Multi-scale Material Growth and Erosion in Extreme Environments

Andrew M Bell

PhD Thesis, 2019

University of Strathclyde

Department of Chemical and Process Engineering

Declaration of Authenticity and Author's Rights

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Signed: A. Bell

Date: 20/9/18

Acknowledgements

I would like to thank my primary supervisor Dr Paul Mulheran. His guidance and supervision throughout my PhD were invaluable. It would not have been possible for me to complete my PhD without his encouragement. I would also like to thank my original secondary supervisor Prof Richard Brown and my second secondary supervisor Dr Tom Scanlon. Their alternative perspective on the work provided a useful insight into other things to consider.

Furthermore, I would like to thank the Chemical and Process Engineering department's IT and administrative staff for their help with any issues I had in the department. I would also like to thank the ARCHIE-WeSt High Performance Computer that was used to perform the computational work for my Molecular Dynamics simulation.

Thank you to the friends I have made over the course of my PhD. Particularly, I would like to thank David C., Scott, Javier, Onorio, Dorin, Evan, Hrvojka, Rab, Martin, Daria, David M., Andrew M., Rueben and Aditi. They made my PhD much more fun and made it enjoyable to come and work in the department.

I would also like to thank Dr Zimbitas and Dr Fletcher. Their encouragement when I was struggling with writing this thesis allowed me to regain motivation and continue.

Finally, I would like to thank my family for supporting me throughout my PhD. I would especially like to thank my Mum and my Dad whose encouragement and support over the course of my life has made me into the person I am now.

Contents

Conference Presentations

Bell, Andrew M; Brown, Richard E; Mulheran, Paul A - July 2015 CCP5 Summer School, University of Manchester, UK

I attended a 9 day summer school on computational methods of simulating chemical systems. While at the summer school, I presented a poster entitled "*Multi-scale Chemistry Modelling for Spacecraft Atmospheric Re-Entry*"

Bell, Andrew M; Brown, Richard E; Mulheran, Paul A - June 2016 ARCHIE-WeSt Showcase Workshop, University of Strathclyde, UK

Poster Presented: "*Multi-scale Chemistry Modelling for Spacecraft Atmospheric Re-Entry*" – Won a prize for best Postgraduate Student poster

Bell, Andrew M; Scanlon, Tom J; Mulheran, Paul A - April 2017 ISSC-21, Royal Northern College of Music, Manchester, UK

Poster Presented: "*Multi-scale Material Growth in Extreme Environments*"

Oral Presentation: "*Material Growth and Erosion in Extreme Environments*"

Bell, Andrew M; Scanlon, Tom J; Mulheran, Paul A - September 2017 CCP 5 AGM, University of Strathclyde, UK

Poster Presented: "*Multi-scale Material Growth in Extreme Environments*"

Abstract

This work sets out to develop a model to analyse surface growth during high energy deposition. This is important to applications such as hypersonics and thin-film growth. We considered Molecular Dynamics (MD), an atomistic model which captures key details but can only simulate small systems at short timescales due to computational cost. We also considered kinetic Monte Carlo (kMC), a mesoscale model lacking a lot of the fine detail but using a vastly reduced computational cost, allowing the analysis of larger systems over longer timeframes.

The many-body Sutton-Chen potential was employed in preference to the Lennard-Jones (a simple pairwise potential) in the MD code, capturing electronic density effects and how they affect the surface atom interactions. How the average surface height and surface roughness was affected by high energy atomic impingement was analysed for a variety of systems. A kMC code was then created that made use of the MD statistics to recreate the surface growth patterns, while allowing for much larger and longer simulations.

Using MD, the average surface height initially decreases before growing linearly. Meanwhile the surface roughness grows rapidly initially before increasing more slowly. Analysis of the effect of polar and azimuthal angles showed that the surface started eroding above a polar angle of 50°, and that using random or fixed azimuthal angles angle only affected the surface substantially at polar angles above 70°. Using kMC, a surface size of 56 by 28 lattice sites served well as a model system for the deposition of 2.5 monolayers while larger surfaces were required to avoid finite size effects during the deposition of 40 monolayers. We conclude that we have developed a model that can be used to simulate the evolution of a surface during high energy deposition, applicable to realistic sizes and timeframes.

Chapter 1 – Introduction

The work in this thesis focuses on surface evolution at the molecular scale, which is important for many different applications and processes. Two of these applications are detailed in sections 1.1 and 1.2 . To analyse surface evolution at the molecular scale, molecular modelling techniques are used and some of these techinques are discussed in detail in sections 1.3 -1.5 .

1.1 Hypersonics

Surface evolution can be important in hypersonic conditions when objects are hurtling through gases at velocities with a Mach number greater than 5. At these speeds, the chemistry of the gases becomes more significant. During atmospheric re-entry, spacecraft travel at speeds close to 10 km/s in rarefied air. At these speeds, molecules of Oxygen and Nitrogen can be dissociated, exited and ionized. The surface of the spacecraft can be impacted by these reactive atoms and ions and be ablated by their reactions with the surface. It is therefore important to simulate how the surface evolves due to the impacts and the reaction chemistry of the impacting atom, ion or molecule.

Due to the scarcity of the air, the Navier-Stokes equations used by Computational Fluid Dynamics (CFD) breaks down, requiring smaller scale methods to simulate in those conditions such as the Direct Simulation Monte Carlo method (DSMC) proposed by Bird in 1994 (1). Work by Scanlon et al (2) has looked at the reaction chemistry in hypersonic flows using a DSMC program called dsmcFoam (3) and using a Quantum-Kinetic chemistry model, also proposed by Bird (4), that wasn't used in previous versions of dsmcFoam. Scanlon et al investigated the hypersonic flow around a simple cylinder and compared with analytical solutions and with solutions produced by another DSMC code known as MONACO (5), which uses the Total Collision Energy chemical reaction model originally introduced in 1979 by Bird (6). The paper then shows that there was a good agreement between dsmcFoam, MONACO and the analytical solutions. Work by Palharini, Scanlon and White (7) builds upon the work by Scanlon et al (2) and investigates the structure of the flowfield inside cavities with chemically reacting hypersonic flows using the

dsmcFoam with the Q-K chemistry. In their work, they find the behaviour of cavities differs significantly from the behaviour in the continuum regime with phenomena being observed at a cavity length-to-depth (L/D) ratio of 3 in hypersonic flows that occurs between ratios of 10 and 14 for the continuum regime. At a L/D ratio of 5, the flow was able to deeply penetrate into the cavity, creating a situation that could be disastrous for re-entry vehicles with excessive heating of the structure beneath the thermal protection system. The use of DSMC in hypersonics is still under development and an improved version of dsmcFoam, referred to as dsmcFoam+, has been benchmarked by White et al. (8).

In DSMC simulations, a computational mesh is made, similar to CFD. Inside the mesh are simulation particles that represent millions of real atoms and molecules. By using particles, it is capable of simulating systems where gases no longer function as a continuous fluid unlike CFD. The cells of the mesh in DSMC is smaller than in CFD and is usually smaller than the mean free path, the average length a particle can travel without colliding with another particle. By making the cells smaller, collisions with the nearest neighbours are encouraged. The timestep used in DSMC is also smaller than CFD and smaller than the mean free time, which is the average time between collisions, to decouple particle movement from particle collisions.

An early aim of the work in this thesis was to try to couple the dynamics seen at the molecular level to the DSMC method to improve the accuracy of simulations using that method, allowing more accurate representations of the surface evolution of an ablative heat shield during atmospheric re-entry. To do this, the results obtained from MD could be used to augment the models used by DSMC for collisions between the surface and the particles. However, these aims evolved during the project and the aim is now to analyse how surfaces evolve during deposition more generally.

1.2 Thin-Film Growth

There are a number of processes used to grow thin films experimentally. These processes can be separated into chemical vapour deposition (CVD) or physical vapour deposition (PVD). As the name implies, CVD involves chemical reactions occurring to form the thin film on the target surface. As those processes rely more

on the chemical reactions and not on the impact of the thin film material on the target surface, simulations dealing with the dynamics of surface evolution are not as useful as simulations dealing with the reaction chemistry.

PVD, on the other hand, involves bombarding the target surface with the thin film material and can be achieved with a variety of methods. One method is pulsed laser deposition, which focuses laser pulses on the thin film material and causes a plume of high energy plasma to be ablated off of the thin film material. The plasma then travels to the target surface and is deposited. This method is not well-suited to simulation as the size of the deposited material is difficult to control, making comparisons to experiment difficult.

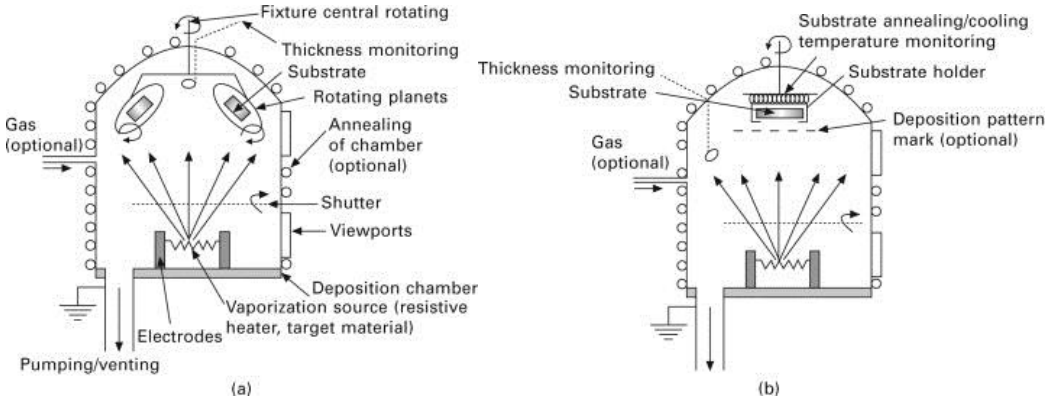


Figure 1: Schematic diagram (not to scale) of two setups for the vacuum thermal evaporation method of deposition for thin-film growth. Taken from (9)

Another method that can be used to achieve PVD is evaporation deposition, where the thin film material is evaporated and the vaporized atoms are directed to the target surface where they are deposited. Figure 1 shows two example set-ups for vacuum thermal evaporation where the thin film material is heated in a low pressure chamber to facilitate greater evaporation rates. Flash evaporation can also be used for deposition by continuously feeding a small amount of powder onto a pre-heated boat, causing it to rapidly evaporate.

A third method for PVD is magnetron sputtering. This works by using a strong magnetic field to ionize a gas in the magnetron chamber. These ions then bombard the thin film material with kinetic energies in the order of kiloelectronvolts (keV), causing the thin film material to be vaporised and sputtered towards the target surface, where it is deposited.

Magnetron sputtering and evaporation deposition are well suited to simulation as the stream of particles being deposited can be treated as atomistic impacts bombarding a surface with kinetic energies in the order of electronvolts (eV) and tens of electronvolts.

1.3 Modelling Methods

The most common methods of modelling simulations can be divided into four groups. These four groups are Quantum, Atomistic, Mesoscopic and Continuum. In Figure 2 the time and length scales that can be used by the different groups of simulation methods are outlined.

Quantum simulation methods such as Density Functional Theory (DFT) are ab initio methods that are capable of determining the electronic structure of atoms. These methods are incredibly accurate but are also incredibly computationally intensive, meaning they cannot simulate much more than 100 femtoseconds and simulations in the order of nanometers.

Atomistic methods such as Molecular Dynamics (MD) sacrifice some level of accuracy, looking only at the interactions between atoms but this reduces the computational cost of the simulations, allowing systems that can be hundreds of nanometers long or simulations that are hundreds of nanometers long. Above those points, the computational costs become too great even with the simplification of the simulations.

Mesoscopic methods such as kinetic Monte Carlo (kMC) and Coarse Grained methods simplify the simulations further. In kMC, the dynamics of the simulation are ignored and at each timestep, random numbers are used to determine if an event happened. kMC can be done using a lattice-based system or using off-lattice methods. kMC are very useful for simulating surfaces during growth and other phenomena where the changes between timesteps are not as important as changes over a long timeframe. Coarse Grained methods simplify simulations by grouping atoms into larger blocks and simulating the dynamics of the blocks instead. Coarse Grained methods are therefore more suited to simulating proteins and other large molecules such as MOFs. The simplifications used make mesoscopic methods less

suitable for simulations less than a nanometer or for times less than a nanosecond but vastly reduce the computational load, allowing even larger and longer simulations up to hundreds of micrometers and hundreds of microseconds.

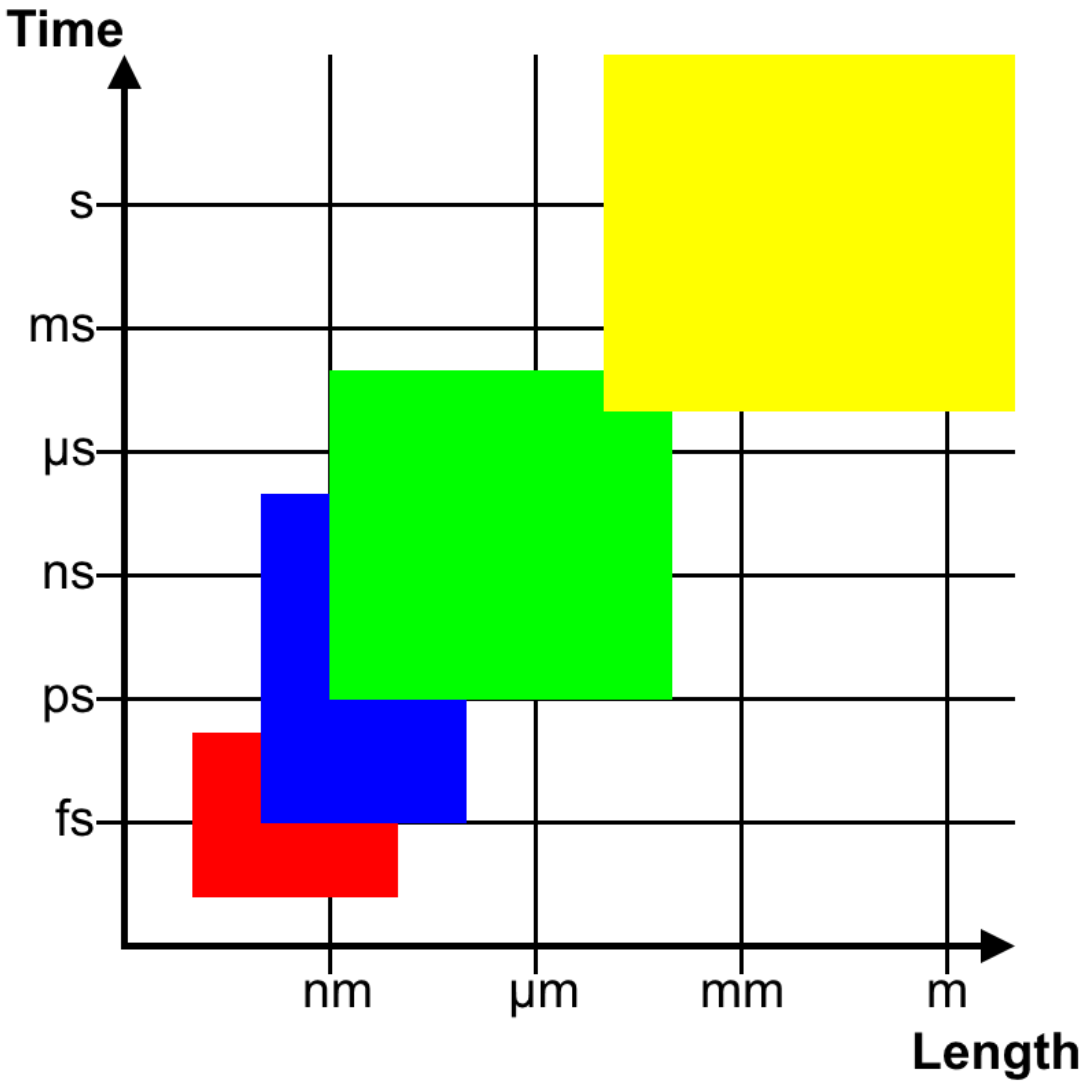


Figure 2: Diagram of the time and length scales that can be simulated with various simulation methods

Continuum methods such as CFD instead of calculating dynamics between atoms, splits the simulation area into regions and integrates the changes in those regions over time. This means that any fine detail is removed completely and continuum methods are unable to simulate anything below the order of micrometers and microseconds. This simplification allows Continuum methods to calculate length scales of meters and timescales of seconds.

In this work, we seek to use dynamics for surface growth seen at lower time and length scales to improve the accuracy of simulations at larger time and length scales, starting with MD as surface growth cannot be simulated at the timescales that are feasible for Quantum methods. From MD, we scale up to lattice-based kMC, which allows us to simulate growth at more useful length scales. Lattice-based kinetic Monte Carlo was chosen over coarse graining as the changes in the surface during deposition are too small to be coarse grained.

1.4 Molecular Dynamics

MD is a simple simulation method that uses a force field to determine how atoms are interacting with each other and uses these interactions to determine the dynamics of the system, advancing the system through time by a single timestep by moving atoms based on their kinetics once their interactions for other atoms are accounted for. This simple method can be used in a large variety of different applications. For example, MD can be used to simulate how a composite alloy of boron nitride nanotubes (BNNT) and aluminium behaves under tensile loading. This was investigated in a recent paper by Cong and Lee (10).

Figure 3 shows representative snapshots of an MD simulation of the alloy at various strains starting with the relaxed structure under no strain (a). At a strain of 0.09 (c), the aluminium structure has begun to distort and at a strain of 0.23 (d), the aluminium has ruptured and separated, while the BNNT remains intact. The BNNT was being tore apart at a strain of 0.28 (e) and had snapped at a strain of 0.32 (f).

Cong and Lee used MD to investigate how the alloy performed at BNNT thicknesses of (4,4) to (10,10) and for BNNT volume fractions of 1% to 5%. In their paper, they show that the BNNT plays a significant role in the alloys load bearing capabilities and that the alloy is much more resilient to strain than pure aluminium. Their MD simulations also suggest that the thickness of the BNNT is less important for the material strength than the volume fraction of the BNNT with the stress response of BNNT at all thicknesses being roughly similar at constant volume fraction while the stress response increased with increasing volume fraction in simulations using the same BNNT thickness. They also found that the elasticity of the alloy improved both by increasing the thickness of the BNNT and by increasing the volume fraction.

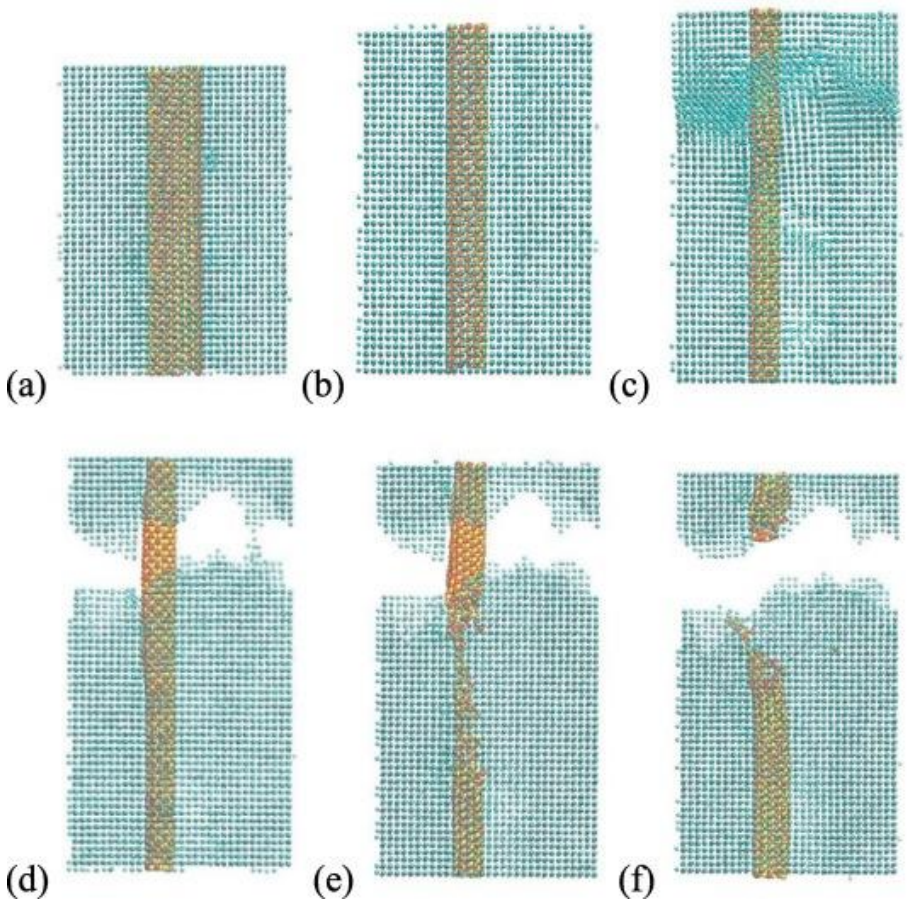


Figure 3: Snapshots of (10,10) BNNT-Al alloy MD simulations at strains of (a) 0, (b) 0.05, (c) 0.09, (d) 0.23, (e) 0.28 and (f) 0.32. Taken from (10)

MD worked well for the purpose of the paper by Cong and Lee because they were simulating on a nanometer length scale and a nanosecond timescale. To calculate the dynamics of the materials they used a variety of potential fields with a simple pairwise Lennard-Jones potential used to describe the interactions between the aluminium and the BNNT. A slightly more complex Tersoff potential (11) is used to describe how the BNNT interacts with itself and an even more complex pair functional is used to describe the aluminium's interactions with itself. Despite their relative complexity compared to pairwise potentials, these are all simple tools that provide powerful results in understanding the behaviours of materials.

MD has been used several times to simulate the mechanical properties of nanoscale structures, looking at the compression of a composite of aluminium and carbon nanotubes (CNT) (12) and the tensile loading of gold and platinum nanowires (13), a composite of aluminium and CNTs (14), composites of copper, zirconium and CNTs (15), epoxy-CNT composites (16) and BNNTs with bamboo-like joints (17).

In this work, we seek to investigate the surface growth of thin films during deposition. We analyse using MD, initially simulating with a simple Lennard-Jones pairwise potential but later move to simulating with a more complex Sutton-Chen pair functional. The simulations in this work are on a nanometer length scale and a nanosecond timescale.

1.4.1 Sutton-Chen

The Sutton-Chen potential (18; 19) can be used to investigate the properties of nanoparticles. A recent paper by van der Walt, Terblans and Swart investigated the properties of copper nanocubes (20). In the paper, they look at a variety of copper FCC nanocubes with a (100) surface from 3x3x3 to 15x15x15. For each cube, they calculated the average cohesive energy per atom at 0K for the perfect cube and each time after stripping a (111) layer off of the cube's corners and a (110) layer off of the cube's edges until they had an octahedron. For each nanocube, they chose the structure with the maximum average cohesive energy. They then calculated the average potential energy over time at a range of temperatures from 50K to 850K. They found that the 3x3x3 favoured a HCP structure at 50K and the 5x5x5 favoured HCP at 450K. For larger nanocubes, they found premelting, especially in the corners, at higher temperatures suggesting that the structure chosen is not as stable at higher temperatures.

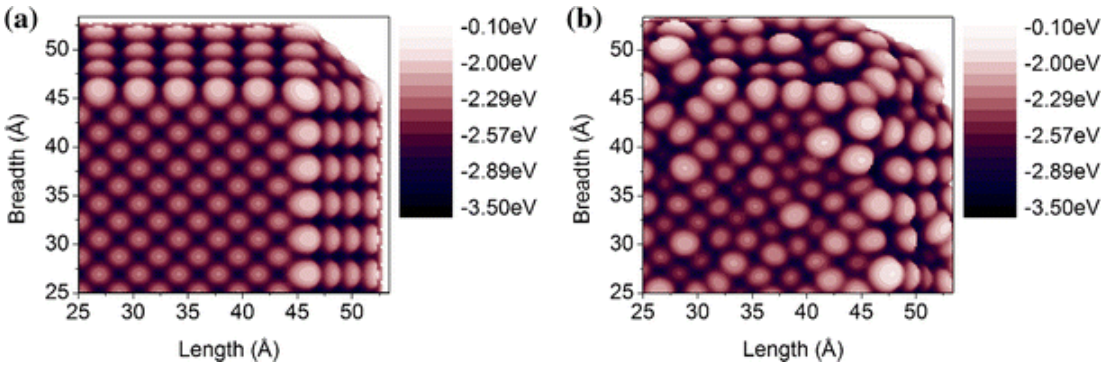


Figure 4: Contour plot of the surface energy of a corner of a 15x15x15 Cu nanocube with 3 layers stripped off at the edges at a temperature of (a) 0K and (b) 800K. Taken from (20)

Figure 4 shows a contour plot of the surface energy at 0K and 800K. In the 0K contour plot, the different surface orientations of (100) on the faces, (110) on the edges and (111) on the corners are easily identifiable. In the 800K contour plot, the

surface has become a lot more disordered and has expanded as the edge of the surface is no longer visible. The corners and edges are the most affected by the disorder with the different surface orientations being a lot harder to identify.

The Sutton-Chen potential was well suited to the work by van der Walt, Terblans and Swart because they were simulating a structure with metallic bonding, which means the structure is affected much more by the density of atoms surrounding an atom than materials like proteins.

Other work done with the Sutton-Chen potential includes finding the global minimum potential energy of transition metal clusters (21), analysing crack propagation through materials with impurities (22), manipulating nanoparticles on a gold substrate with a silver tip (23) and assess the validity of the Cauchy-Born hypothesis (24).

In this work, surface growth on a metal thin film will be simulated so the Sutton-Chen potential is an appropriate choice for the potential field used in the MD code.

Modifications of the Sutton-Chen such as Quantum Sutton-Chen by Çağin et al (25), have also been used to model glass formation, crystallisation and liquid properties for materials such as Nickel (26; 27; 28), Palladium-Nickel alloys (29; 30), Platinum-Palladium alloys (31; 32), Copper-Silver and Copper-Nickel alloys (33), and pure Palladium, Silicon and their alloys (34).

1.4.2 Deposition

When analysing the surface growth of thin films by deposition, the rates at which atoms attach to the surface and the rate at which they cause atoms from the surface to be ejected are important and how those rates are affected by changes in the conditions of deposition is also important. This is investigated for aluminium and nickel by Hanson et al. (35). In their paper, they simulated on a crystal that they claimed had 972 atoms of nickel or aluminium and had 12×8×9 atoms per side (which would be 864 atoms) with periodic boundary conditions in the x and y dimensions to represent an infinite surface and was equilibrated to 300K. Their simulations used a Voter/Chen potential, modified to reproduce DFT dimer potentials for interparticle spacings below 1.4Å for aluminium or 1.6Å for nickel.

They ran five simulations of a nickel (111) surface being impacted 50 times by nickel atoms and ran another five simulations on an aluminium (111) surface being impacted 50 times by aluminium atoms to determine sputter yield and sticking probability for a particular impact energy and incidence angle. The sputter yield and sticking probability are defined in equations 1 and 2:

$$Y_{Sp} = \frac{n_{Sp}}{n_{impact}} \quad (1)$$

$$P_{St} = \frac{n_{St}}{n_{impact}} \quad (2)$$

where Y_{Sp} is the sputter yield, n_{Sp} is the number of atoms sputtered, n_{impact} is the total number of impacts, P_{St} is the sticking probability and n_{St} is the number of impacting atoms that stuck to the surface. It is unclear how n_{Sp} and n_{St} are defined in the paper by Hanson et al.. In this thesis, n_{St} is incremented for each impacting atom that does not leave the surface before the subsequent impacting atom impacts the surface while n_{Sp} is incremented for each atom that is ejected from the surface excluding the most recent impacting atom. This means that if an impacting atom ejects an atom that has previously impacted and stuck to the surface, the ejected atom is counted as a sputtered atom and not as an impacting atom that did not stick to the surface.

In the simulations by Hanson et al., after each of the 50 impacts, the equations of motion were run for 0.6ps without a thermostat. The sets of five simulations were repeated for multiple impact energies and incidence angles. They also predicted sputter yields and sticking probability for an amorphous aluminium surface being impacted by aluminium atoms.

Another aspect of the sputter yield investigated by Hanson et al. is shown in Figure 5. In the figure, they analyse how the sputter yield at an incidence angle of 0° changes with increasing impact energy. The results obtained by Hanson et al. are plotted on the dotted line as open triangles though there may be an error with the figure as they appear to be closed in (b). This is compared to an empirical formula by Yamamura et al. (36), represented by the solid line. The results for aluminium, (a), are also compared to MD simulations by Hansen et al. (37), represented by closed squares, and experimental results by Hayward and Wolter (38), represented by closed triangles. The results for Nickel, (b), are compared to experimental results by Hechtl et al. (39), Hechtl et al. (40) and Fontell and Arminen (41), represented by

closed squares, closed triangles and closed diamonds, respectively. In (a), it can be seen that their simulations overestimate the sputter yield for aluminium but are still in reasonable agreement with experimental results. In (b), their simulations for nickel are in very good agreement with experimental results up to 75eV but, at 100eV and above, overestimate the sputter yield once again.

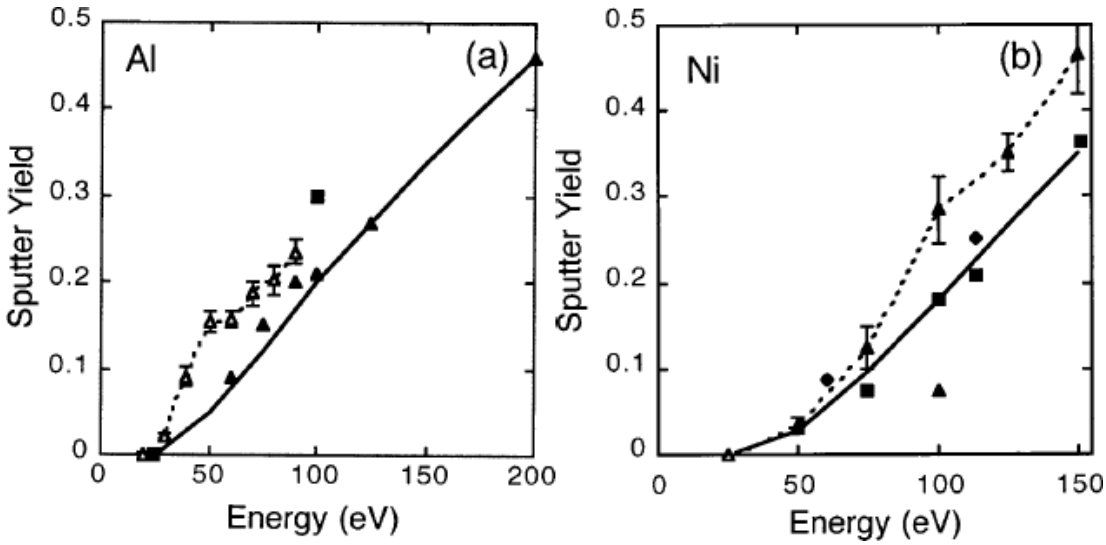


Figure 5: Sputter Yield at an incidence angle of 0° as a function of impact energy for (a) Aluminium and (b) Nickel. Taken from (35)

The results obtained by Hanson et al. of the sputter yield and sticking probability for nickel at various impacting energies and incidence angles are compared to results obtained by developmental versions of the code used in this work and are discussed in more detail in sections 3.1 and 3.2 of this work.

The work by Hanson et al. looked at Aluminium on Aluminium and Nickel on Nickel but there have been other papers that have analysed the sputter yields and sticking probabilities for Aluminium on Aluminium (37; 42; 43; 44; 45; 46) as well as other systems such as Copper on Copper (47; 48; 49), Argon on Copper (47) and noble gases on Magnesium Oxide and Magnesium Hydroxide (50; 51; 52).

1.4.3 Thin film growth with MD

Molecular Dynamics has also been used for simulating thin film growth via deposition, such as the work by Joe et al. (53), which analysed the growth of an amorphous carbon film using MD. In their work, carbon atoms were deposited on a

large diamond (001) slab with an impact energy of 75eV at a variety of incidence angles. There was also 4.25ps between depositions. The slab in their work had 4 fixed monolayers, 17 monolayers acting as a heat bath with a Berendsen thermostat (54) equilibrating the system to 300K and 20 free-moving monolayers. The number of free-moving layers was chosen to avoid deposited atoms being implanted in the layers acting as a heat bath.

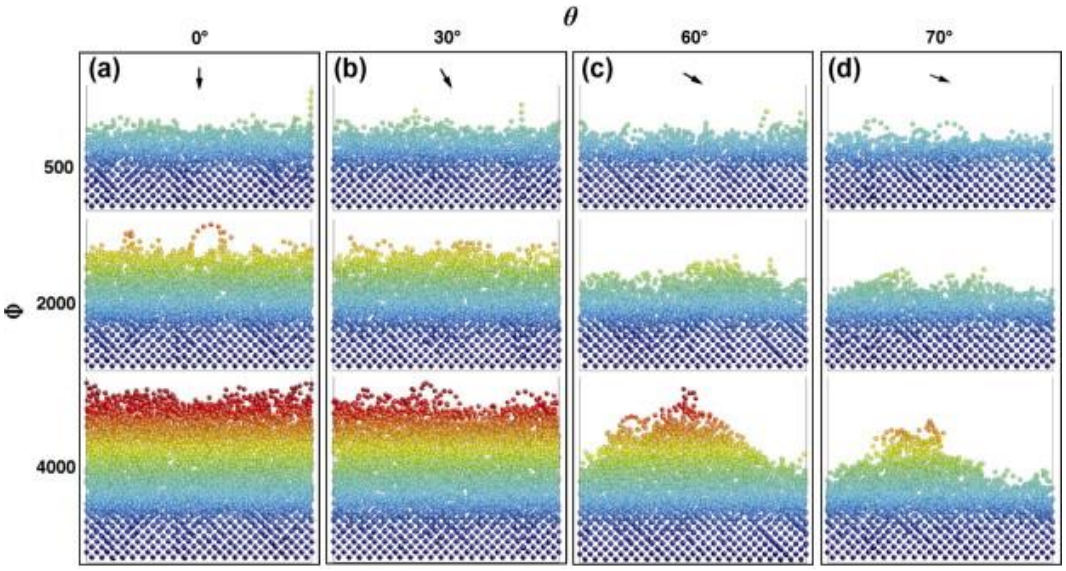


Figure 6: Cross-section of film after 500-4000 impacts, Φ, with an incidence angle, θ, of (a) 0°, (b) 30°, (c) 60° and (d) 70°. Colour encodes height of atoms. Taken from (53)

An example of their simulations is shown in Figure 6. Looking at the simulations with an incidence angle of 0° and 30°, the films that grew were mostly similar with the film grown from depositions at an incidence angle of 30° being slightly smaller. Both of these films were quite smooth with very little surface roughness. On the other hand, depositions at an incidence angle of 60° caused the film to grow slower and produced a significantly rougher surface with a large difference in height across the surface. The large bump also formed with depositions at an incidence angle of 70° but the bump was smaller. This bump formed as the high incidence angle caused the deposited atoms to favour growth in the direction of the source of atoms. This created a sloped surface that effectively created a shadow over the surface behind this slope. Furthermore, when impacting a sloped surface, depositions with a smaller incidence angle favour downhill movement of atoms while larger incidence angles cause uphill movement of atoms, meaning that if a slope was formed on a surface, depositions with smaller incidence angles would tend towards smoothing the surface while larger incidence angles reinforce the sloped surface.

Joe et al. verified that the structure seen for an incidence angle of 60° (Figure 6 (c)) was accurate by simulating 36000 impacts on a slab with a surface area that was approximately nine times larger. This larger slab produced a surface with features that were essentially the same as the features seen in Figure 6 (c).

The surface roughness's dependence on the incidence angle used during deposition has later been demonstrated in similar conditions experimentally by Lei et al. (55), who grew amorphous films of carbon on wafers of N-type silicon (111).

The work by Joe et al. differed from previous works as it looked at how the surface structure evolved at the different incidence angles while others just analysed the atomic structure. (56; 57; 58)

In the work carried out in this thesis, simulations of atoms impacting a slab will be carried out. Like the paper by Joe et al., the slab in these simulations will have fixed layers, thermostated layers acting as a heat bath and free-moving layers. Analysis of finite-size effects will also be carried out within this thesis. The simulations in the work in this thesis also observe the effects of changing the polar and azimuthal angles of impact. While the paper by Joe et al. used a set time between impacts designed to allow the system to be equilibrated between impacts, the optimum length of the delay was not known for the simulations carried out in this thesis so an analysis was performed to identify this optimum delay between impacts.

1.5 Kinetic Monte Carlo

1.5.1 Lattice-based kinetic Monte Carlo

Lattice-based kinetic Monte Carlo has been used by Chugh and Ranganathan (59) to simulate surface growth of gallium nitride (GaN) using the wurtzite crystal structure and a deposition flux that mimics the deposition in a molecular beam epitaxy chamber.

In their simulations, each lattice site of their HCP structure has its height stored and a variable is used to determine if it is occupied by gallium or nitrogen atom. The variable also contains information about the bonding of the occupying atom. The system used prevents a vacant site from occurring with the crystal and prevents

atoms from occupying the site above another atom of the same type. To simulate deposition, a random site is selected at regular intervals and an atom is deposited to that site. If both the deposited atom and the atom occupying the site are the same type, the three nearest neighbours are searched for an alternative site that the deposited atom can adsorb to. If none of those sites are suitable for adsorption, the deposition fails. To simulate diffusion, a catalogue of diffusion rates was calculated using the Arrhenius equation. The energy barriers to the various diffusion events were determined using DFT and the prefactor, which acts as the attempt frequency, was taken to be 10^{12} s^{-1} for both gallium and nitrogen. The rates obtained were then calculated relative to a rate at a reference energy and events with a lower barrier than the reference energy took place every timestep as they were given a probability of 1. The events with a higher barrier use the relative rate to determine the probability that they occur during a timestep.

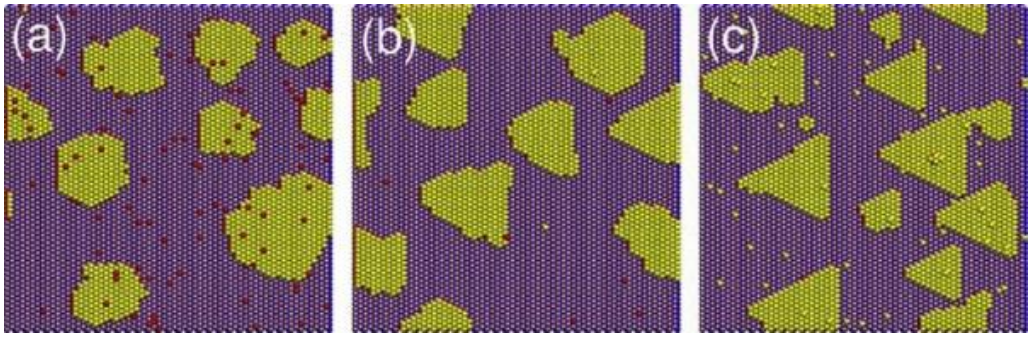


Figure 7: Different surface morphologies obtained after 0.6ML are deposited at a rate of 0.02ML/s when using Ga/N ratios of (a) 1:1.15, (b) 1:1 and (c) 1.15:1. Taken from (59)

Figure 7 shows how the ratio of gallium deposited to nitrogen deposited has a large effect on the surface morphology. When there is an excess of nitrogen, islands grow in irregular shapes and numerous nitrogen adatoms are isolated. The irregular islands under excess nitrogen flux are also seen experimentally. As the flux of gallium is increased, the islands become more triangular and the nitrogen adatoms are more likely to diffuse. The nitrogen diffuses as the barrier to diffuse decreases in the presence of a gallium adatom. When there is an excess of gallium, the islands are very triangular and there is numerous isolated gallium adatoms.

Lattice-based kMC works well for this paper as the system analysed is a very stable HCP crystal meaning atoms are likely to adhere to the HCP crystal structure making off-lattice kMC unnecessary.

As well as gallium and nitrogen deposition, lattice-based kMC has also been used to simulate chemical vapour deposition of Diamond (60; 61; 62; 63).

1.5.2 Off-lattice kinetic Monte Carlo

Off-lattice kMC is used by Clements et al (64) to model the porosity of ice, modifying the MIMICK model, which was developed from a model by Garrod (65) and designed for ices formed on dust grains in interstellar clouds and in proto-planetary disks, to replicate experimental results obtained on ices created in laboratory, such as the work by Brown et al. (66), and then comparing the laboratory ices to interstellar ices.

In their simulations, all molecules are tracked. Each timestep, only the molecule that was deposited or diffused has its position optimised by minimizing the sum of the pairwise potentials. This is done as minimizing all atoms each timestep is prohibitively expensive in terms of computational load.

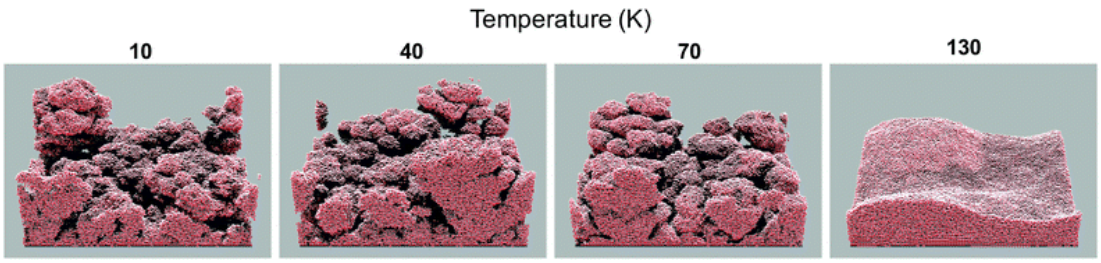


Figure 8: Simulated ices at various temperatures after the deposition of 25 monolayers at a deposition flux of 10^{13} molecules cm^{-2} s^{-1}. Taken from (64)

Figure 8 shows some simulated ices at various temperatures after deposition. The molecules deposited retained 70% of their energy after deposition and molecules that diffused retained 75% of their energy if diffusion causes them to settle in a potential well. It can be seen that the porosity of the ice decrease as the temperature of the system is decreased with the 130K ice having no visible pores while 10K, 40K and 70K ices have lots of pores and cavities all over them though the 70K ice looks smoother than the other porous ices.

When the model was used to model interstellar ices, it was seen that the ices were denser, suggesting the ices on interstellar objects are less porous than the ices created in laboratory conditions.

Using Off-lattice kMC instead of lattice-based kMC is necessary for these simulations as the system uses amorphous ice, which cannot be simulated reasonably by a lattice-based system. Even in a crystalline system, lattice-based kMC would not be able to accurately represent the formation of pores inside the structure, which is a key focus of the paper by Clements et al.

In this thesis, the system analysed is a highly-ordered crystal structure so we chose to investigate using a lattice-based kMC. Another advantage to using lattice-based kMC is that it is much less computationally intensive than off-lattice kMC as lattice-based kMC only needs to calculate all potential moves once at the start of the simulation or have them calculated before the simulation using a more accurate model while off-lattice kMC has to determine the rates for each event at every timestep. Off-lattice kMC also needs the position of molecules optimised after moving which adds more computational load as the sum of the potentials has to be minimized.

1.5.3 Saddle Points

To improve the scales that can use kMC, efforts have been made to accelerate kMC. One method used is to perform on-the-fly saddle point searches.

Saddle point searches look for events that are much more infrequent on the timescales used by MD, looking at states where the system has become stuck in a local minimum and identifying the paths out of this minimum. Once the paths are known, the rates for those paths can be determined with the Arrhenius equation and kMC can be used to select a path. Once a path is selected, the system moves to a new minimum, time is advanced and the process begins again.

A paper by Henkelman and Jónsson (67) aimed to accelerate kMC by combining it with the dimer method to perform on-the-fly saddle point searches. The dimer method, developed by Henkelman and Jónsson (68), starts with two images of the system, referred to as the “dimer”. The two images have almost the same set of coordinates but are separated by a fixed distance. The dimer is rotated along the midpoint to minimize the energy of the dimer as the direction that minimizes the energy is along the minimum curvature mode. The dimer is then translated along the minimum curvature mode. The net translational force acting on the images will pull

towards a minimum but this force is inverted as the saddle point lies at the maximum along the minimum curvature mode. To find multiple saddle points, the saddle point searches can be started at numerous random points surrounding a minimum.

Despite their usefulness, saddle point searches are computationally expensive so the use of these searches to accelerate time may restrict the length scale used in simulations. The computational cost can also grow significantly if there are a number of low barrier events that can cause the system to quickly flicker between a number of states without allowing the system to evolve. It is possible to reduce the computational cost by creating a catalogue of events storing previous saddle point searches and using methods that treat multiple states with low energy barriers as one large basin (69).

As the work in this thesis has focused on crystalline surfaces suited to lattice-based kMC, a method involving saddle point searches is unnecessary for the simulations performed. However, saddle point searches could be of future interest if the surface evolution of more amorphous systems is analysed.

1.6 Thesis Aims and Objectives

The aim of this thesis is to create a model to analyse surface growth during high energy deposition. To fulfil that aim, MD was used as it is an atomistic model that can capture key details during simulations. However, due to the computational cost, kMC was also used as it is a mesoscale model that while lacking fine detail, had substantially reduced computational costs. To create these models, the following objectives had to be met:

For the MD simulations:

- Select a suitable system design to regulate the temperature of the system
- Select a potential model that is suitable for the simulations of high-energy impacts on the system
- Optimize the number of calculations required for simulations while maintaining the integrity of the potential model
- Develop an algorithm to produce the systems being used for simulations

- Establish the optimum method to curtail long range forces without creating unrealistic behaviours in the potential energy and forces

For kMC simulations:

- Select data from the MD that the kMC can use to recreate the key features of the MD
- Develop algorithms that seek to either prevent features not seen in MD simulations or replicate features that are seen in MD simulations
- Make the algorithms adjustable to allow optimization of the algorithms based on how well the MD is replicated

By completing these objectives, it will be possible to analyse how the surface growth of a system is affected by the polar angle and kinetic energy of the incoming atom, the number of impacts on the evolving surface, and the size of the system, all using MD. With the kMC objectives met, it will be possible to reasonably replicate the MD results and then expand to much larger systems too computationally intensive for the MD model.

1.7 Summary

In this chapter, a number of different modelling techniques and some of the applications that these modelling techniques can be applied to have been reviewed. As demonstrated in section 1.4, MD allows the simulation of surface deposition and film growth with atomistic detail. One caveat is that the accuracy of the simulations is dependent on the accuracy of the force field. The computational cost also restricts the size of the system and the timescale that can be simulated over. On the other hand, kMC allows the simulation of surface deposition and film growth at the expense of atomistic detail as the dynamics are simplified to a set of rules. The advantage of the simplified dynamics is that the computational cost is vastly reduced allowing for much larger simulation over much longer timescales. The set of rules used in kMC requires the rates at which some events occur. Kinetic Monte Carlo is used over ordinary Monte Carlo as MC does not deal with the dynamics of the surface evolution and is better suited to determining the equilibrium state that a system would reach.

