the lattice depending on the energy of the PKA as stated in section 4.3. The system was set up at 300K and bombarded by a PKA with energy varying from 100eV to 1keV. As before, the number of surviving Frenkel pairs and the average distance of defects from the point of impact was measured

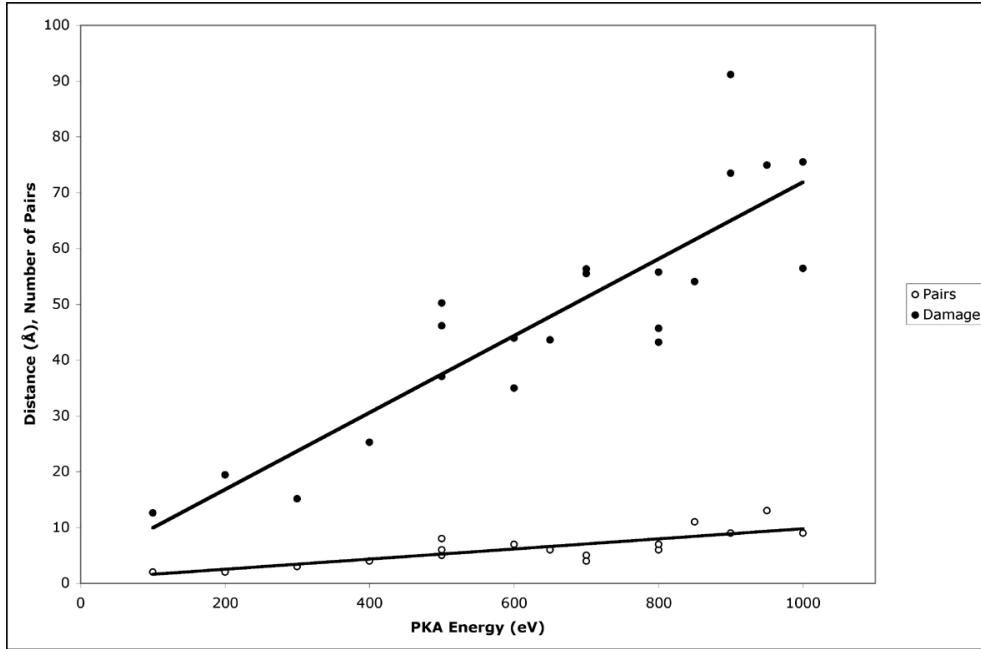


Figure 4.7: Surviving Frenkel Pairs and Average Defect Distance against PKA Energy

in the final configuration. The results of 21 simulations are shown in figure 4.7.

This graph shows that the number of surviving Frenkel pairs increases with the energy of the PKA. As mentioned in section 4.3.1, this is due to the fact that every defect created results in an increase in the energy stored in the lattice. This energy is supplied by the kinetic energy of the PKA, so it follows that the higher the PKA energy, the more defects must be created to dissipate this energy. An empirical formula for the number of defects produced at a given energy is shown by Bacon *et al* [16] to be

$$N = 6.01(E_{pka})^{0.8}$$

for titanium. Figure 4.8 shows how this formula compares with the results obtained from this series of simulations. The curve appears to be a plausible description of the data but is slightly low. This may be because given a long enough simulation run, more of the defects may eventually recombine.

The average distance of defects from the point of impact also increases with the energy of the PKA. After impact the energy of the PKA must be dissipated into the crystal until the extra energy per atom is no longer enough to displace atoms from their lattice sites, and the cascade ends. For a larger energy input, the volume over which the energy must dissipate before defects stop forming must