In this thesis, we set out to model the surface evolution during high-energy impacts. We begin by looking at MD, first detailing the some of the technical aspects used and then how the MD code was developed. Then the results obtained by running simulations with the MD code developed were analysed, looking for important behaviours and phenomena that are occurring at the atomistic level. The development of the kMC code is then detailed, including some comparisons to the MD made during development, until eventually 3 production versions of the kMC code have been produced. With the three production versions, many simulations were run to choose the version of the code and parameters that best represents the behaviours seen in the MD. Simulations scaling up the surface and number of impacts were then run with the system chosen, analysing how the surface evolution is affected by longer simulations and larger systems. Finally, we reflect on what conclusions can be drawn from the work in this thesis and what could be done in future work.

Chapter 2 – Molecular Dynamics Methodology

Earlier work completed by the author as part of his Master of Engineering degree predominantly used MD. Some of the theory used in the report for that project is reused in this section as it is relevant to the project although the material is substantially augmented as required here.

Two things are required in MD; equations of motion and a potential model. The equations of motion dictate how the atoms in a system move over time using the interactions between the atoms while the potential model dictates how the atoms are interacting at a particular moment in time using the position of the atoms at that moment in time. In this work, the velocity-Verlet algorithm, (70) which is a variation of Verlet's integration of Newton's equations of motion (the Verlet algorithm), was used to solve the equations of motion. The Verlet algorithm and its variations are similar to the Taylor series expansion of f(t+δt), which is,

$$f(t+\delta t) = f(t) + f'(t) \times \delta t + \frac{1}{2} f''(t) \times \delta t^2 + \cdots \quad (3)$$

where f(t), f'(t) and f''(t) are a generic function at t, the first derivative of the function at t and the second derivative of the function at t respectively and δt is a small change in t. In the Verlet algorithm, the function represents the positions of the atoms, the derivatives represent the velocity of the atoms and the acceleration of the atoms respectively and t represents the time. The Verlet algorithm eliminates the velocity term by combining the equation with the Taylor series expansion of r(t+(-δt)), which makes the final equation,

$$r(t+\delta t) = 2 \times r(t) - r(t-\delta t) + a(t) \times \delta t^2 \quad (4)$$

Here r(t) is the position of the atoms at time t, a(t) is the acceleration of the atoms at time t and δt is a small increment in time. To obtain the velocity of the atoms at time t, Verlet used,

$$v(t) = \frac{r(t+\delta t) - r(t-\delta t)}{2 \times \delta t} \quad (5)$$

where v(t) is the velocity of the atoms at time t. This could also be used to get the velocity at time t+δt by using r(t) instead of r(t-δt) and dividing by δt instead of 2δt. A problem with the Verlet algorithm is that the positions of the atoms must be known for two consecutive timesteps before it can be used. This is why the variation known as the velocity-Verlet algorithm was used instead. This variation doesn't eliminate the velocity term when calculating the position of the atoms at time t+δt, i.e.,

$$r(t+\delta t) = r(t) + v(t) \times \delta t + \frac{1}{2} a(t) \times \delta t^2 \tag{6}$$

To obtain the velocity at time t+δt, the acceleration of the atoms at time t+δt is recalculated using the potential model at the new time to get the forces acting upon the atoms at that time. The following equation is then used,

$$v(t+\delta t) = v(t) + \frac{a(t) + a(t+\delta t)}{2} \times \delta t \tag{7}$$

This could then be repeated to using t+δt as the new value of t.

2.1 Berendsen Thermostat

Typically in MD, when simulating an NVT ensemble as we do in this work, a thermostat is used to equilibrate the system to a certain temperature by manipulating the kinetic energy of the thermostatically controlled atoms. One relatively simple thermostat that could be used is the Berendsen thermostat. (54) This scales the velocity of all thermostatically controlled atoms by a factor, χ, which is determined by,

$$\chi = \left(1 + \frac{\delta t}{t_T}\left(\frac{T}{\tau} - 1\right)\right)^{\frac{1}{2}} \tag{8}$$

where δt is the timestep, t_T is the relaxation time, T is the desired temperature of the system and τ is the current kinetic temperature of the system, given by,

$$\tau = \frac{2\left\langle \frac{1}{2} m v^2 \right\rangle}{f k_b} \tag{9}$$

Here the numerator represents the average kinetic energy of the system, m and v are the mass and velocity of the atoms in the system being averaged, k_b is the Boltzmann constant and f is the number of degrees of freedom.

In the simulations performed in this work, the relaxation time was set to 100 femtoseconds. The value chosen has significant effects on the evolution of the surface as a longer value means that the system will take longer to relax to equilibrium while a shorter value would prevent the natural evolution of the surface dynamics. The value of 100 fs was chosen as this is a good compromise between relaxing to equilibrium in a reasonable timeframe and allowing the natural surface dynamics to evolve.

To make sure the Berendsen thermostat didn't adversely affect the kinetics of the system, the velocity distribution of a slab thermostated to 300K was compared to the expected velocity distribution, the Maxwell-Boltzmann.

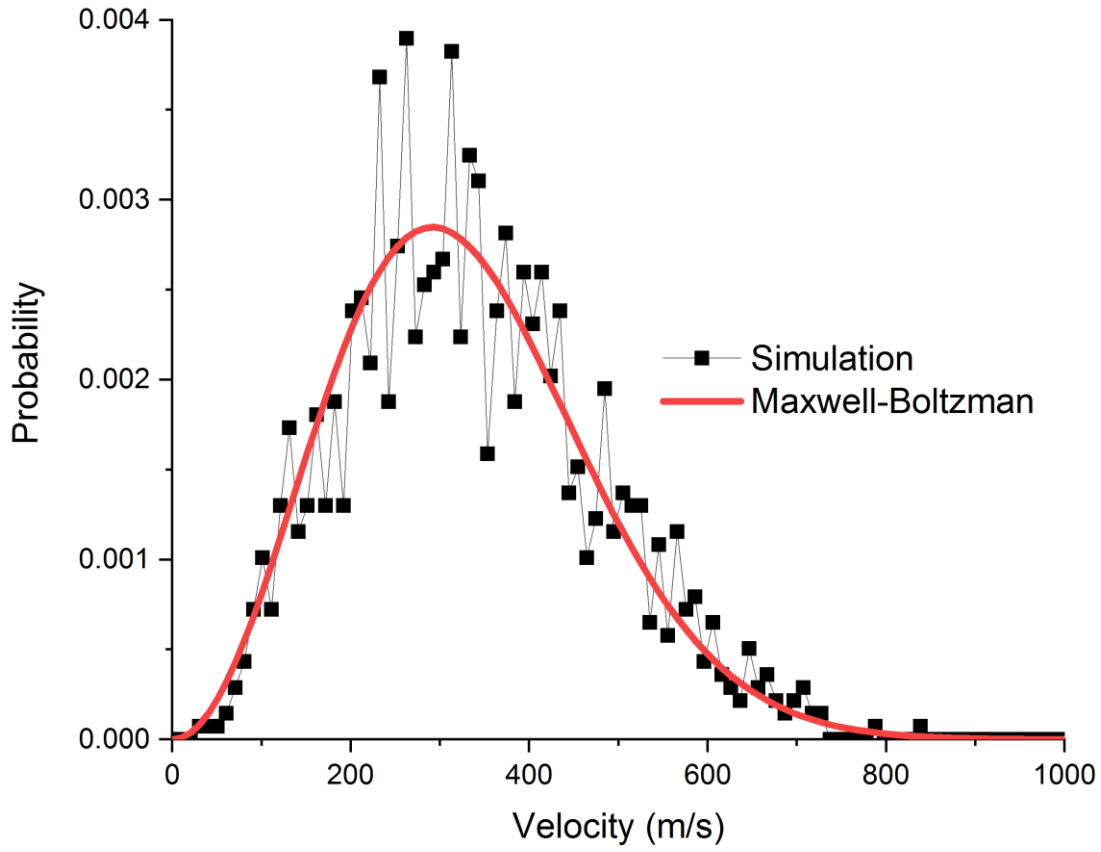


Figure 9: Velocity distribution produced using the Berendsen thermostat and the equivalent Maxwell-Boltzmann distribution

As shown in Figure 9, the simulated system does follow the trend of the Maxwell-Boltzmann but there is a large amount of stochastic noise due to the system being used having only ~1500 atoms and the statistics were only generated for a single point in time.

2.2 Lennard-Jones Potential

The Lennard-Jones potential model was chosen as a starting point for this project due to the simple nature of the model as it only uses a single equation for the potential,

$$u(r_{ij}) = 4\epsilon\left(\frac{\sigma^{12}}{r_{ij}^{12}} - \frac{\sigma^{6}}{r_{ij}^{6}}\right) \tag{10}$$

Here $u(r_{ij})$ is the potential between atoms i and j at distance r_{ij}, ε is the potential minimum, σ is the inter-particle spacing where the potential is zero and r_{ij} is the distance between atoms i and j. The force acting on atom i in direction k can therefore be represented by,

$$f_{ij,k}(r_{ij}) = \frac{-u'(r_{ij}) \times r_{ij,k}}{r_{ij}} \tag{11}$$

where $f_{ij,k}(r_{ij})$ is the k component of force acting on atom i due to atom j at distance r_{ij}, $u'(r_{ij})$ is the derivative with respect to r_{ij} of the potential at distance r_{ij} and $r_{ij,k}$ is the k component of the inter-particle spacing of atoms i and j. From Newton's Third Law,

$$f_{ji,k}(r_{ij}) = -f_{ij,k}(r_{ij}) \tag{12}$$

Although the simplicity of this model and other pair potential models make them an ideal starting point for the project, it also makes them unable to explicitly capture some more complex effects like bond bending and 3-body terms.

Despite the simple nature of this potential model, it is still computationally intensive so to reduce the computational load, a cut-off (r_{cut}) is applied to the system. Beyond this cut-off, the potential and the force are set to zero. While this reduces the number of calculations, it can still be computationally intensive as it still requires the distance between every pair of atoms to be calculated at each point in time. Section

2.4 explores how neighbour lists can be used to further cut down the computational load while section 2.6 provides more details about the cut-off, examining different methods and how they affect the dynamics of the system.

2.3 Sutton-Chen Potential

The Lennard-Jones potential was found to be unsuitable for giving an accurate description of a metallic crystal so the next potential model considered for the project was the Sutton-Chen potential (18; 19). The Sutton-Chen potential is a form of the embedded atom method which is described by the equation,

$$U = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} V_{ij}(r_{ij}) + \sum_{i=1}^{N} F(\rho_i) \tag{13}$$

Here U is the potential of the system, $V_{ij}(r_{ij})$ is a pair potential, $F(\rho_i)$ is a functional that describes the energy of embedding an atom into the local density, ρ_i which represents the density of electrons due to the surrounding atoms. It is supposed to capture the many body nature of metallic bonding. This term takes the form of

$$\rho_i = \sum_{j=1, j\neq i}^{N} \rho_{ij}(r_{ij}) \tag{14}$$

where ρ_{ij} is a pair potential. For the Sutton-Chen potential, the pair potentials and the functional take the form of,

$$V_{ij}(r_{ij}) = \epsilon\left(\frac{a}{r_{ij}}\right)^n \tag{15}$$

$$\rho_{ij}(r_{ij}) = \left(\frac{a}{r_{ij}}\right)^m \tag{16}$$

$$F(\rho_i) = -c\epsilon\sqrt{\rho_i} \tag{17}$$

where c, m and n are dimensionless constants, ε is a constant with units of energy and a is a constant with units of length. Due to the equation for the functional, the Sutton-Chen potential is also a type of Finnis-Sinclair potential which was inspired

by the tight-binding theory for the electronic structure of solids as Finnis and Sinclair used the second-moment approximation of this theory to argue that the functional of the Embedded Atom Method should take the form of a square root (71).

2.4 Neighbour Lists

To further reduce the number of calculations, a Verlet neighbour list (72) was implemented initially. To set up the Verlet neighbour list, a second, larger cut-off (r_{list}) is required. If the distance between two atoms is less than this cut-off, then the atoms are considered neighbours. The distance between every pair of atoms is checked to determine how many sets of neighbours there are, what set of neighbours was the first set involving atom i and what atom was atom i's neighbour when there was N neighbours. Then for subsequent timesteps, only the distances between neighbours is calculated. The neighbour list will be recalculated after a number of timesteps as new atoms may become neighbours while some atoms may stop being neighbours.

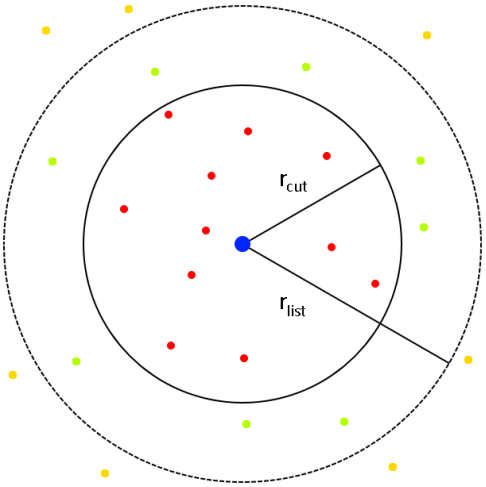


Figure 10: Representation of the Verlet neighbour list

The frequency of Verlet neighbour list updates affects how significantly the computational load is reduced but the frequency of updates may be too low for some atom pairs; the distance between a pair of atoms may be greater than the r_{list} during one update but less than r_{cut} during the next, which means that some interactions would not have been calculated when they should have. If this affects only a small number of atom pairs, it will not be worth increasing the frequency of

updates due to the increase in computational load. An alternative would be to implement an exclusion list and add the fast-moving particles to the list. The exclusion list is set up such that any atom on the exclusion list is treated as the neighbour of all other atoms. The increase in computational load caused by the exclusion list is much smaller than the increase in computational load that an increase of the frequency of updates would cause. The increase can also be offset by further decreasing the frequency of updates or by reducing the value of r_{list}.

As the size of the system increases, it may be found that Verlet neighbour list updates are too computationally intensive to be viable for large systems. If this is the case, the Verlet neighbour list and the exclusion list should be replaced by a cell index neighbour list (73; 74). In this method, the slab is divided into a number of cells where the cells' x, y and z-dimensions are larger than r_{cut}. This allows each atom to only check every atom in 27 cells (or only 14 using Newton's third law) for all potential interactions instead of checking every atom of the slab, which drastically reduces the computation time needed. The cells are mapped to an index to track which cells are neighbours. This index accounts for periodicity in all directions in its most basic form but can be altered to account for any type of periodicity. Every time the forces need to be calculated, the atoms are assigned to cells based on their position. Once assigned to a cell, the atom is added to a linked list, where the atom becomes the head atom of the cell it is assigned to and the previous head atom is stored at the current atom's element in a list array. This allows force calculations to proceed sequentially through the atoms in each cell.

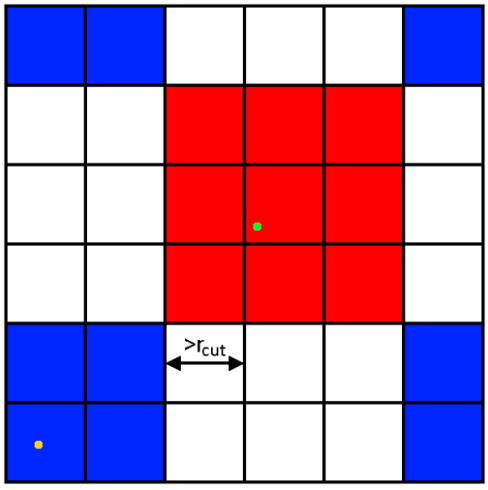


Figure 11: Representation of the cell index neighbour list

Figure 11 provides two examples of how the cell index would work with a particular atom. For the green atom, it would proceed through the linked list of the cell that contains the atom to look for interactions then it would look through the linked list of the neighbouring cells, shown in red. For the yellow atom, because it's in a corner cell, the neighbouring cells, shown in blue, loop over to the opposite side of the slab to account for the slab's periodicity.

2.5 Slab Generation

To begin with, the slab generation code was set to produce a crystal with a (100) surface on the z-axis. It did this by generating a small repeating unit and duplicating the repeating unit, with each duplicate offset by a multiple of the lattice parameter, a. The repeating unit is made up of four atoms with coordinates at (0, 0, 0), (0.5a, 0.5a, 0), (0, 0.5a, -0.5a) and (0.5a, 0, -0.5a). Figure 12 shows a diagram the original method of slab generation.

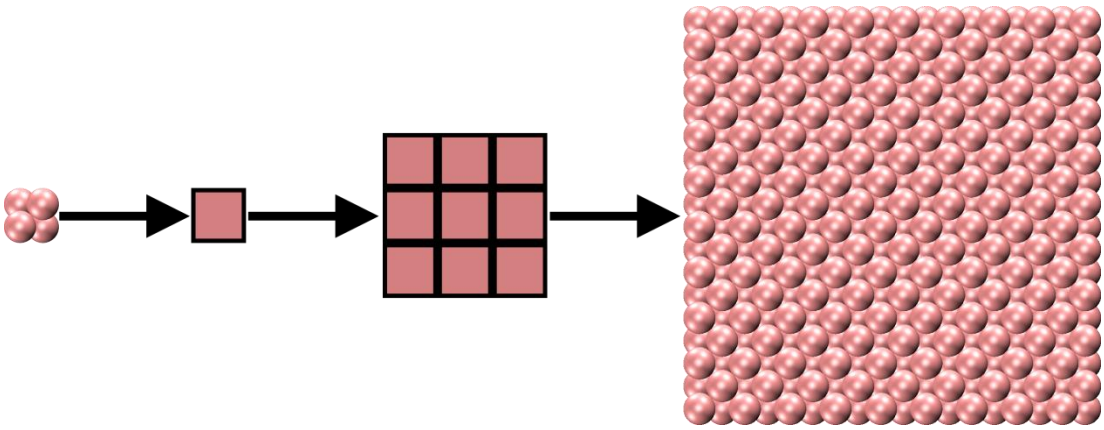


Figure 12: Schematic diagram of the originally developed method of (100) slab generation

It was later seen that the (100) surface might not be the surface that is impacted in a real scenario so it became prudent to allow the slab generation code to use other surfaces on the z-axis. At first, new surfaces were created by manipulating the original (100) repeating unit to produce a repeating unit for the new slab to be made. The original (100) repeating unit and the repeating units for a (110) surface and a (111) surface are shown in Figure 13.

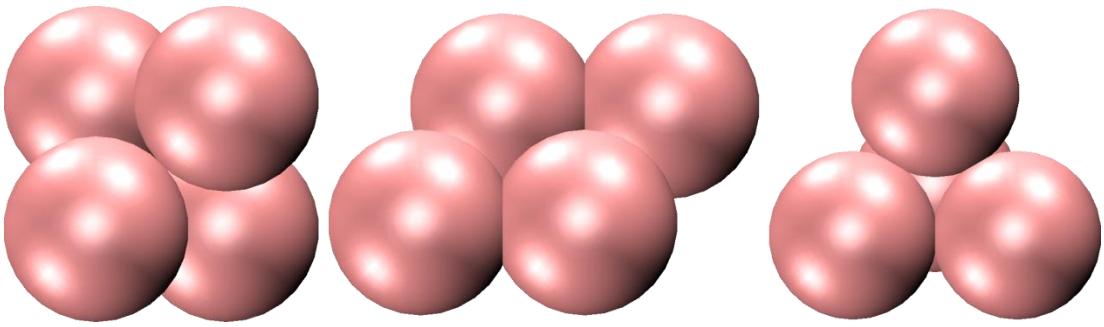


Figure 13: The repeating units used to create (100), (110) and (111) surfaces

However, this repeating unit manipulation was shown to be inadequate when trying to generate a repeating unit for the slab with a (111) surface as there were problems with tessellation. This was due to the three-layered nature of the (111) surface not being represented in the repeating unit. Furthermore, the two layers that were represented had uneven representation, causing the higher layer to be overrepresented and the lower layer to be underrepresented.

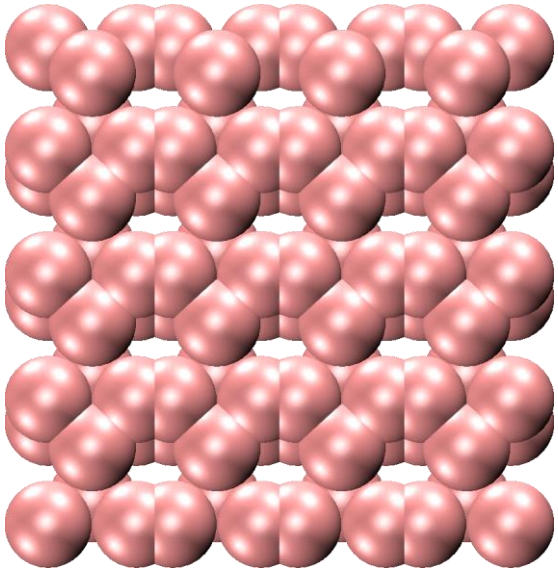


Figure 14: An incorrectly configured (111) slab with poor tessellation

To fix this, instead of manipulating the repeating unit, the slab generation code was rewritten to generate a (100) supercell (with dimensions 4a×4a×4a) from the (100) repeating unit. The supercell was reduced to only atoms with a dot product between 0 and a cut-off factor for three Miller indices. These Miller indices represent the surface shown on each axis with the Miller index for the z-axis being the surface used in the molecular dynamics simulations. Table 1 details the Miller indices used for the three surfaces that were studied in this work.

Table 1: Miller indices used for slab generation

x	y	z
(100)	(010)	(001)
$(\bar{1}10)$	$(00\bar{1})$	(110)
$(\bar{1}10)$	$(\bar{1}\bar{1}2)$	(111)

The atoms that remained are then converted into the repeating unit, again using the dot product of the atom's coordinates and the relevant miller index to obtain the equivalent coordinate for that index. The new repeating unit is then duplicated to produce the system with the desired surface. A diagram of the current process showing the generation of a system with a (111) surface is shown in Figure 15.

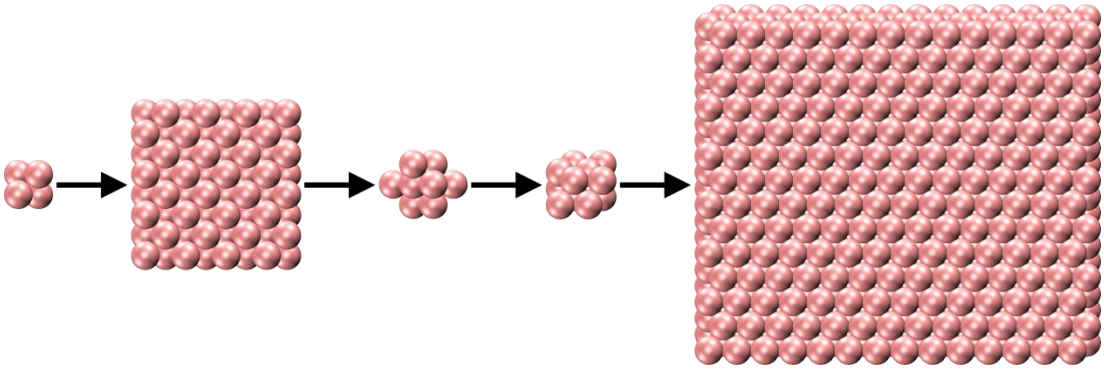


Figure 15: Schematic diagram showing the process used for slab generation. In this diagram, a system with a (111) surface is generated.

An external setup file is used to control the slab generated by the code. This file sets the lattice parameter used during generation (usually set in reduced length units, allowing the same slab to be used for multiple simulation conditions), the Miller index of the surfaces, how many layers the slab should have in each direction, the distance between these layers (normalised by a) and the maximum value of the dot product when reducing the supercell (normalised by a). An example of the external setup file can be found in Appendix E. It should be noted that the slabs generated with this code are not equilibrated to a temperature as equilibration is performed later by the Molecular Dynamics code.

2.6 Potential Cut-Off

When using a potential cut-off, the potential and force equations below the cut-off affect how accurately the system is represented. If the equations are unaltered below the cut-off, the sudden step in the potential and the force will prevent the total force on an atom and the potential energy of the system being smooth functions of time (there will be sudden steps in these equations as the distance between an atom pair crosses r_{cut}). Shifting the potential equation by the value of the potential at the cut-off ensures that the potential goes smoothly to zero, removing the sudden steps in the potential energy of the system as a function of time. However, the total force on an atom will still have sudden steps as the force equation remains unaltered. If a smoothing function is applied on the potential function between $r_{cut} - \delta$ and r_{cut}, the potential and the forces will both go to zero at r_{cut} but while the potential goes smoothly to zero, the equation for the forces becomes a polynomial with a minimum at $r_{cut} – 0.5\delta$. This means that at $r_{cut} - \delta$, the forces will suddenly decrease until reaching the minimum, going below zero and becoming repulsive before returning to zero. This causes the forces in this region to be unrealistic.

Using a shifted force cut-off (75; 76), where the force equation is shifted by the force at the cut-off, ensures both the potential and the force go smoothly to zero, meaning that the total force on an atom is a smooth function of time. This significantly improves the energy conservation in a system and the accuracy of the dynamics of the system.

Figures 16 and 17 highlight the differences of the different cut-offs for both the potential and force, respectively. In Figures 16 to 23 the distance was normalised by a (the lattice parameter). This is the reduced length units usually used for the Sutton-Chen potential while the Lennard-Jones potential typically normalises distances by σ. It can be seen in Figure 16 that the potentials for the smoothing function, the simple cut-off and true equation are exactly the same until a value of 2.64 reduced length units. At that value, the smoothing function begins to increase the potential smoothly to zero. At 2.84, the simple cut-off jumps to zero. The potentials when using the shifted-potential and shifted-force cut-offs are always slightly different from the potentials calculated by the true equation.

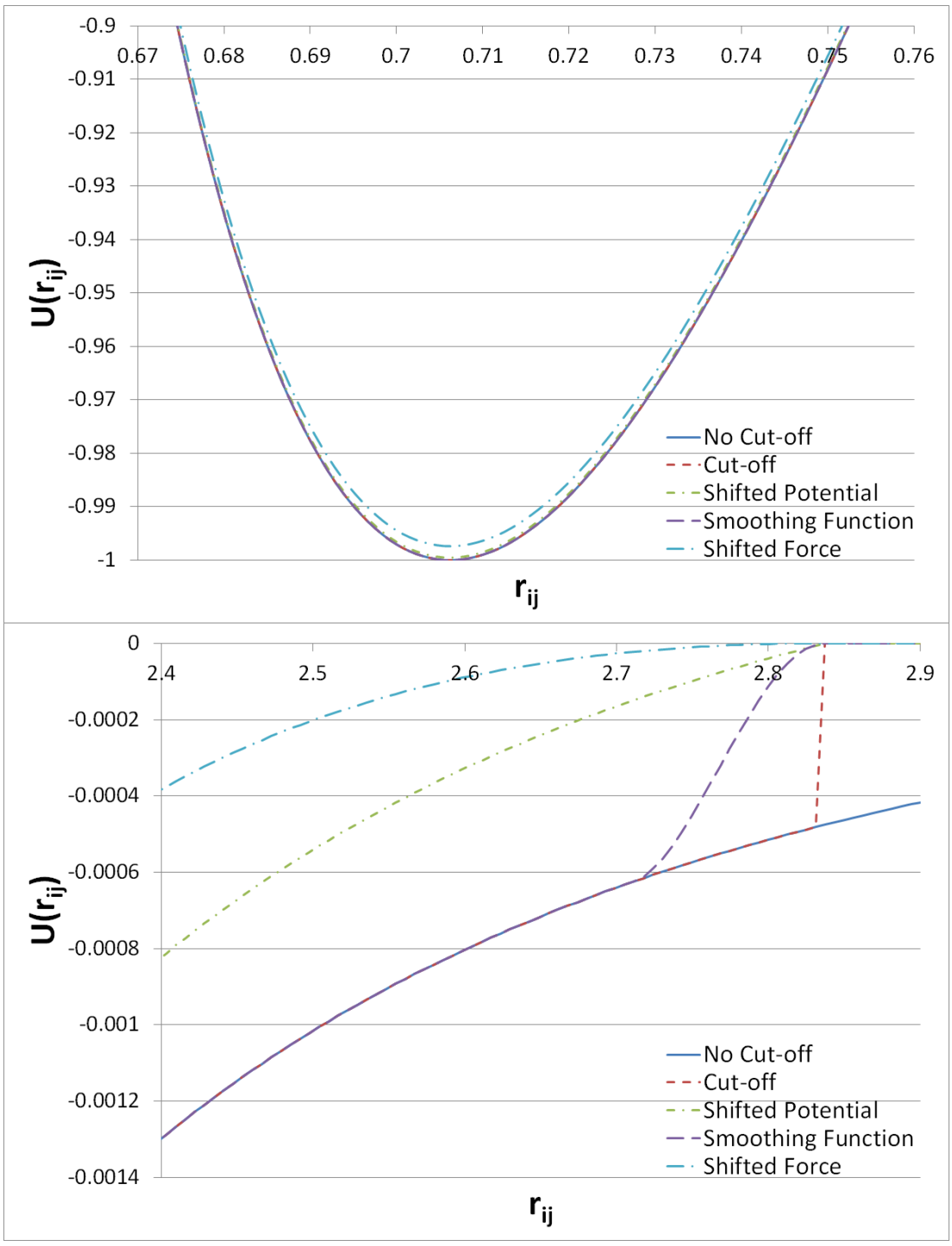


Figure 16: Potential curves using the Lennard-Jones potential for the different cut-offs at the minimum potential and around the cut-off

The same features are mostly present in Figure 17. One distinct change is that between 2.64 and 2.84, the forces for the smoothing function spikes downward, creating a repulsive force instead of going smoothly to zero. The Shifted Forces cut-

off was used for simulations using the Lennard-Jones potential as it produced the smoothest transition to zero at the cut-off.

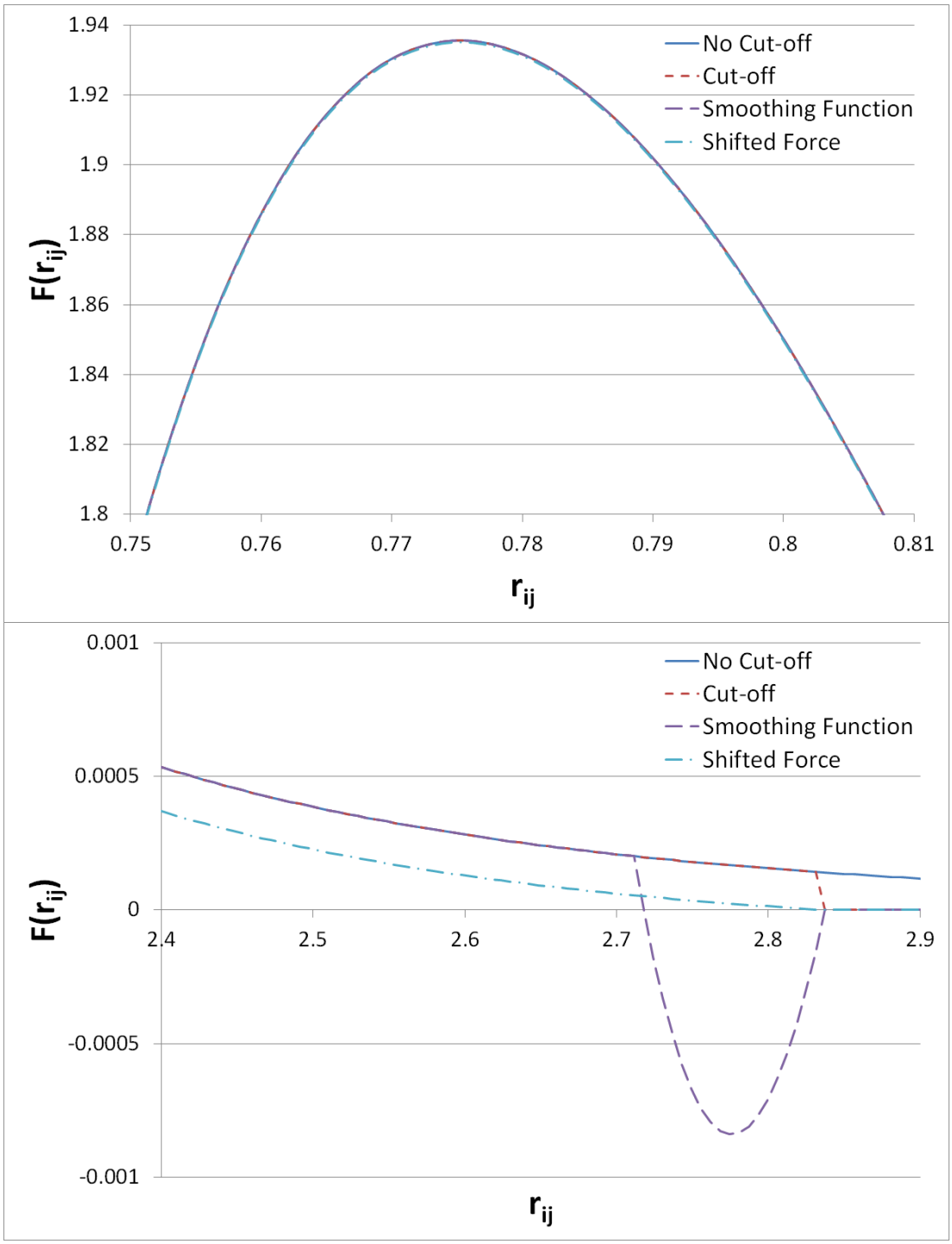


Figure 17: Force curves using the Lennard-Jones potential for the different cut-offs at the maximum force and around the cut-off

However, when the code was adapted to use the Sutton-Chen potential, the values around the cut-off are significantly different due to the many-body nature of the potential. Due to this, shifting the second derivative of the potential was tried as well as the forces and the potential. The cut-off with a smoothing function was not used for the Sutton-Chen as it was difficult to adapt it to the potential and it produced highly unrealistic forces for the Lennard-Jones potential.

Figures 18 and 19 are analogous plots to Figures 16 and 17 showing the potential and force curves for a pairwise interaction using the Sutton-Chen potential. Looking at the potentials in Figure 18, all three cut-offs involving a shift produce weaker potentials with the shifted second derivative produces the weakest potential. Close to the cut-off, it can be seen that the shifted potential accelerates towards zero instead of slowly declining towards zero.

The pairwise force curves for the Sutton-Chen potential are shown in Figure 19. It should be noted that while it has no effect on the forces in the Lennard-Jones potential, the shifted potential cut-off has an effect on forces in the Sutton-Chen potential. This is because the derivative of the potential doesn't remove the constant shifting the potential within the density functional. Due to this, the density functional tends to 0 at the cut-off causing forces to spike close to cut-off. It can also be seen that for the shifted force cut-off, the forces have a step-change at the cut-off instead going smoothly to zero. The shifted second derivative cut-off does go to zero but it appears to accelerate as it gets closer to the cut-off.

As mentioned earlier, Figures 18 and 19 show the potential and forces based on a pair-wise interaction but as the Sutton-Chen potential is a many-body potential, these figures don't show fully representative potential and forces.

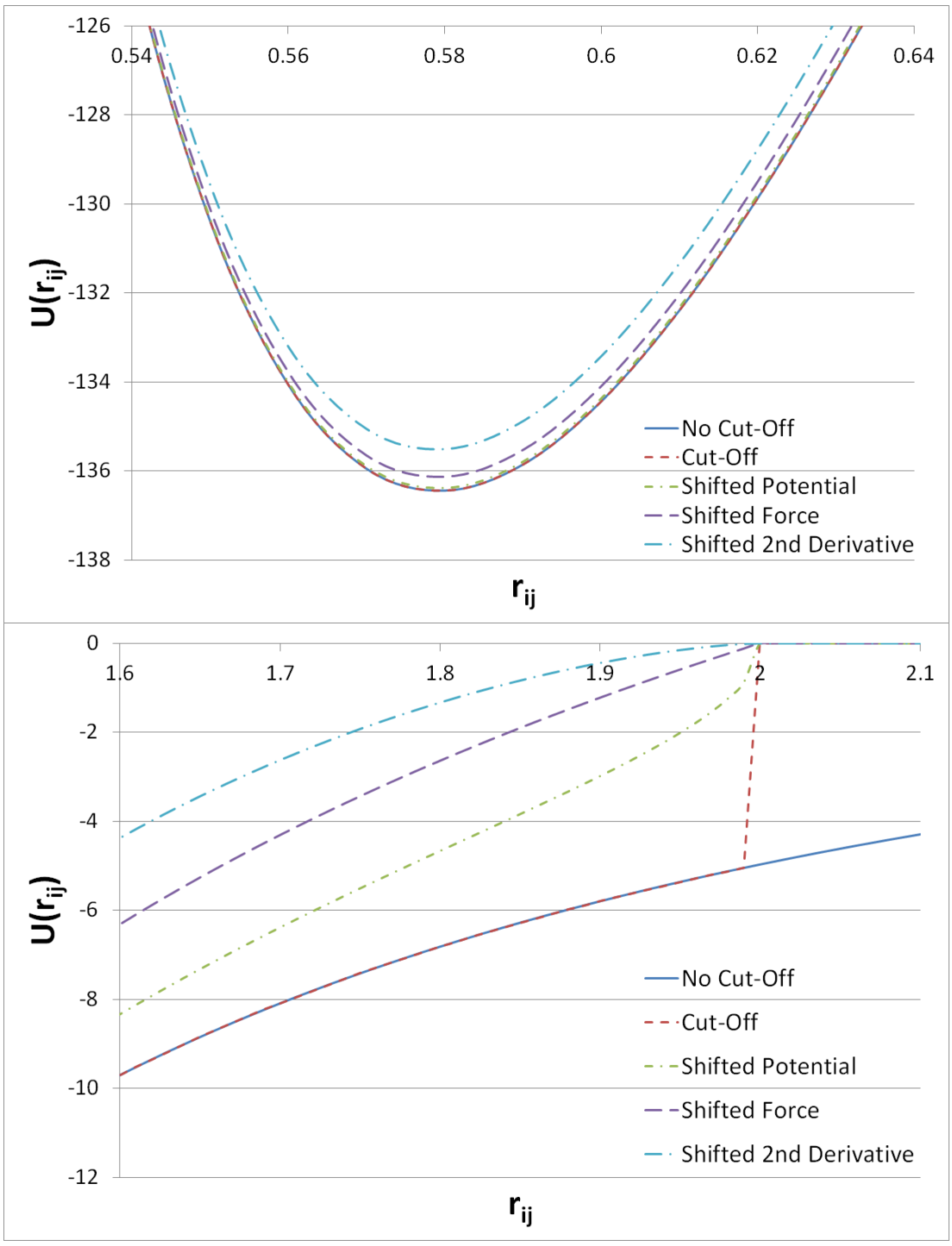


Figure 18: Pairwise potential curves using the Sutton-Chen potential for the different cut-offs at the minimum potential and around the cut-off

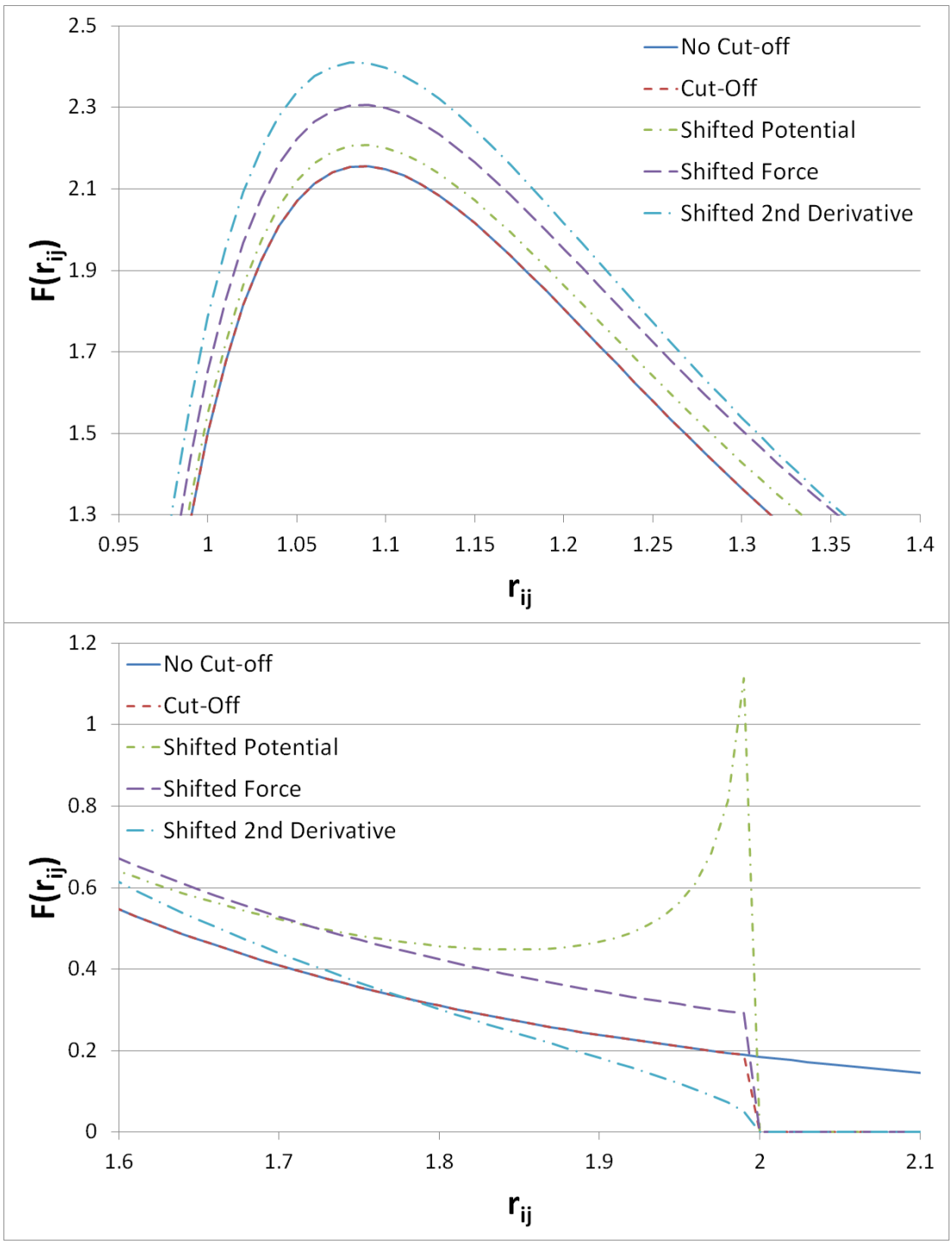


Figure 19: Pair-wise force curves using the Sutton-Chen potential for the different cut-offs at the maximum force and around the cut-off

The potential caused by an atom approaching the surface and the force on that atom in the z direction were plotted against its distance from the average surface height.

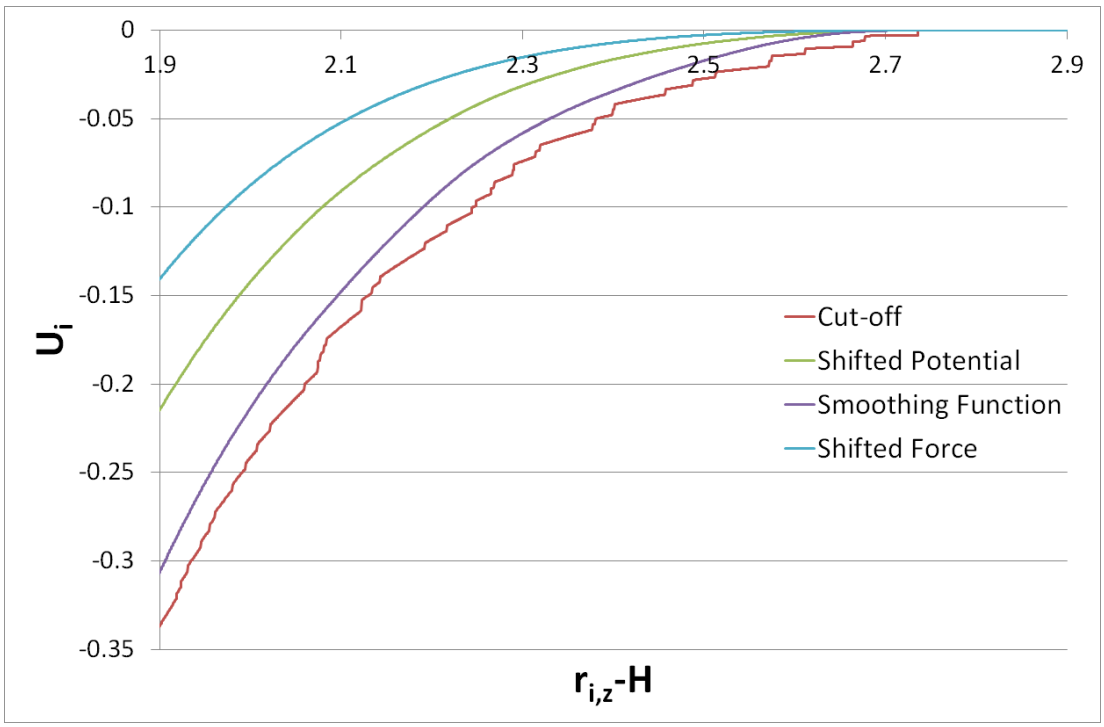


Figure 20: Potential of an atom using the different cut-offs for a system using the Lennard-Jones potential as the atom moves away from a surface and approaches the cut-off

Figure 20 shows the potential energy caused by an atom, i, as it moves away from a surface and approaches the cut-off for the different cut-offs used in a system using the Lennard-Jones potential. These were generated by positioning atom i above the surface and moving the atom in small steps, recording the potential and the forces at each step. The distance is plotted as the difference between the z coordinate of atom i, $r_{i,z}$, and the average surface height, H. The potential energy caused by atom i, U_i, is calculated by subtracting the total potential energy of the system without atom i from the total potential energy of the system with atom i. It should be noted that the potential energy becomes zero before the cut-off of 2.84a because atom i was not directly above an atom on the surface. The simple cut-off produces several step changes and the shifted potential and shifted force cut-off fall smoothly to zero as expected. The cut-off with a smoothing function appears to decline more naturally to zero than it did in the pairwise interaction.

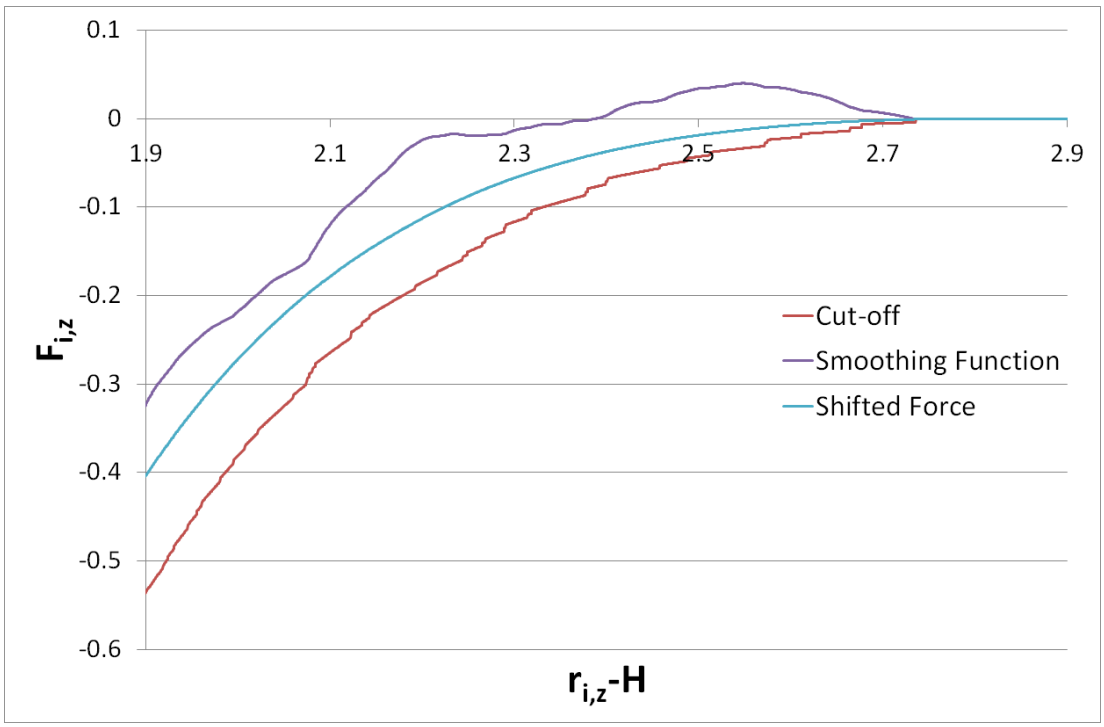


Figure 21: Force acting on an atom using the different cut-offs for a system using the Lennard-Jones potential as the atom moves away from a surface and approaches the cut-off

Figure 21 shows the total force acting on atom i in the z direction, $F_{i,z}$, as it moves away from a surface. The simple cut-off and the shifted potential cut-off (not plotted as it is identical to the simple cut-off) both saw several step-changes as more neighbouring atoms crossed the cut-off and the shifted force cut-off declined smoothly to zero. The cut-off with a smoothing potential had numerous unnatural fluctuations in the forces, producing a repulsive force close to the cut-off.

Figure 22 is the analogous plot to Figure 20 for the Sutton-Chen potential. Using a simple cut-off, it can be seen that there are numerous step changes to the potential as neighbouring atoms cross the cut-off, changing the value of the density functional. The shifted potential cut-off seems to fall smoothly until neighbouring atoms begin crossing the cut-off, at which point the potential energy begins decelerating and then accelerating as the next neighbouring atom approaches the cut-off. The shifted forces and shifted second derivative appear to fall smoothly to zero.

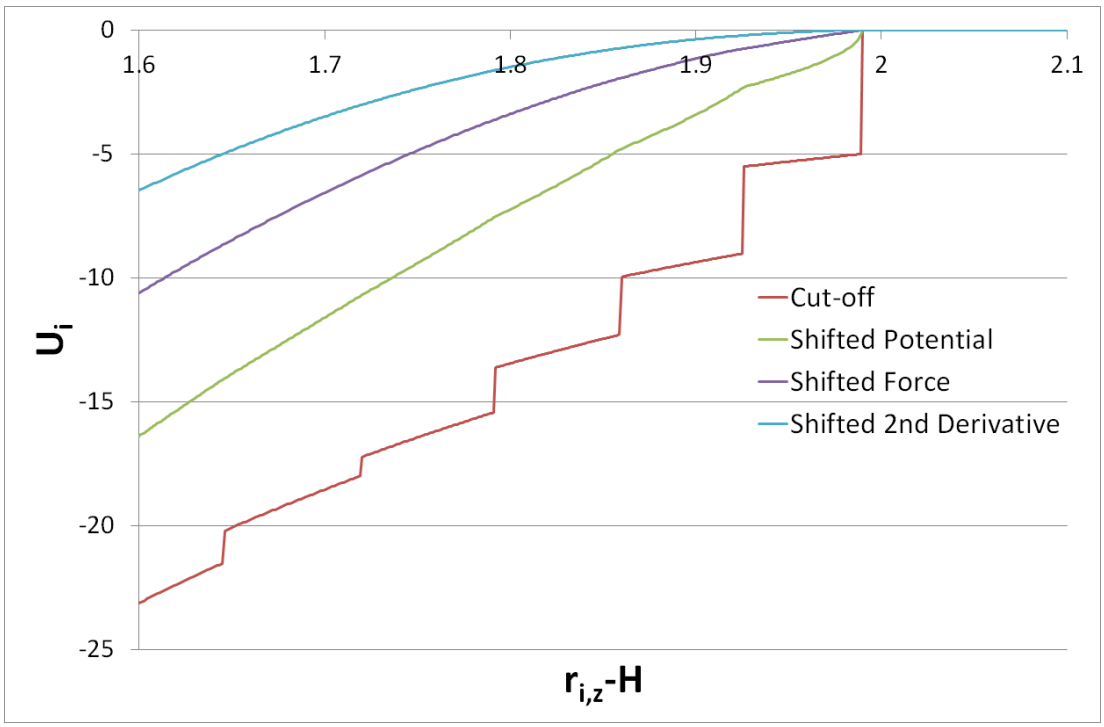


Figure 22: Potential of an atom using the different cut-offs for a system using the Sutton-Chen potential as the atom moves away from a surface and approaches the cut-off

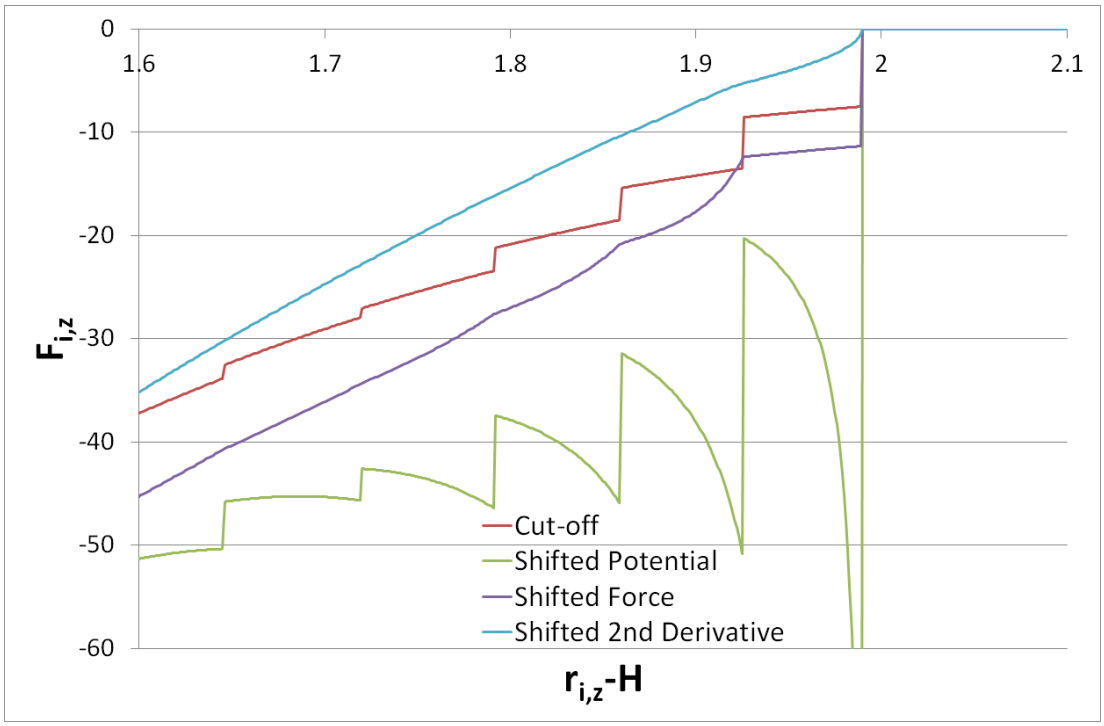


Figure 23: Force acting on an atom using the different cut-offs for a system using the Sutton-Chen potential as the atom moves away from a surface and approaches the cut-off

Figure 23 shows the total force acting on atom i in the z direction, $F_{i,z}$, as it moves away from a surface. Like the potential energy, the simple cut-off has numerous step-changes. The shifted potential cut-off creates a highly unrealistic force curve where the forces begin to increase exponentially as a neighbouring atom approaches the cut-off. This creates a peak of -182 in reduced force units, which is the force normalised by ε÷a for the Sutton-Chen potential, at the cut-off (not shown to keep other force curves distinguishable). The shifted forces cut-off, unlike for the Lennard-Jones potential, does not fall smoothly to zero as the atom approaches the cut-off and instead has an unwanted step-change. Meanwhile the shifted second derivative cut-off falls slowly and then more sharply towards zero at the cut-off.

Since the shifted second derivative cut-off was the only cut-off without a step change in the forces at the cut-off, that cut-off was used for the Sutton-Chen potential.

2.7 Lennard-Jones Code Development

2.7.1 Program Creation

Starting with the MD code from the author's 5th year project, all subroutines and code related to the potential model used in that project were removed except for the subroutine used to generate images. A new subroutine was made to calculate the potential and the forces acting on the atoms using two loops to calculate the distance between every atom. As the potential between two atoms only required one calculation and the forces on atom j due to atom i was the negative of the forces on atom i due to atom j, the code only calculated the distances when j was greater than i. After calculating distance, it checked if the atoms were neighbours and if they interacted before using the Lennard-Jones equation and its derivative to calculate the potential and the forces.

After some test simulations, it was noted that the system was exhibiting peculiar behaviour. Closer inspection of the code revealed that forces between the slab and the slab images weren't being calculated. To rectify this, another loop was created to replicate the position of an atoms image and calculate the distance between the image of an atom and an atom in the main slab. However, due to the way the

neighbour list added sets of neighbours, the slab only interacted during neighbour list updates. To correct this, the arrays for the neighbour list were expanded to store image-specific data. This required the distance to be calculated for every value of i and j as the image of atom j may react with atom i but the image of atom i won't interact with atom j. Due to this, all forces were calculated for every atom instead of using Newton's third law to calculate half of all forces for every atoms. The potentials were still only calculated when i was greater than j since the potential energy would report a value double the size of the actual value if it was calculated for every atom.

As explained in section 2.4, the next step is to make an exclusion list to eliminate the need to update the neighbour list more frequently to maintain accurate forces for a small number of fast moving atoms. First, arrays were made to store the max velocities of all atoms, then a conditional statement was added to the force subroutine so that any atoms that had a maximum velocity big enough to travel the difference of r_{cut} and r_{list} in 2 updates or less would be put into the exclusion list. Any atom on the exclusion list would then be added to the neighbour list of every atom.

Preliminary simulations revealed some problems with the current code, primarily that the simulation broke the first law of thermodynamics (the law of conservation of energy). It was unclear what was causing this so to rule out erroneous units, the system was changed to use reduced units, which are dimensionless units based on system parameters like σ and ε. To further simplify the code, some of these parameters were set to 1. However, despite the reduced units, the simulations still broke the first law.

Due to how the energy was fluctuating, it was suggested that the system was not at its minimum potential energy. To find the slab with the interparticle separation where the energy was at the minimum, a separate version of the code was made with a scaling factor that scaled all interparticle separations. This code was set to only calculate the initial potential energy at numerous scaling factors assuming the system was periodic in 3 dimensions. When the minimum potential energy was found, a simulation was run using the slab with the interparticle separation where the minimum was found. This conserved energy better than the preliminary simulations but the conservation was still deemed unacceptable.

It was then realized that the lack of energy conservation could be due to the sudden step at the cut-off in both the potential and force equations. To deal with this, alternative implementations of the Lennard-Jones equation were sought after from literature and examined. As noted in section 2.6, the shifted-force equation ensured the force and potential went smoothly to zero so this equation was implemented into the code. After fixing some errors with the implementation, the minimum potential energy of the slab had to be found again as the interparticle separation that produced the minimum was sensitive to any change in the potential calculation. Once the interparticle separation that produced the new minimum was used, acceptable energy conservation was achieved and it was found that the conservation of energy was approximately proportional to the inverse of the timestep squared (e.g. decreasing the timestep by a factor of 10 increased the conservation of energy by a factor of 100).

After this, numerous simulations were run to look at how slabs reacted to various impacts. Eventually, larger slabs were used and the ability to impact the slab with multiple atoms was added. However, when using even larger slabs, the time taken to run part of the simulation suggested that the complete simulation would take many days and that the simulation time had increased by a factor greater than the factor that the number of atoms had increased by. The reason for this increase was that while the computation time of the forces was proportional to the number of atoms, the computational time taken to update the neighbour list was proportional to the number of atoms squared. To counteract this, a cell index using linked-lists was implemented to replace the neighbour list. At first, the results produced were inconsistent with the results using the neighbour list but this was due to only half the forces being calculated and the interactions with slab images being ignored. Newton's third law was used to calculate the other half of the forces and a function was used to make the distance calculated be the smaller value of either the separation of the atoms if both were in the same slab or the separation of the atoms if one atom was in the slab and the other was in an image so that the interactions with slab images were calculated.

2.7.2 Scattering Simulations

With reasonable simulations of single atom impacts achieved, work began on multi-atom impacts. For angled impacts, the code used ratios of x and y velocity to z velocity to determine the incident angle. This was changed to use a polar angle as the incident angle and an azimuthal angle to determine the direction the atom was travelling as some simulations required a fixed incident angle but a random direction of travel for each of the atoms impacting the slab. Using the two angles made this much simpler to achieve and removed the potential for human error when calculating the velocity ratios. After code was implemented for random azimuthal angles, the code was also applied to the positions of atoms in the x and y directions. This allowed atoms to randomly impact anywhere on the slab in any direction.

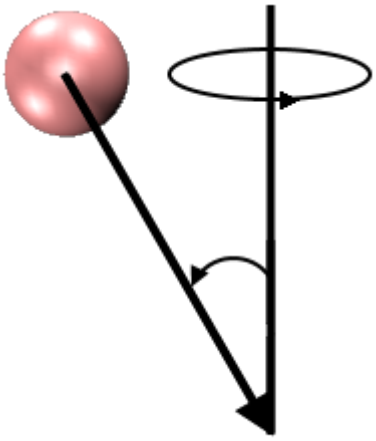


Figure 24: Polar angle of incidence and azimuthal angle

An example of the polar and azimuthal angles is shown in Figure 24. The thick vertical line represents the normal to the surface and the thick line with an arrow represents the atom's trajectory. The polar angle is the angle between the normal and the trajectory and the azimuthal angle is the angle between the atom's position, the normal and the x-axis.

Figure 25 shows an example sputtering simulation. The think lines with arrows represent the trajectory of the atoms. In this example, the bottom two layers, in blue, are fixed and the six above them, in green, are thermostated to 300K while the one layer above those, in pink, is free-moving after initially being equilibrated to 300K. The other pink atoms, either surface atoms that have been displaced or atoms that have impacted the surface, are also free-moving. An impacting atom is rapidly

approaching the surface. Some of the atoms that impacted previously have stuck to the surface and there are a few ejected atoms, either sputtered from the surface by an impact or an impacting atom that failed to stick.

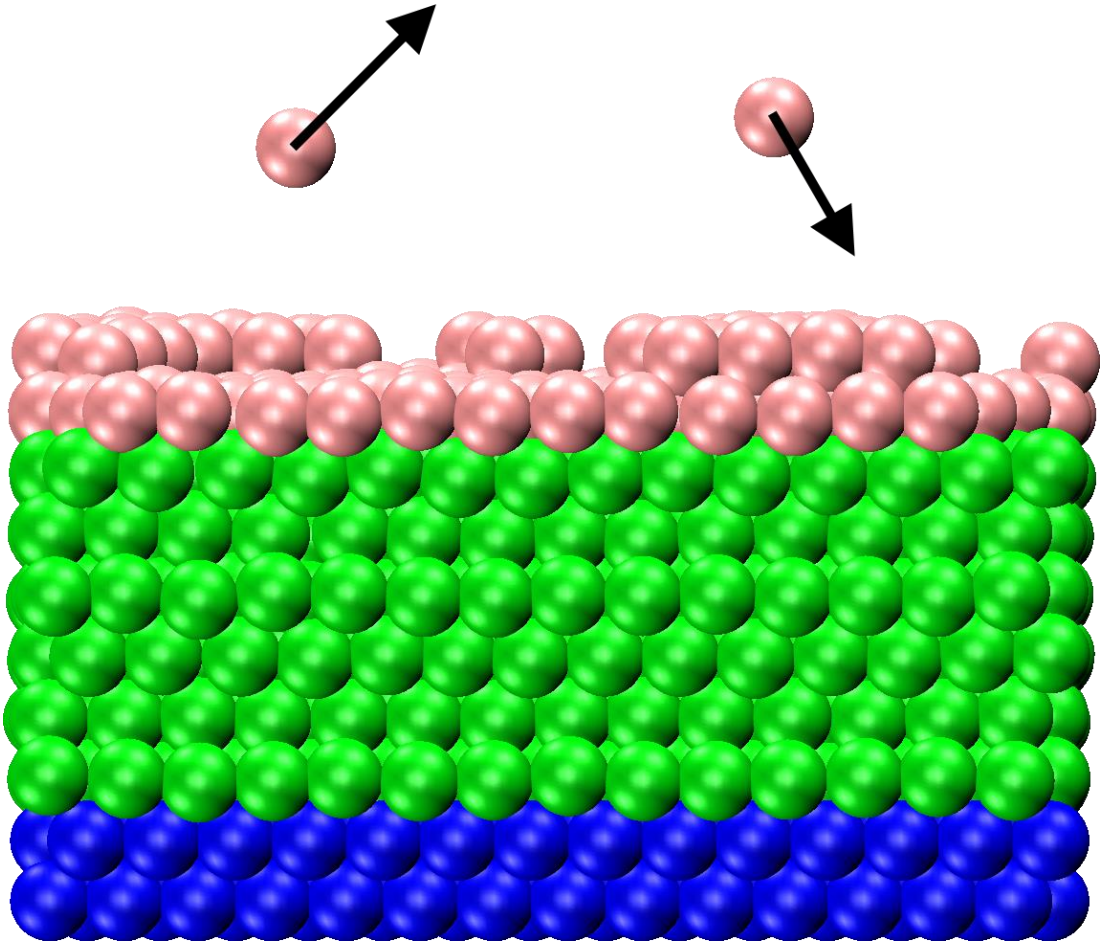


Figure 25: Diagram of an example sputtering simulation. Blue atoms are atoms in the non-moving fixed layers, green atoms are in the thermostated layers and pink atoms are free-moving. The arrows show the trajectory of the atoms above the surface, one of which is an impacting atom and the other is an atom that was either sputtered by an impacting atom or an impacting atom that failed to stick.

To minimize the need of large trajectory files, code was created to generate a small text file that was used to monitor when atoms were ejected from the surface during a simulation. The file provides the time the atom was deemed to be ejected and the position, velocity and acceleration of the atom at that time. The file was later updated to include the point in time when an atom approached the surface and when an ejected atom returned to the surface, which can potentially occur after a collision above the surface.

Early multi-atom impact simulations revealed that the compiler used for the code used the same set of random numbers for every simulation. It was determined this was due to the compiler using the same starting seed for the random number generator in every simulation. To prevent this, code was added to determine the starting seed using the wall-clock time at the start of the simulation as the basis.

The validation simulations for multi-atom impacts required the slab to be preheated to 300K so the Berendsen thermostat already in the code was modified to be a subroutine so that it only worked on thermalized layers when the simulation was running and worked on all non-fixed layers during preheating. After a test simulation, the slab was exhibiting strange behaviour. A wave was travelling up and down the slab, causing some atoms to be ejected from the surface well after impact. At first, this was assumed to be due to the slab no longer being minimized so the slab minimization code, which was incredibly out-dated, was updated and merged with the simulation code. The merging of the codes meant that further updates to the simulation code would also be applied to the slab minimizer at the same time. However, the slab minimizer did not fix the wave travelling through the slab. It was later realized that the wave was an artefact of the forces present during slab relaxation/minimization being amplified by the thermostat, preventing the dissipation of these forces. To remove this artefact, code was created to randomly distribute velocities to the non-fixed atoms before pre-heating.

2.8 Sutton-Chen Code Development

2.8.1 Surface Algorithm

After the Sutton-Chen model was implemented, it was decided that the properties of the surface should be monitored to determine how it was affected by impacts. The properties that were determined to be the most useful were the average surface height and the surface roughness. To obtain these statistics, the code had to accurately capture the surface. At first, the code was altered to look at all atoms and count the number of obscuring atoms. An obscuring atom was defined as an atom within 6.336Å of the observed atom with an angle greater than 30° between the potential obscuring atom, the observed atom and the normal to the z axis. The distance used was chosen to avoid impacting or sputtered atoms from “obscuring” a surface atom. After looking at all atoms, a surface atom was defined as having fewer

than three obscuring atoms. If an atom met this condition, its current height was added to the surface height and a counter was incremented by 1. Once all atoms had been checked, the surface height was divided by the counter to obtain the average surface height, which was then used to find the standard deviation in the height. This standard deviation is treated as the roughness of the surface. It was found that while it reasonably captured the surface for an amorphous slab, it captured an extra layer of the crystalline slab.

Next, the code was modified to count the number of neighbour atoms, which was defined as an atom within 3.168Å of the atom being looked at. After looking at all atoms, a surface atom was defined as an atom with between 4 and 11 neighbour atoms. The lower limit was to avoid classifying sputtered or impacting atoms as surface atoms and the upper limit was intended to stop atoms beneath the surface but missing a few neighbours from being identified as surface atoms. However, it was found that while it perfectly captured the surface of the crystalline slab, it captured too many atoms after a number of impacts as the slab started to become more amorphous.

Since both seemed to capture the surface reasonably for different structures but were too lenient with the other structure, the code was adapted to look at both the number of neighbour atoms and obscuring atoms. However, this was still too lenient in places but too strict in others so the numbers used for the definitions of neighbour, obscuring and surface atoms were revised until it was found that all surfaces were reasonably captured when an obscuring atom had an angle greater than 60° normal to the z axis and a surface atom had no obscuring atoms.

2.8.2 Lattice Parameter Optimization

While analysing the effects different impact parameters had on the surface, an odd behaviour was seen in a 900K impact simulation. It was noted that the surface height of the system grew rapidly at an unrealistic rate near the beginning of the simulation. After analysing the trajectory file of the system, it was seen that around this time, some surface atoms rapidly moved to a higher layer and other atoms near these higher layer islands were elevated from the original surface position. Due to the system constraints, it was theorized that this was caused by the lattice parameter being too small for that system temperature, causing the system to try to

expand. As the system was periodic in the X and Y directions and had a fixed lower layer, this expansion could only occur at the surface.

To rectify this, it was necessary to optimize the lattice parameter being used at a given temperature. To optimize the lattice parameter, it was decided that the pressure of the system should be close to atmospheric pressure. To calculate the pressure of the system, the system was simulated in fully periodic boundary conditions and the following equation was used

$$\mathcal{P} = \rho k_B \tau + \mathcal{W}/V \tag{18}$$

where $\mathcal{P}$ is the instantaneous pressure, ρ is the density, k_B is the Boltzmann constant, τ is the instantaneous temperature, V is the volume and $\mathcal{W}$ is the instantaneous Virial coefficient. The Virial coefficient can be calculated using

$$\mathcal{W} = \frac{1}{3}\sum_i \sum_{j>i} \boldsymbol{r}_{ij} \cdot \boldsymbol{f}_{ij} = -\frac{1}{3}\sum_i \sum_{j>i} w(r_{ij}) \tag{19}$$

where $\mathbf{r}_{ij}$ is the position of an atom i relative to atom j, $\mathbf{f}_{ij}$ is the force on atom i due to atom j and $w(r_{ij})$ is the pairwise virial function at a molecular separation of r_{ij}, which is given as

$$w(r_{ij}) = r_{ij}\frac{du(r_{ij})}{dr_{ij}} \tag{20}$$

where $u(r_{ij})$ is the pairwise potential function at a molecular separation of r_{ij}.

Each lattice parameter was simulated for 1000 timesteps at 100K intervals of temperature up to 900K. When a large enough variety of lattice parameters had been simulated, polynomial curves were fit to the pressure against temperature to obtain the temperature at which each lattice parameter crossed 0 GPa. This was due to the size of the simulation, as the fluctuations in pressure were too large to accurately determine the point at which they were at atmospheric pressure. Once the temperature at which the pressure was 0 GPa was obtained for all lattice parameters, a polynomial was then fit to the lattice parameter against these temperatures. The optimal lattice parameters for given temperatures were then simulated under these conditions to verify that they produced 0 GPa at that temperature. Results from the verification simulations were also used to further refine the fit.

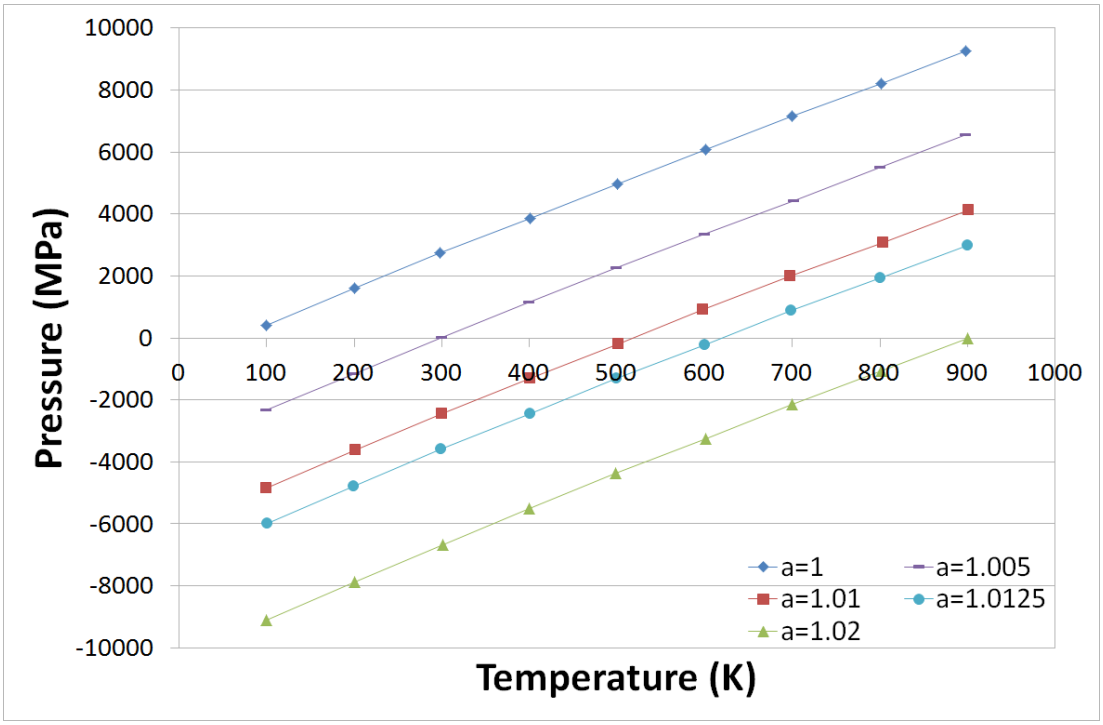


Figure 26: Pressure against Temperature for various lattice parameters, normalised by the initial lattice parameter

Figure 26 shows the pressure experienced by a fully periodic system at various temperatures and various lattice parameters, normalised by the initial lattice parameter. It can be seen that for the initial lattice parameter, the system experiences a large amount of pressure at 900K but also experiences some pressure at 300K. A system with a lattice parameter 0.5% larger was able to obtain a pressure of ~0MPa at 300K while a system with a lattice parameter 2% larger is needed to obtain a pressure of ~0MPa at 900K. However, the lattice parameter that produced the optimal pressure in a fully periodic system was only used for 900K simulations. For 300K simulations, the initial lattice parameter was still used as this is the value that was fit in the literature by Sutton and Chen (18) and was still used for the truncated potential by Rafii-Tabar and Sutton (19).

2.9 Final MD Code Overview

Initially in the MD code, a seed used for random number generation is generated at run-time. The code then reads in parameters from an input file called “conditions.dat” and reads arguments supplied at run-time, two of which specify the

files containing the data for the surface being impacted and the data for the atoms impacting the surface. A third argument supplied at run-time specifies how output files should be named. The typical values of the parameters in the input file can be seen in Table 2.

While n_{atom} is set to the atomic number of titanium, 22, instead of the atomic number of nickel, 28, this has no effect on the simulation and was used purely for aesthetic purposes in visualisation software.

Numerous variables are currently expressed in a non-conventional format. The energy variables $e_{k,in}$ and ε have been expressed in terms of their equivalent temperature. The two variables for times, δt and t_T, have been expressed in reduced time units. Time can be reduced using the equation described by Allen and Tildesly,

$$\dot{t} = t \times \frac{\sqrt{\frac{m_{Atom}}{\varepsilon \times e_{unit}}}}{a \times r_{unit}} \tag{21}$$

where ṫ is reduced time, t is time, ε is the energy parameter in energy units and the rest of the symbols are as seen in Table 2. The cut-off, r_{cut}, is also expressed in dimensionless units, which is generated for lengths by dividing the length by $a \times r_{unit}$, effectively expressing the cut-off in terms of the lattice parameter, a.

From the parameters supplied in the input file and the arguments, the program generates all of the impacting atoms, reading the file detailing the impacting atom and then duplicating it until the number of impacting atoms matches the value of n_{impact}, which was 1000 in most simulations. These impacting atoms are moved to an area where interactions are ignored and they are each given kinetic energy that mimics the energy that they will impact the surface with.

If the slab has not been equilibrated to the desired temperature, all atoms in the slab, except atoms in the fixed layers, are given random initial velocities. The free-moving layers are then temporarily thermostated and the slab is simulated for 150 picoseconds to bring the slab to the desired temperature.

Table 2: List of parameters and their typical value in input file for MD simulation code

Parameter	Value	Explanation
δt	4.56499×10^{-4}	timestep (in dimensionless units)
i_{loop}	1	first loop size counter
j_{loop}	5	second loop size counter
k_{loop}	100	third loop size counter
N_{Thread}	Depends on CPU	Number of threads for parallelisation
r_{cut}	2	Interaction cut-off (in dimensionless units)
r_{unit}	1×10^{-10} m	Length units
a	3.52	Lattice parameter
ε	182.3418	Energy parameter (as temperature equivalent)
c	39.755	constant
q	6	exponent for cohesive N-body term
u	9	exponent for repulsive term
n_{atom}	22	Atomic number (for visualisation purposes)
m_{Atom}	9.7463×10^{-26} kg	Atomic mass
t_T	4.56499×10^{-2}	relaxation time for Berendsen (in dimensionless units)
T	300 K	Target Temperature
n_{impact}	1000	number of impacting atoms
id	8	number of loops between impacts
$e_{k,in}$	6366.588	initial kinetic energy (as temperature equivalent)
e_{unit}	1.6022×10^{-19} J	Energy units
$r_{x,in}$	-200	Initial position, x axis
$r_{y,in}$	-200	Initial position, y axis
$r_{z,in}$	9	Initial position, z axis
θ_x	0	rotation of impacting atom around x axis
θ_y	0	rotation of impacting atom around y axis
θ_z	0	rotation of impacting atom around z axis
θ	10°-80°	Polar angle
φ	-180	Azimuthal angle

2.9.1 Impact

At the beginning of each impact event, the initial position and the angles of the impacting atom are determined by $r_{x,in}$, $r_{y,in}$ and $r_{z,in}$ in the input file and the angles of the impacting atom are determined by θ_x, θ_y, θ_z, θ and φ in the input file. The typical values shown in Table 2 for these mean that the initial position is randomly generated in x and y while always being above the surface. As well as allowing random positions in x and y, the parameters allow for the impacting atom to be centered above the system. The first three angle parameters are not applicable to single atoms so they are left at 0 but θ and φ, the polar and azimuthal angles respectively control the velocity vector of the impacting atom. The typical value for φ causes the azimuthal angle to be randomly generated for each atom but other values allow the azimuthal angle to be fixed to the same angle for all impacting atoms. Polar angles were always fixed for every impact.

Once positioned, the atom will begin approaching the surface. When the atom is within a threshold of $1.5 \times (\bar{h} + r_{cut})$, where $\bar{h}$ is the average surface height and r_{cut} is the cut-off, and its acceleration in the z direction has become positive, the atom is considered to have impacted the surface. This point was chosen as the acceleration in the z direction becoming positive means that the atom is being repulsed, suggesting it is very close to another atom and the threshold was chosen to limit impacts being counted after collisions above the surface. The threshold is also allowed to move as the surface grows to prevent impacts no longer being counted once the surface grows too much.

After the atom impacts the surface, the system continues to advance in time for a duration chosen in the input file. During this time, the system checks for any atoms that cross the threshold. If an atom leaves the threshold, it is checked if it was the last impacting atom. If it was, the impacting atom is considered to have not stuck and the sticking probability is suitably adjusted. If it was not, then the atom crossing the threshold is considered to have been a surface atom that was sputtered and the sputter yield was adjusted. Once the simulation has advanced by the chosen amount of time, the next impacting atom is prepared. This process is repeated until all impacting atoms have impacted the surface.

2.9.2 Surface

To determine which atoms are part of the surface, defined as those that can be directly impacted, the system looks at all pairs of atoms and checks if the pairs are within 1.8 lattice parameters (a from Table 2, typically 3.52Å) or within 0.9 lattice parameters, chosen to capture up to next-next-nearest neighbours and nearest neighbours, trying to account for how much the atoms move. Pairs within 1.8 lattice parameters are then checked to see if the distance between the pair is mostly in the z direction. If over 90% of the distance between them is in the z direction, the higher atom will be counted as an obscuring atom of the lower atom, meaning that the higher atom is blocking impacting atoms from reaching the lower atom. Any atoms that have an obscuring atom therefore cannot be a surface atom. If a pair of atoms are within 0.9 lattice parameters, a count of nearest neighbours is incremented for both atoms. After going through all pairs, any atom with fewer than 4 nearest neighbours are likely to be atoms above the surface but not part of the surface while atoms with 12 or more are likely to be completely surrounded by atoms and inaccessible to impacting atoms. As such, only atoms with more than 3 and less than 12 neighbours are considered as surface atoms.

The average surface height is calculated from the average of the z-coordinate of the identified surface atoms. The surface roughness is then defined as the root-mean-square difference of surface atom z-coordinates and the average surface height.

2.9.3 Timestep

In the algorithm for advancing a timestep, the positions of the atoms are updated first using their current velocities and accelerations. Any atom that should be moved past a periodic boundary is moved to the equivalent position on the other side of the system and a counter is incremented to reflect it has moved across the periodic boundary. The counter is used to keep track of how far the atom has travelled away from the initial position of the system. The algorithm then updates the linked lists that determine which cell an atom is in. Then the subroutine to calculate the forces is called. Once the forces are calculated, the current acceleration on each atom is calculated and from that the velocity is calculated. For fixed atoms, these are ignored and set to zero and for atoms in the thermostated layers, the temperature of

the atoms is calculated using these velocities and the velocities are scaled by χ, the factor set out in equation 8.

2.9.4 Forces

To calculate the forces, initially, all atoms are looped over to determine the potential energy due to attractive forces acting on each atom. This is done by looping over all cells and then selected the head atom of the linked list of that cell. The distance between that head atom and every other atom in that cell and the surrounding cells is calculated and the force on the atom by each of the other atoms is summed up. This is then repeated for the next atom in the linked list until it has been calculated for every atom in the current cell, at which point the subroutine moves onto the next cell and repeats the process.

Half of all atoms are then looped over again using the energy calculated from the first loop to calculate the total potential energy and the forces acting on each atom. Only half of the atoms need to be looped over due to equation 12.

Chapter 3 – Molecular Dynamics Results

3.1 Results obtained for the Lennard-Jones potential

When the simulation code was able to simulate test runs of an atomic impact using the Lennard-Jones model without producing any noticeable errors, it was decided that the results from the code would be compared to the results of a paper by Hanson et al. (35) to attempt to validate the potential model. In this paper, they simulated ion self-sputtering using Nickel and Aluminium and analysed the sputtering rate and the sticking probability of 50 impacts at various kinetic energies and polar angles.

In the paper, they claimed to use a fcc slab with a 111 surface and 972 atoms but with 12x8x9 atoms in the x, y, z directions (which would be 864 atoms). Due to how a slab with a 111 surface was generated with our code, a slab with 12x8x9 atoms per side could be created. However, when attempting to simulate the system, it was found that the slab was too small to have periodic boundary conditions in the x and y direction using our code as any given atom would have to interact with an atom and that atom's image at the same time due to the cut-off distance used for our simulations. To prevent interactions with an atom and its image, a larger slab of 14 by 16 by 9 atoms was created. This slab was chosen as it was the smallest possible slab that could have periodic boundary conditions in the x and y direction without atoms interacting with other atoms and the images of those atoms at the same time. One of the other conditions of the slab in the Hanson paper was that the bottom two layers were fixed. It was also mentioned that the top layer was a free plane but it was unclear if this meant all other layers were thermostatically controlled. In the code in this thesis, the thermostat is applied to the 6 layers beneath the surface layer and above the two fixed layers. The slab was preheated to 300K, which led to problems detailed in section 2.7.2.

The atom impacts started by taking an atom and placing it randomly above the surface. The atom is assumed in the Hanson paper to be an “ion” that is neutralized by charge transfer before impact. The assumption was based on a paper by Kimmel and Cooper (77) that analysed resonant charge transfer and neutralisation for Li, Na and K scattered from a Cu (100) surface. The Hanson paper positioned the atom just within their cut-off but the atoms were placed outside the cut-off in the code in this work. This ensures there would be no discrepancies; only the first atom would’ve started within the cut-off and, due to the way the simulation code worked at the time, the subsequent atoms would need to cross the cut-off before impact. The impacting atom’s velocity vectors will be set to give a specific kinetic energy and polar angle but a random azimuthal angle, giving it a random vector parallel to the slab surface.

For each polar angle and each kinetic energy considered, there are five simulations that consist of 50 atomic impacts. It is unclear how the Hanson paper set up its batch of 50 impacts but they allowed the slab to move with no thermostat for 0.6ps between impact events. It is unclear if the thermostat is applied during an impact event. In our code, the 50 impacting atoms were spaced out to allow the impact to happen and let the slab settle for 0.6ps before the next atom approached the slab. The thermostat was not used at any point in the simulation. One disadvantage of this method is that we could not replicate the 90° results (grazing incidence) from the Hanson paper as our atoms would never approach the slab.

When simulating the atomic impacts, our code was set to use Nickel for the slab and impacting atoms by altering the LJ parameters, σ and ε, to the values from literature that were given for Nickel (78). On the ARCHIE-WeST HPC, the simulations were stored in groups of Kinetic Energy of the impacting atoms (25 eV, 50 eV, 75 eV and 100 eV). The job script was then set-up so that it ran 5 simulations at an incident angle of 0°, increased the incident angle to 10°, ran 5 simulations, etc. up to 80°.

From Figure 27, it can be seen that there are discrepancies between the sputter yields we obtained and the sputter yields Hanson et al obtained. For example, the 50 eV sputter yield peaks at ~0.2 at an angle of 30° in the Hanson paper but peaks above 0.5 at an angle of 20° in the results from our code. While both papers show a steep decrease in the sputter yield when moving from an incident angle of 50° to an incident angle of 70°, our results don’t show the sputter yield as zero for incident angles of 70° or more. It should be noted that one of the five simulations for 50 eV

impacts at 10° crashed before finishing for an unknown reason so the results for those conditions were less accurate. Due to the significant differences observed, it was decided that there was little value in repeating that simulation or to continue using the simulation code to perform the simulations at kinetic energies of 75 eV and 100 eV.

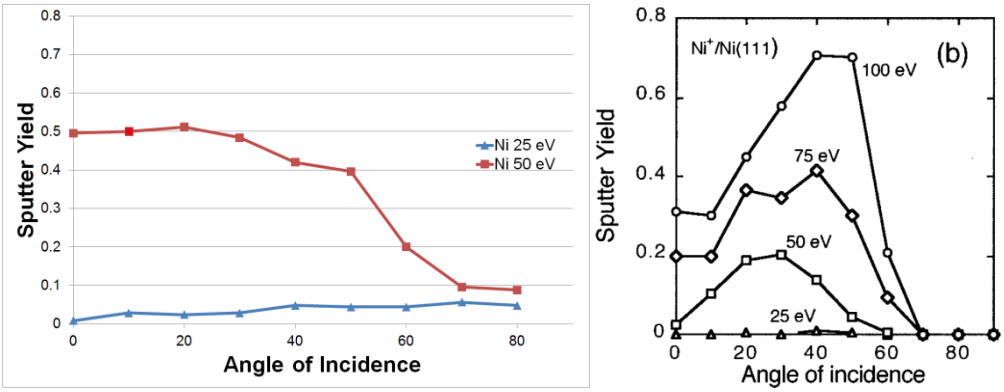


Figure 27: Sputters Yields obtained from our code for 25 and 50 eV (left) and taken from Hanson et al (35)(right)

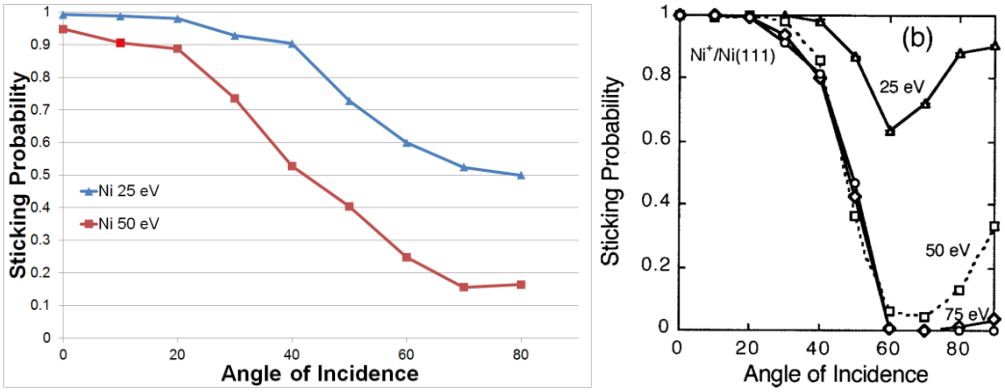


Figure 28: Sticking Probabilities obtained using our code for 25 and 50 eV (left) and taken from Hanson et al (35)(right)

Comparing the two graphs in Figure 28, variations can be seen in the sticking probabilities but the variations are less severe than those seen for the sputter yields. The Hanson paper shows that all “ions” stick to the surface at all energies for an incident angle of 30° or less and that the sticking probability rapidly decreases at an incident angle above 40°. Meanwhile, our code suggests that some “ions” don’t stick for all incident angles and energies and that the decrease in sticking probabilities is more gradual at 50 eV but more severe at 25 eV. It can also be seen that the

recovery in sticking probabilities at higher incident angles in the Hanson paper is not seen in our results.

From these results, it was concluded that the Lennard-Jones potential model was not suitable for capturing the behaviour of metal surfaces during material erosion and that a new model was needed. The Hanson paper used the many-body Voter-Chen potential (79), modified below 1.4 Å for aluminium and below 1.6 Å for nickel to reproduce density functional dimer calculations. We chose to look at the Sutton-Chen potential, which is a many-body potential that was developed later than the Voter-Chen potential.

3.2 Results obtained for the Sutton-Chen potential

After reworking the Molecular Dynamics code to use the Sutton-Chen potential, we once again compared our MD results to the paper by Hanson et al.

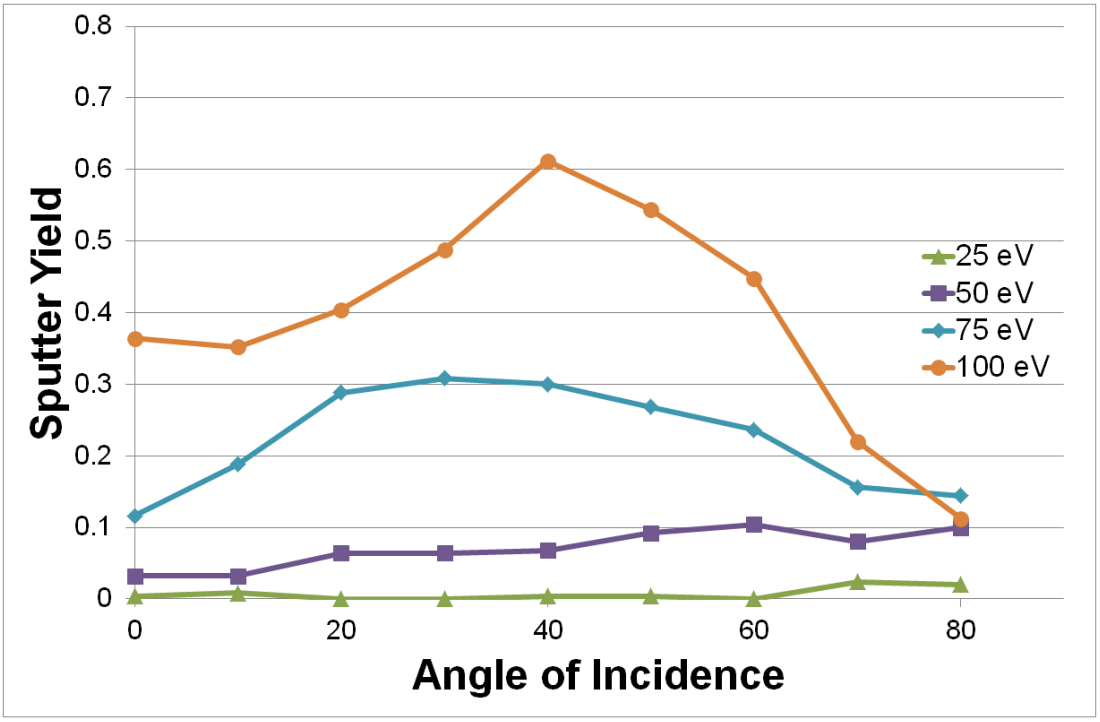


Figure 29: Sputter Yields obtained using our code when using the Sutton-Chen potential

In Figure 29, we show the sputter yields obtained at the various incident angles and kinetic energies used in Figure 27. It can be seen that for 25 and 50 eV, the MD code has much less sputtering at all angles when using the Sutton-Chen potential than it did with the Lennard-Jones potential. However, there were still discrepancies with the Hanson et al paper. Notably, our code was often showing less sputtering at incident angles between 10° and 60° while it was showing more sputtering at incident angles of 70° and above. It is unclear what caused these discrepancies as there were a number of unknown factors in the Hanson et al. paper that could not be replicated as not enough detail was supplied. Nevertheless, the trends in behaviour in the Hanson paper are similar to those we find here.