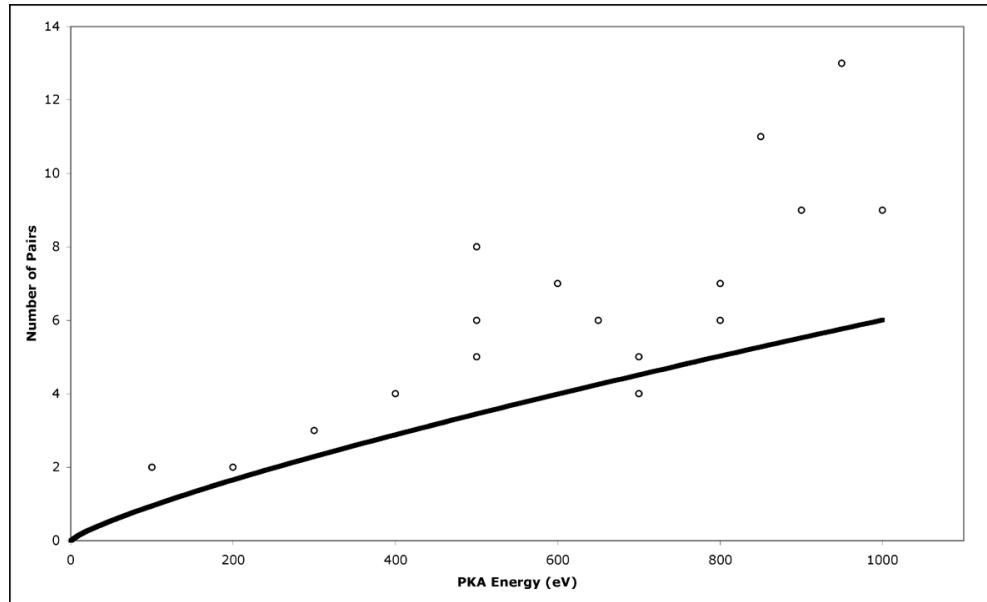


Figure 4.8: Surviving Frenkel Pairs against PKA Energy

therefore be greater, hence the distance which defects can reach from the initial point of impact increases.

# 5. Conclusion

## 5.0.3 Discussion

The first objective of this project was to adapt MDCASK to simulate titanium using a Finnis-Sinclair potential. As well as implementing the new potential, this also required generalising the initialisation routine to allow titanium's hexagonal close-packed structure to be used. Secondly, the scaling behaviour of MDCASK was to be examined on both serial and parallel computer systems. Also, software was to be developed to analyse the raw data generated by MDCASK and convert it to information that could be used to visualise the results of simulations.

Once these objectives had been achieved, MDCASK was to be used to carry out radiation damage simulations of titanium, under varying conditions of temperature and impact energy. The results of these simulations were then to be analysed and visualised at the atomic level to investigate defect production in titanium crystals under irradiation.

Several difficulties were encountered during the execution of the project. Firstly, the lack of documentation of MDCASK meant that it was difficult to understand the effects that the various parameter of the code would have on simulations. For example, it took some time to find the most efficient size of link cells, such that the code would run as quickly as possible, but that the cells were not so small that atoms did not correctly find all their neighbours. Because of this, the report includes a fairly detailed section documenting the input and output of MDCASK, which would be of use to another user of MDCASK who did not want to have to read the code to discover what a particular variable did.

The serial version of MDCASK was simple to compile and execute on several operating systems including Darwin 7.8 (Mac OS X), Redhat Linux 7.1, and Solaris 2.9. However, there were two major issues with the parallel code that delayed the use of MDCASK in parallel. Firstly, the design of the code for broadcasting data between nodes made the assumption that data blocks declared one after each other would also be contiguous in memory, which is not actually guaranteed to be the case on `lomond`. Secondly, a bug was eventually found in the `teacup` routine which was causing communication between nodes to return incorrect values, resulting in the timestep always being set to its smallest possible value. Although this does result in a physically correct simulation, it takes far too long to evolve the system to equilibrium before and after the radiation damage event, when a long time step is needed. Because of this many of the early experiments in the project, including most of the experiments investigating defect production at varying temperatures, were executed by running many different experiments at

once as background jobs on separate workstations in the Physics CPLab. Doing this caused each simulation to take longer which meant that the iterative cycle of experimenting and solving problems was also longer. Furthermore, because the CPLab is a shared resource, several simulations were killed during execution because another user was using the same machine.

Another issue was that the potential as given was unsuitable for simulating radiation damage because it had been designed for calculating surface and defect energies and did not provide a hard enough repulsion to high energy particles which might approach each other closely. This was simple to overcome by fitting a Biersack and Ziegler screened electrostatic potential to model the correct short range interaction.