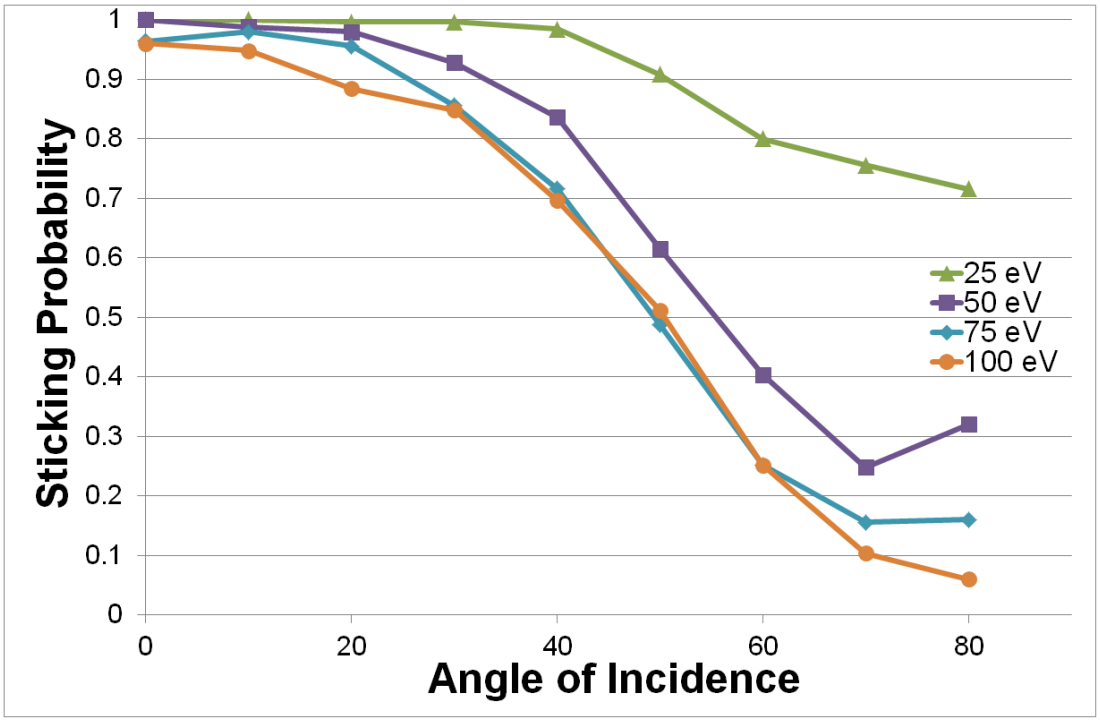


Figure 30: Sticking Probabilities obtained using our code when using the Sutton-Chen potential

Figure 30 shows the sticking probabilities of impacting atoms at the various incident angles and kinetic energies used in Figure 28. Once again, the Sutton-Chen code produced results that more closely aligned to the results by Hanson et al but still have significant discrepancies. In our code, the probability does not plummet all the way to 0 at 60° and 70° for 75 and 100 eV impacts. Our code also does not show the recovery in the sputtering probability for 25 eV impacts seen at 70° and 80° while the recovery in the probability for 50 eV impacts occurred at a higher incident angle and recovered much faster.

Analysing the Hanson et al. paper to try to better replicate the simulations, we concluded that it was not possible to adequately replicate conditions that can have a drastic effect on the statistics produced such as the thermostat due to a lack of detail. Another issue with the paper was that the crystal used in the paper was too small for our potential as, if we had used a crystal that size, atoms would be able to interact with an atom and its image at the same time. It is unclear how the Hanson et al. paper could have avoided this issue. It's also unclear how impacts in the Hanson et al. paper were affected by the impacting atom being placed within the boundary where atoms would begin to interact. This is what allowed the Hanson et al. paper to include impacts at an incident angle of 90° which our simulations lacked. Due to these rather significant issues with simulations in the Hanson et al. paper, it was decided that it was not feasible to continue making our code aim to produce their statistics.

3.2.1 Surface Impact Analysis

Despite this, it was decided that as the Sutton-Chen appeared to produce more reasonable results than the Lennard-Jones, development of the MD code would continue with the Sutton-Chen potential. It was also realised that although the sputtering and sticking statistics are important, these alone do not capture the effects of the impacts on the surface. To further capture the effects, a surface algorithm was developed. To verify that the surface algorithm was working, a simulation was run using conditions used previously to compare against the Hanson et al. paper. The conditions used had 50 impacts at a kinetic energy of 100 eV, an incident angle of 10° and a random azimuthal angle. This simulation was chosen as it produced the most amorphous surface of the previously run simulations, making it the most difficult to accurately capture. The surface obtained using the surface algorithm was compared to the trajectory file to look for major discrepancies between it and the surface that should be found.

From Figure 31, it can be seen that the algorithm perfectly captures the surface of the crystalline surface at the beginning of the simulation while in Figure 32, it captures the surface quite well even when it has become amorphous with only a few surface atoms being missed and a small amount of atoms being seen as surface atoms when they shouldn't be.

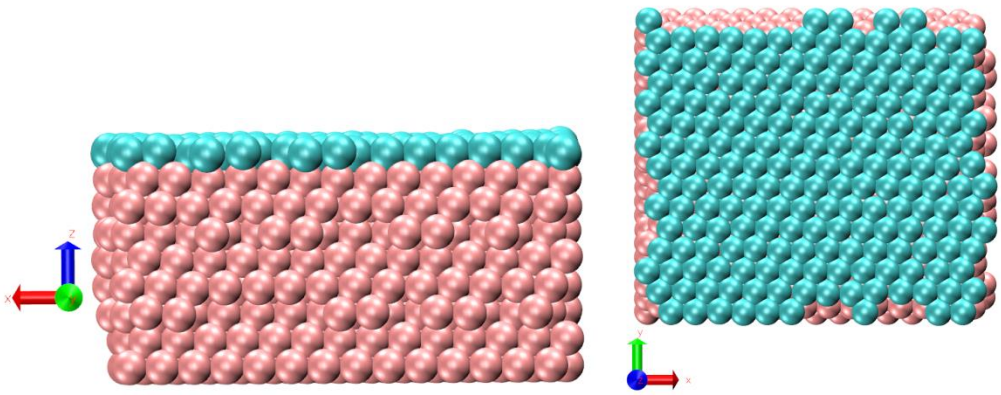


Figure 31: Side-view (left) and Top-view (right) of the comparison of the surface algorithm and a trajectory file at the beginning of a simulation. Surface atoms are in blue.

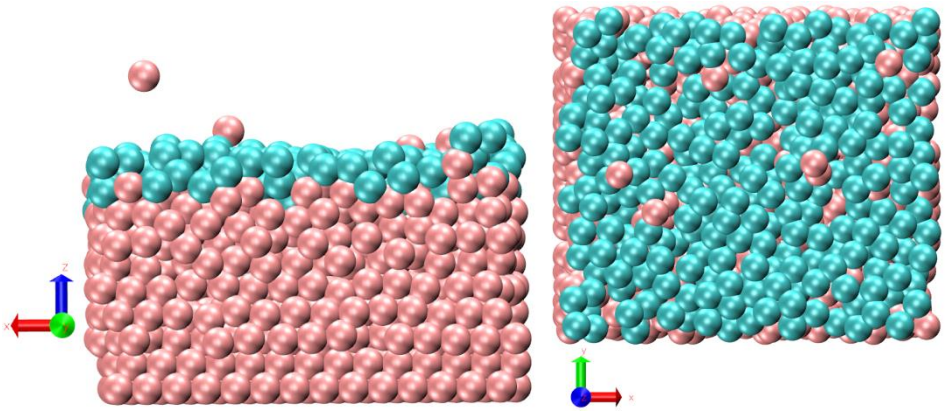


Figure 32: Side-view (left) and Top-view (right) of the comparison of the surface algorithm and a trajectory file at the end of a simulation. Surface atoms are in blue.

After checking the surface statistics were sensible when compared to the overall trajectory of the simulation, the focus became to observe how the statistics varied when changing some simulation settings. For one simulation, the slab was rapidly heated to 1800K and then rapidly cooled to 300K to create an amorphous surface before running the default simulation conditions on it. Figures 33 and 34 compare the average surface height and the surface roughness for the amorphous surface to the original surface.

In Figure 33, the average surface height was plotted against the number of monolayers deposited for simulations using a crystalline surface and an amorphous surface. The average surface height has been normalised against the height of a monolayer while the number of monolayers deposited is obtained by normalising the time taken against the flux of the impacting atoms. It can be seen that the amorphous surface had a higher initial average surface height at 0.3 monolayers

compared to 0.1 for the crystalline surface. Initially, the difference in average surface heights grew but the difference began to shrink and from 0.1 monolayers deposited onwards, the random fluctuations in height was the dominating factor for the magnitude of difference at a given point in time with the difference appearing to average at the equivalent of about 0.1-0.15 monolayers. It is believed that the crystalline surface deformed and became more amorphous as it was impacted as the average surface height increased by the equivalent of about 0.55 monolayers despite only being impact by the equivalent of about 0.25 monolayers.

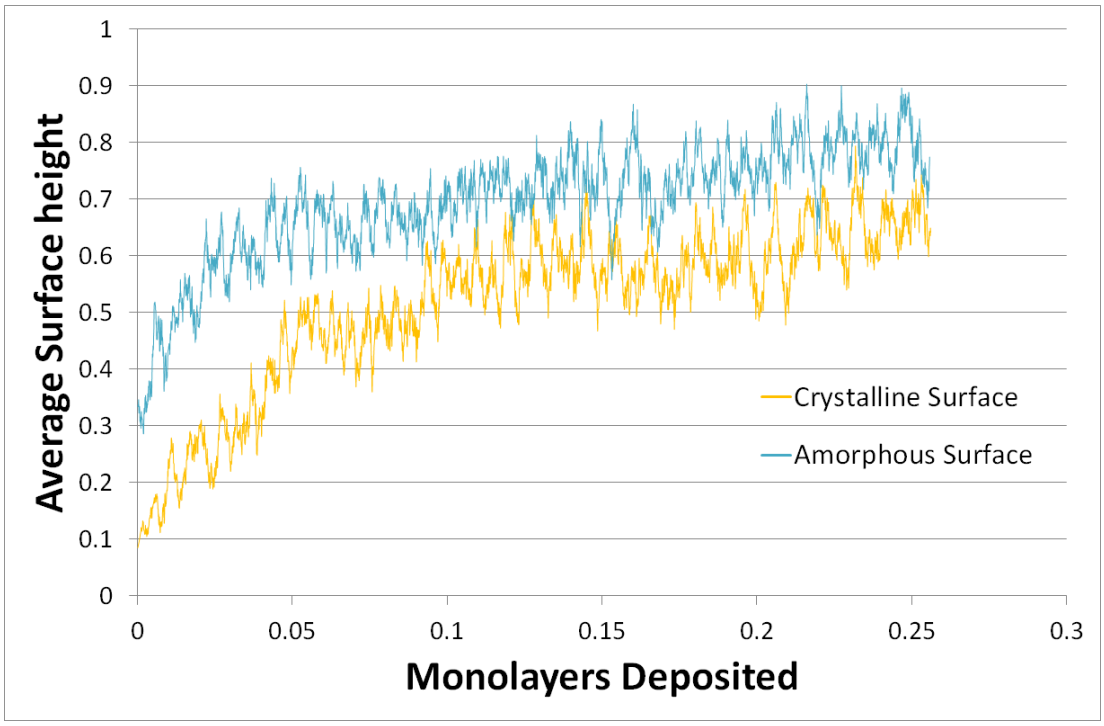


Figure 33: Comparison of average surface height between an amorphous initial surface and a crystalline initial surface

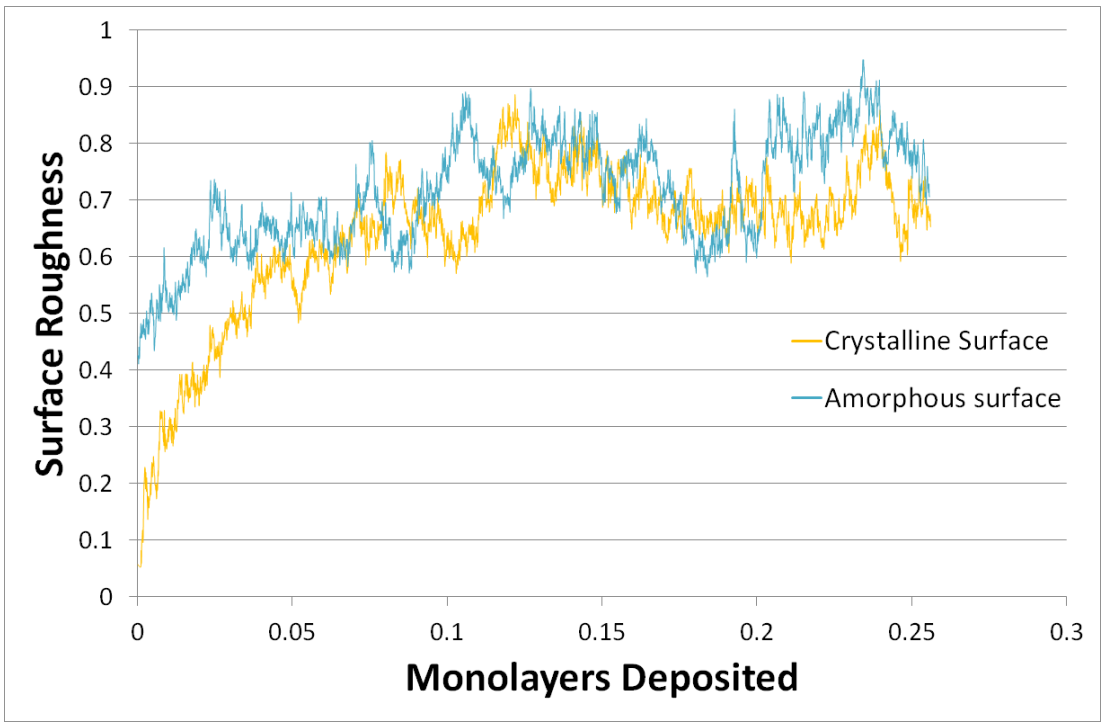


Figure 34: Comparison of surface roughness between an amorphous initial surface and a crystalline initial surface

The surface roughness was plotted against the number of monolayers deposited for simulations using a crystalline surface and an amorphous surface in Figure 34. The surface roughness, which is defined in this work as the standard deviation in the surface height across the surface at a given time, has been normalised against the height of a monolayer and like Figure 33 the number of monolayers deposited is obtained by normalising the time taken against the flux of the impacting atoms. It can be seen that both surfaces settled in the same range of 0.6-0.9 monolayers of surface roughness but the amorphous surface reached this range much quicker than the crystalline surface as the amorphous surface started at a surface roughness of 0.4 monolayers compared to the crystalline surface's initial roughness of less than 0.1 monolayers.

Other simulations looked into the effects of the size of the surface with the same conditions applied to a system with twice as many layers in the x direction and a system with twice as many layers in both the x and y directions, which doubled and quadrupled the size of the surface, respectively. The surface statistics obtained can be seen in Figure 35 below.

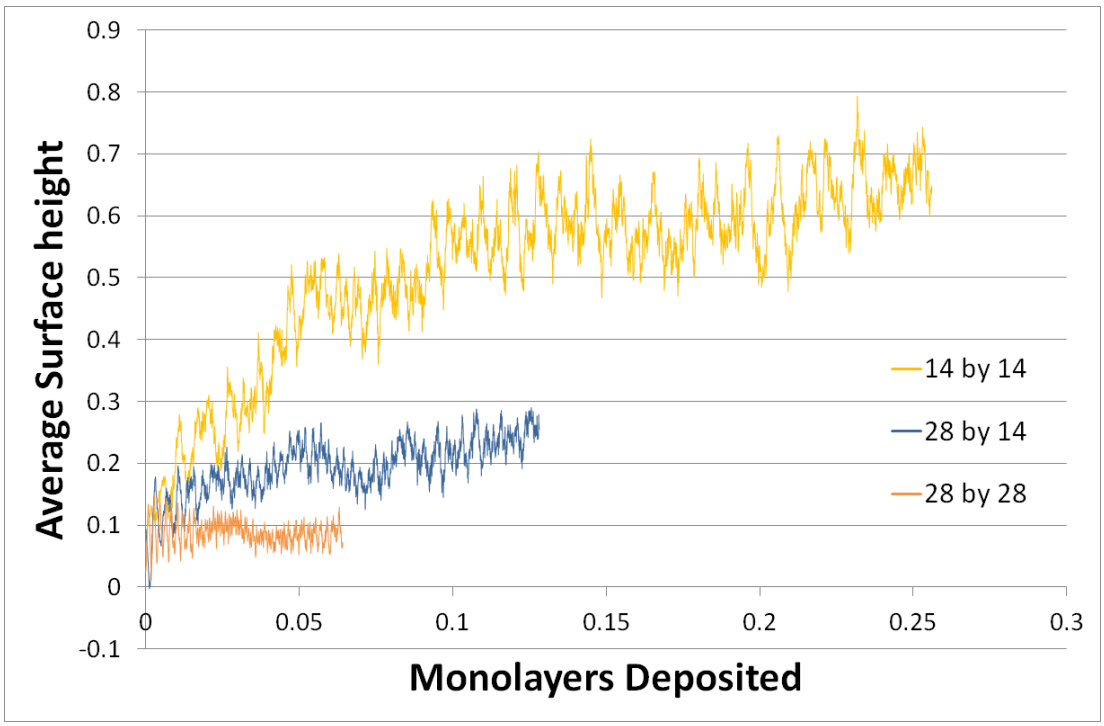


Figure 35: Comparison of average surface height between the 14 by 14, 28 by 14 and 28 by 28 surfaces after 50 impacts, equivalent to depositing 0.255, 0.128 and 0.064 monolayers, respectively

In Figure 35, it can be seen that the larger surfaces saw substantial reductions in average surface height when compared to the 14 by 14 surface with the 28 by 14 surface growing less than half as much and the 28 by 28 surface seeing little to no growth. It was theorized that this was due to the width of the 14 by 14 surface being too small, in comparison to the diameter of the shockwave that ripples through the slab after an impact, causing significant and unrealistic interactions between an impact shockwave and its image, which would further distort the surface. This would also explain the substantially different growth rates for the average surface height between the 28 by 14 surface and the 28 by 28 surface as the 28 by 14 surface would still be partially affected by the unrealistic interactions. It is unclear if these finite size effects are completely gone in the 28 by 28 surface of if a larger surface would be required.

In Figure 36, it was noted that the larger surfaces were being roughened at a slower rate and that the 14 by 14 surface was erratically changing roughness with sudden dips and spikes changing the surface roughness by the equivalent of over 0.1 monolayers.

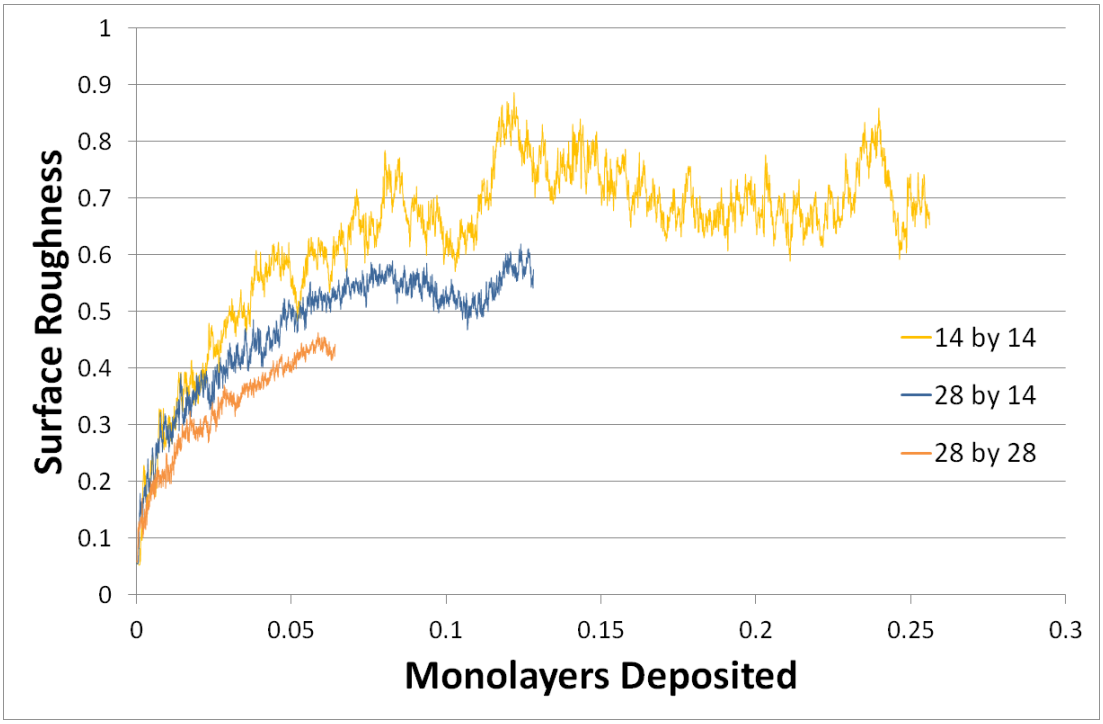


Figure 36: Comparison of surface roughness between the 14 by 14, 28 by 14 and 28 by 28 surfaces after 50 impacts, equivalent to depositing 0.255, 0.128 and 0.064 monolayers, respectively

To determine what range of values the surface roughness would settle in, the number of impacts could be increased. Planned analysis of the effects of an increased number of impacts on the 14 by 14 surface by running a simulation with 500 impacts instead of 50 was repeated for the two larger surfaces. This would be equivalent to depositing 2.551 monolayers on the 14 by 14 surface, 1.276 monolayers on the 28 by 14 surface and 0.638 monolayers on the 28 by 28 surface.

In Figure 37, the effect the increased number of impacts has on the average surface height was investigated. It can be seen that the finite size effects only causes non-linear growth in the surface height during early impacts. While it is unclear if the 28 by 28 surface is affected by the finite size effects, the surface height held roughly constant during the early impacts. After the equivalent of ~0.15 monolayers are deposited, the rate of growth for all simulations became linear as expected. However, it was noted that the surface growth rate was larger for the 14 by 14 surface than it was for the 28 by 14 and 28 by 28 surfaces. The 28 by 14 surface also appeared to grow at a slightly faster rate than the 28 by 28 surface. This suggests that the finite size effects influence the surface growth rate during later impacts but not as significantly as it did during the earlier impacts.

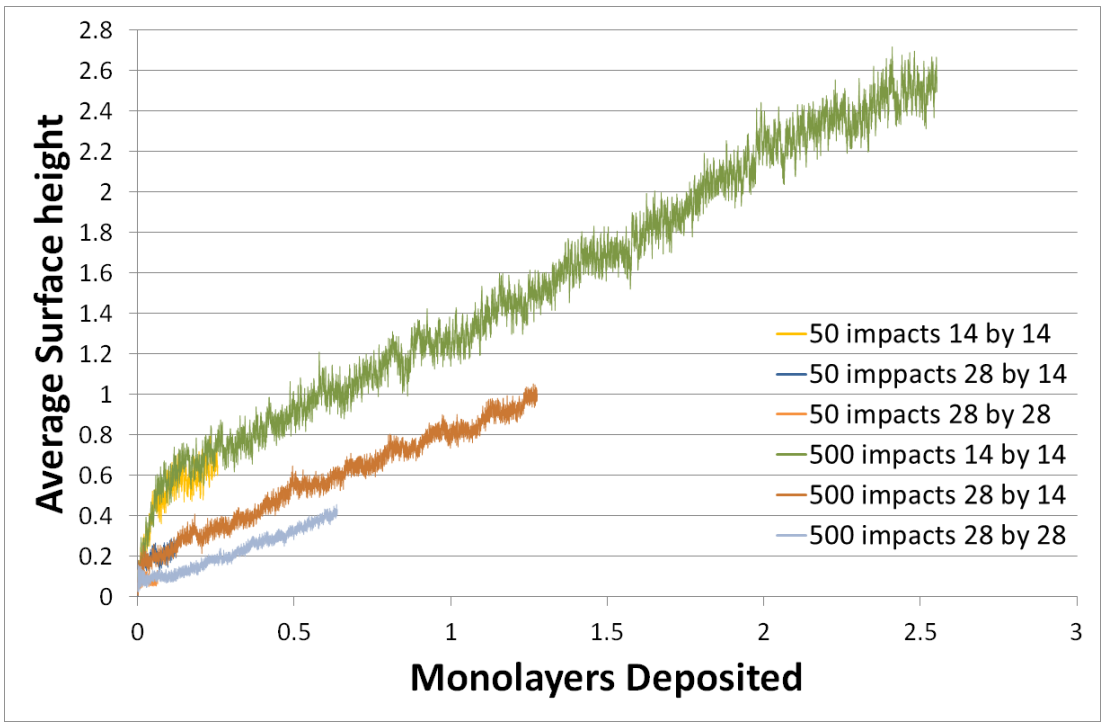


Figure 37: Comparison of average surface height between surfaces after 50 impacts and surfaces after 500 impacts

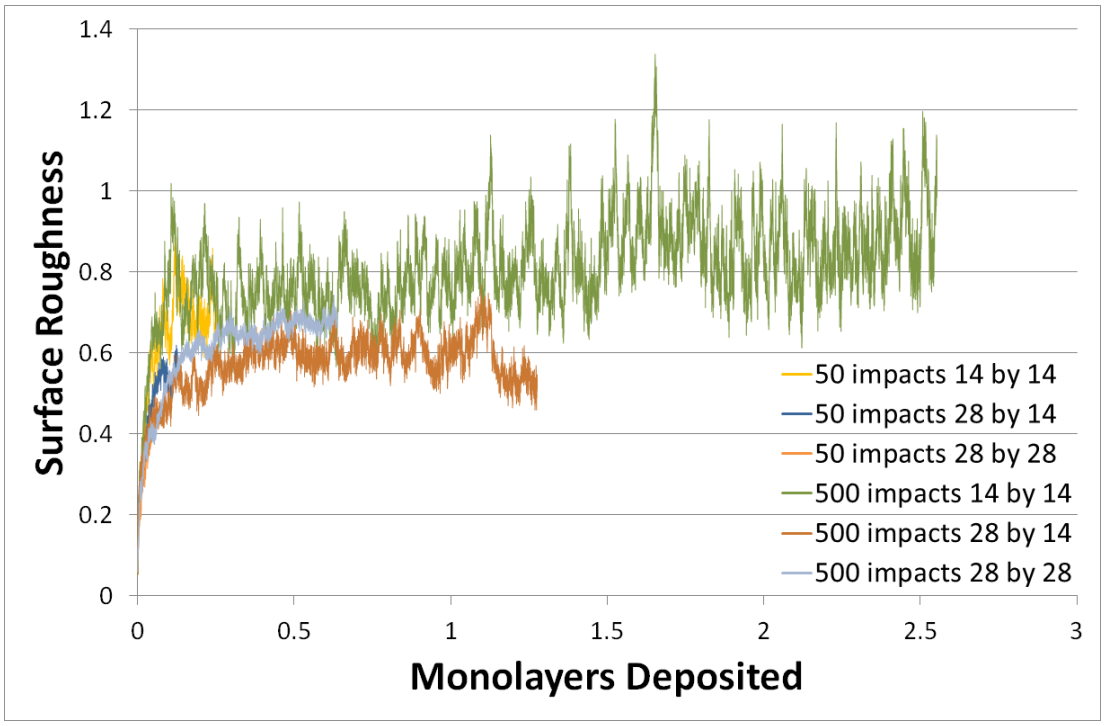


Figure 38: Comparison of surface roughness between surfaces after 50 impacts and surfaces after 500 impacts

The effect the increased number of impacts has on the surface roughness was analysed in Figure 38. The influence of the using random number generation for the position and azimuthal angle of impacts can be seen with a portion of the 28 by 14 surface differing by the equivalent of over 0.1 monolayers between the 50 impact and 500 impact simulations. It can also be noted that as anticipated earlier, the larger surfaces do not settle into the same range of values for surface roughness as the 14 by 14 surface. However, it was unexpected that the 28 by 28 surface would roughen more than the 28 by 14 surface.

3.2.2 Thermostat Analysis

We now analyse the effects the overall delay between impacts and the amount of thermostated time between impacts had on the surface. To begin with, 1 picosecond was non-thermostated and a few overall times were tested. From this, it was decided that 4 ps must be thermostated to allow the system to cool back to the original temperature between impacts. Next, the overall time between impacts was altered without changing the amount of thermostated time. The amounts of non-thermostated time chosen were 0 ps, 0.6 ps and 1 ps. The effects of these changes compared to the original simulation, which had no thermostated time and 0.6 ps non-thermostated time, can be observed in Figure 39 and Figure 40 below.

The effect that different delays between impacts have on the average surface height was investigated in Figure 39. It can be seen that the overall time between impacts made little difference to the average surface height with all three simulations that were using a 4 ps thermostated impact delay all following similar patterns of surface growth. The most noticeable difference between these three was that the simulation using 4.6 ps between impacts (blue) had much greater fluctuations in the average surface height than the 4 ps (red) and the 5 ps (green). It is unclear why this was observed when using 4.6 ps but not when using 5 ps. While the overall time between impacts had little effect, the increase in thermostated time has led to a substantial reduction in the average surface height compared to the original simulation (yellow), which uses no thermostated time and only 0.6 ps between impacts. It was also observed that the average surface height decreases during the early impacts for the three with the 4ps thermostated delay. The cause of this decrease is discussed later in Figures 41, 42 and 43.

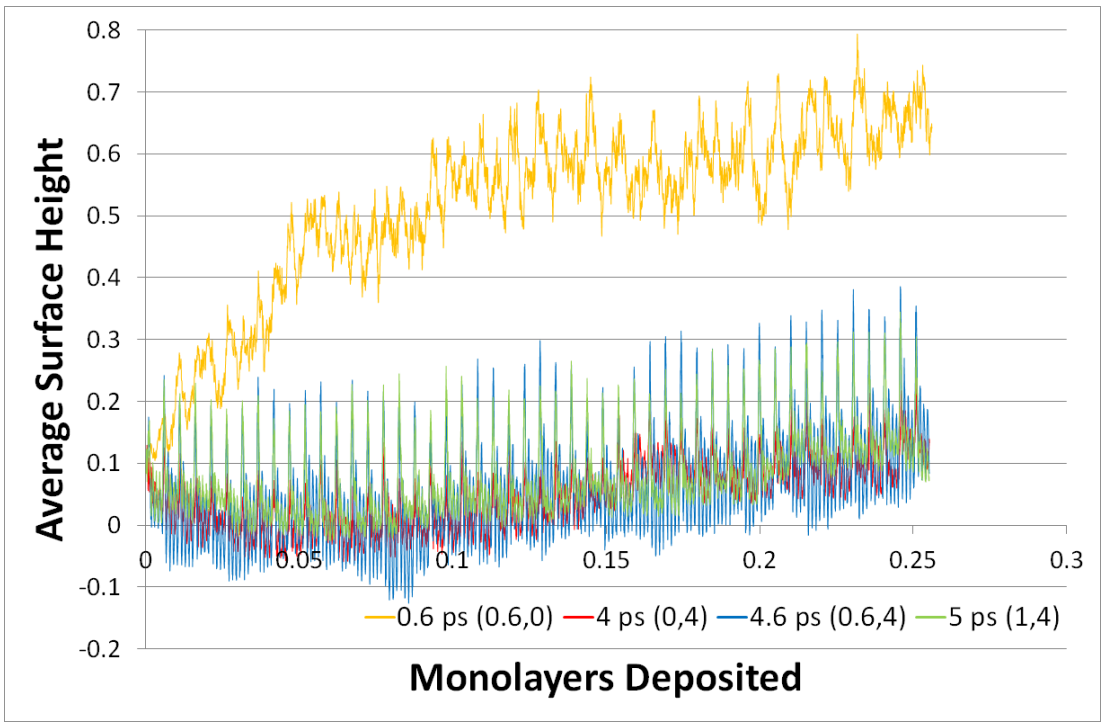


Figure 39: Comparison of average surface height between surfaces with different delays between impacts. The legend details the time between impacts with the bracketed numbers referring to the time in picoseconds where the thermostat is not active and the time that the thermostat is active, respectively

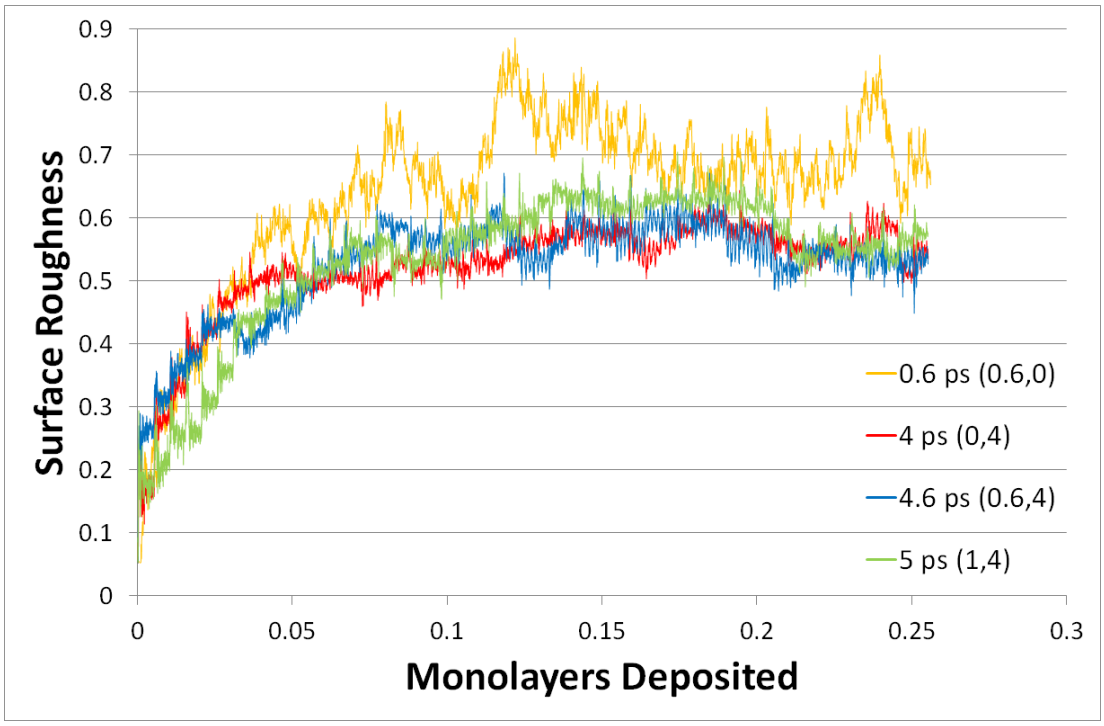


Figure 40: Comparison of surface roughness between surfaces with different delays between impacts

In Figure 40, the effect that different delays between impacts have on the surface roughness was analysed. While there is a substantial difference between the average surface height between the simulations with the 4 ps thermostated delay and the original simulation with only the 0.6 ps delay without a thermostat, the surface roughness of the simulations are quite similar with the longer simulations having a slightly lower range of values and being less prone to large fluctuations.

It can be concluded that the optimal delay between impacts is 4 ps with the thermostat as this reduces simulation time as much as possible without causing the unrealistic interactions across periodic boundaries. It also seems reasonable that, in a real system, the temperature would always be regulated by the system dissipating the energy from the impact.

The effects of more impacts and larger surfaces were re-examined using the optimal impact delay. To start with, 2 even larger surfaces (56 by 28 and 56 by 56, 8 and 16 times larger than the original surface, respectively) were included but the 56 by 56 surface was removed as it was too costly to run with the run-time constraints we were dealing with. The larger slabs were also to be impacted much more so that the equivalent number of monolayers deposited would be 2.551 (e.g. 1000 impacts for the 28 by 14 surface), which is the same as the amount deposited on the 14 by 14 surface, but again due to run-time constraints, this had to be scaled back for the 28 by 28 and 56 by 28 surfaces, which were run for 1000 and 500 impacts instead of 2000 and 4000 impacts, respectively. We decided to run a second simulation for both of these surfaces, again using 1000 and 500 impacts for the 28 by 28 and 56 by 28 surfaces, respectively, using the final surface obtained from the first simulation as the starting point of the second. This meant that overall the 28 by 28 surface had been impacted the desired number of times while the 56 by 28 surface had still only had a quarter of the desired number of impacts.

In Figure 41, the average surface heights were plotted during deposition of 500 atoms on the 14 by 14, 28 by 14 and 28 by 28 surfaces using the original 0.6 ps between impacts with no thermostat and the average surface heights during deposition of 500 atoms on the 14 by 14 surface, 1000 atoms on the 28 by 14 and 56 by 28 surfaces and 2000 impacts on the 28 by 28 surface using the 4 ps thermostated impact delay. It can be seen that when using 4 ps between impacts, there is a small equilibration period where the average surface height of all surfaces

shrank near the start of the simulation before the surface growth becomes flux-dependent.

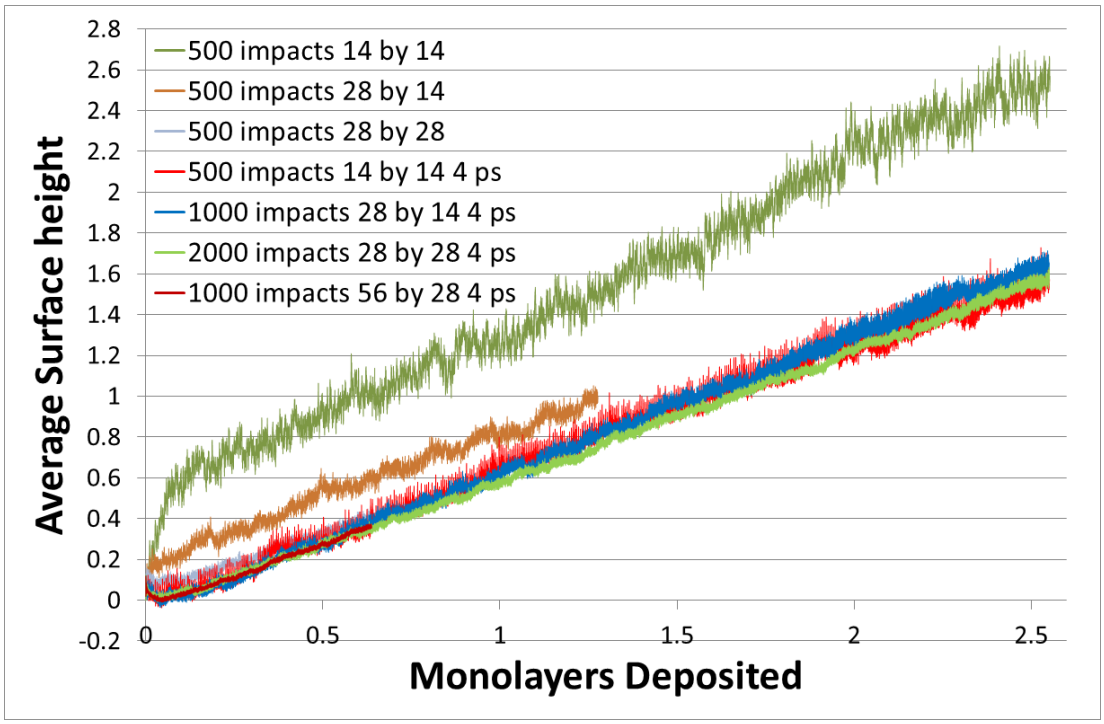


Figure 41: Comparison of average surface height between surfaces using 4 ps thermostated impact delay and surfaces that used the original 0.6 ps impact delay

It is believed that the decrease in surface height is seen using the surface identification algorithm due to the initial roughening of the surface causing subsurface atoms to be exposed by vacancies in the original surface layer, causing the atoms to be seen as surface atoms in the algorithm. The total number of surface atoms increases because for every atom on the original surface that is sputtered, up to three subsurface atoms are exposed by the resultant vacancy. The overall change to the surface detected by the algorithm is effectively that one of the surface atoms is lowered and up to two more atoms at this lower height are added. The total number of surface atoms is also increased by impacting atoms that stick to the surface as these adsurface atoms do not initially cover any of the original surface atoms below them, causing them to still be identified as surface atoms by the algorithm. However, this increase in surface atoms does not fully offset the decrease in surface height caused by sputtering. As the surface becomes roughened, the total number of surface atoms is affected less by sputtering and sticking. Sticking has less of an effect as adsurface atoms are more likely to entirely cover surface-level atoms, at which point the surface-level atoms are no longer identified as part of the

surface by the algorithm. The effect of sputtering is also reduced as atoms that sputtered are more likely to be on the edge of an island or the edge of a vacancy. Sputtering these atoms exposes less subsurface atoms. As the change in the total number of surface atoms decreases, the surface height is affected more by the overall rate of growth or erosion, which is determined from the sputter yield and the sticking probability. Eventually, the total number of surface atoms only slightly fluctuates and the rate of growth or erosion becomes the only significant factor in determining the surface height.

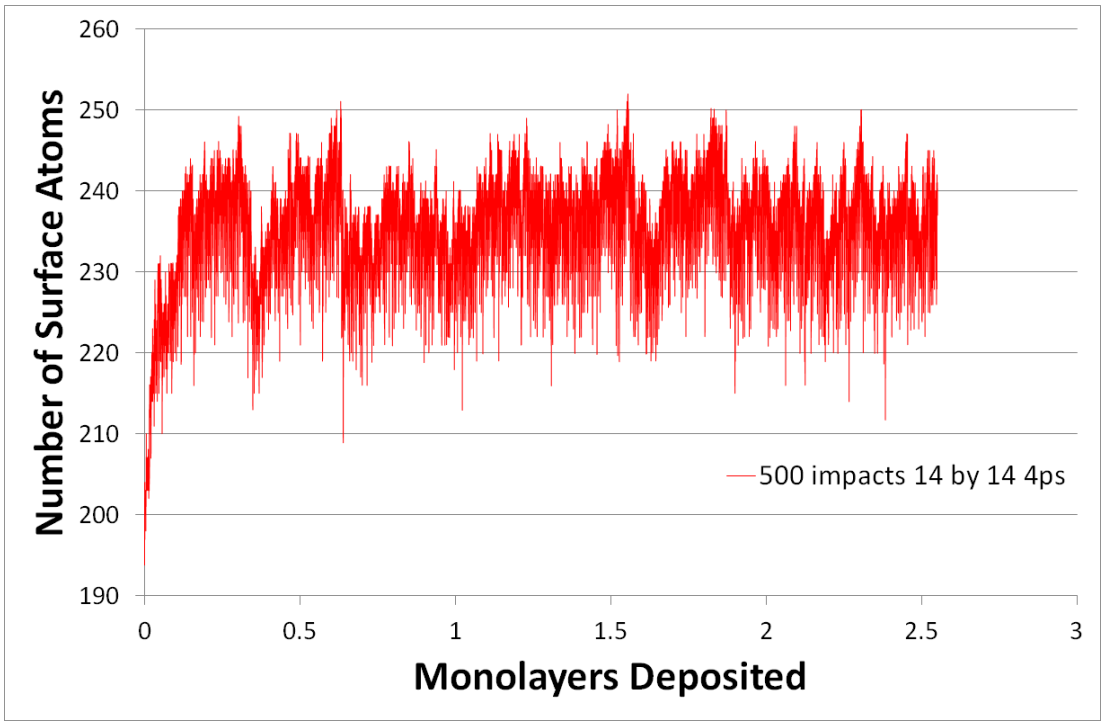


Figure 42: Number of surface atoms in an MD simulation

An example of how the number of surface atoms changes throughout a simulation can be seen in Figure 42. The rapid increase from the starting number of 196 surface atoms to the range of 220-250 coincides with the decline in the average surface height.

Figure 43 shows the surface of the 56 by 28 system initially and at a point after the deposition of some atoms where the average surface height had decreased. Comparing the two, it can be seen that a large number of islands and vacancies have formed on the surface. It can also be seen that while there are more island atoms than there are vacancies, there are a larger number of atoms exposed by the vacancies then there are island atoms.

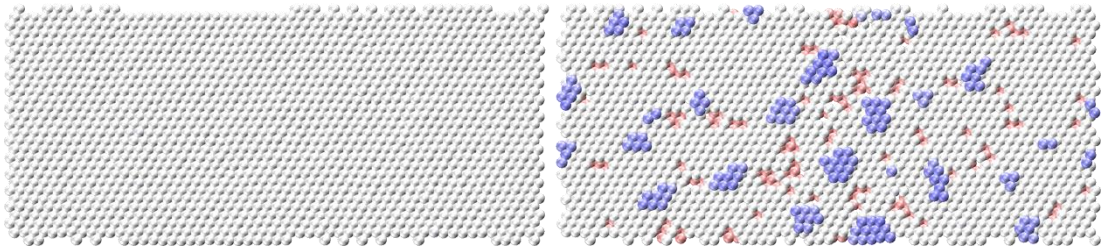


Figure 43: Initial 56 by 28 surface and the surface when the average surface height decreased. A colour gradient denotes the height of atoms relative to the original surface with the original surface layer in white. The gradient starts two layers below the surface in red and ends two layers above the surface in blue.

Looking back at Figure 41, the linear growth is inversely proportional to the size of the surface as the height of each surface is approximately the same after a given number of monolayers are deposited. It can also be seen that for 0.6 ps between impacts, the growth is dependent on slab size as well as the flux with smaller surfaces growing significantly during the equilibration period. Despite this, the 28 by 28 surface has a similar growth pattern to the simulations using 4 ps with marginally higher surface heights. This confirms that the rapid surface growth for the smaller surfaces is a finite size effect and further suggests it is caused by interactions with an impact shockwave and its image. However, as the 28 by 28 surface still has some differences with the simulations using 4 ps, it is possible there are still some finite-size effects and an even larger surface would be needed to fully remove the effects. However, it was considered much more likely that the depth of the system is too small to adequately dissipate the energy of the impact, allowing the impact energy to propagate in a wave to the fixed layers before being reflected back toward the surface, which is unrealistic. If this was the case, the figure shows that using the thermostat damped down the waves, preventing them being reflected. This would be expected as the thermostat time constant was set to 100 fs but the mimimum time that a sound wave would take to travel through the nickel system's thermostated layers (~12Å) is ~199 fs taking the speed of sound for nickel as 6,040 m/s (80). Another possible solution would be increasing the slab depth but this was not considered to be feasible as it would increase the computational cost significantly.