Finally, once MDCASK was correctly running on `lomond` it became clear that because of the large number of users of the machine, disk space was very limited. For this reason it was not possible to store enough snapshots of each simulation run to generate a smooth video of the cascades. However, the final state of defects resulting from the cascades was recorded and used for analysis. In order to have enough simulations for the results to be reliable, it was necessary to concentrate on smaller systems, and thus lower energy cascades. This was also an advantage as because of the queuing system on `lomond` there are much fewer time slots available for jobs which take longer than 8 hours or use more than 8 processors.

Despite these setbacks, MDCASK was modified such that the initial objectives of the project were achieved, allowing a total of 52 radiation damage simulations of crystals of up to 256000 titanium atoms to be performed under varying conditions. Analysing the data files generated allowed the identification of defect locations within the crystal. Comparing the results of the simulations showed that the number and spread of defects increases with the PKA energy, broadly in line with the power law form in section 4.3.2, while increasing temperature does not affect the number of defects created, but tends to decrease the penetration of defects away from the point of impact. This confirms effects seen in a variety of metals to date including aluminium, iron, copper and zirconium, as described by Bacon *et al* [5] in a review paper.

Unfortunately, due to the low energies studied during the course of the project, defect clustering was not observed in the final states of any of the simulations. This was to be expected and is comparable with the lack of defect clustering observed by Wooding *et al* [17] in  $\alpha$ -zirconium at energies of less than 1 keV.

This project clearly shows that MDCASK is a very useful tool for studying molecular dynamics and in particular the effects of radiation damage to metals. Of course no system is perfect, but by describing solutions to the problems encountered, demonstrating how MDCASK can be customised to suit a particular type of simulation and providing a detailed guide to its use, the project will hopefully

provide a solid base for further use of MDCASK at Edinburgh University.

#### 5.0.4 Further Work

There are several areas of interest in this project that would merit further work. Firstly, because of the added versatility to MDCASK, it would be easy to do similar experiments for many other metals or alloys, provided that they can be described by a potential of the form described in section 1.2.

Another study would be to investigate the dependence of defect production on the incident angle of the PKA. For most directions it is expected that the resulting cascade should be broadly similar to those seen in this project. However, if the incident angle is parallel to a plane of atoms in the crystal, it is possible that the cascade would be confined to spread mainly in that plane, rather than spreading uniformly around the direction of the PKA's motion.

Thirdly, and most interestingly, would be to increase the energy of the PKA far enough to see sub-cascades and defect clustering occurring. Of course, high energy cascades need a large simulation volume to contain the resulting damage, as well as longer simulation runs to give the system time to return to equilibrium. These requirements are very demanding of both computational power and storage space. However, the recent installation of a IBM Blue Gene system at Edinburgh University would make it realistically possible to do some cutting-edge molecular dynamics research on an entirely new scale.



# Bibliography

- [1] Furio Ercolessi. A molecular dynamics primer. <http://www.fisica.uniud.it/ercolessi/md/md.pdf>, 1997.
- [2] M.S. Daw and M.I. Baskes. Embedded-atom method: Derivation and application to impurities and other defects in metals. *Phys. Rev. B*, 29(12):6443, 1984.
- [3] M.W. Finnis and J.E. Sinclair. A simple empirical n-body potential for transition metals. *Phil. Mag. A*, 50:45–55, 1984.
- [4] G.J. Ackland. Theoretical study of titanium surfaces and defects with a new many-body potential. *Phil. Mag. A*, 66(6):917–932, 1992.
- [5] D.J. Bacon, F. Gao, and Y.N. Osetsky. Atomic-scale computer simulation of primary irradiation damage effects in metals. *J. Computer-Aided Materials Design*, 6:225–237, 1999.
- [6] R.E. Voskoboinikov, Y.N. Osetsky, and D.J. Bacon. Atomic-scale simulation of defect cluster formation in high-energy displacement cascades in zirconium. *J. Phys.:Condens. Matter*, 16, 2004.
- [7] Aaron Harwood. Embarrassingly Parallel. 2004.
- [8] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard. <http://www mpi-forum.org/docs/mpi-11-html/mpi-report.html>, 1995.
- [9] Sun Microsystems Inc. Sun ONE Grid Engine Administration and User’s Guide. <http://www.sun.com/products-n-solutions/hardware/docs/pdf/816-2077-12.pdf>, 2002.
- [10] Lawrence Livermore National Laboratory. The MDCASK Benchmark Code. <http://www.llnl.gov/asci/purple/benchmarks/limited/mdcask/>, 2002.
- [11] Gear. *Numerical initial value problems in ordinary differential equations*. Prentice Hall, 1973.
- [12] J.F. Ziegler, J.P. Biersack, and U. Littmark. *The Stopping and Range of Ions in Solids*. Pergamon Press, 1985.
- [13] Arrays (Java 2 Platform SE v1.4.2)  
. [#sort\(java.lang.Object\[\],%20java.util.Comparator\).](http://java.sun.com/j2se/1.4.2/docs/api/java/util/Arrays.html)
- [14] QTSuperImageSequencer 1.1 - VersionTracker  
. <http://www.versiontracker.com/dyn/moreinfo/macosx/22193>.

- [15] F. Gao, D.J. Bacon, P.E.J. Flewitt, and T.A. Lewis. The effects of electron-phonon coupling on defect production in  $\alpha$ -iron. *J. Nucl. Mater.*, 249:77, 1997.
- [16] D.J. Bacon, A.F. Calder, F.Gao, V.G. Kapinos, and S.J.Wooding. Computer simulation of defect production by displacement cascades in metals. *Nucl. Instrum. Meth. B*, 102(1):37–46, 1995.
- [17] S.J. Wooding, L.M. Howe, F. Gao, A.F. Calder, and D.J. Bacon. A molecular dynamics study of high-energy displacement cascades in  $\alpha$ -zirconium. *J. Nucl. Mater.*, 254:191–204, 1998.

# Appendix A. Input Files

## A.1 mold.in

```
MDCASK
Basic MD simulation of an orthorhombic lattice with F-S (Ti)

4.173 7.228 6.627          ALATT (in ANGSTROMS) BLATT CLATT
.T   .T   .T                  PBCX PBCY PBCZ (PERIODIC BOUNDARY CONDITIONS)
.F   0                         REMOVELAYERS NSURFLUS
.F   0                         STATIC NSTATIC
.F   .F   .F   0              TEMPCONT ZTC XYTC NTCLAYERS
.F   0                         SCALEVELOCITY NSCALE
2                           NELEM
47.90      47.90            MASS (Ti) MASS (Ti)
1.0E-15                DELTA (TIMESTEP IN SECONDS)
.F
0.1E-14 0.1E-16 0.10        CHNGDT
600.00000001               DTMAX DTMIN DXMAX
10.0    0.0    0.0           TEMPRQ (TEMPERATURE IN K)
0.0     10.0   0.0           B0
0.0     0.0    10.0          B0
500 500 0                   NSTEPS NWRCONF NIN
.T 500 -8.8                 WCONF NPRINT EPCUTOFF
-12345                      SEED
.F
5000.0 95.0 32.0           INCLUDEPKA
-2 0.00 0.00 0.00 1        EPKA THETA PHIANG
1                           NPKA XPKA YPKA ZPKA INPKA
IDNPKA
```

## A.2 eamdata

Ti functions (F-S potential)

```
22      47.90      4.173
0.00102      0.0020      5.1000000000000005
25.887386250471003      -10.832537432434474      25.887386250471003
```

```
-10.832537432434474
25.87633822545922 -10.8302567045316 25.87633822545922
-10.8302567045316
25.86529252680151 -10.827975953816651 25.86529252680151
-10.827975953816651
```

### A.3 latt.in

```
4 ATOMS IN BASIS
0.000 0.000 0.000 FRACTIONAL COORDS ( 0 .. 2 )
1.000 1.000 0.000
1.000 0.333 1.000
0.000 1.333 1.000
```