In Figure 44, the surface roughness for the same simulations shown in Figure 41 was plotted. All simulations appear to follow a similar trend with rapid roughening at the beginning before the roughness seems to level off as more monolayers are deposited. The simulations using the 0.6 ps impact delay have significantly more

fluctuations than those using the 4 ps impact delay but again, the 28 by 28 surface using the 0.6 ps impact delay more closely follows the trend of the 4 ps impact delay simulations.

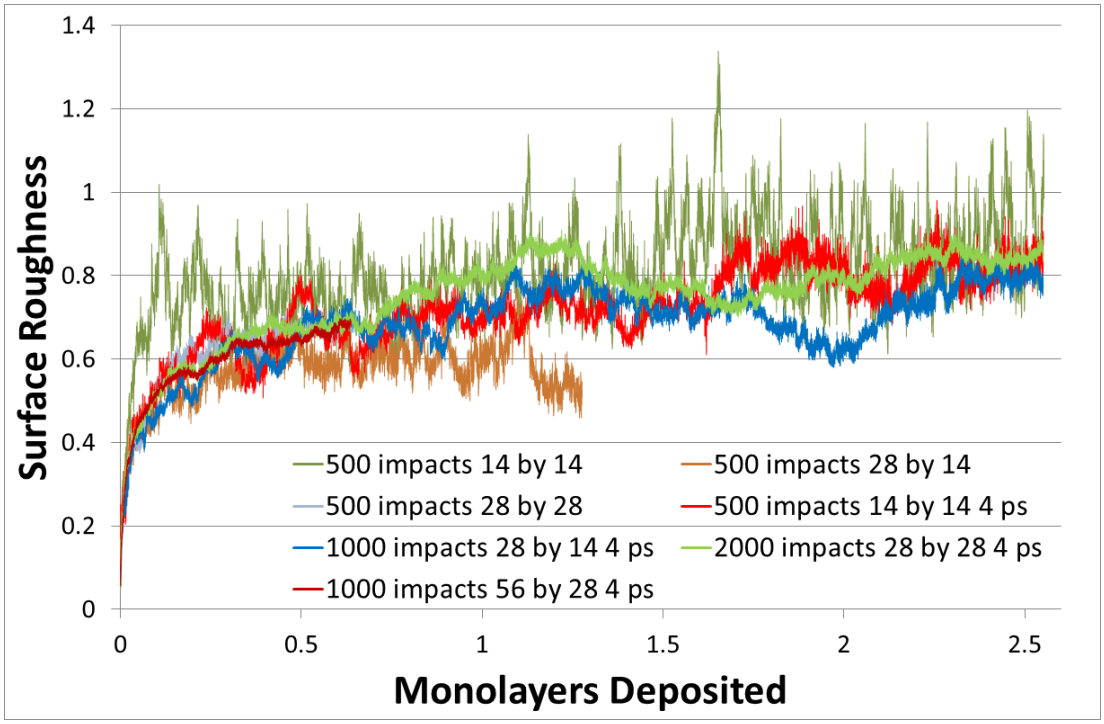


Figure 44: Comparison of surface roughness between surfaces using the 4 ps impact delay and surfaces that used the original 0.6 ps impact delay

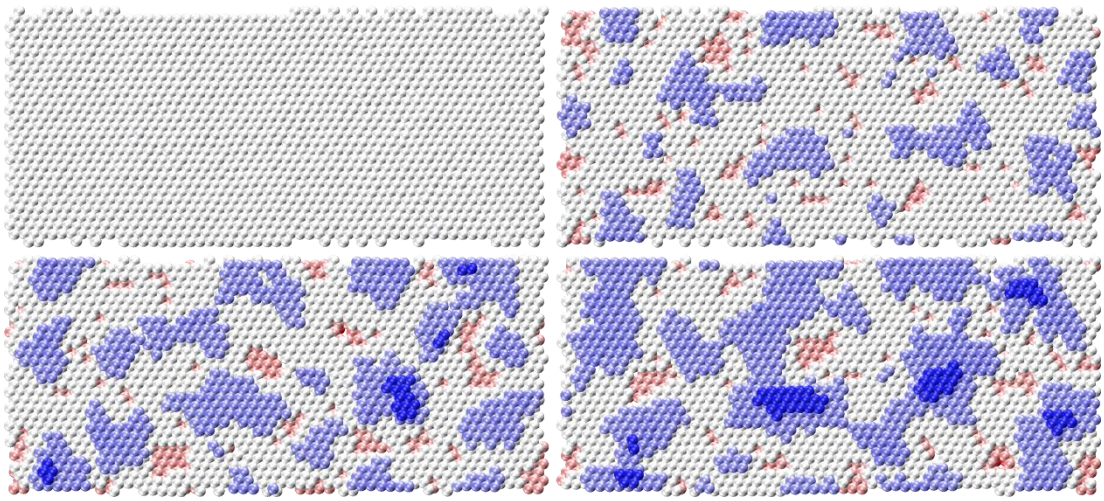


Figure 45: Surface of the 56 by 28 system during the deposition of 1000 atoms. This figure uses the same colour gradient to denote the height of an atom as Figure 43.

Figure 45 shows the surface of the 56 by 28 system initially and after the deposition of ~333, ~667 and 1000 atoms. It can be seen that after approximately 333 atoms

were deposited, several islands have formed across the surface alongside lots of smaller vacancies. After more atoms are deposited, approximately 667, the islands have become larger and islands have begun forming atop islands producing multi-layer islands. The vacancies have also become larger but have reduced in quantity. At the end of the simulation, after the deposition of 1000 atoms, the islands have become even larger and have begun merging with each other. The multi-layered islands have also grown both in size and in number as the islands grew. Vacancies, on the other hand, have begun to shrink and decrease in number.

The simulations were later repeated using the 4 ps impact delay on the 28 by 28 and 56 by 28 surfaces 4 times to analyse how stochastic fluctuations affect the surface statistics. These repeats only used 1000 impacts for the 28 by 28 surface and 500 impacts for the 56 by 28 surface.

Figure 46: Comparison of average surface height for multiple simulations on the 28 by 28 surface using the same input conditions

In Figure 46, the comparison of five simulations run using the 28 by 28 surface and the same conditions is shown to analyse how the stochastic nature of the simulations affects the average surface height. It can be seen that all five simulations follow the same trend. The variation between the simulations is small with each simulation being on average within 0.02-0.05 monolayers of all of the

others. The second simulation (red) appears to grow faster than the other simulations after the equivalent of one monolayer is deposited with the surface height approaching 0.8 monolayers at the end of the simulation while the rest have surface heights around 0.7-0.75 monolayers.

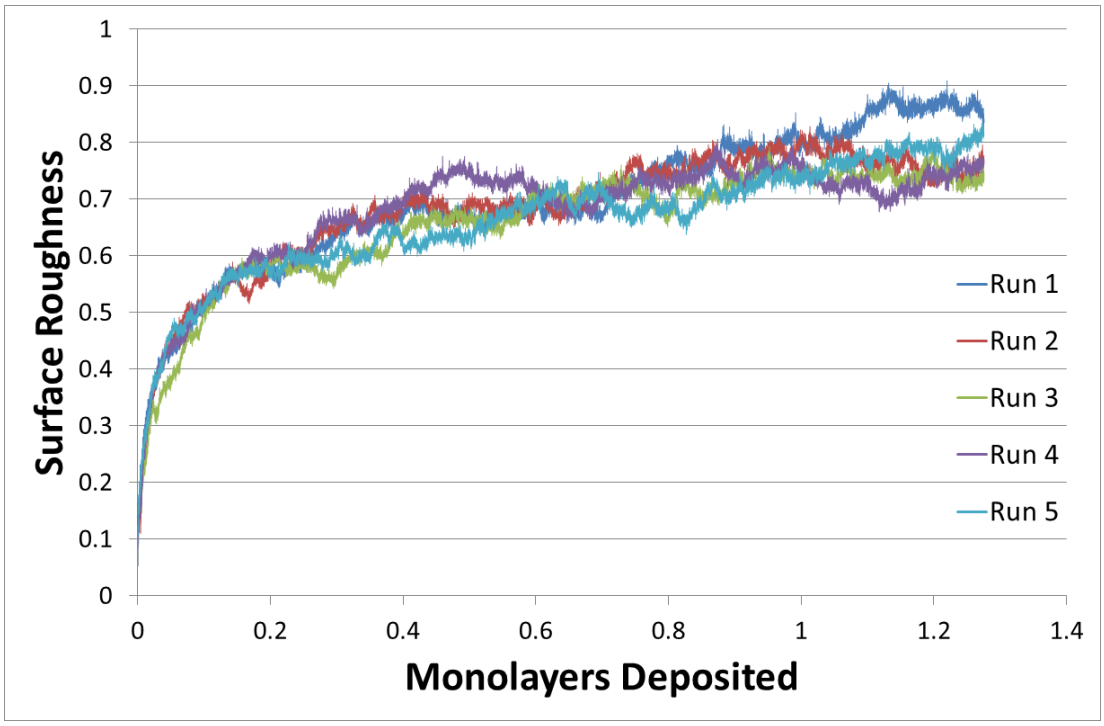


Figure 47: Comparison of surface roughness for multiple simulations on the 28 by 28 surface using the same input conditions

The comparison of five simulations run using the 28 by 28 surface and the same conditions to analyse how the stochastic nature of the simulations affects the surface roughness is shown in Figure 47. All five simulations appear to follow the same trend but the difference between the surface roughnesses of each simulation is much greater than the difference in surface heights. The fourth (purple) and fifth (cyan) simulations had a difference of the equivalent of about 0.15 monolayers at ~0.5 monolayers deposited while the fourth and first (blue) simulations were close to a difference of 0.2 monolayers at ~1.15 monolayers deposited.

In Figure 48, the effect of the stochastic nature of the simulations on the average surface height is analysed by comparing five simulations run using the 56 by 28 surface and the same conditions. Similarly to the 28 by 28 surface, the 5 simulations seem to follow the same trend from surface growth, staying within 0.02-0.05 monolayers of one another. After 0.25 monolayers are deposited, the fourth

simulation (purple) appears to start growing faster than the rest of the simulation. This would suggest that the balance of island atoms and vacancies on the surface has tipped towards more island atoms per vacancy. This is expected to only be temporary and the average surface height of the fourth simulation will likely fall back into the same range as the rest of the simulations.

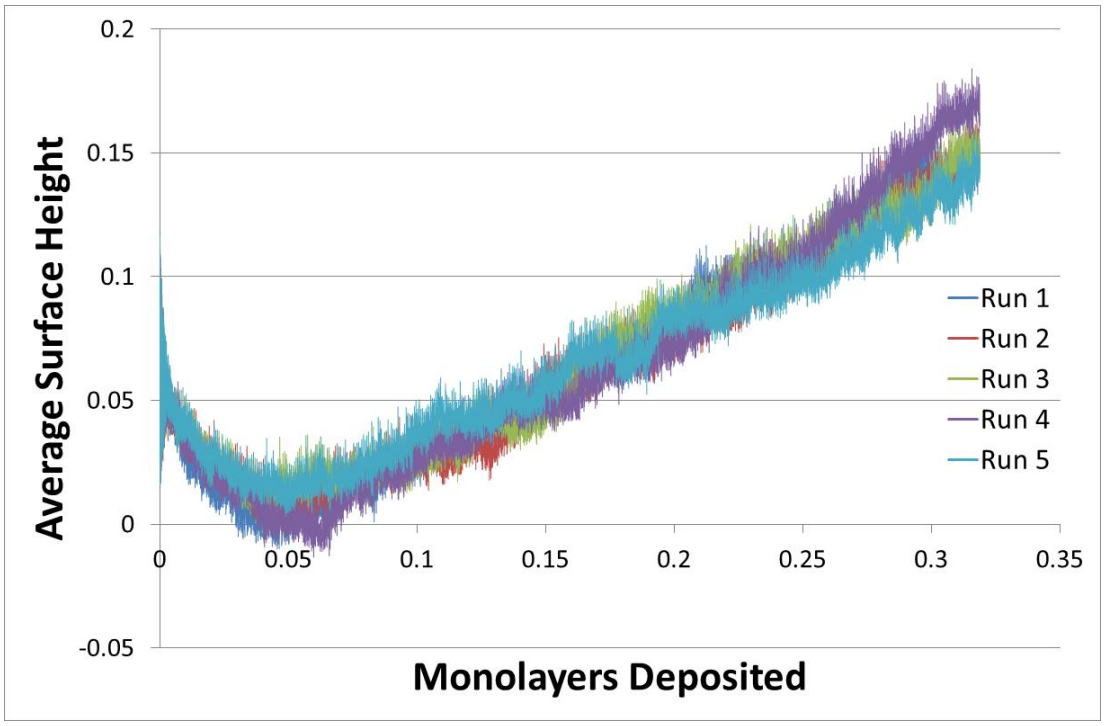


Figure 48: Comparison of average surface height surface roughness for multiple simulations on the 56 by 28 surface using the same input conditions

How the stochastic nature of the simulations affects the surface roughness was analysed in Figure 49 by comparing five simulations run using the 56 by 28 surface and the same conditions. Like the 28 by 28 surface, all 5 simulations appear to follow the same trend and the magnitude of the differences between the simulations is much greater for surface roughness than it is for the average surface height. There is an odd kink in the fourth simulation (purple) after 0.25 monolayers deposited where the surface roughness drops by the equivalent of ~0.03 monolayers. This is clearly linked to the sudden increase in the surface growth rate seen at roughly the same point of the simulation in Figure 48. This could be caused by a sudden change in the surface such as a large vacancy in a surface layer becoming filled. When a vacancy is filled, the surface becomes smoother and locally increases its height while also reducing the number of atoms seen as surface atoms.

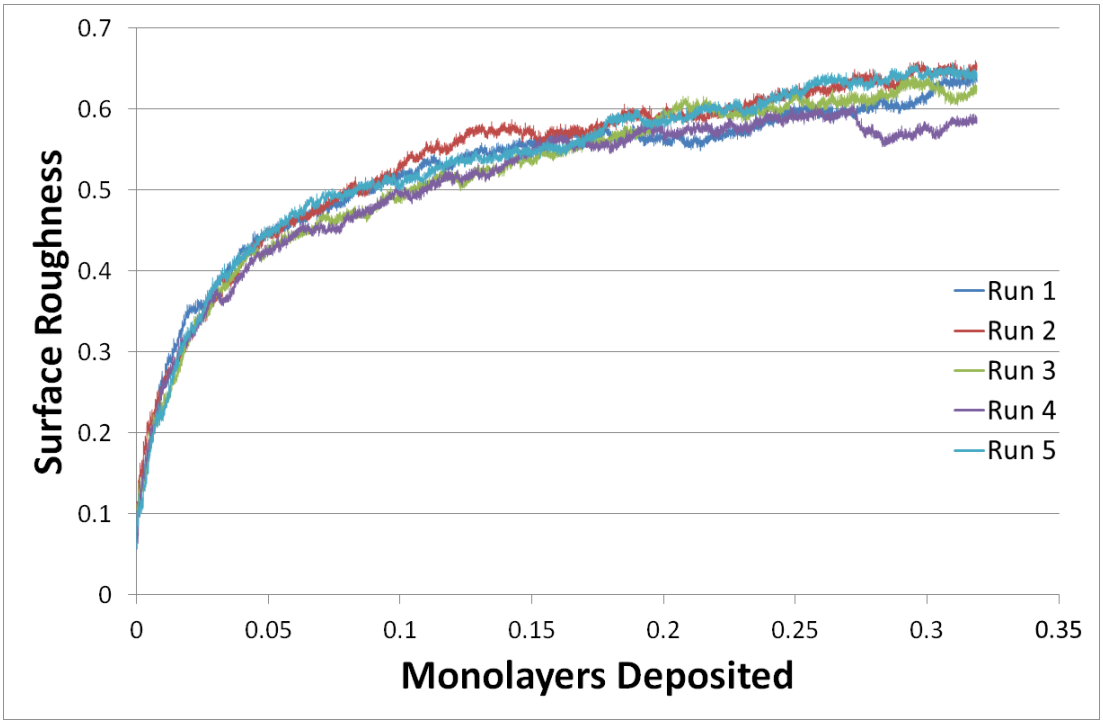


Figure 49: Comparison of surface roughness for multiple simulations on the 56 by 28 surface using the same input conditions

3.2.3 Impact Angle Analysis

It was then decided to analyse how changing the polar angle (see Figure 24) of the impact affected surface growth. Simulations of the 56 by 28 surface being impacted 1000 times successively were run at polar angles from 10° to 80° using 100 eV impacts and a 4 ps thermostated delay between impacts, like previous simulations. It was now possible to use 1000 impacts on the 56 by 28 surface within the run-time constraints because a compiler flag was discovered that, when used, caused the code to run more efficiently and almost halved the time needed for simulations.

Figure 50 analyses the effect of the polar angle on the average surface height. For polar angles of 10°, 20° and 30° (blue, red and green, respectively), the surface growth exhibits the same trend of a slight decrease during equilibration and a linear increase afterwards but the rate of growth decreases with increasing polar angle. When using a polar angle of 40° (purple), it was seen that there was a much sharper decline in the surface height during equilibration before the surface begins to grow very slowly. The same sharp drop in surface height is also seen for 50° (cyan) but

the surface has stopped growing and now erodes as more is deposited. Erosion is also observed at 60° (orange) and the rate of erosion has increased but the surface height does not fall as sharply during the equilibration as it did for the 50° simulation, though it still decreased more than the 10°-30° simulations. The erosion rate appears to remain about the same for 70° (light blue) and 80° (pink) but the rapid decrease in the surface height seen during early impacts in other simulations appears to be replaced with a more gradual decline over a longer period with the decline appearing to be greater than the erosion rate of later simulations at 70° and less than the erosion rate of later simulations at 80°.

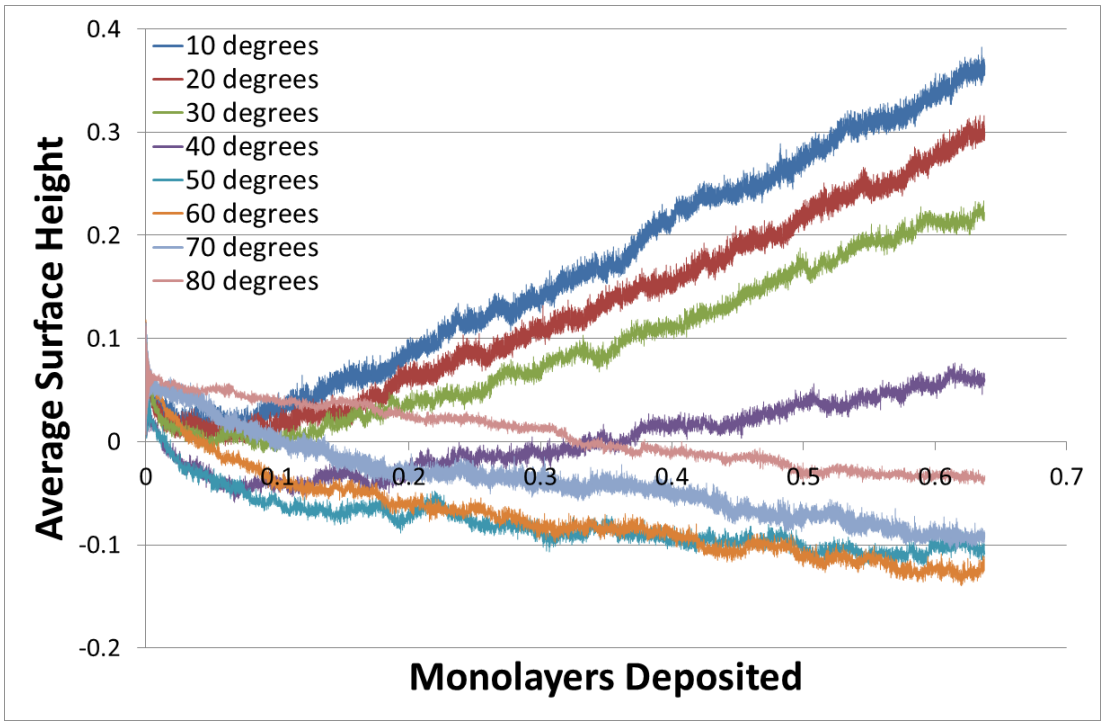


Figure 50: Average surface height at various polar angles

Figure 51 displays the effect of the polar angle on the surface roughness. It can be seen that as the polar angle increases, the surface roughness decreases but this effect is less dominant than random fluctuations in the roughness as the surface evolves. This is evident when comparing the 10°-50° simulations as, while the 10° simulation (blue) is most often the most roughened and the 50° simulation (cyan) is often the least roughened, the 20°, 30° and 40° simulations (red, green and purple, respectively) all briefly become the most and least roughened at various points during the simulation. The effect of increasing the polar angle becomes more prominent at 60° where the difference in surface roughness between 50° and 60° is

often greater than the difference between 10° and 50°. At 70° and 80°, the surface roughened even slower and unlike the other systems, rapid roughening was not seen near the beginning of the simulations.

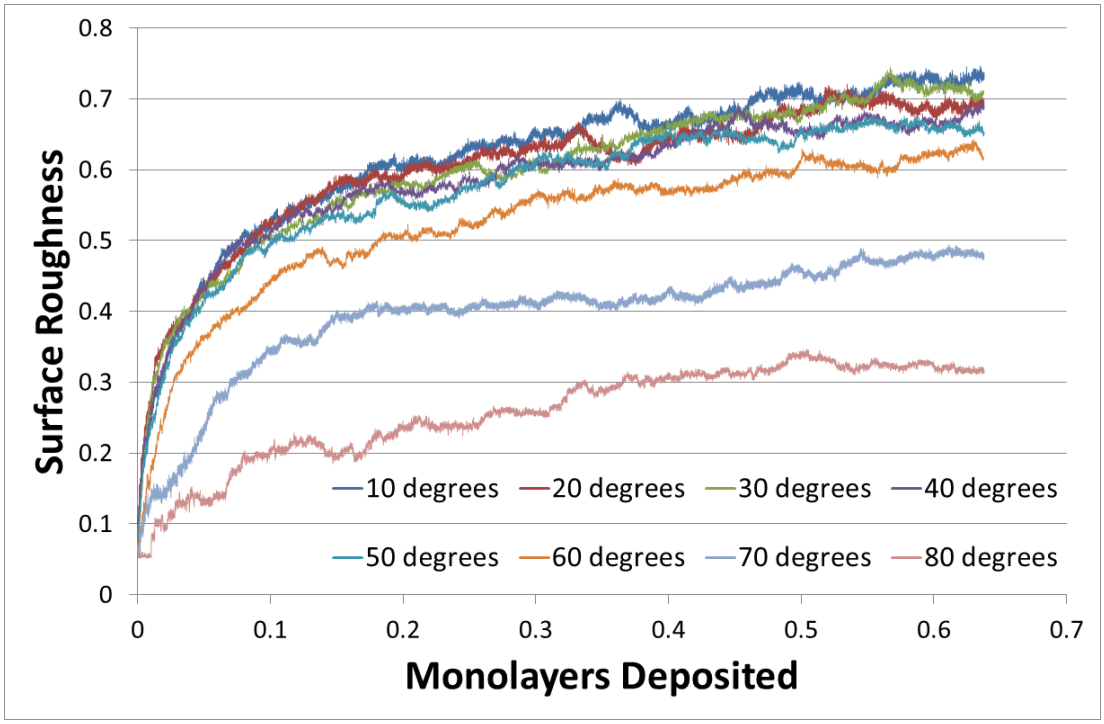


Figure 51: Surface roughness at various polar angles

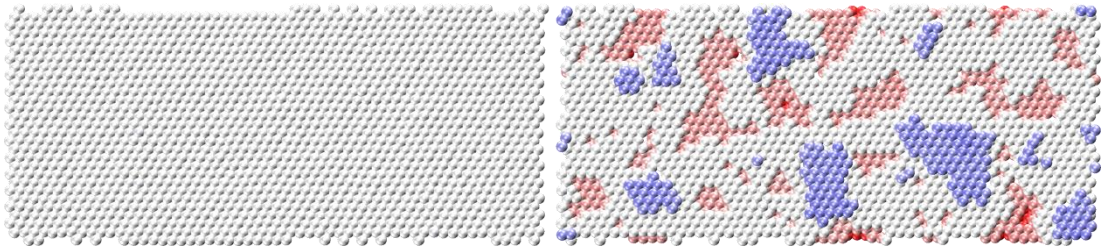


Figure 52: The initial surface and an eroded surface after 1000 impacts at a polar angle of 60°. This figure uses the same colour gradient to denote the height of an atom as Figure 43.

The surface of the 56 by 28 system initially and after 1000 atomic impacts at a polar angle of 60° is shown in Figure 52. Compared to the surface after 1000 depositions shown in Figure 43, the islands are significantly smaller while the vacancies are much larger and more prevalent than the islands. It can also be observed that numerous vacancies also have further vacancies revealing a layer even further below the original surface. In a real world application such as re-entry spacecraft, the exposure of deeper layers would mean that the layer of thermal and chemical protection is becoming thinner and less effective. Continued erosion can lead to a

gap in the protective layer forming and lead to erosion of the internal layers with disastrous consequences.

The effect of the azimuthal angle (see Figure 24) was also analysed by first simulating at each polar angle with each impacting atom being given a random azimuthal angle, which is how preceding MD simulations have handled the azimuthal angle. Another three simulations were then run where every impacting atom used the same azimuthal angle. Three simulations were run for each in an attempt to account for the stochastic variability of the MD simulations.

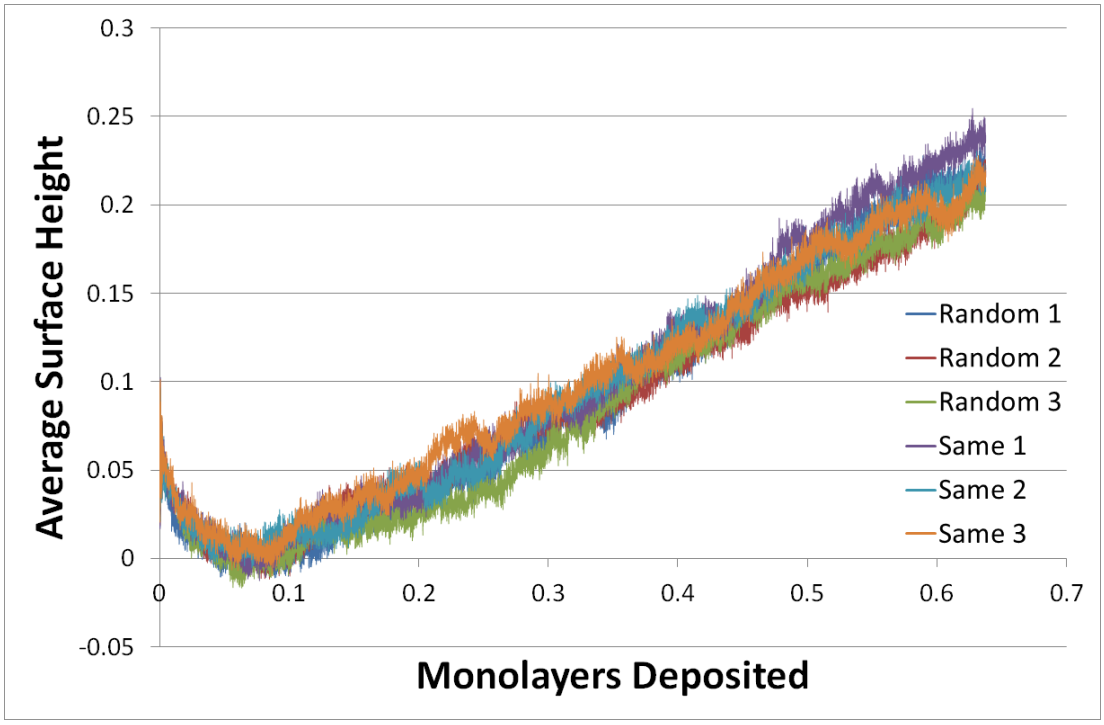


Figure 53: Comparison of average surface height for simulations of a polar angle of 30° with 3 using random azimuthal angles and 3 using the same azimuthal angle

In Figure 53, the comparison of the average surface height at a polar angle of 30° when using random azimuthal angles and the same azimuthal angle is shown. The simulations using the same azimuthal angle (purple, cyan and orange) appear to be growing faster than those using random azimuthal angles (blue, red and green) but the difference is very small and can be accounted for by random fluctuations.

The comparison of the surface roughness at a polar angle of 30° when using random azimuthal angles and the same azimuthal angle is shown in Figure 54. The simulations using random azimuthal angles (blue, red and green) appear to be a

little more roughened than the simulations using the same azimuthal angle (purple, cyan and orange) but, like the average surface height in Figure 53, the difference in surface roughness is very marginal and the effect could be negated by random fluctuations.

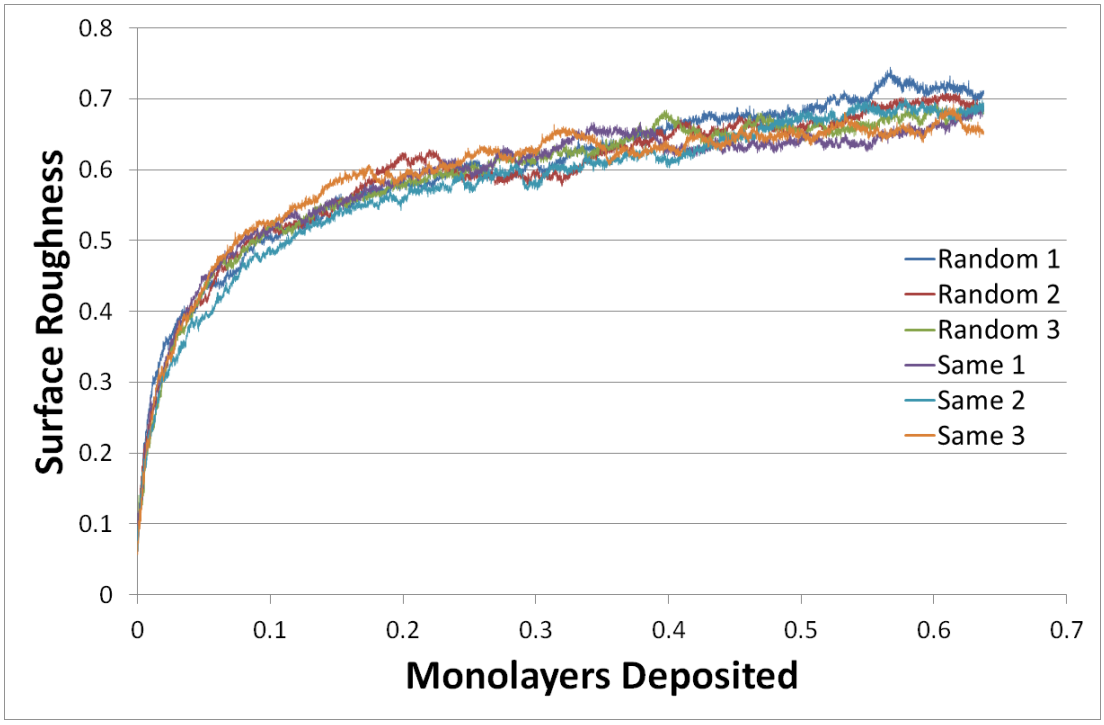


Figure 54: Comparison of surface roughness for simulations of a polar angle of 30° with 3 using random azimuthal angles and 3 using the same azimuthal angle

The comparison of the average surface height at a polar angle of 80° when using random azimuthal angles and the same azimuthal angle is shown in Figure 55. Unlike Figure 53, there is a much clearer difference between the simulations using the same azimuthal angles (purple, cyan and orange) and the simulations using random azimuthal angles (blue, red and green) with those using the same angle eroding less than those using random angles. This is potentially caused by the azimuthal angle used for the simulations with the same azimuthal angle (180°, where 0° represents atoms travelling down along the x-axis and 90° represents atoms travelling down along the y-axis) creating impacts that are more likely to glance off the surface and transfer less kinetic energy. With less kinetic energy, the surface height would evolve at a slower rate.

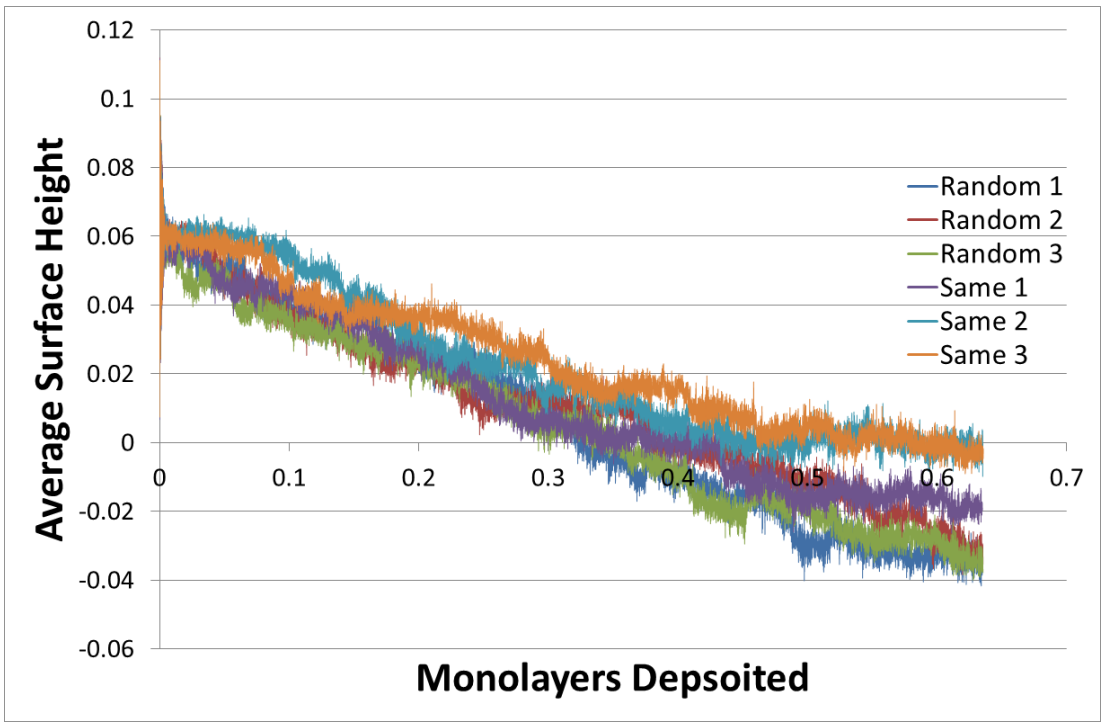


Figure 55: Comparison of average surface height for simulations of a polar angle of 80° with 3 using random azimuthal angles and 3 using the same azimuthal angle

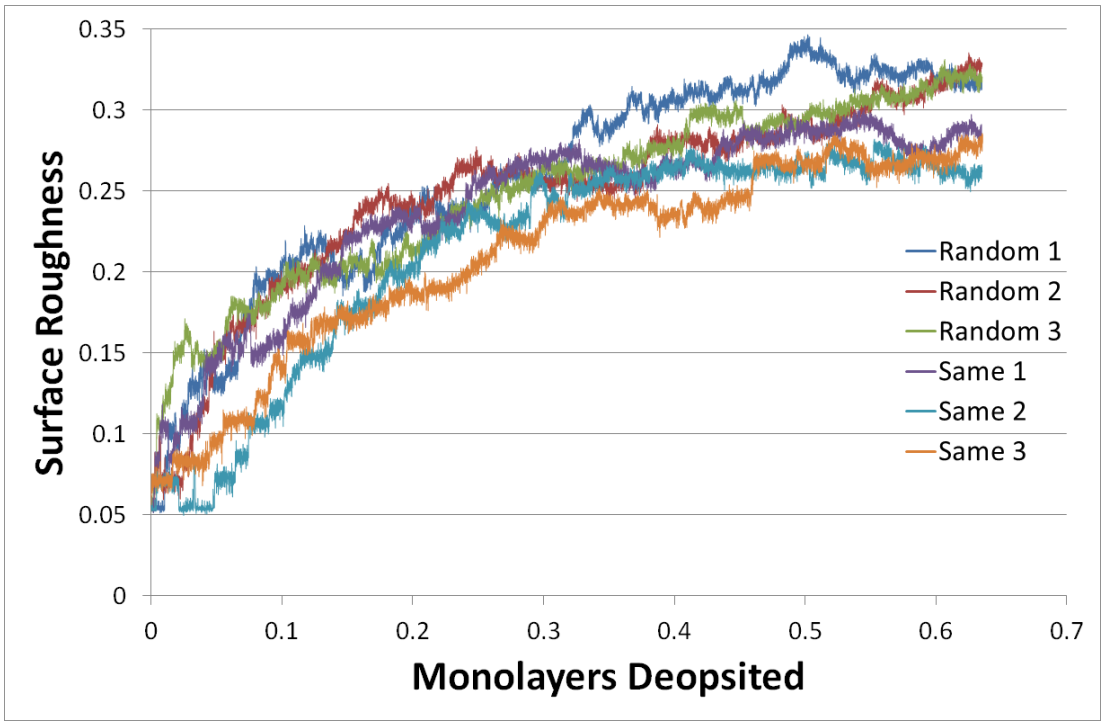


Figure 56: Comparison of surface roughness for simulations of a polar angle of 80° with 3 using random azimuthal angles and 3 using the same azimuthal angle

In Figure 56, the comparison of the surface roughness at a polar angle of 80° when using random azimuthal angles and the same azimuthal angle is shown. As in Figure 55, the simulations using random surface angles (blue, red and green) are more clearly distinguishable from the simulations using the same azimuthal angle (purple, cyan and orange) with those using random angles roughening more than those using the same angle. Again, this is potentially caused by the angle used transferring less kinetic energy to the surface an atom impacts the surface. With less kinetic energy, atoms on the surface are less likely to move, meaning the surface would roughen more slowly.

Graphs for the other polar angles simulated can be found in the Appendix. All of them appear to show very marginal differences in the average surface height and the surface roughness except for the polar angle of 70°, which is more similar to the differences seen at 80° with less erosion and less surface roughness for those using the same azimuthal angle. However, the difference in average surface height is less distinct.

The sputter yields and sticking probabilities were obtained from each of the azimuthal angle simulations and the yields and probabilities of each group of three simulations were averaged.

In Figure 57, the sputter yields and sticking probabilities were plotted against the polar angle for simulations using random azimuthal angles (blue and red squares) and the same azimuthal angle (green and purple diamonds). The squares and diamonds represent the average value of three simulations and the error bars visualise the likely range of values that could be obtained based on the 3 simulation sample, showing ±2 standard deviations.

Looking back at Figure 50, the decreasing surface growth rate, the change from surface growth to surface erosion and the change in the surface erosion as the polar angle increased can be explained by the difference between the sputter yield and the sticking probability. At lower angles, the sputter yield is much lower than the sticking probability. As the polar angle increases to 40°, the sputter yield increases and the sticking probability decreases, causing the growth rate to shrink. At 50°, the sputter yield begins to fall but it is now marginally higher than the sticking probability, causing the surface to erode. As the polar angle increases further, both the sputter

yield and sticking probability falls but the sticking probability begins to level off as it approaches zero.

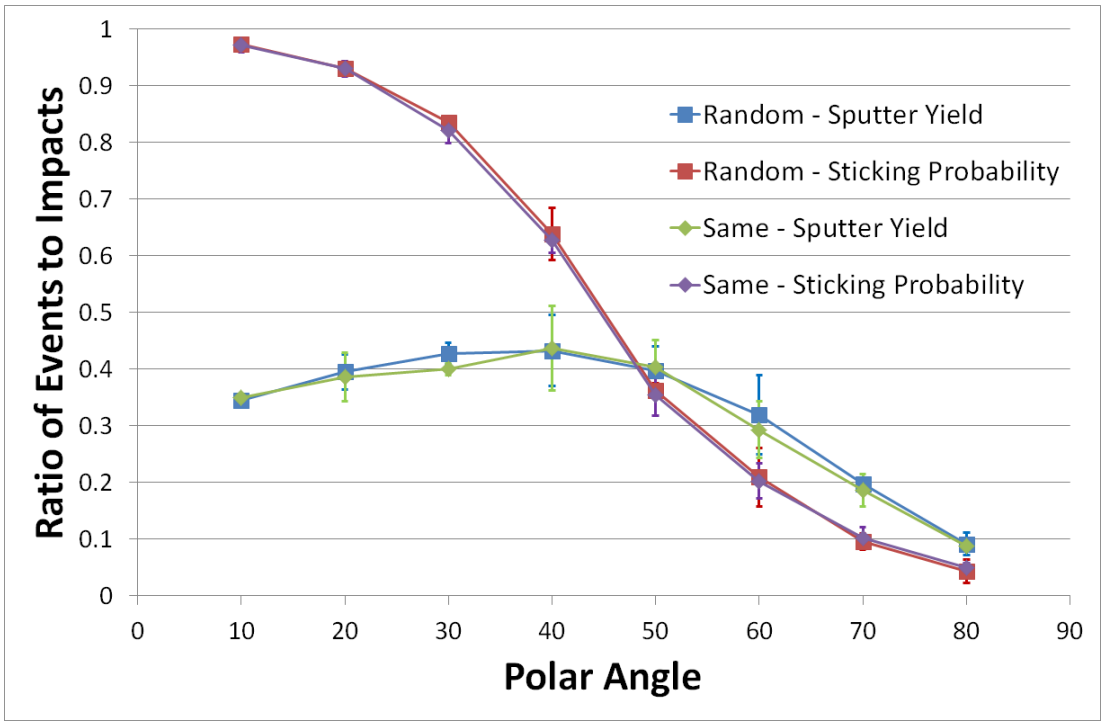


Figure 57: The sputter yields and sticking probabilities at various polar angles using both random azimuthal angles and the same azimuthal angle

Figure 57 can also partially explain the behaviour seen in Figure 51. At low polar angles, there is a lot of sputtering and sticking events, creating lots of opportunities for the surface to be roughened. As the polar angle increases, the number of events decreases and so does the surface roughness. At 60°, the large drop coincides with a large drop in the sputter yield so far and the same is seen at 70° and 80°. However, there are some behaviours that cannot be explained by the impact statistics alone as the difference in surface roughness between simulations using random azimuthal angles and the same azimuthal angle at a polar angle of 80° as the simulations with the same azimuthal angles has marginally more surface events.

3.3 Summary

From our MD simulations, we found that the Lennard-Jones potential was unsuitable for the impact of our Nickel system. With the Sutton-Chen potential, we found that

the surface suffers from finite size effects from the propagation of the kinetic energy after an impact. The finite size effects can be mitigated most effectively by thermostatting the system for 4 ps between impacts. Using the 4 ps thermostated delay, we see that the system decreases its surface height during early impacts before growing linearly at a rate that is proportional to the number of monolayers deposited. The surface roughness, on the other hand, increases rapidly during the early impacts before increasing at a much slower rate during the rest of the simulation.

When analysing the effects of the polar angle, we found that the growth rate decreased with increasing polar angle until 50°, when the surface began to erode instead. At polar angles above 70°, the surface equilibration does not occur. For surface roughness, the roughness decreased with increasing polar angle but the effect became more pronounced at polar angles above 60°. The effect of using random or fixed azimuthal angles only substantially affected the average surface height and the surface roughness when also using polar angles above 70° as below that, the difference in surface height and surface roughness can be accounted for by random fluctuations of the simulations.

Plotting the average sputter yield and sticking probability for the simulations with random azimuthal angles and simulations with the same azimuthal angle, we saw that the difference between random and fixed azimuthal angles was very small. The sputter yield and the sticking probability were also shown to be major contributing factors in the average surface height and surface roughness obtained. However, there were other unknown factors that contributed to the surface roughness.

Experimental data in the literature was sought to compare against the simulations carried out in this work. However, no experimental data was found and comparisons were made with other simulation works where possible. Due to the lack of detail in those works, it was not possible to accurately compare against the results obtained in this work.

While this chapter does not have many images of surfaces from the MD simulations, MD surface images are analysed and used to help develop the kMC code in Chapters 4 and 5.

Chapter 4 – Kinetic Monte Carlo Methodology

While MD produces highly accurate models of surface behaviour, the computational time required is extremely prohibitive. Kinetic Monte Carlo is used in this work in an effort to vastly reduce computational time while seeking to maintain a similar level of accuracy of the MD simulations. In the current chapter, the base kMC code will be detailed in section 4.1. This will be followed in section 4.2 by an initial assessment of an earlier kMC code, made with comparisons to the MD, and in section 4.3 by an assessment of the base kMC code and four variants, made with comparisons to longer timescale simulations of the MD code. From there, section 4.4 will detail improvements that were made to the kMC code, including the addition of a number of algorithms designed to capture features seen in the MD before finally detailing the production kMC codes made in this work in section 4.5.

4.1 Basic Lattice-based kMC set-up

The base code is made of a loop that runs until the number of impacts matches the desired number. It begins by generating two random numbers using intrinsic functions to determine if an impact occurs and to determine if surface relaxation occurs. The probability of an impact was set to mimic the average deposition rate in the MD. When it was determined that an impact will occur, two more random numbers are used to determine the x and y coordinates of the impact site and there are four possible scenarios for the deposition of an atom. These scenarios are the impacting atom doesn't stick but causes sputtering, the impacting atom sticks and causes sputtering, the impacting atom sticks but does not cause sputtering and finally, the impacting atom neither sticks nor causes sputtering. Sputtering without sticking has a probability of the sputter yield multiplied by one minus the sticking probability or $Y_{Sp}\times(1-P_{St})$, both of which are obtained from MD simulations, and during this event, the height of the impact site was reduced as the base code treated the impact site as the site being sputtered. Sticking without sputtering has a probability of $(1-Y_{Sp})\times P_{St}$ and the lattice site has its height increased during this

event. Sticking with sputtering has a probability of $Y_{Sp} \times P_{St}$ but the initial code only incremented the number of impacts during these events and during events with no sticking or sputtering, which has a probability of $(1-Y_{Sp}) \times (1-P_{St})$. The scenarios, their probabilities and the action taken by the base code are detailed in Table 3.

Table 3: Impact Scenarios in base code

Interaction	Probability	Action
Sputtering	$Y_{Sp} \times (1 - P_{St})$	Height of impact site decreases
Sticking	$(1 - Y_{Sp}) \times P_{St}$	Height of impact site increases
Displacement	$Y_{Sp} \times P_{St}$	None
Glancing impact	$(1 - Y_{Sp}) \times (1 - P_{St})$	None

In the cases of sputtering and sticking, the site chosen as the impact site is optimized based on the change to the surface height to try and prevent unrealistic overhanging atoms. This is further explained in section 4.1.1 and represented by Figure 58.