# Appendix B. Output Files

## B.1 mdyn.con

```
      500
NUMBER OF PKA =      0
  0.5000000E-12    3493.876      0.1015783E+09    46772.73
  36.03000          0.000000     0.000000
  0.000000          36.03000     0.000000
  0.000000          0.000000     36.03000
 -0.17771195       -0.20875305   -0.23206769
        4000

      1 1    -5.117    -4.970    -4.783 -0.2947E+01    0.7457E+00    0.2442E+03
  0.4499E+03        46
      2 1    -4.213    -5.395    -5.198 -0.3051E+01    0.1747E+00    0.2916E+03
  0.1809E+03        40
      3 1    -3.055    -4.826    -4.940 -0.3111E+01    0.1116E+00    0.5712E+02
  0.5655E+02        44
      4 1    -2.390    -4.639    -4.898 -0.2757E+01    0.1682E+00    0.3544E+03
  0.6171E+03        40
      5 1    -0.889    -4.958    -4.988 -0.3099E+01    0.1214E+01    -0.5877E+02
  0.1872E+02        44
      6 1    -0.099    -4.622    -4.826 -0.2603E+01    0.9230E+00    0.4675E+03
  0.5532E+03        42
      7 1     0.944    -4.836    -5.245 -0.2850E+01    0.7482E+00    0.5145E+03
  0.8533E+02        45
      8 1     1.928    -4.954    -4.809 -0.2831E+01    0.5720E+00    0.3369E+03
  0.2287E+03        44
      9 1     3.290    -4.907    -5.386 -0.2129E+01    0.4221E-01    0.6207E+03
  0.7262E+03        43
```

## B.2 mdyn.ene

```
  0.10000E-02    0.79924E+04    0.41328E+04    0.47126E+01    0.41375E+04
 -0.98225E+04    0.88249E+06
  0.20000E-02    0.79650E+04    0.41186E+04    0.18879E+02    0.41375E+04
 -0.98225E+04    0.13218E+07
```

0.30000E-02	0.79193E+04	0.40949E+04	0.42522E+02	0.41375E+04
-0.98225E+04	0.17586E+07			
0.40000E-02	0.78551E+04	0.40618E+04	0.75676E+02	0.41375E+04
-0.98225E+04	0.21919E+07			

# Appendix C. Data

## C.1 Serial Performance Test

Atoms	CPU Time
864	14.73
1372	20.56
2048	31.9
2916	43.83
4000	61.69
5324	78.32
6912	99.82
8788	130.02
10976	157.97
13500	198.55
16384	238.57
19652	285.49
23328	343.14
27436	403.06
32000	471.11

nlc	Time
64	1079.25
125	563.92
216	346.23
343	234.06
512	169.42
729	129.38
1000	104.78
1331	87.21



# Appendix D. Fitting the Biersack and Ziegler potential

The conditions that the potential and its first derivative must be continuous at  $r_1$  and  $r_2$  give rise to four equations:

$$\begin{aligned} V_{Ti}(r_2) &= \exp(B_0 + B_1 r_2 + B_2 r_2^2 + B_3 r_2^3) = a \\ \frac{dV_{Ti}}{dr}|_{r=r_2} &= (B_1 + 2B_2 r_2 + 3B_3 r_2^2) \exp(B_0 + B_1 r_2 + B_2 r_2^2 + B_3 r_2^3) = b \\ V_{screened}(r_1) &= \exp(B_0 + B_1 r_1 + B_2 r_1^2 + B_3 r_1^3) = c \\ \frac{dV_{screened}}{dr}|_{r=r_1} &= (B_1 + 2B_2 r_1 + 3B_3 r_1^2) \exp(B_0 + B_1 r_1 + B_2 r_1^2 + B_3 r_1^3) = d \end{aligned}$$

Substituting  $r_1 = 1$  and  $r_2 = 2$ , gives a system of linear equations which can be written in matrix form as

$$\begin{bmatrix} 1 & 2 & 4 & 8 \\ 0 & 1 & 4 & 12 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} = \begin{bmatrix} \ln(a) \\ \frac{b}{a} \\ \ln(c) \\ \frac{d}{c} \end{bmatrix}$$

This can be solved by inverting the matrix to give equations for  $B_0 \dots B_3$ .

$$\begin{bmatrix} 5 & -2 & -4 & -4 \\ 12 & 8 & -12 & 5 \\ -9 & -5 & 9 & -4 \\ 2 & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} \ln(a) \\ \frac{b}{a} \\ \ln(c) \\ \frac{d}{c} \end{bmatrix} = \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

Inserting appropriate values for  $a$ ,  $b$ ,  $c$  and  $d$  obtained from evaluating the potentials and their derivatives at  $r = 1.0$  and  $r = 2.0$  gives the result that  $B_0 = 7.112596228$ ,  $B_1 = -4.393990755$ ,  $B_2 = 0.9407536255$ ,  $B_3 = -0.1197498592$ .