In between events of particle deposition, the surface is allowed to relax. The surface is relaxed by first selecting a site on the surface using random numbers. This site represents an atom trying to diffuse. The area around the chosen site was checked to see how many of the surrounding sites were at the same height as the site chosen. This is used to represent the number of atomic interactions the site has with a higher value representing a more energetically favourable position than a lower value. This is more simplistic than atomic interactions in MD as the distance between sites is not accounted for. A second site was then chosen within a short distance of the atom being moved. The equivalent of the number of interactions was calculated at the new site and then the two were compared. If the new site had the same number or more interactions than the old site, the height at the old site was decreased by one while the height at the new site was increased by one, which represents the atom moving from the old site to the new site. If the old site had more interactions, stochastic methods that accounted for the temperature of the system determined if the move happened. In this way, surface relaxation favours moves towards larger groups of islands or filling vacancies in the lattice but moves away from larger groups are still possible and more likely at higher temperatures.

Initially, the code started with a simple lattice that tracked the height of each site on the lattice. This would appear similar to a (100) surface when the surface is in its initial state but instead of a new atom landing on the surface, the height for the site representing the point of impact (or simply, the impact site) would just increase by one. Prior to the development of the base code, the code was changed to make the lattice function more like a (111) surface like the one used in the MD. To do this, algorithms were added to file output, the interaction counter and the distance calculation to transform the position of the sites into the locations used by a (111) surface. The transforms used are shown in Equations 22, 23 and 24.

$$x_i = X_i + \frac{1}{2}(Y_i + Z_i) \tag{22}$$

$$y_i = \frac{\sqrt{3}}{6}(3Y_i + Z_i) \tag{23}$$

$$z_i = \frac{\sqrt{3}}{2}Z_i \tag{24}$$

Here x_i, y_i and z_i are the Cartesian coordinates of lattice site i, X_i and Y_i are the lattice indices of site i and Z_i is the lattice height of site i. This initially produced an issue as the surface was now angled since changes to the y lattice coordinate and changes to the height also affected the x Cartesian coordinate and both the x and y Cartesian coordinates, respectively. To make the surfaces rectangular again, the effect of the y lattice coordinate on the x Cartesian coordinate was restricted. The effect of the lattice height was not changed at this time because the surfaces being simulated didn't have significant growth and it was unclear how to modify the system to correctly use the new coordinates. In the code, Y_i in Equation 22 was replaced with MOD(Y_i,2). MOD(A,B) is an intrinsic function of the Fortran compiler used to calculate the remainder after the division of A by B. This effectively means that Equation 25 is now used instead of Equation 22.

$$x_i = X_i + \frac{1}{2}\left(\left(Y_i - \frac{2Y_i}{|Y_i|}\left\lfloor\frac{|Y_i|}{2}\right\rfloor\right) + Z_i\right) \tag{25}$$

After each impact or after a significant number of loops, set in the base code to 100,000, the average surface height and surface roughness are calculated. To calculate them, the system iterates over every site and adds the height of the site to the sum of the heights. The code also checks neighbouring sites to identify exposed

atoms beneath the atom represented by the current height of the site. Exposed atoms are represented by red atoms in Figure 58. When an exposed atom is identified, its height is also added to the sum of the heights and the total number of surface atoms is incremented. Once all sites have been iterated over, the surface height is calculated as the average of the heights and the surface roughness is defined as the root-mean-square difference of the heights and the average surface height, similarly to the MD.

4.1.1 Lattice Site Optimisation

At this point, it was noted that the surface contained a lot of unrealistic overhangs that are not observed in the MD simulations. To correct this, an algorithm was created to optimize the site chosen when sputtering or sticking. A graphical representation of what the algorithm does is shown in Figure 58. In this algorithm, the initial height of site chosen was compared with two neighbouring sites.

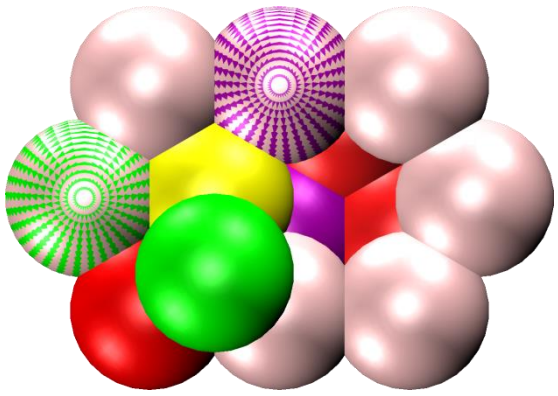


Figure 58: Representation of a surface during an impact in kMC. The yellow atom is the originally selected impact site, the purple atom is the optimal impact site for sticking events, the green atom is the optimal impact site for sputtering events. Atoms with patterns are other sites considered for the same events and red atoms are not present in kMC simulations but represent exposed atoms in the same lattice site as the atom on top of them and to the right.

When choosing a site to sputter, the chosen site, represented by the yellow atom in Figure 58, was compared to the site to its left and the site below it in lattice coordinates, represented in Figure 58 by the atom with the green pattern and the atom in green, respectively. Of these three sites, the site with the highest height was selected as the optimal site for sputtering, which in the case of Figure 58 was the green atom. If a site other than the site with the highest height was chosen,

sputtering the chosen site would cause a gap to appear underneath the highest site in Cartesian space, creating an unrealistic overhang. In the event two or all three sites had the highest height, the site chosen originally was used or random numbers were used to pick between the other two sites if the original site had the smallest height.

When choosing where the representation of an impacting atom would stick to the surface, the algorithm instead compares the chosen site to the site to its right and the site above it in lattice coordinates, represented in Figure 58 by the purple atom and the atom with the purple pattern, respectively. Of those two sites and the chosen site, the site with the smallest height was selected as the optimal site for sticking events, which for Figure 58 was the purple atom. If the site chosen did not have the smallest height, the impacting atom would be hanging unrealistically over a gap in Cartesian space.

4.2 **Initial Assessment of Methodology**

Prior to the development of the base code, the kMC code was used to simulate using the conditions of a simulation previously run using MD at 900K. This earlier version of the kMC code lacked the algorithm to identify exposed atoms, causing gaps to appear in the visual representations of the surfaces produced. The sputter yield and sticking probability data obtained from the MD simulation was used in the kMC simulation. The results obtained from the kMC were compared to the MD results. When comparing the two, the kMC highlighted a peculiar feature in the average surface height in the MD. The behaviour noted didn't appear to be realistic and it was speculated that it was potentially caused by the lattice parameter used for the simulation being too small for the 900K crystal. If it was too small, the crystal would become too constrained and, as the lower layers are fixed and the crystal has periodic boundary conditions in the X and Y directions, the crystal would only be able to expand upwards. We returned to our MD code and performed simulations on a fully periodic crystal to analyse the pressure of the crystal at various temperatures and lattice parameters. A graph of pressure against temperature was then plotted comparing how each lattice parameter was affected. Using that graph, a lattice

parameter 2% larger than the original was selected. The MD simulation was then run again with the corrected value of the lattice parameter for 900K.

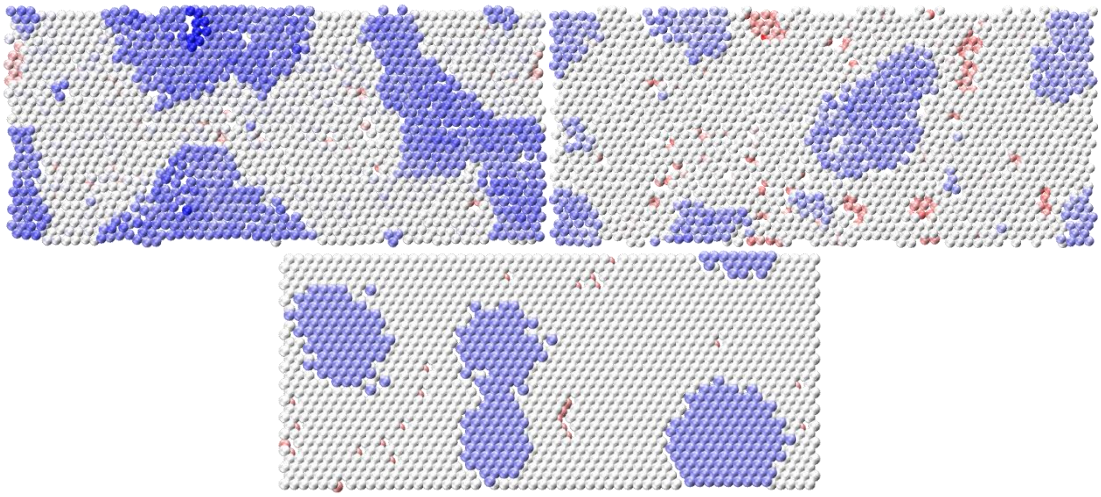


Figure 59: The final surface observed after 1000 impacts at 900K using MD (top left), the same simulation with corrected lattice parameter for 900K (top right) and a kMC simulation using the same conditions (bottom). This figure uses the same colour gradient to denote the height of an atom as Figure 43.

Figure 59 compares the surface seen from the MD simulation, both before and after correcting the value of the lattice parameter, and the kMC simulation. In the MD simulation without the corrected lattice parameter (top left), the surface is dominated by two very large islands that have grown on the surface. There is also a small vacancy and numerous mobile islands and vacancies. The MD simulation with the corrected lattice parameter (top right) has three large islands and a few smaller islands. It also has much more groups of vacancies. The kMC (bottom) doesn't compare well with the original MD as it has much less atoms above the original surface with three large islands and a lot of mobile vacancies. However, it compares much better with the corrected MD simulation having the same number of large islands and a similar number of atoms above the initial surface. There are still differences, most notably the well-rounded nature of the kMC islands and the lack of vacancies compared with the corrected MD.

Both MD and kMC were used to simulate 1000 impacts on a 56 by 28 surface (equivalent to 0.638 monolayers) at 900K and in Figure 60, the average surface height from the simulations were compared. The kMC roughly follows the same trend as the MD but, as mentioned earlier, it was noted that there was a peculiar kink in the MD simulation between 0.05 and 0.1 monolayers deposited, causing the

average surface height to rapidly increase. This led to a new lattice parameter being used for the simulation that was more suitable a temperature of 900K Using the corrected lattice parameter, it can be seen that instead of growing at the beginning, the surface remains roughly constant until approximately 0.1 monolayers are deposited, at which point it grows at a similar rate to the original MD and the kMC simulations. It can be seen that the kMC appears to level off towards the end of the simulation. It is unclear why this occurred but it is expected that this change would have been temporary.

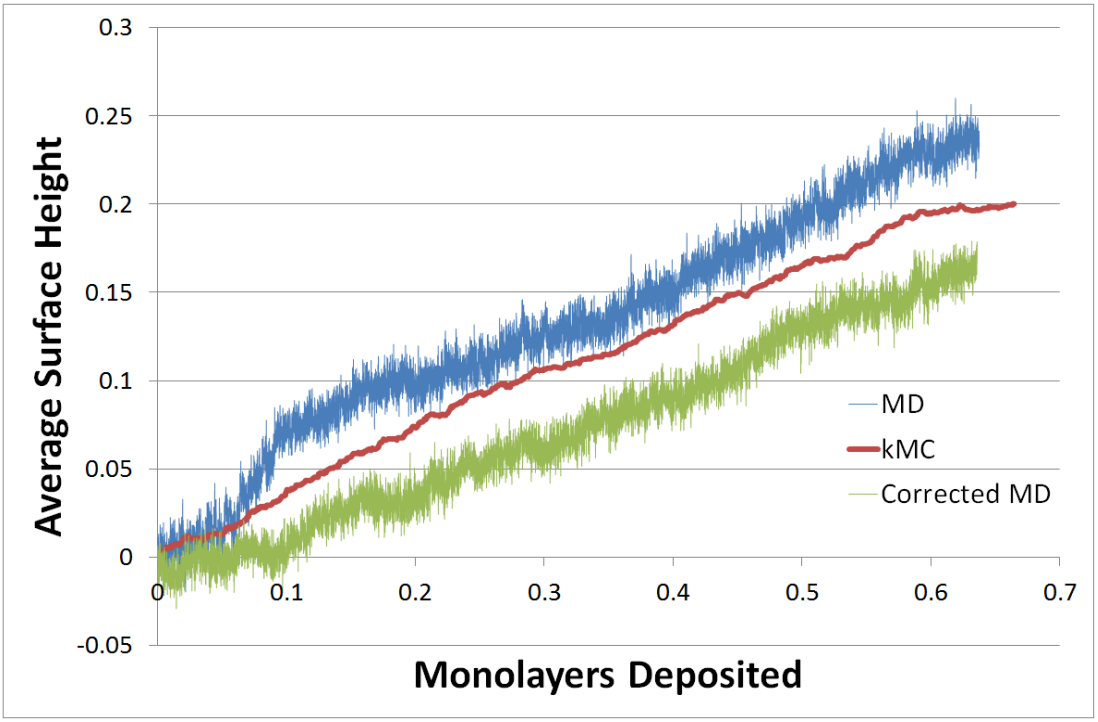


Figure 60: Comparison of MD, before (blue) and after (green) correcting the lattice parameter, and kMC (red) of average surface height against monolayers deposited.

Figure 61 compares the surface roughness obtained from the MD and kMC simulations of 1000 impacts on a 56 by 28 surface at 900K. In the MD simulation, the surface roughness begins to level off at around 0.35 monolayers after the equivalent of half of a monolayer has been deposited. This behaviour is not seen in kMC as the surface continues to roughen, climbing towards 0.45 monolayers of roughness. Unlike the MD, the kMC grew slower than a power law. A potential cause of this may be that there is not enough surface mobility in kMC. It is unclear if the corrected MD simulation will level off as although it is similar to the MD simulation up to 0.3 monolayers deposited, the roughness deviates with the surface

becoming less rough for a small period before increasing again and eventually reaching the same level of roughness as the original MD simulation at around 0.6 monolayers deposited.

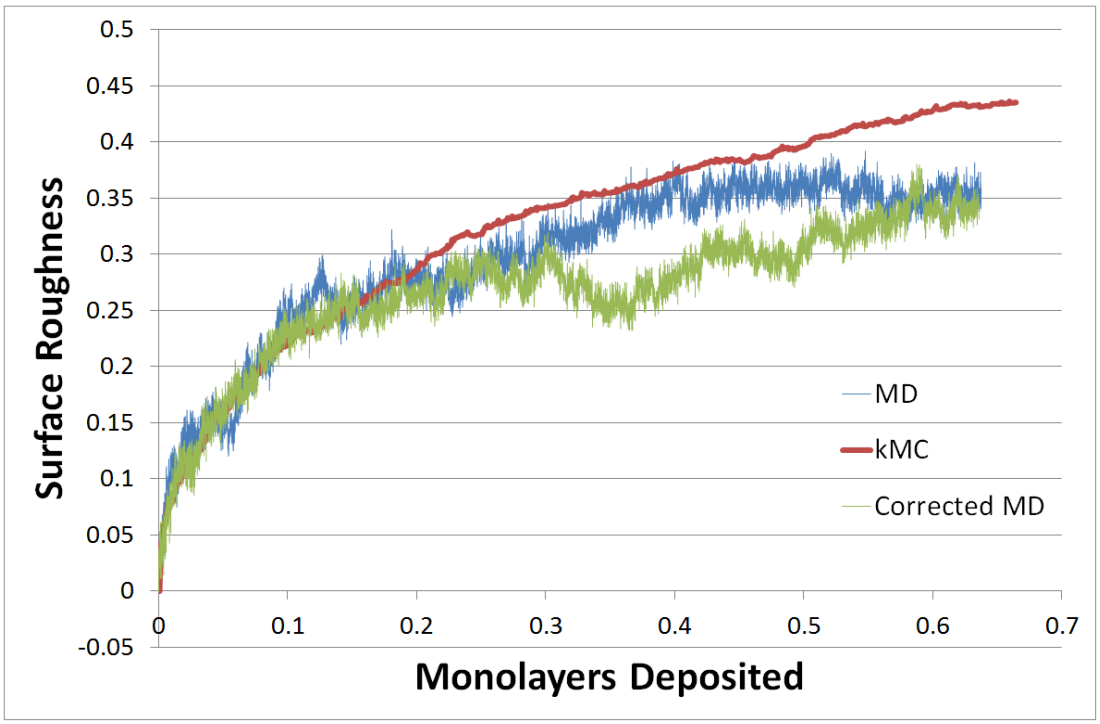


Figure 61: Comparison of MD, before (blue) and after (green) correcting the lattice parameter, and kMC (red) of surface roughness against monolayers deposited

4.3 **Longer Time Scales**

After comparing the MD and kMC at 900K, comparisons between them were made over a longer time period. Due to the computational cost, the smallest MD surface that had been used previously, the 14 by 14 surface, was chosen and simulated the deposition of 8000 atoms, the equivalent of 40.816 monolayers, which was the most impacts that could be simulated with the resources available in a reasonable timeframe. We were also interested in the dynamics of the growth of a layer and captured a number of shots representing the growth of a layer in MD and in three of five kMC codes tested.

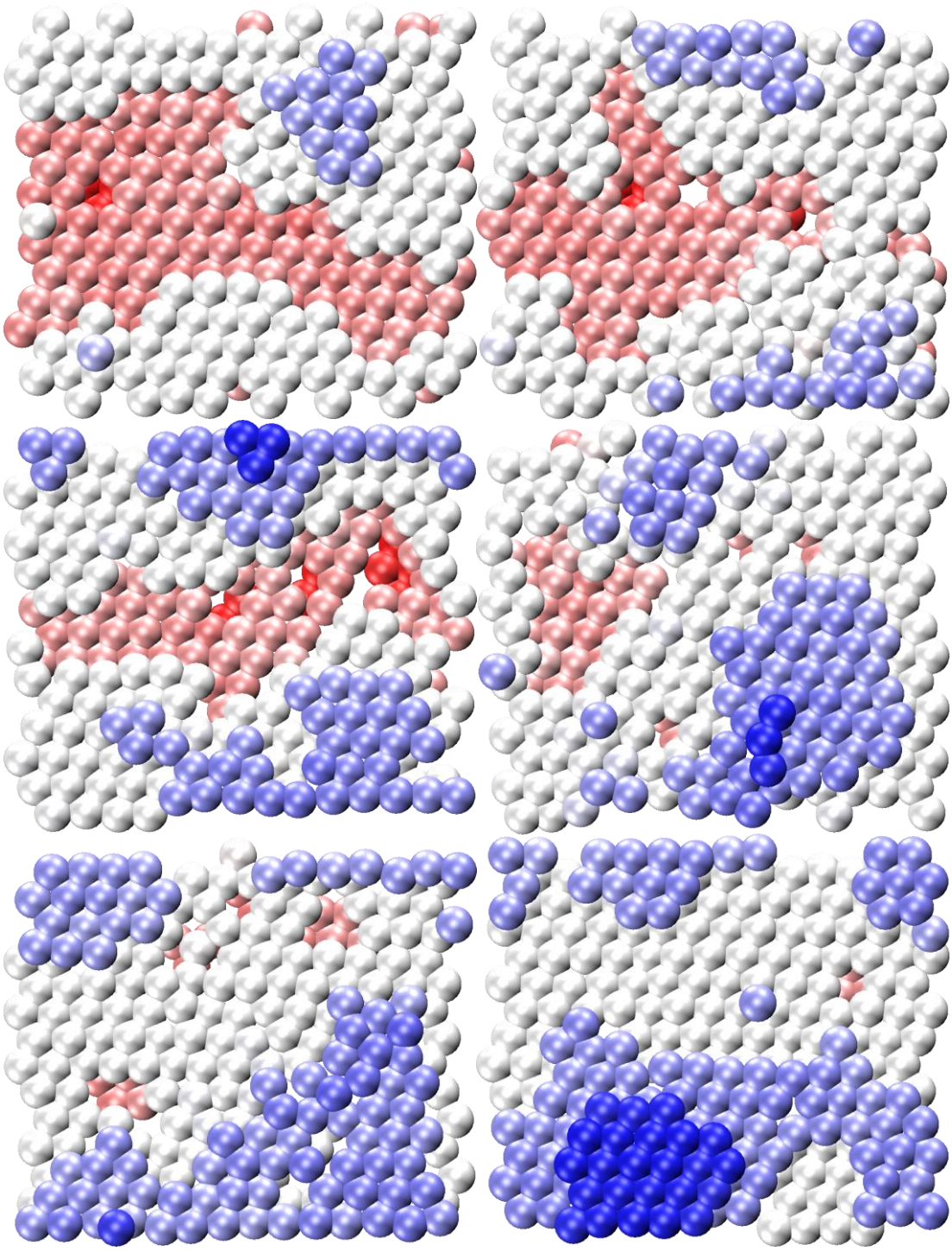


Figure 62: Snapshots of a layer growing on a 14 by 14 surface in MD. A colour gradient denotes the height of atoms relative to the highest layer with that layer in blue. The gradient starts four layers below this layer in red and transitions from red to white and from white to blue at the layer in the middle of the range.

Figure 62 shows the growth of a layer in MD with the growth progressing from the top left to the top right, middle left, middle right, bottom left and finally the bottom right. Due to the highly complex nature of the surface growth, it is difficult to determine accurately where a layer begins to form and when it has been completed so the first and last points were chosen based on the point where one layer and the layer above had almost stopped being a surface layer as it had become almost completely covered. It can be seen that islands and vacancies form very frequently as more deposition occurs.

While setting up longer simulation lengths for the kMC, the base kMC code was developed, creating the algorithm that identified exposed atoms represented in red atoms in Figure 58. The visualisation of surfaces generated by the base kMC code more accurately represents the surface produced in the kMC simulations. This can be seen by looking at the kMC surface in Figure 59, as there are a number of visible gaps along the left and lower edges of islands and along the right and upper edges of vacancies due to only one atom being represented by each site on the lattice. The gaps are where exposed atoms would have been represented if they were identified. These gaps are not present in Figure 63 and in all subsequent surfaces produced by the base kMC code or its variants.

Figure 63 shows the growth of a layer using the base kMC code. The first and last points are chosen similarly to MD but also use the first appearance of a new layer. Islands and vacancies do not form as frequently as the MD but the pattern of surface growth is a good match for the MD. However, in kMC the islands forming are not moving while even the largest islands move significantly in MD.

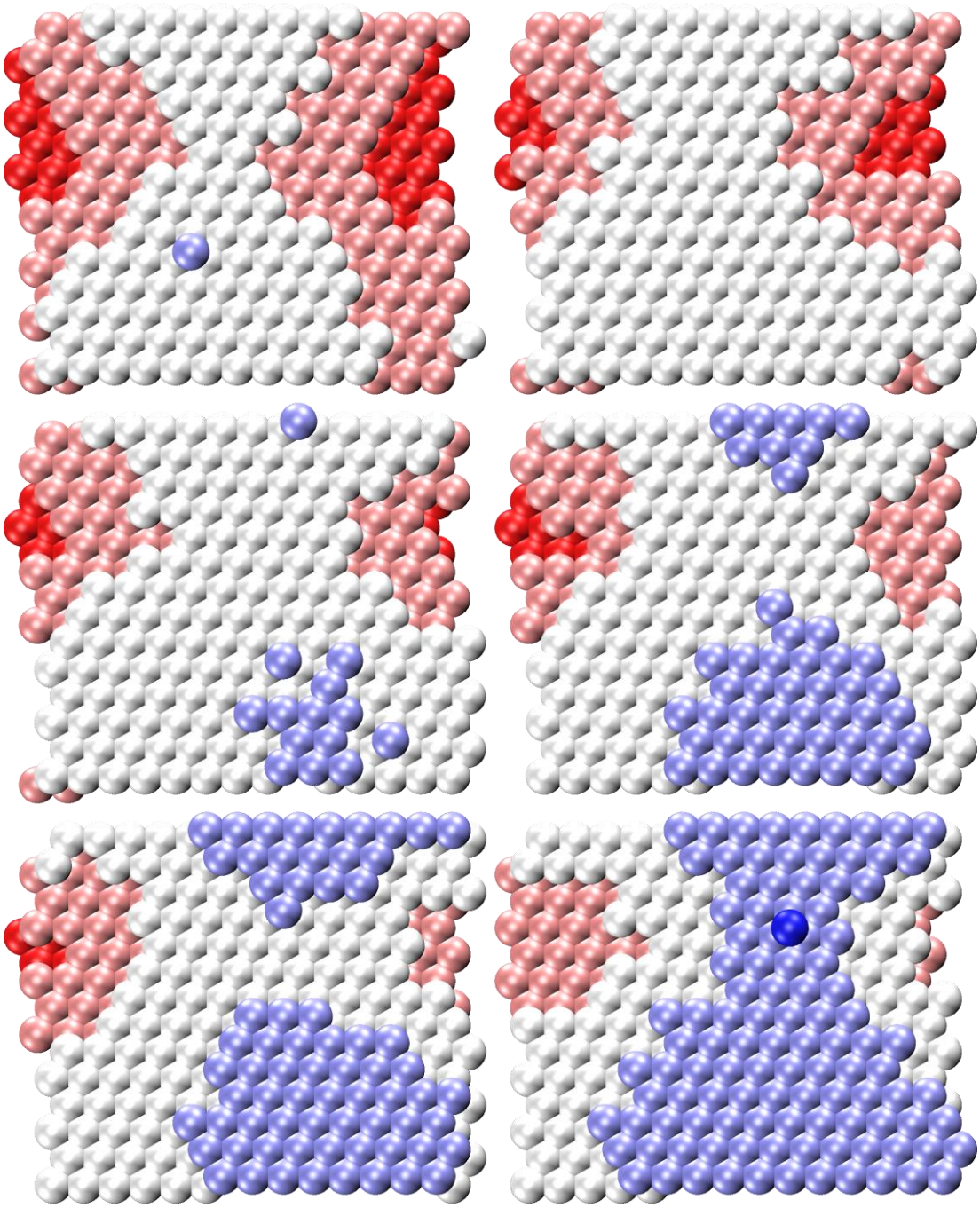


Figure 63: Snapshots of a layer growing on a 14 by 14 surface using kMC. This figure uses the same colour gradient as Figure 62.

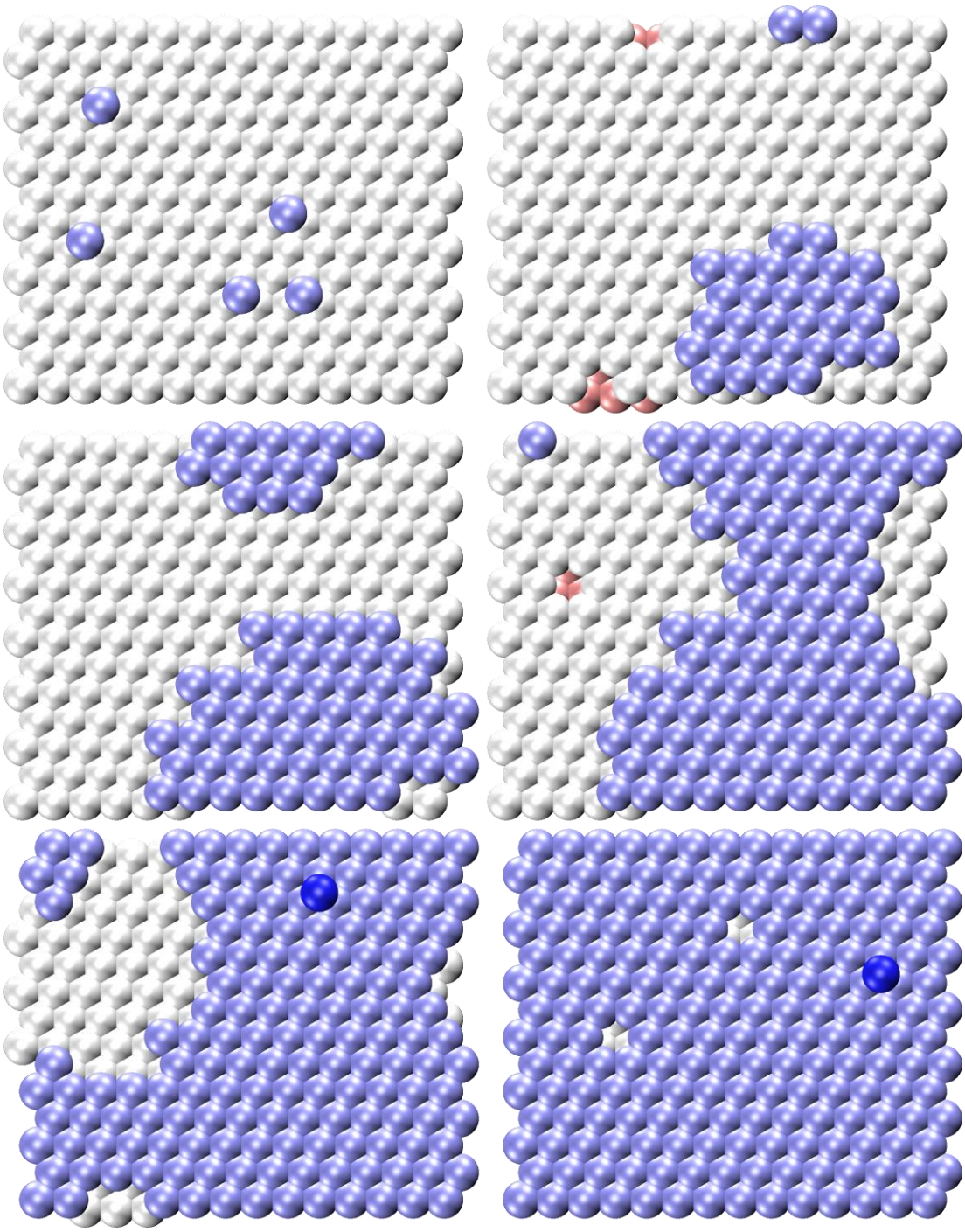


Figure 64: Snapshots of a layer growing on a 14 by 14 surface using kMC. This figure uses the same colour gradient as Figure 62.

In Figure 64 we show another example of layer growth using another version of the kMC code. In this version of the code (Recip), the counting algorithm was adapted to make the strength of interactions dependent on the reciprocal of distance. This change is based on a simplification of the long-range potential energy and force curves for an atom which decay with growing distance in MD. The system has a drastically different growth pattern from the MD and the previous kMC code, growing layer by layer. This made the start and end points very clearly defined as one layer only begins growing when the previous layer has almost fully grown. However, this system is a poor match for the MD and it can be seen that the formation of islands and vacancies is very rare.

A very similar growth pattern was observed for the two more versions of the kMC code tested in these conditions. The first, called Expv1, further altered the counting algorithm to use an exponential decay based on the distance between two sites instead of the reciprocal of the distance as an exponential decay is a more accurate simplification of the long-range potential energy and force curves for the MD. The next, called Expv2, also used the exponential decay but also tried accepting all moves if the height differential was greater than one. The exponential decay was tested as the interactions between atoms in MD tails off very quickly.

Figure 65 shows the growth of a layer using another version of the kMC code, called Expv3, which utilizes the exponential decay but the algorithm was further altered so that there was a representation of interactions with atoms beneath the surface, which is calculated in the MD. The growth of the layer is much more complex than in the MD. The beginning and end points were chosen based on the approximate number of atoms shown in the three highest layers. It can be seen in this surface that vacancies are very difficult to fill while islands are relatively easy to form.

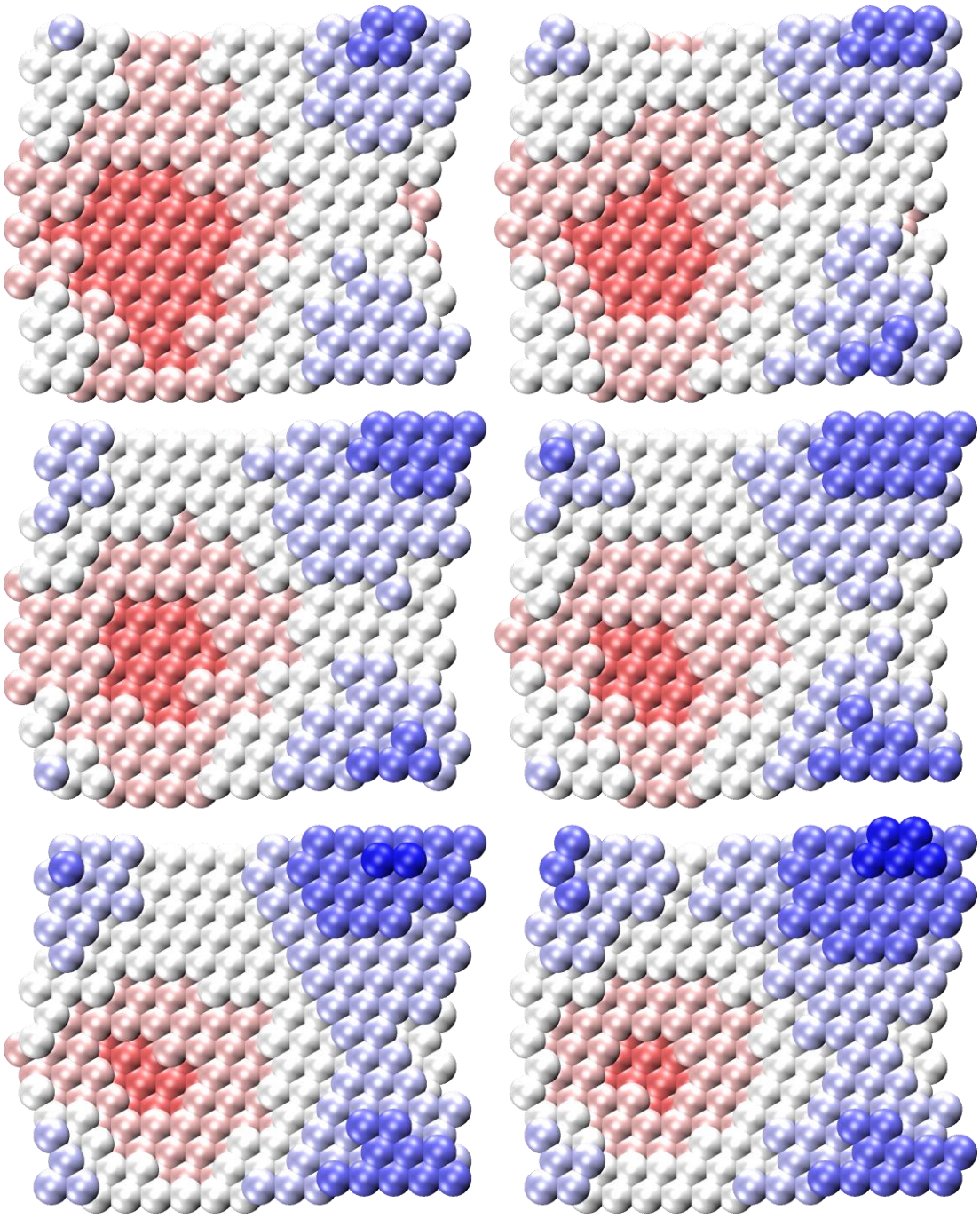


Figure 65: Snapshots of a layer growing on a 14 by 14 surface using kMC. This figure uses a similar colour gradient to Figure 62 with the gradient starting six layers below the highest layer instead of four. The white layer still denotes the middle of the range.

After comparing the methods of surface growth, how much the surface grew and how much it roughened during the course of the simulations was compared.

Figure 66: Average surface height during the deposition of the equivalent of 40 monolayers using MD and multiple variants of kMC code

In Figure 66, the average surface height during the deposition of the equivalent of 40 monolayers is compared using the MD code and the last five variants of the kMC code. The average surface height is obtained in angstroms and is normalised against the distance between monolayers for an ideal system. However, the actual distance in MD between monolayers varies due to temperature increases as more layers are deposited, so different values would be obtained at different points in time. It can be seen that the MD simulation grows by roughly two monolayers per five monolayers deposited while the kMC simulations generally grow slower. However, this may be due to the distance between monolayers growing in MD as the temperature increases. After the equivalent of ~35 monolayers deposited, the growth rate of the MD begins to slow down. It is unclear if this was a temporary slowdown or if it would've persisted if more monolayers were deposited. In the kMC simulations, Expv3 began diverging from the rest of the kMC simulations at ~25 monolayers, growing even slower, suggesting that it doesn't correctly capture the behaviour of the MD.

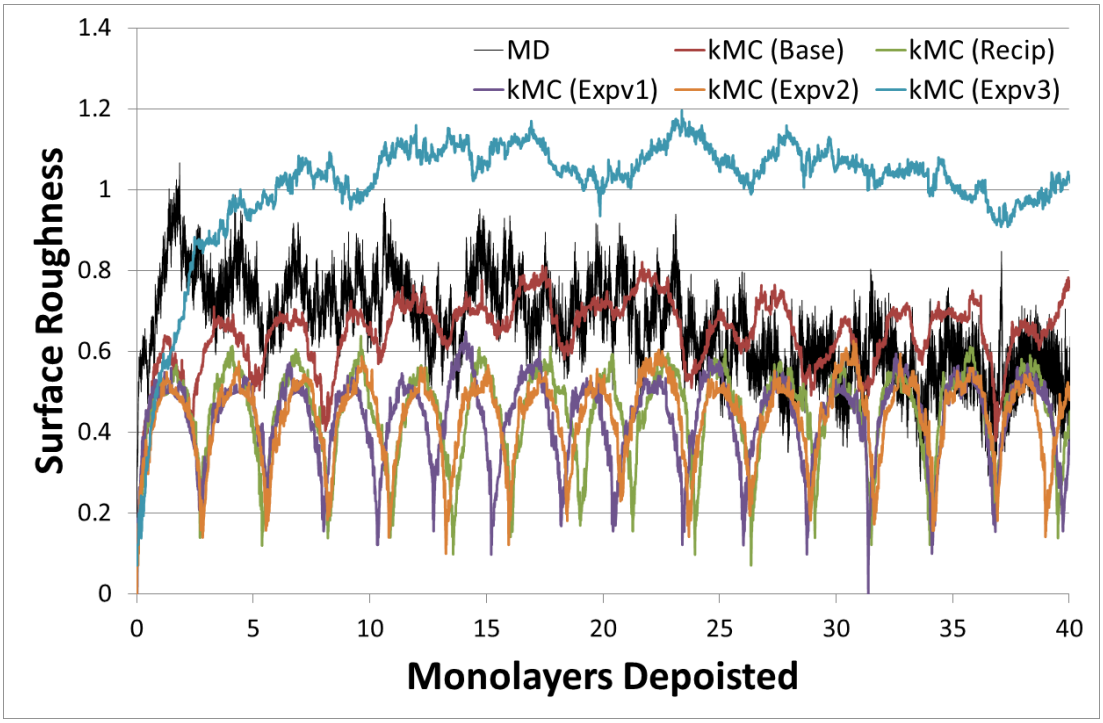


Figure 67: Surface roughness during the deposition of the equivalent of 40 monolayers using MD and multiple variants of kMC code

In Figure 67, the surface roughness during the deposition of the equivalent of 40 monolayers is compared using the MD code and the last five variants of the kMC code. The surface roughness is obtained in angstroms and is normalised against the distance between monolayers for an ideal system. However, the actual distance in MD between monolayers varies due to temperature increases as more layers are deposited, so different values would be obtained at different points in time. In the MD simulation, the surface roughness quickly peaks after the equivalent of about two monolayers is deposited before declining and oscillating. After~28 monolayers, the surface roughness appears to be oscillating in a manner that suggests the surface is growing layer-by-layer, with the next layer of the surface mostly growing after the current layer has finished growing. The change in how the surface grows is likely the cause of the growth rate of the average surface height slowing down in Figure 66. For the kMC simulations, Expv3 roughens much more than the MD and stays roughened throughout the simulation though it takes longer for it to reach a surface roughness equivalent to the maximum surface roughness of the MD. The other kMC simulations that applied weighting when calculating the number of interactions (Recip, Expv1 and Expv2) had layer-by-layer growth throughout the simulation. The base kMC code was the only simulation that appeared to have a

surface roughness that was comparable to the MD but this was only after the deposition of approximately 10 monolayers.

4.4 **Algorithm Developments**

From Figures 66 and 67, it became clear that the kMC variants tested were not able to correctly capture the surface growth dynamics of the MD but, of the codes tested, the base kMC code was the closest. However, it could also be seen that the behaviour of the MD had likely been affected by finite size effects. Due to this, it was decided that the kMC comparisons would return to comparing against the longest simulation possible with this work's run-time constraints on the largest system simulated in this work with MD, which was the 56 by 28 surface that was impacted by 1000 atoms.

Unless otherwise specified, figures showing surfaces in the rest of this chapter use the same colour gradient to denote height as Figure 43, with the original surface layer in white and the gradient ranging from two layers below the original surface in red to two layers above the surface in blue.

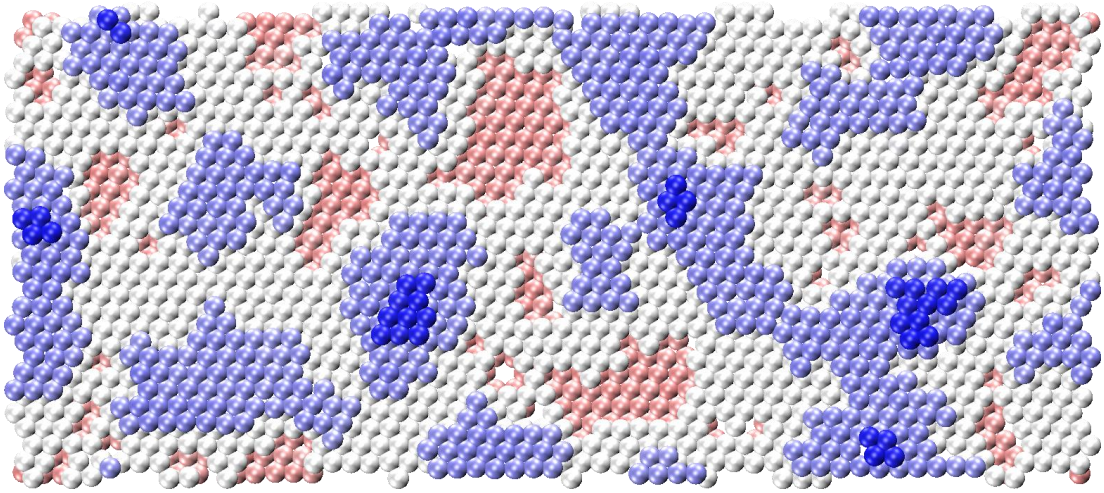


Figure 68: Surface seen after the equivalent of 0.638 monolayers are deposited using MD code

Figure 68 shows an example of the surface obtained after depositing the equivalent of 0.638 monolayers when using the MD code. The simulation conditions for this simulation are the same as the conditions used for the random azimuthal angle simulations in Figure 53. The surface has numerous islands and vacancies in the

surface layer all with irregular shapes. One group of islands has merged into one long island that has stretched across the periodic boundary in the Y axis. A few islands are multi-layered as they have smaller islands on top of them. It also appears that one vacancy is multi-layered but the surface algorithm failed to identify the atoms exposed by the sub-surface vacancy.

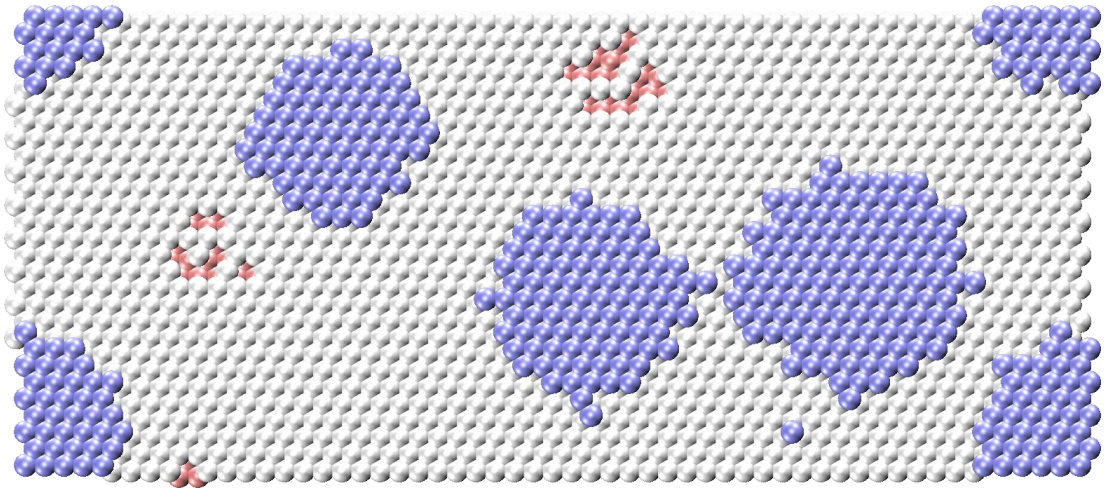


Figure 69: Surface seen after the equivalent of 0.638 monolayers are deposited using base kMC code

Figure 69 shows the surface obtained using the base kMC code with simulation conditions equivalent to the conditions used by the MD simulation that produced the surface in Figure 68. Comparing both figures, it can be seen that the kMC, produces a much less roughened surface, with few vacancies in the surface and significantly fewer islands which are larger but more round. There are also no multi-layered islands or vacancies. To better capture the surfaces seen in MD, a variety of different changes were tried with the kMC code.

4.4.1 **Schwoebel Barrier Effects**

The Schwoebel barrier is an energy barrier encountered in MD simulations and in real-world applications that limits interlayer transport by restricting atoms from diffusing between layers at a step change. This barrier appears because the attractive force pulling atoms across the diffusion barrier is weaker at the step change as there is a vacancy on the other side of the barrier. One of the first changes made to the base kMC code was trying to introduce a representation of the

Schwoebel barrier. The first attempt at this created an extra barrier to atom movement when an atom tried to move down to a lower layer, using an exponential decay based on how many layers it was dropping down. The algorithm that determines whether a relaxation movement is accepted or rejected is summarised in equation 26

$$\text{move}\begin{cases}\text{reject } if\ e^{(h_d-h_r)}-R\geq 0\\ \text{accept } if\ e^{(h_d-h_r)}-R<0\end{cases} \tag{26}$$

where h_r is the height of the site with the relaxing atom, h_d is the height of the destination site that the relaxing atom is moving to and R is a random number in a uniform distribution between 0 and 1. The kMC code that utilized this algorithm during relaxation movements is labelled Swbv1.

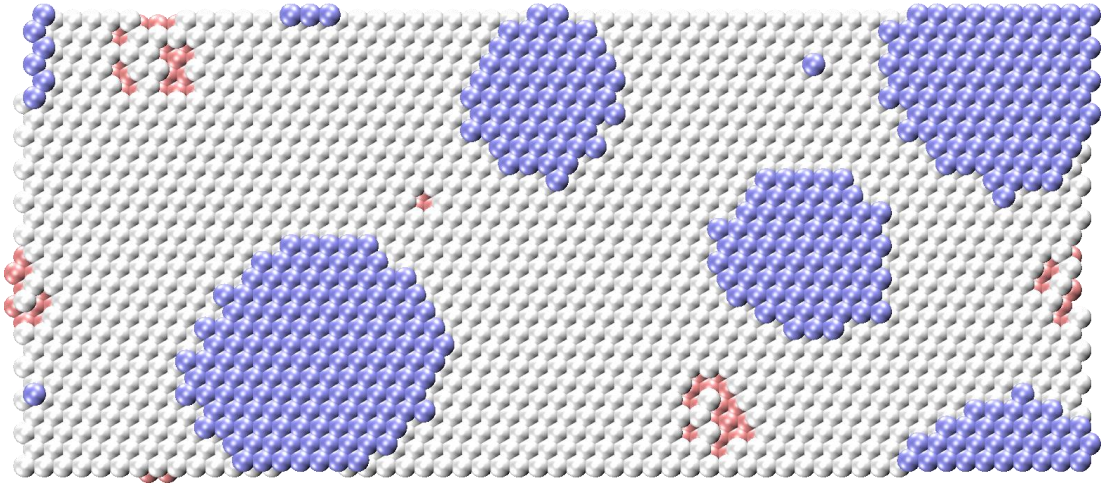


Figure 70: Surface seen after the equivalent of 0.638 monolayers are deposited using the Swbv1 kMC code

The surface produced using the Swbv1 code can be seen in Figure 70. This surface is marginally more representative of the MD surface than Figure 69 as the vacancies are larger but the surface is still significantly smoother than the MD.

An alternate representation of the Schwoebel barrier was then tried that, rather than rejecting all moves on the same probability, took the amount of interactions for the moving atom before and after the move into account. This variant of the barrier algorithm is shown in equation 27.

$$\text{move}\begin{cases}\text{reject } if\ nb_d - nb_r - S_w * (h_d - h_r) \leq 0 \\ \text{accept } if\ nb_d - nb_r - S_w * (h_d - h_r) > 0\end{cases} \quad (27)$$

Here nb_d and nb_r are the number of bonds and S_w is a constant that determines the strength of the barrier. A version of the kMC code that uses this version of the algorithm and set S_w to 2 is labelled Swbv2. This essentially treats the Schwoebel barrier as being equivalent to the energy barrier that must be overcome to move away from two neighbouring atoms.

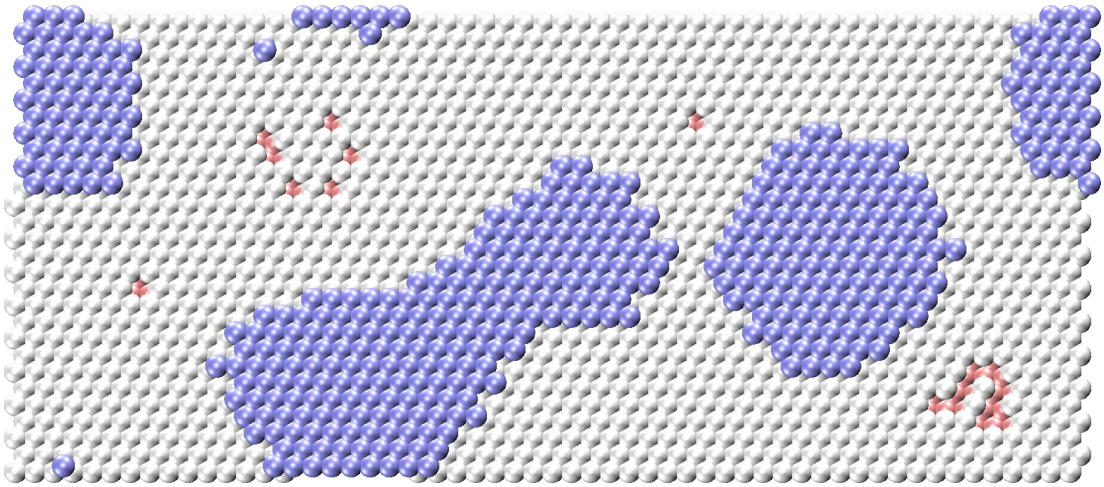


Figure 71: Surface seen after the equivalent of 0.638 monolayers are deposited using the Swbv2 kMC code

Figure 71 shows the surface obtained using the Swbv2 code. Compared to Swbv1 in Figure 70, the vacancies are smaller and less common but the islands are more connected. However, this is likely due to how close the islands formed to one another rather than the change to the code.

Figure 72 shows the surface obtained using the Swbv3 code, which compared to the base kMC code uses the barrier algorithm shown in equation (27 and sets S_w to 5. The vacancies produced are larger than those seen for Swbv2 in Figure 71 but less prominent than those for Swbv1 in Figure 70. The islands are more connected on this surface than the previous two surfaces but there are significantly less islands than the amount seen in MD.

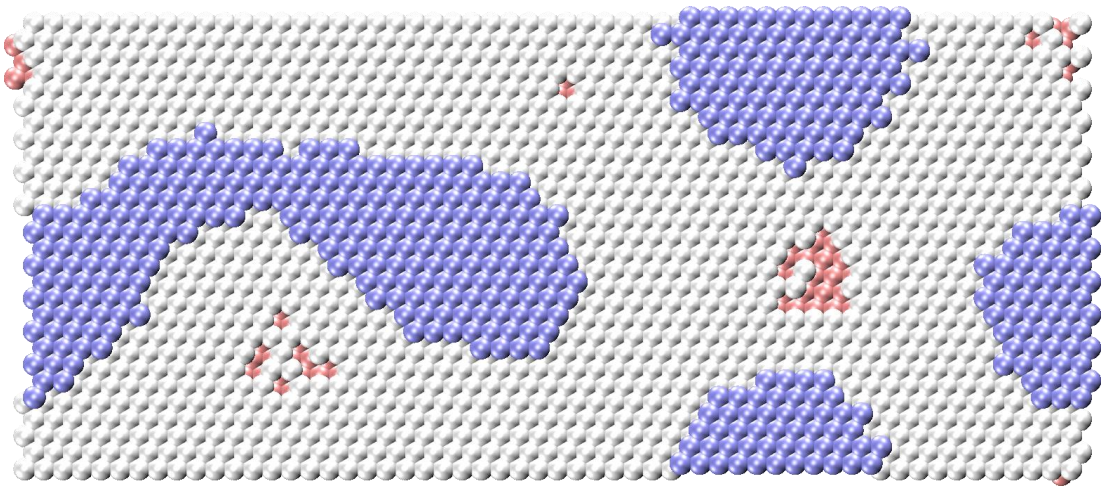


Figure 72: Surface seen after the equivalent of 0.638 monolayers are deposited using the Swbv3 kMC code

4.4.2 **Decoupling Impact Events**

While the representations of the Schwoebel barrier had improved how accurate the kMC was to the MD, the MD was still significantly rougher. To improve the roughness of the kMC, the part of the code that dealt with impacts was reworked. Originally, an atomic impact was ignored if the impact was determined to have both stuck to the surface and caused sputtering as these two events were treated as being cancelled out and making no change to the surface. This method does not match the behaviour seen in MD as any impact can cause the impacting atom to stick and other surface atoms to sputter. To change the current kMC method and decouple sputtering and sticking events, the selection of the impact site was decoupled from the sputtering and sticking algorithms. The impact site was now selected before determining if there would be sputtering or sticking after an impact. Once the impact site was determined, this was used by the sticking algorithm and the sputtering algorithm. In the sputtering algorithm, more random numbers were generated using intrinsic functions to determine which site within a certain range of the impact site, determined by D_1, would be chosen as the site being sputtered. This was done to ensure that sputter and sticking events would rarely cause the surface to have no change as this could only happen when the randomly generated numbers selected the impact site as the site being sputtered.

A version of the kMC code, called Sepv1, was developed using the modified impact event algorithm as well as the barrier algorithm in equation 27 and, like Swbv3, S_w was set to 5. For the modified impact events in Sepv1, D_1 was set to 2, meaning that any atom being sputtered had to be within 2 interparticle spacings of the impact site. Table 4 shows the impact scenarios using the Sepv1 code. The probabilities of the scenarios are not shown as these are identical to the probabilities in the base code, shown in Table 3.

Table 4: Impact Scenarios in Sepv1 kMC code

Interaction	Action
Sputtering	Height of site within 2 interparticle spacings of impact site decreases
Sticking	Height of impact site increases
Displacement	Height of impact site increases and height of site within 2 interparticle spacings of impact site decreases
Glancing impact	None

After some simulations were run using this and future variants of the kMC code, it was noted that there was an error in the modified impact event algorithm that was allowing anisotropic sputtering to occur. This was possible as the algorithm wasn't checking that the distance of the site being sputtered was equal to or less than the maximum distance the site being sputtered could be from the impact site on a single axis. The code was altered to check the distance between the impact site and the site being sputtered and a new site was chosen to be sputtered if the site previously selected was too far from the impact site.

The surface in Figure 73 is produced using the Sepv1 kMC code. Islands and vacancies are much more prominent on this surface than all of the previous kMC surfaces but vacancies are still much less common than seen in the MD. This surface was the first to produce multi-layered island atoms and vacancies.

As the separation of sputter and sticking events made a drastic difference to how the surface grew, Sepv2 was developed to use the modified impact event algorithm and the original Schwoebel barrier algorithm, summarised in equation 26, to analyse how the surface was affected when these two algorithms were combined.

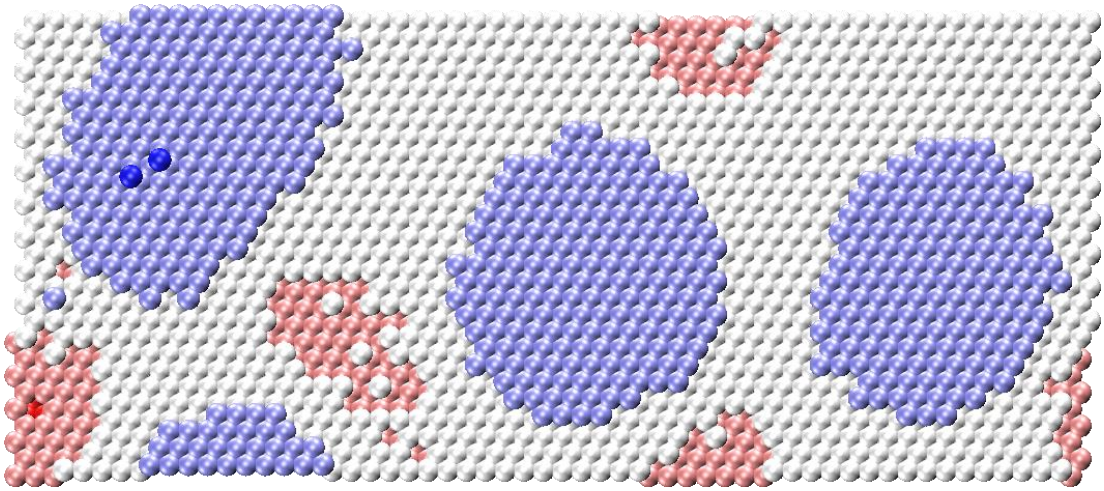


Figure 73: Surface seen after the equivalent of 0.638 monolayers are deposited using the Sepv1 kMC code

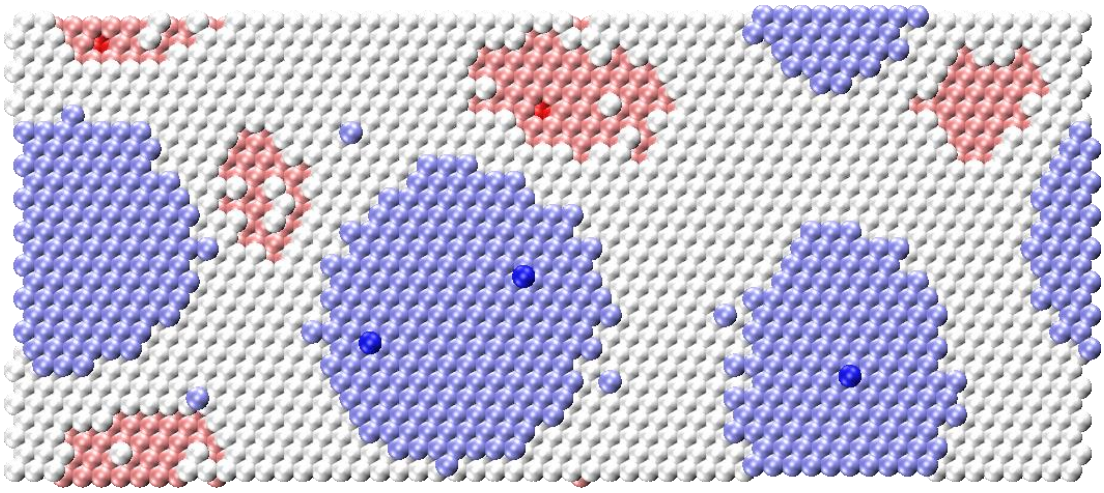


Figure 74: Surface seen after the equivalent of 0.638 monolayers are deposited using the Sepv2 kMC code

The surface obtained from the Sepv2 kMC code in Figure 74 is largely similar to the surface obtained from the Sepv1 kMC code in Figure 73 with multi-layered island atoms and vacancies. This would suggest that for the surface currently being produced, the method of the Schwoebel barrier isn't particularly important. Changing the Schwoebel barrier is later revisited in Chapter 5.

The next change made to the code was first to convert multiple uses of the lattice site optimisation algorithm into a single subroutine. This simplifies the code, making it easier to add the algorithm into other sections of the code and making it possible to simultaneously alter all uses of the algorithm. In the kMC code called OptSR, the subroutine was added to the surface relaxation steps, optimizing the lattice site of

the atom chosen to move and the site chosen to be the atom's destination. The OptSR code also used the original Schwoebel barrier algorithm and the modified impact event algorithm like the Sepv2 code.

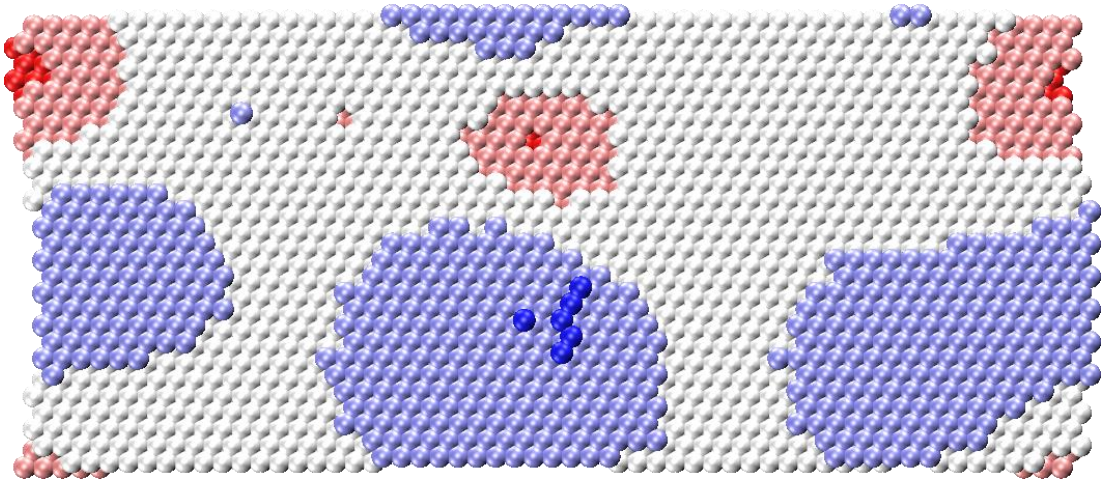


Figure 75: Surface seen after the equivalent of 0.638 monolayers are deposited using the OptSR kMC code

The surface obtained from the OptSR code in Figure 75 produced marginally more multi-layered adatoms but appears to have produced smaller vacancies. Despite this, one of the vacancies had a larger multi-layered vacancy than previously seen.

The next change aimed to keep surface relaxations within close proximity of the impact site. In the MISSR kMC code, the site of the atom being moved was chosen to be the impact site and after each move, the next move used the destination of the previous move. Other than this change, the code used the same parameters and algorithms as the OptSR code as the MISSR code was an evolution of the OptSR code.

The surface obtained using the MISSR code in Figure 76 is very disorganized with numerous island atoms and small island clusters interspersed with small vacancies. While there were a number of multi-layered atoms, several of these were in unrealistic positions and it was unclear how these overhangs were appearing. These unrealistic overhangs and their potential causes are discussed further in Chapter 6. It was deemed that the changes made to produce the MISSR code were detrimental to the accuracy of the kMC so the next code developed was another evolution of the OptSR code.

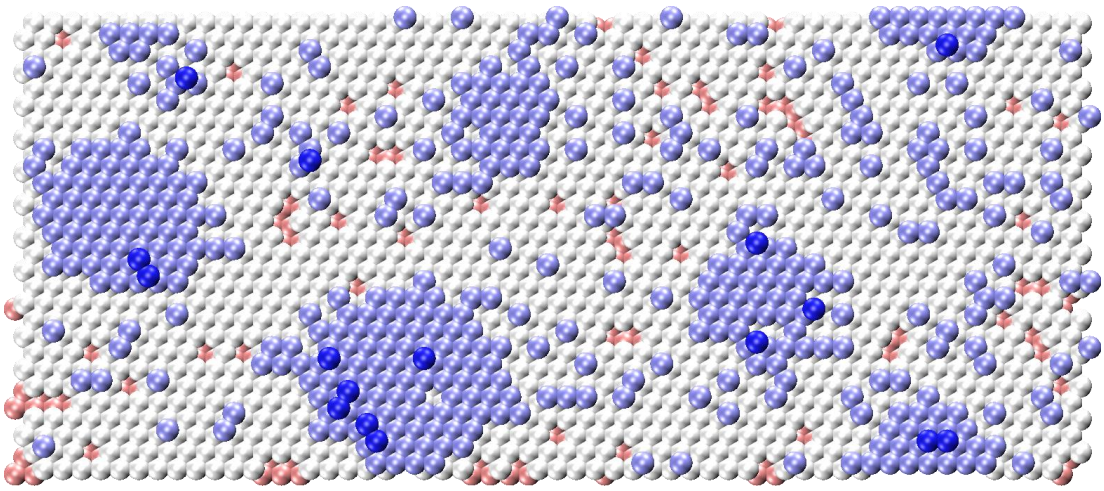


Figure 76: Surface seen after the equivalent of 0.638 monolayers are deposited using the MISSR kMC code

4.4.3 **Enhanced Surface Relaxation**

In an effort to replicate the heightened temperatures and increased activity on the surface immediately after an impact seen in the MD simulations, the impact event algorithm was further modified so that following an impact, the surface attempted to relax itself by moving atoms numerous times. A variety of methods were chosen for this, initially making the extra movements by looping through the surface relaxation algorithm but this was later separated into a new algorithm that only occurs after impact. In this algorithm, the site of the atom being moved and the destination site are chosen as sites within a distance of the impact site, determined by D_2 and D_3. The movements between these sites are also less constrained than surface relaxations, with a movement always occurring. To avoid some unrealistic surface movements, the site of the atom being moved and the destination site were swapped if the original destination was higher than the original site of the atom being moved.

The PIRv1 code was developed to utilise the new impact event algorithm with post-impact relaxations. The number of post-impact relaxations per impact, determined by N_{PI}, was set to 10 in this code. As mentioned previously, this code was an evolution of the OptSR code, meaning it also used the original Schwoebel barrier algorithm and used the lattice optimisation algorithm during surface relaxations. The impact scenarios when using the PIRv1 code is detailed in Table 5.

Table 5: Impact Scenarios in PIRv1 kMC code

Interaction	Action
Sputtering	Height of site within 2 interparticle spacings of impact site decreases and 10 post-impact relaxations occur
Sticking	Height of impact site increases and 10 post-impact relaxations occur
Displacement	Height of impact site increases, height of site within 2 interparticle spacings of impact site decreases and 10 post-impact relaxations occur
Glancing impact	10 post-impact relaxations occur

As shown in Table 5, the PIRv1 code was the first kMC code developed in this work that includes events near the impact site after a glancing impact where an atom impacts the surface and bounces away without sputtering the surface. As some kinetic energy would transfer from the impacting atom to the surface in the MD, the inclusion of these events in the kMC code is expected to better account for the consequences of the energy transfer in the MD.

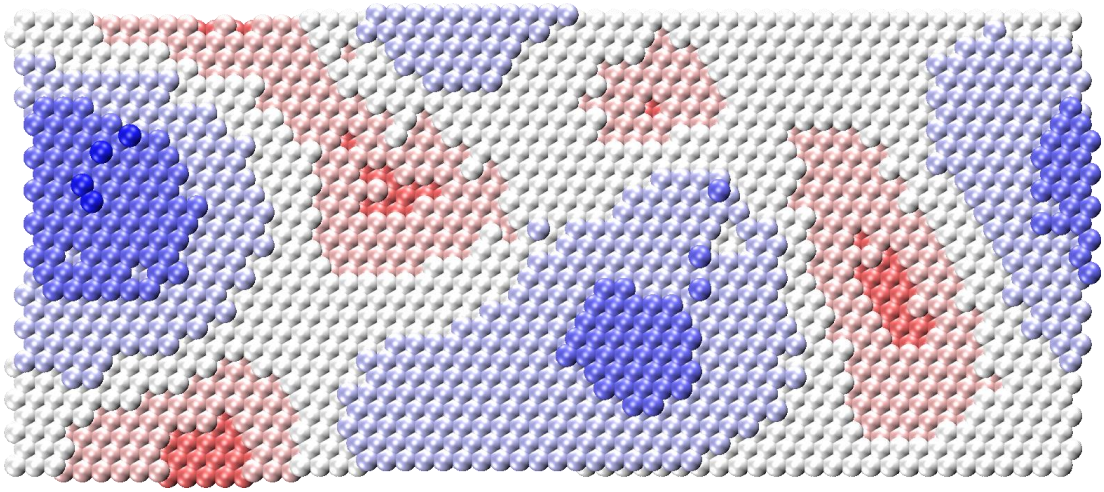


Figure 77: Surface seen after the equivalent of 0.638 monolayers are deposited using the PIRv1 kMC code. This figure uses a similar colour gradient to Figure 43 but ranges from three layers below the original surface in red to three layers above the original surface in blue.

The surface produced using the PIRv1 code is shown in Figure 77. It can be seen that the surface has produced two large islands and a second island layer has begun to grow, though it was noted another unrealistic overhang was present. The surface also has adatoms beginning a third island layer. Two large vacancies and a

smaller vacancy have also formed on the surface and the number of multi-layered vacancies is higher than what was seen in other kMC simulations.

It was noted that the simulation appeared to have some anisotropy with the vacancies seeming to favour vertical erosion. Analysing the code revealed that multiple algorithms were failing to account for a shift in the X-axis coordinate that was applied to every second row on the Y-axis. This was causing the site optimisation algorithm to select the wrong neighbouring atom and causing errors when calculating the distance between two atoms.

It was also realised that the lattice optimisation algorithm was creating more anisotropy because it was being used after checking the distance the site chosen initially was from the relevant site. As the algorithm has the potential to increase or decrease this distance, the algorithm would need to be used before the distance calculation. The PIRv1 code was modified to remove these sources of anisotropy, becoming the PIRv2 code.

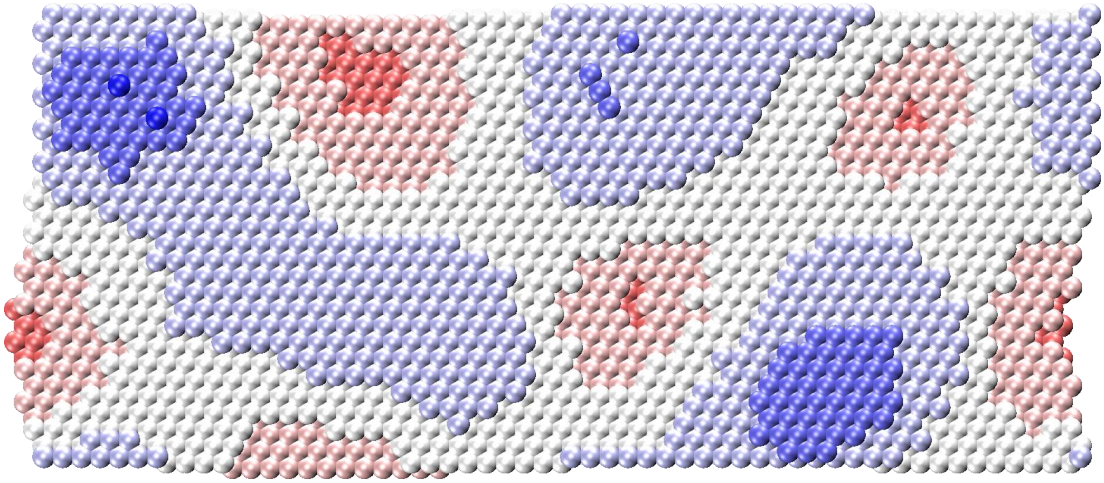


Figure 78: Surface seen after the equivalent of 0.638 monolayers are deposited using the PIRv2 kMC code. This figure uses the same colour gradient to denote height as Figure 77.

Figure 78 shows the surface produced by the PIRv2 code. The surface still has adatoms beginning a third island layer seen in the surface produced by the PIRv1 code but the size of the vacancies and the number of multi-layered vacancies has decreased.

Another method to make the surface relaxations between impacts stay close to the last impact site was then tried. In this method, the site of the atom being moved was selected to be with a given distance of the impact site, determined by D_4. The

destination site was chosen with the usual method, meaning the site was within a given distance of the site of the atom being moved, determined by D_5. This local surface relaxation algorithm was used to develop the LSRv1 code using the PIRv2 code as a base.

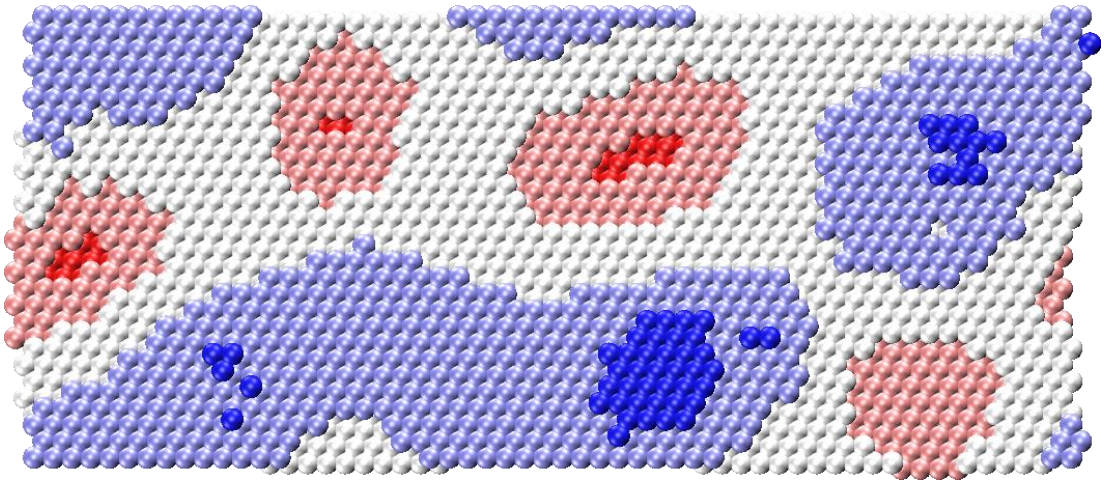


Figure 79: Surface seen after the equivalent of 0.638 monolayers are deposited using the LSRv1 kMC code

Using the LSRv1 code, the surface shown in Figure 79 was obtained. The surface has one large island that spans across the periodic boundary on the X axis. Multi-layered islands are smaller than those in Figure 78, lacking the adatoms beginning a third island layer, but are larger than most other kMC simulations. Vacancies are also smaller with less multi-layered vacancies.

4.5 **Final Algorithm**

It was then decided the effect of altering parameters should be looked at in an attempt to find the optimal parameters to use. To assist this, a new setup subroutine was created to read an input file at runtime that shared the same codename as the output files. With the new setup subroutine, three production versions of the code, Prdv1, Prdv2 and Prdv3, were created. Algorithm and code development was stopped to focus on running simulations with these three versions of the code.

Compared to the base code, all three versions of the production code make use of the modified impact event algorithm with post-impact relaxations, detailed in Table 6. All three also include the lattice optimisation algorithm during surface relaxations

and the algorithm fixes to account for the shift in the X-axis coordinate dependent on the Y-axis coordinate to avoid anisotropic behaviour. Still compared to the base code, Prdv1 makes use of the original Schwoebel barrier algorithm, meaning surface relaxations follow equation 26. Meanwhile, Prdv2 and Prdv3 use the altered Schwoebel barrier algorithm presented in equation 27. Compared to the base code, Prdv3 also uses the local surface relaxation algorithm.

Table 6: Impact Scenarios in Prdv1, rdv2 and Prdv3 kMC codes

Interaction	Action
Sputtering	Height of site within D_1 interparticle spacings of impact site decreases and N_{Pl} post-impact relaxations occur
Sticking	Height of impact site increases and N_{Pl} post-impact relaxations occur
Displacement	Height of impact site increases, height of site within D_1 interparticle spacings of impact site decreases and N_{Pl} post-impact relaxations occur
Glancing impact	N_{Pl} post-impact relaxations occur

The parameters of the simulations run are detailed in the following chapter.

Chapter 5 – Kinetic Monte Carlo Results

Three versions of the production code, Prdv1, Prdv2 and Prdv3, were created and were deemed suitable to try a number of parameter variations. Each code was used to run 10-18 simulations. In this section, each simulation is denoted by a letter followed by vX where X is the same number as the version of the code used.

Table 7: Overview of parameters used in simulations Av1 to Jv1

Simulation	N_{PI}	D_1	D_2	D_3	D_5
Av1	2	4	4	4	2
Bv1	2	4	1	1	2
Cv1	2	1	4	4	2
Dv1	2	4	4	4	1
Ev1	2	4	4	4	4
Fv1	5	4	4	4	2
Gv1	5	4	1	1	2
Hv1	5	1	4	4	2
Iv1	5	4	4	4	1
Jv1	5	4	4	4	4

Table 7 provides an overview of the simulations run on the Prdv1 kMC code. As mentioned in Chapter 4, the N_{PI} parameter determines the number of post-impact relaxations that occur. D_1 is the maximum distance the atom selected to be sputtered can be from the impact site. D_2 is the maximum distance that the site of the atom being moved during a post-impact relaxation can be from the impact site and D_3 is the maximum distance that the destination site of that atom can be from the impact site. D_5 is the maximum distance that the destination site of an atom, which is randomly selected as the atom that will move during a surface relaxation, can be from the site of the atom. All of the distance-based parameters are normalized by the interparticle spacing. A visual representation of these distances can be seen in Figure 80.

The simulations are split into two groups of five simualtions: Av1-Ev1 and Fv1-Jv1. In each group, the first simulation, shaded in green, is treated as a baseline. From

this baseline, the second and third simulations, shaded in orange, varies distances related to actions taken immediately after an impact, with the second reducing D_2 and D_3 to 1 and the third reducing D_1 to 1. The fourth and fifth simulations, shaded in blue, varies distance used during the surface relaxation steps between impacts with the fourth reducing D_5 to 1 and the fifth increasing D_5 to 4.

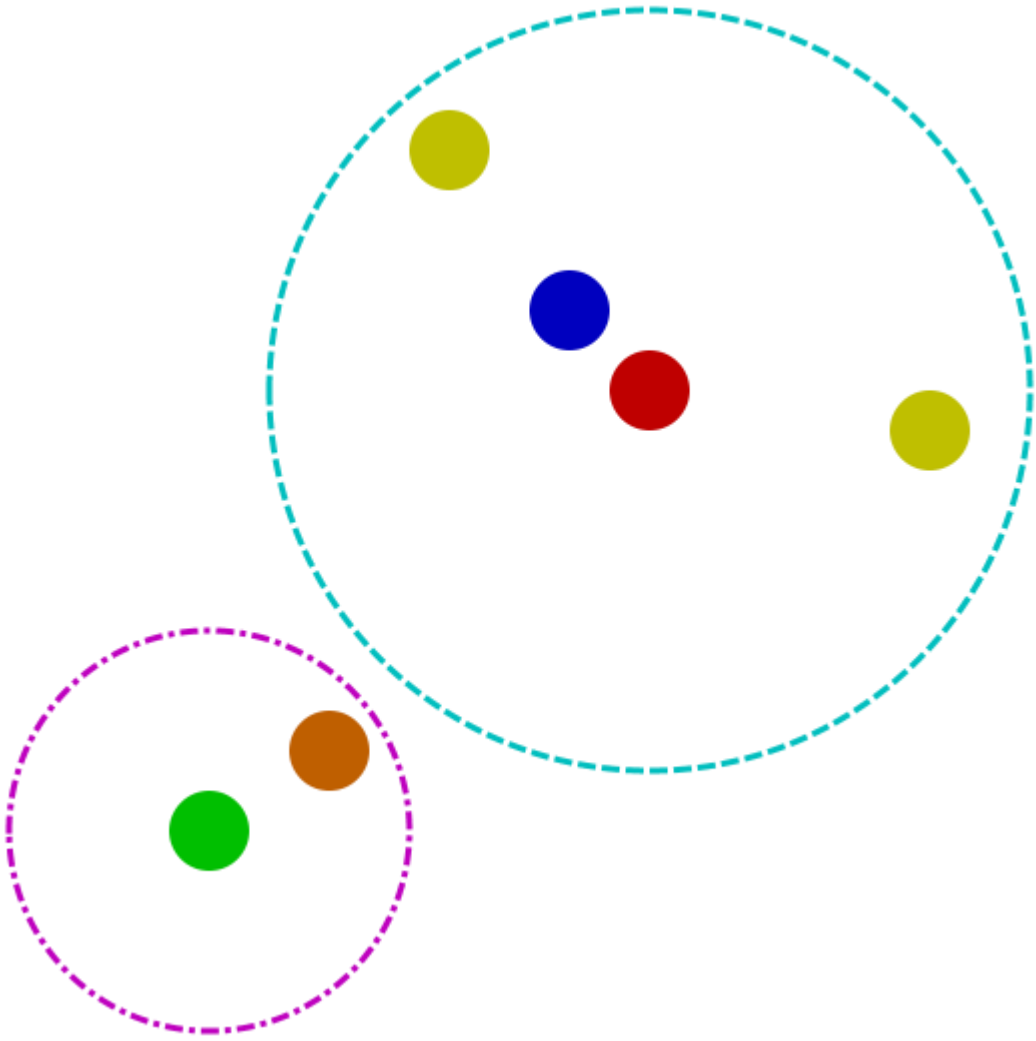


Figure 80: Schematic diagram of the parameters D_1 – D_3 and D_5

Figure 80 is a visual representation of the distances D_1-D_3 and D_5 used in both the Prdv1 and Prdv2 codes. In this figure, the red circle represents the impact site and the cyan dotted circle represents the limits of D_1, D_2 and D_3, which are all based on the distance from the impact site. The blue circle and the two yellow circles represent the chosen atoms limited by these distances, respectively. The blue circle,

limited by D_1 is the atom chosen to be sputtered. The yellow circles, limited by D_2 and D_3, are the atoms chosen for post-impact relaxations. The green circle is a randomly selected atom that is chosen to move during surface relaxations. The position of this atom is limited by D_4 in Prdv3 but in Prdv1 and Prdv2, D_4 is effectvely set to infinity as the atom's position is unrestricted. The magenta dotted circle represents the limit of D_5. The orange circle is the atom, limited by D_5, that is chosen as the destination of the relaxing atom.

Throughout this chapter, unless otherwise stated, all figures showing surfaces use the same colour gradient to denote height as Figure 43. Remember that the original surface layer is in white and the gradient ranges from two layers below the original surface in red to two layers above the surface in blue.

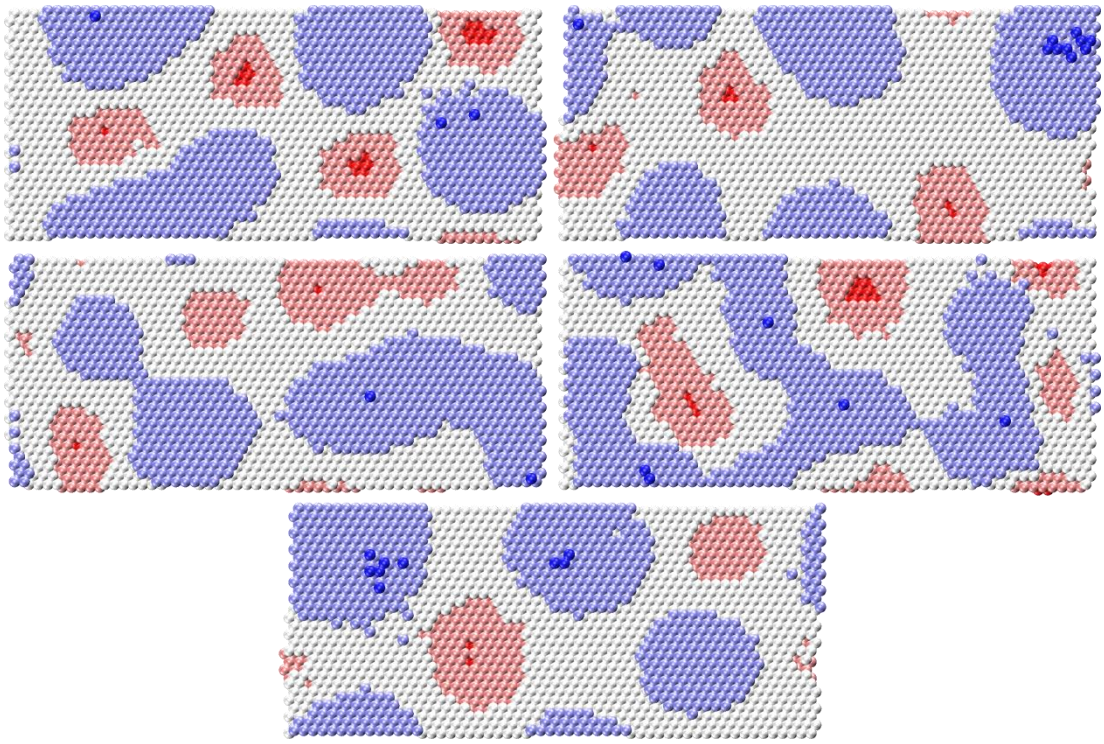


Figure 81: Simulations Av1 (top left), Bv1 (top right), Cv1 (middle left), Dv1 (middle right) and Ev1 (bottom)

In Figure 81, snapshots from simulations Av1-Ev1, using the Prdv1 kMC code, are shown. All five simulations have 2 post-impact relaxations after each impact. Simulation Av1 produced 3 large islands and 4 small vacancies. Despite the vacancies being small, there were significantly more multilayered vacancies than multilayered islands. In Bv1, the proportion of multilayered islands to multilayered vacancies was reversed. However, the islands are smaller compared to Av1.

Simulation Cv1, meanwhile, produced larger islands and vacancies but had almost no multilayered islands or vacancies.

The effect of varying D_5, which is the maximum distance that the relaxing atom's destination can be from the randomly selected relaxing atom, was now consided. Simulation Dv1 produced one large island that spans the periodic boundaries. There were also a few larger vacancies. Like Av1, it favoured the production of multilayered vacancies. Ev1 produced more but smaller islands and fewer but larger vacancies. There were not many multilayered islands and almost no multilayered vacancies.

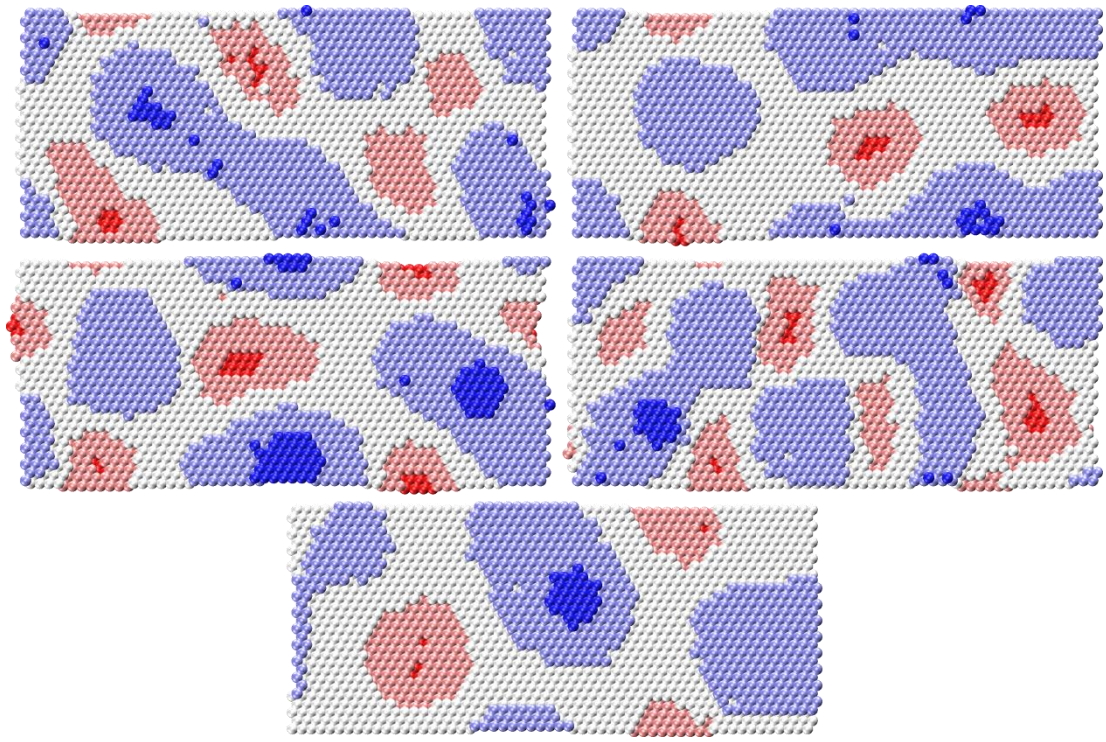


Figure 82: Simulations Fv1 (top left), Gv1 (top right), Hv1 (middle left), Iv1 (middle right) and Jv1 (bottom)

In Figure 82, snapshots from simulations Fv1-Jv1 are shown. These employ the same conditions as Av1-Ev1 except each impact has 5 post-impact relaxations instead of 2. Compared to Av1, Fv1 has larger islands and vacancies. It also has more multilayered islands. Gv1 produced a very large island by merging three smaller ones. Compared to Bv1, there is also more multilayered vacancies. Hv1 produced very large multilayered islands and also larger vacancies with more multilayered vacancies.

The effect of varying D_5 was considered again. In simulation Iv1, the islands produced were unable to fully merge with each other like in Dv1. However, Iv1 produced significantly more multilayered island atoms and more multilayered vacancies as well. Jv1 produced larger and more rounded vacancies than Ev1. It also produced a relatively large multilayered island on one of the two large islands.

Comparing Fv1-Jv1 with Av1-Ev1, it is clear that increasing the number of post-impact relaxtions from 2 to 5 made it easier for vacancies and multilayered islands to be formed and generally increased the surface roughness. It is unclear how it affected island merging as while some simulations had increased island merging, others had decreased island merging.

Table 8: Overview of parameters used in simulations Av2 to Ov2

Simulation	S_w	N_{PI}	D_1	D_2	D_3	D_5
Av2	5	2	4	4	4	2
Bv2	5	2	4	1	1	2
Cv2	5	2	1	4	4	2
Dv2	5	2	4	4	4	1
Ev2	5	2	4	4	4	4
Fv2	5	5	4	4	4	2
Gv2	5	5	4	1	1	2
Hv2	5	5	1	4	4	2
Iv2	5	5	4	4	4	1
Jv2	5	5	4	4	4	4
Kv2	10	5	4	4	4	2
Lv2	10	5	4	1	1	2
Mv2	10	5	1	4	4	2
Nv2	10	5	4	4	4	1
Ov2	10	5	4	4	4	4

Table 8 provides an overview of the simulations run on the Prdv2 code. The most significant difference between the Prdv1 and Prdv2 codes is the Schwoebel barrier is changed from an exponential decay to being treated as equal in energy to a given number of interactions, which is determined by the S_w parameter. See Equation 27 for details. Apart from this, Prdv1 and Prdv2 are the same so the N_{PI} parameter is still the number of post-impact relaxations per impact. D_1 is still the maximum distance the sputtered atom can be from the impact site. D_2 is the maximum distance that the atom moving during a post-impact relaxation can be from the

impact site and D_3 is the maximum distance that the destination site of that atom can be from the impact site. D_5 is still the maximum distance that the destination site of the randomly selected atom moved during a surface relaxation can be from the site of the atom. As was the case for the Prdv1 code, a visual representation of these distances can be seen in Figure 80.

Due to the addition of the S_w parameter, the simulations for the Prdv2 code are split into three groups of five simualtions: Av2-Ev2, Fv2-Jv2 and Kv2-Ov2. In each group, the first simulation, shaded in green, is treated as a baseline. From this baseline, the second and third simulations, shaded in orange, varies distances related to actions taken immediately after an impact, with the second reducing D_2 and D_3 to 1 and the third reducing D_1 to 1. The fourth and fifth simulations, shaded in blue, varies distance used during the relaxation steps between impacts with the fourth reducing D_5 to 1 and the fifth increasing D_5 to 4.

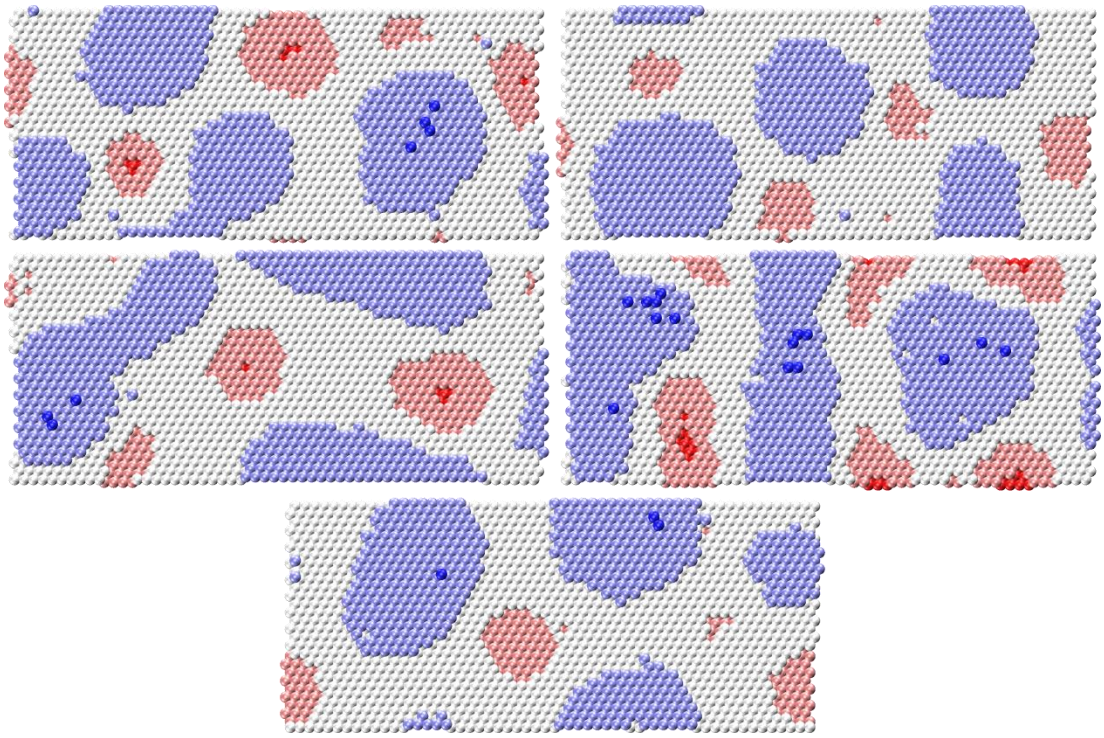


Figure 83: Simulations Av2 (top left), Bv2 (top right), Cv2 (middle left), Dv2 (middle right) and Ev2 (bottom)

Snapshots from simulations Av2-Ev2, using the Prdv2 kMC code, are shown in Figure 83. All five take the Schwoebel barrier to be equal to 5 neighbours and have 2 post-impact relaxations after each impact. Av2 produced three large islands and four vacancies. Av2 has very few multilayered island atoms and multilayered

vacancies. Bv2 produced no multilayered islands or multilayered vacancies. It also produced small islands, making it signifcantly less roughened than Av2. While Cv2 produced multilayered islands and multilayered vacancies, they were less frequent than Av2. Cv2 islands were larger.

The effect of varying D_5, which, as stated previously, is the maximum distance that the relaxing atom's destination can be from the randomly selected relaxing atom, was considered once again. Simulation Dv2 ehhibited peculiar behaviour with two islands growing mostly in the Y direction, stretching across the periodic boundary. Two of the vacancies also favoured growth in the Y direction but weren't large enough to stretch across the periodic boundary. Ev2 produced larger islands than Av2 and smaller vacancies but produced no multilayered vacancies and marginally less multilayered island atoms. The islands produced by Ev2 were much more rounded than the islands in Av2.

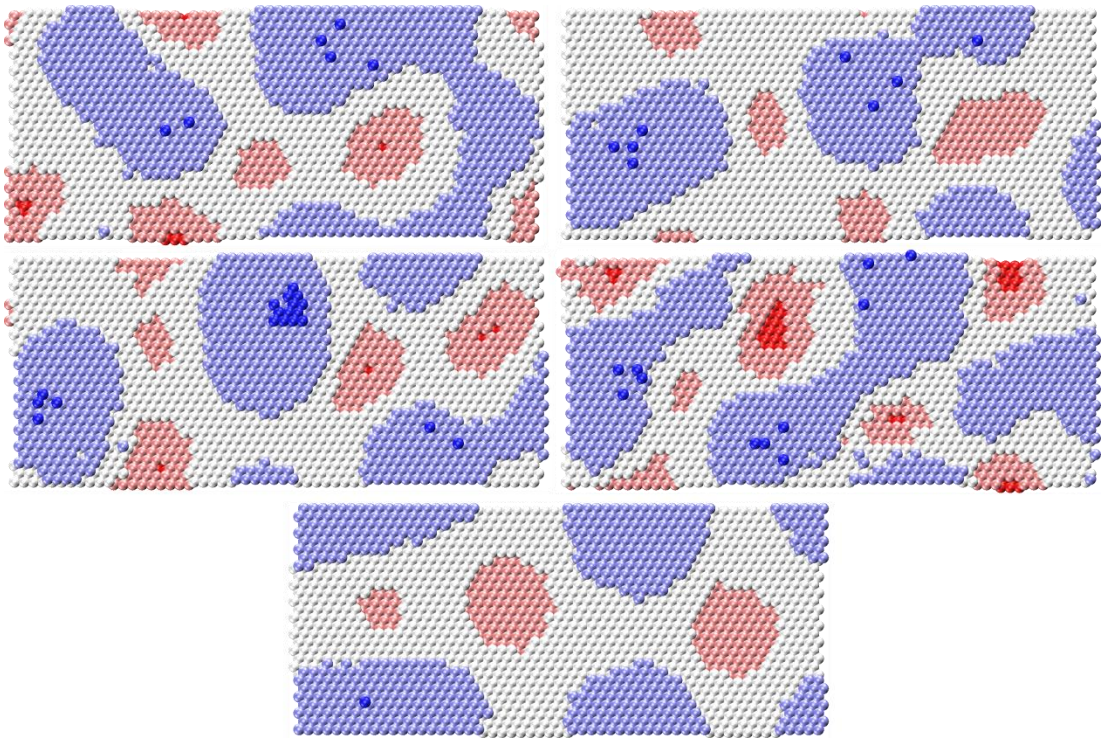


Figure 84: Simulations Fv2 (top left), Gv2 (top right), Hv2 (middle left), Iv2 (middle right) and Jv2 (bottom)

In Figure 84, snapshots from simulations Fv2-Jv2 are shown. These employ the same conditions as Av2-Ev2 except each impact has 5 post-impact relaxations instead of 2. Simulation Fv2 produced large islands that appear to have been formed by merging multiple smaller islands. Compared with Av2, it also produced

smaller vacancies. Simulation Gv2 has much more multilayered islands atoms than Bv2, which failed to produce any multilayered islands or vacancies. Gv2 also has larger islands and vacancies, greatly increasing the surface roughness. Simulation Hv2 produced larger vacancies than Cv2 but smaller islands. Despite the islands being smaller, it also produced more multilayer island atoms and formed a multilayered island of 14 atoms.

Simulation Iv2 was substanially different from Dv2 and did not replicate the peculiar anisotropic growth in the Y direction. It produced larger multilayered vacancies but less multilayered island atoms. Iv2 also produced a large island formed by merging all of the other islands. Compared to Ev2, Jv2 produced larger islands and larger vacancies but only has one multilayered island atom, but this is only marginally less than Ev2.

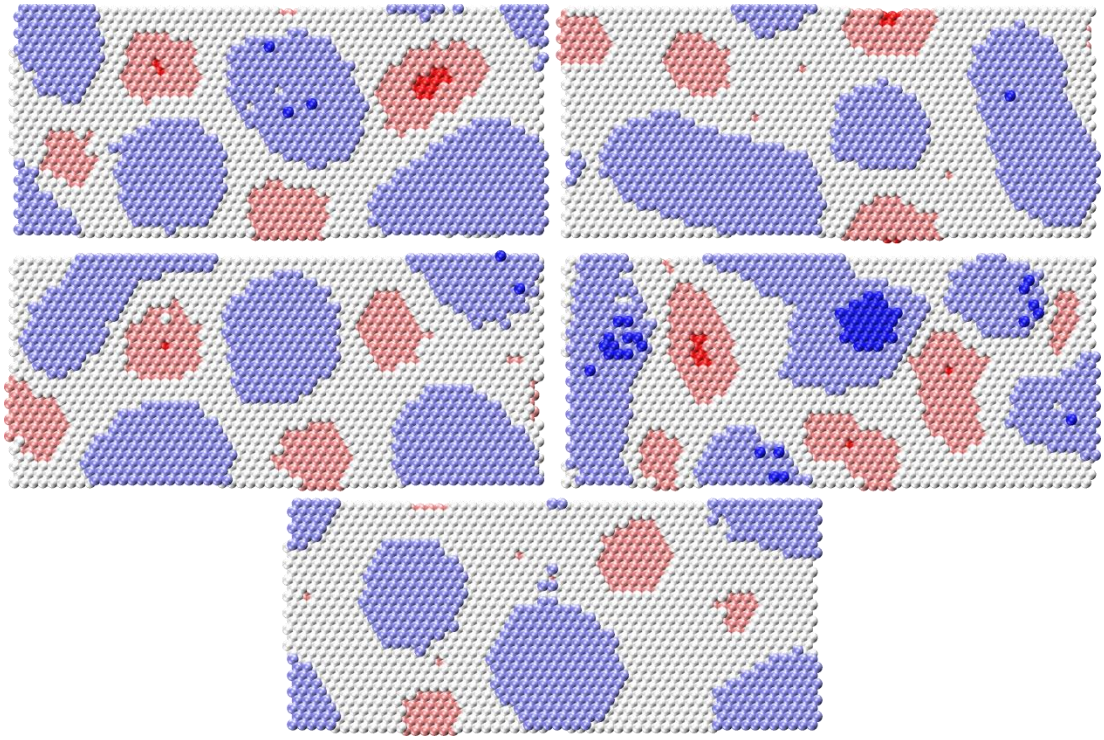


Figure 85: Simulations Kv2 (top left), Lv2 (top right), Mv2 (middle left), Nv2 (middle right) and Ov2 (bottom)

Snapshots from simulations Kv2-Ov2 are shown in Figure 85. These employ the same conditions as Fv2-Jv2 except the Schwoebel barrier is taken to be equal to the energy barrier that needs to be overcome to move away from 10 nearby atoms instead of 5 nearby atoms. Compared to Fv2, simulation Kv2 produced smaller islands. This seems counter-intuitive as the stronger Schwoebel barrier is expected

to increase island and vacancy production. This is likely the cause of the large multilayer vacancy. When comparing Lv2 to Gv2, there is a clear shift from mutlilayered islands to multilayered vacancies but the vacancies and islands only appear to have become more rounded. Similarly, Mv2 produced almost no multilayer island atoms and smaller islands compared to Hv2.

Nv2 produced much more multilayer island atoms and much less multilayer vacancies than Iv2. However, it also produced larger vacancies and smaller islands. Ov2 produced no multilayers and had smaller islands and vacancies than Jv2.

With the exception of Nv2, when comparing simulations Kv2-Ov2 to Fv2-Jv2, there is a clear shift from mutlilayered islands to multilayered vacancies. This seems logical as the doubling in strength of the Schwoebel barrier means atoms that can occupy vacancies or multilayered vacancies are much more likely to be restricted by the Schwoebel barrier due to the energy of their current position being close to the energy of their destination while multilayer island atoms have significant differences in energy between their current position and their destination, causing the Schwoebel barrier to have a negligible effect.

Table 9: Overview of parameters used in simulations Av3 to Rv3

Simulation	S_w	N_{Pl}	D_1	D_2	D_3	D_4	D_5
Av3	5	2	4	4	4	6	2
Bv3	5	2	4	1	1	6	2
Cv3	5	2	1	4	4	6	2
Dv3	5	2	4	4	4	1	2
Ev3	5	2	4	4	4	6	1
Fv3	5	2	4	4	4	6	4
Gv3	5	5	4	4	4	6	2
Hv3	5	5	4	1	1	6	2
Iv3	5	5	1	4	4	6	2
Jv3	5	5	4	4	4	1	2
Kv3	5	5	4	4	4	6	1
Lv3	5	5	4	4	4	6	4
Mv3	10	5	4	4	4	6	2
Nv3	10	5	4	1	1	6	2
Ov3	10	5	1	4	4	6	2
Pv3	10	5	4	4	4	1	2
Qv3	10	5	4	4	4	6	1
Rv3	10	5	4	4	4	6	4

Table 9 provides an overview of the simulations run on the Prdv3 kMC code. The most significant difference between the Prdv2 and Prdv3 codes is that in Prdv3 the atoms that move during surface relaxations are now restricted to being within a certain distance, defined by D_4, of the impact site, wheras in Prdv2 (and Prdv1) they were randomly chosen from anywhere on the surface. In this way, Prdv3 is based on the idea that local diffusion driven by the impact energy is more significant than thermally induced diffusion across the surface.

Like the Prdv2 code, the S_w parameter determines how many interactions are treated to have the same energy barrier as the Schwoebel barrier and more detail can be found in section 4.4.1. The N_{Pl} parameter determines the number of post-impact relaxations that occur. D_1 is still the maximum distance the sputtered atom can be from the impact site. D_2 is the maximum distance that the atom moving during a post-impact relaxation can be from the impact site and D_3 is the maximum distance that the destination site of that atom can be from the impact site. A visual representation of these distances, including D_4 and D_5, can be seen in Figure 86.

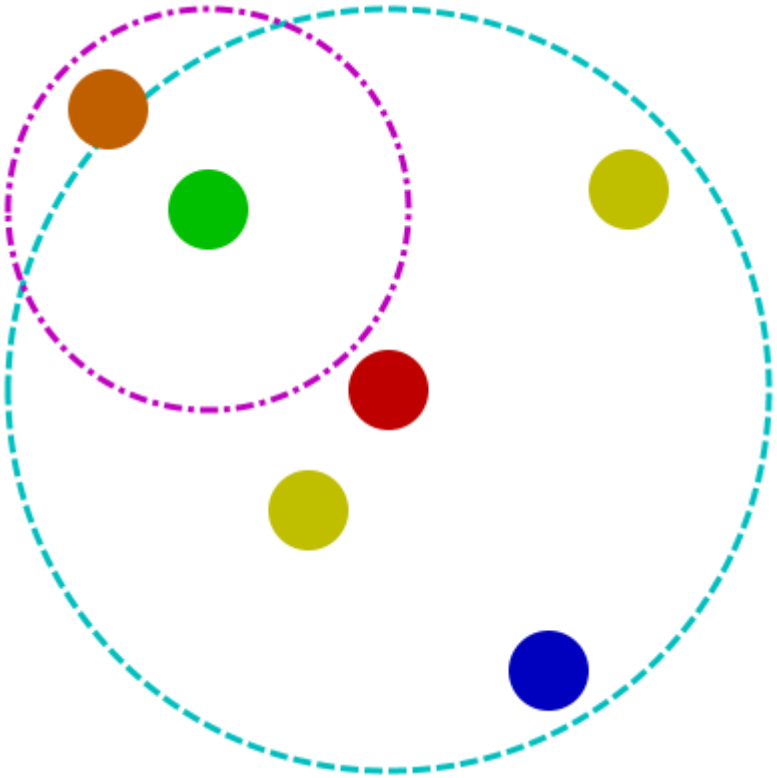


Figure 86: Schematic diagram of the parameters D_1 - D_5

The simulations are split into three groups of six simualtions: Av3-Fv3, Gv3-Lv3 and Mv3-Rv3. In each group, the first simulation, shaded in green, is treated as a baseline. From this baseline, the second and third simulations, shaded in orange, varies distances related to actions taken immediately after an impact, with the second reducing D_2 and D_3 to 1 and the third reducing D_1 to 1. The fourth, fifth and sixth simulations, shaded in blue, varies distance used during the relaxation steps between impacts with the fourth reducing D_4 to 1, the fifth reducing D_5 to 1 and the sixth increasing D_5 to 4.

Figure 86 is a visual representation of the distances D_1-D_5 used in the Prdv3 code. The major difference between this figure and Figure 80 is that the atom that moves during surface relaxations, represented by the green circle, is now limited by D_4, which is based on the distance from the impact site.

Like in Figure 80, the red circle represents the impact site and the cyan dotted circle represents the limits of D_1, D_2, D_3 and now D_4, which are all based on the distance from the impact site. The blue circle, limited by D_1 is the atom chosen to be sputtered. The yellow circles, limited by D_2 and D_3, are the atoms chosen for post-impact relaxations. The magenta dotted circle represents the limit of D_5 and the orange circle is the atom, limited by D_5, that is chosen as the destination of the relaxing atom.

In Figure 87, snapshots from simulations Av3-Fv3 for the Prdv3 code are shown. All six take the Schwoebel barrier to be equal to the energy barrier that would need to be overcome when moivng away from 5 nearby atoms and perform 2 post-impact relaxations after each impact. The first simulation, Av3, produced three large islands, one of which appears to be three smaller islands merged. It also produced 4 vacancies. There are very few multilayered islands and there are only two relatively tiny multilayered vacancies. Bv3 produced the same number of islands and vacancies but they appear to be smaller than Av3. Bv3 also produced almost no mutlilayered islands or vacancies. This is likely due to the restirction causing the area surrounding the impact site to become more smooth, filling vacancies and shrinking islands. (Recall in B that the distances D_2 and D_3 were reduced from 4 to 1, making the post-impact relaxations much more local to the impact site). Cv3 also produces a similar result with only marginally bigger vacancies and more multilayered islands. From this, it can be seen that restricting how far from the impact site that atoms are sputtered, like in Cv3 which restricted the distance from 4

like in Av3 to 1, or relaxed shortly after impact, like in Bv3 which restricted the distances from 4 like in Av3 to 1, also restricts the roughness of the surface.

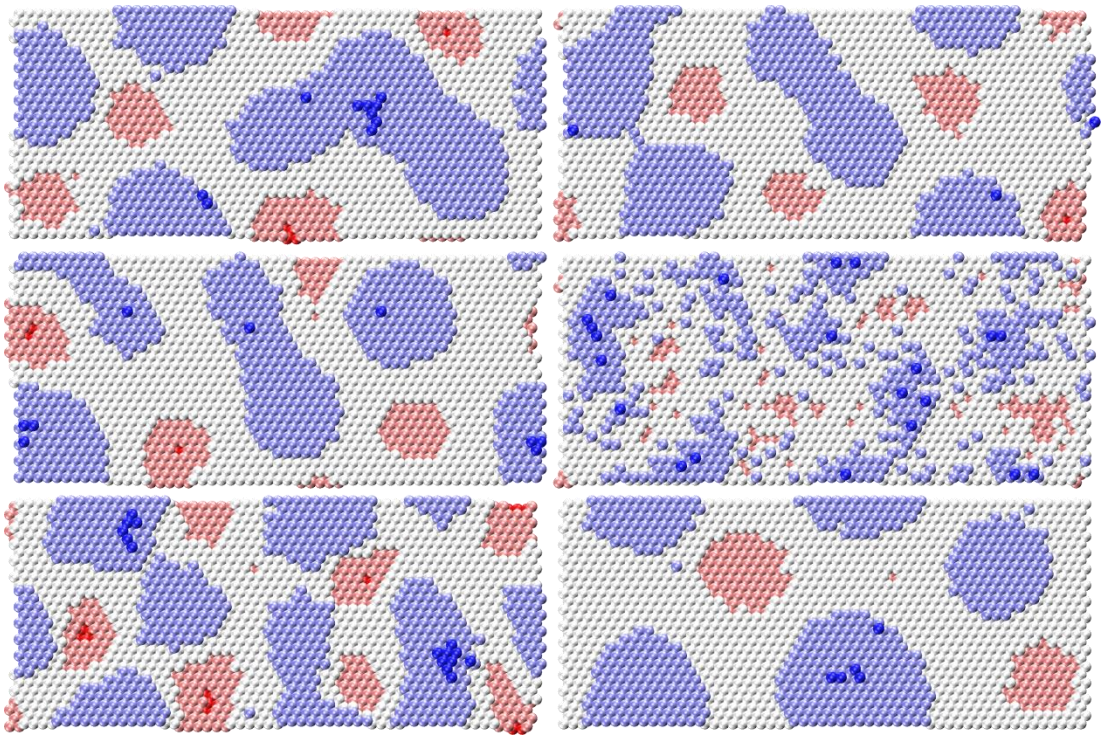


Figure 87: Simulations Av3 (top left), Bv3 (top right), Cv3 (middle left), Dv3 (middle right), Ev3 (bottom left) and Fv3 (bottom right)

The effect of varying D_4, which is the maximum distance that atoms moved during surface relaxations can be from the impact site, and D_5, which is the maximum distance that the destination site of the atom being moved can be from the current site of that atom, were then considered. Simulation Dv3 produces many small islands and vacancies that are more similar to the kind seen in MD but they are too scattered compared to MD. Ev3 produces more vacancies and islands than Av3 and also has more multilayered islands. Fv3 on the other hand has few islands and vacancies and they are very large and rounded. This shows that roughness is greatly affected by restrictions to the atoms chosen during relaxation.

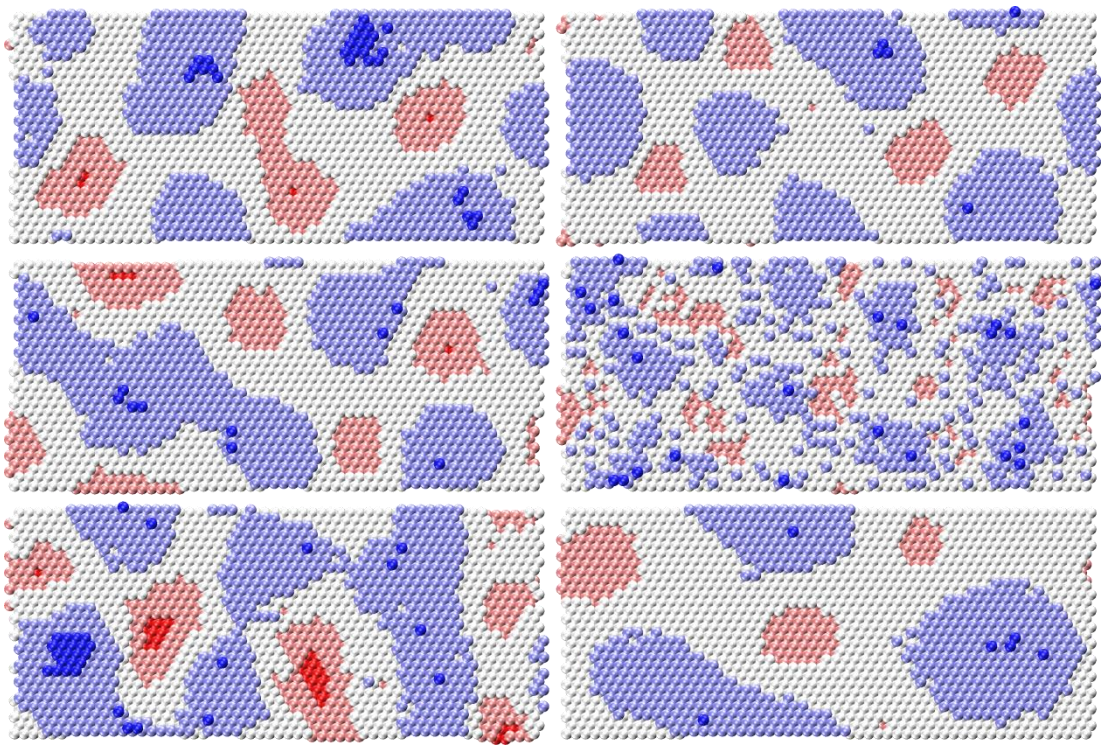


Figure 88: Simulations Gv3 (top left), Hv3 (top right), Iv3 (middle left), Jv3 (middle right), Kv3 (bottom left) and Lv3 (bottom right)

In Figure 88, snapshots from simulations Gv3-Lv3 are shown. These employ the same conditions as Av3-Fv3 except 5 post-impact relaxations occur after each impact instead of 2. Compared to Av3, Gv3 produced larger multilayered islands although the islands produced are smaller. Hv3 produced smaller and more frequent islands and vacancies compared to Gv3 but compared to Bv3, the main difference is less island merging, which is potentially due to the increased relaxations at short range preventing islands from coming into contact. Iv3 on the other hand has much more island merging, compared to Cv3 or Gv3, and has more frequent vacancies.

Jv3 has a large increase in the frequency of vacancies and islands giving it an even rougher surface compared to Dv3. However, the islands also appear to be much more fragmented while in MD, the islands tend to be merged. In simulation Kv3 it can be seen that all islands have merged into one large island spanning across the surface. In comparison with Fv3, Lv3 appears to favour more vacancy growth and less island growth.

When Gv3-Lv3 are compared to simulations Av3-Fv3, which, as stated earlier, had 5 and 2 post-impact relaxations, respectively, it can be seen that the extra movements make islands and vacancies more likely to form overall. It also appears to make

islands more likely to merge into one larger island with Iv3, Kv3 and Lv3 having a maximum of two islands on the first layer above the surface.

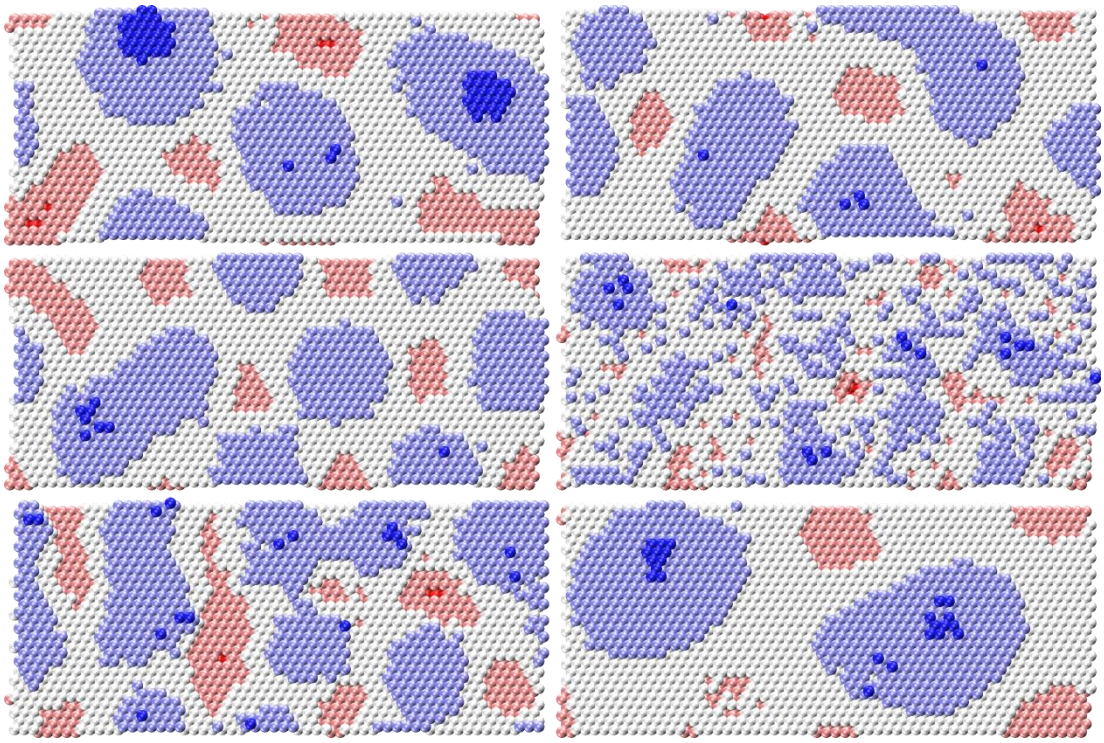


Figure 89: Simulations Mv3 (top left), Nv3 (top right), Ov3 (middle left), Pv3 (middle right), Qv3 (bottom left) and Rv3 (bottom right)

In Figure 89, snapshots from simulations Mv3-Rv3 are shown. These employ the same conditions as simulations Gv3-Lv3 except the Schwoebel barrier is taken to be equal to the energy barrier that needs to be overcome to move away from 10 nearby atoms instead of 5 nearby atoms. In comparison to Av3 and Gv3, Mv3 has produced more rounded islands and has substanially larger multilayered islands. Simulation Nv3 produced a greater number and smaller islands than Mv3 it created larger but fewer islands than Hv3, due to the increased Schwoebel barrier preventing some smoothening. Simulation Ov3 produced signicantly more but smaller islands and vacancies than Cv3 and Iv3.

Pv3 is even more complex than Dv3 and Jv3 but it appears to have created less vacancies and significantly more islands. Qv3 produced oddly-shaped islands and vacancies. Multilayer islands and vacancies were also much less common compared to Ev3 and Kv3. Simulation Rv3 produced islands that are larger and more rounded than Fv3 and Lv3. There was also an increase in the size of

multilayer islands, likely due to the increase in the Schwoebel barrier preventing atoms moving from high to low sites.

Compared to Gv3-Lv3, simulations Mv3-Rv3 favour the creation of islands and vacancies due to the doubling of the strength of the the Schwoebel barrier, which restricts interlayer transport.

Comparing the surfaces produced from the simulations detailed in Tables 7 to 9, some trends can be seen in the effects of changing the parameters.

By increasing the number of post-impact relaxations from 2 to 5, the surface roughness produced in simulations is increased. This is due to the increased amount of surface interactions.

Decreasing the D_2 and D_3 parameters (the maximum distance the two atoms involved in a post-impact relaxation can be from the impact site) produced surfaces that were smoother with less islands and vacancies. This is due to the post-impact relaxations being constrained locally to the previous impact site, potentially causing some relaxations to effectively cancel each other out.

Reducing the D_4 parameter in the Prdv3 code (the maximum distance a relaxing atom can be from the impact site) produced surfaces that had more islands but the islands were smaller and more scattered. This is because the surface relaxation is more heavily based on the surface conditions local to the impact site.

Reducing the D_5 parameter (the maximum distance a relaxing atom can travel from its initial location) produced islands that were more stretched and were more likely to merge with nearby islands. This is due to the surface relaxations being based more locally on the conditions around the relaxing atoms. Meanwhile, increasing the same parameter produced islands that were more rounded as the surface relaxations were able to account for the conditions further away from the relaxing atoms.

Increasing the strength of the Schwoebel barrier tended to create smaller and more numerous islands but this was not always observed.

The effect of reducing the D_1 variable (the maximum distance a sputtering atom can be from the impact site) appears to be inconsistent suggesting that the simulations may not be sensitive to changes to D_1.

The N_{PI} and D_5 parameters might be worthy of further investigation in the future with a broader range of MD simulation substrate temperatures. For the purposes of this study, the results show that the parameters and the processes they mimic from the atomistic MD simulations can be distinguished and help to define a preferred set of simulations with which the length and time scales of the simulations can be extended.

5.1 Selecting the optimal parameters

From visual inspection of the images presented in Figures 81 to 89, it was determined that the simulations that best reproduced the Molecular Dynamics behaviour were simulations Dv3, Jv3 and Pv3, shown in Figures 87, 88 and 89, respectively. These simulations used the Prdv3 code and, as explained in Table 9 and Figure 86, restricted how far from the last impact site that an atom could be chosen during a surface relaxation step. This implies that the relaxation process driven by the the impact energy is more important than the general thermally induced surface diffusion.

Most of the other simulations were producing signifcantly more mutlilayered vacancies than what was seen in MD and many others produced islands that were substanially more rounded than the islands produced in MD. The islands were often larger than those seen in the MD with the size of the multilayer islands being inconsistent and the number of multilayer islands typically being smaller than the MD. Some other simulations were better at replicating the amount of islands merging, nevertheless Dv3, Jv3 and Pv3 seem the best options to study further and this further study is shown below.

Figure 90 compares the average surface height obtained during an MD simulation with the average surface height obtained during simulations Dv3, Jv3 and Pv3 using the third version of the production kMC code. All three kMC simulations produce the small decrease in surface height seen at the start of the MD but to differing levels of accuracy (notwithstanding the statistical nature of the comparisons being made). The cause of this decrease was discussed earlier in Figures 41 to 43.

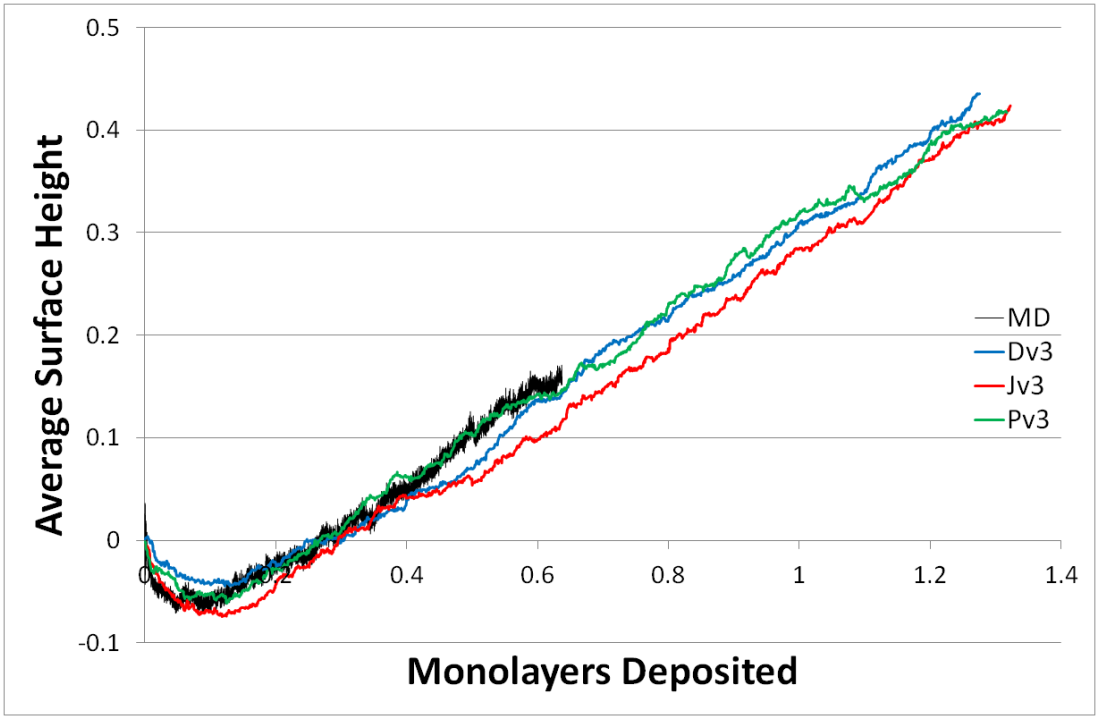


Figure 90: Average surface height of an MD simulation and kMC simulations Dv3, Jv3 and Pv3

Referring back to Figure 90, Jv3 (the red curve) has the approximately the same magnitude of decrease but takes longer to recover from the dip. Dv3 (the blue curve) doesn't decrease as much or as fast as the MD. Pv3 (the green curve) best matches the MD as it gets the correct magnitude of height decrease and begins increasing in height at roughly the same time as the MD however it is marginally slower to decrease than the MD and Jv3.

As both the kMC and the MD runs will be affected by random fluctuations, multiple simulations were run and the average results of these runs were also compared to minimise the effect of randomness, showing the underlying trends. This can be seen in Figure 91.

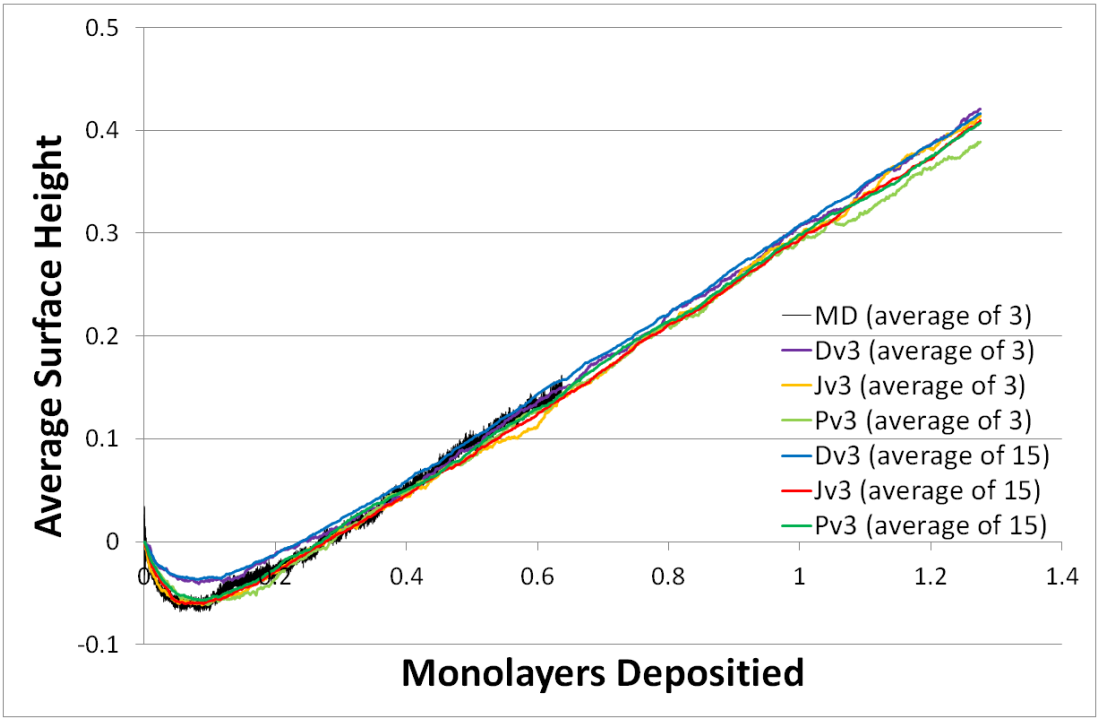


Figure 91: Average surface height of an MD simulation and kMC simulations Dv3, Jv3 and Pv3, averaged over 3 runs and 15 runs

From Figure 91, it can be seen that the underlying trends of the average surface height in Jv3 and Pv3 both match the trend seen in the MD very well, with both producing a very similar initial decrease in the surface height. Meanwhile, Dv3 had a much more limited decrease in the average surface height. When the average surface height began to grow linearly, all simulations followed a similar trend.

Figure 92 compares the surface roughness seen in an MD simulation to the surface roughness seen in simulations Dv3, Jv3 and Pv3 using the third version of the production kMC code. Initially, Jv3 and Pv3 roughened much quicker than the MD but as the simulations progressed, the surface roughness of Jv3 and Pv3 became more aligned with the surface roughness of the MD. Dv3, on the other hand, began with a surface roughness that was very close to the roughness of the MD but over time roughened slower than the MD. It can be seen that after the end of the MD simulation, the surface roughness of all three kMC simulations began to converge and closely matched each other at the end of those simulations.

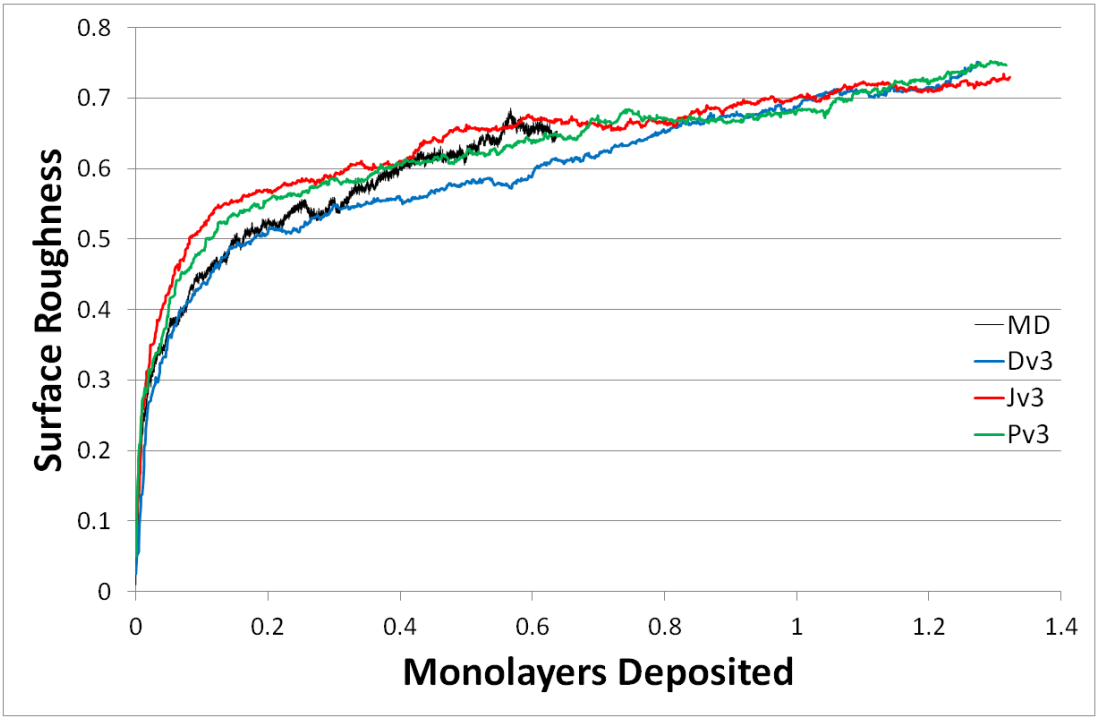


Figure 92: Surface roughness of an MD simulation and kMC simulations Dv3, Jv3 and Pv3

Like the average surface height, the surface roughness is also affected by random fluctuations during both the MD and the kMC. To ensure the comparisons are not restricted by noise, the surface roughness was averaged over 3 runs for the MD and 3 and 15 runs for the kMC in Figure 93.

From Figure 93, it can be seen that underlying trend of the surface roughness in Dv3 is very close to the surface roughness observed in the MD during the rapid roughening phase at the beginning but as the simulation progresses the MD roughens faster than Dv3. The underlying trends of the surface roughness in Jv3 and Pv3 show that these simulations tend to roughen faster than the MD throughout the simulation. As seen in Figure 92, after the MD simulation ended, the surface roughness in the three kMC simulations begin to converge.

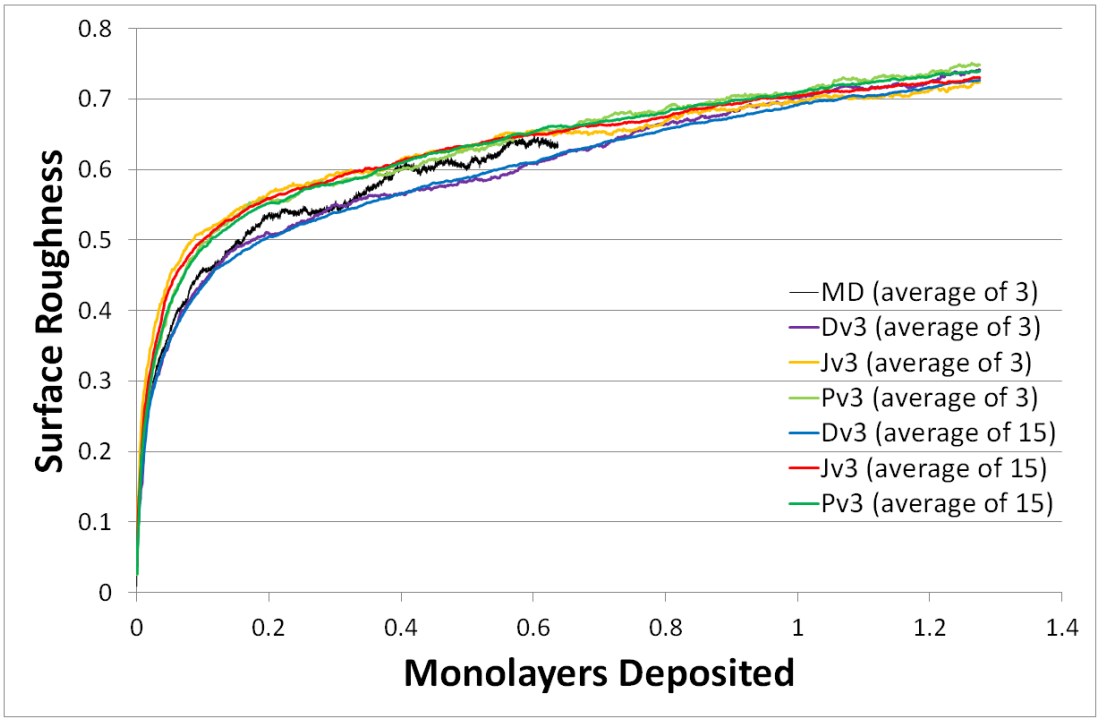


Figure 93: Surface roughness of an MD simulation and kMC simulations Dv3, Jv3 and Pv3, averaged over 3 and 15 runs

By the end of the kMC simulations, Dv3 and Jv3 are likely to have converged while the surface in Pv3 tends to be slightly rougher. Recall from Table 9 that Pv3 uses a Schwoebel barrier that is treated as equivalent to the energy barrier that needs to be overcome to move away from 10 nearby atoms instead of 5 nearby atoms like in Dv3 and Jv3. This increase may be restricting the filling of vacancies which could be causing the surface to be slightly rougher than Dv3 and Jv3. The slower surface roughening at the beginning of the simulation in Dv3 compared to Jv3 and Pv3 is likely caused by Dv3 only having 2 impact movements instead of 5 like Jv3 and Pv3. These movements can cause the formation of vacancies and islands so the decreased amount of impact movements means islands and vacancies are formed slower leading to slower surface roughening.

As the surface statistics shown are unable to definitively determine which simulation is the most accurate representation of the MD, the island and vacancy size distributions were investigated.

5.1.1 Island Size Distributions

Although the average surface height and surface roughness provides some information about how accurately the kMC simulations replicate the MD simulations, the visual comparison appears to be the most useful for determining if the surfaces are similar. As visual comparisons are quite subjective, a more qualitative measure was sought.

A new subroutine was added to the Prdv3 kMC code to generate the size distribution of islands and vacancies on each of the layers of the surface. To achieve this, the system cycled through each layer of the surface except the lowest layer. The lowest layer is excluded as the distributions it produced were consistent in all circumstances. The subroutine then searches across every site of the system. If a site has a height lower than the layer being investigated, it is registered as a vacancy, while sites with heights equal to or greater than the current layer are registered as islands. The selected site's neighbouring sites are then checked to determine if they are part of the same island or vacancy as the selected site.

To avoid islands or vacancies counting the same sites multiple times or incorrectly reporting large islands or vacancies as multiple smaller islands or vacancies, when a new island or vacancy is registered, a linked-list is generated. As new sites are added, these new sites are connected to the previously added sites. Then, if the neighbour of a site in an island or vacancy is found to be part a different island or vacancy, the system moves through the linked-list of the island or vacancy, updating each site so that the two detected islands or vacancies are correctly identified as one merged island or vacancy.

To produce the distributions for MD simulations, a program was made that converted the positions of atoms to the equivalent sites of a kMC lattice. Due to the way that MD crystals were generated and the way island growth occurred, the conversion to lattice sites had to be able to account for both directions of the (111) surface, ABC packing and CBA packing. The two directions of the (111) surface are illustrated in Figure 94. In the kMC, the surface is assumed to always grow with ABC packing like the surface on the left, with atoms supported by a trio of atoms pointing up on the y axis. However, in the MD, it was seen that the (111) surface generated had CBA packing like the surface on the right, with atoms supported by a trio of atoms pointing down. It was also seen that islands could form in either direction,

meaning that some layers could form HCP-like stacking with CBA-B packing instead of the expected FCC-like stacking of CBAC. This change in packing was observed multiple times in the MD simulation with the deposition of the equivalent of 40 monolayers (see Figure 62 in section 4.3) with the 15 ad-layers growing CBACBA-BCABC-BAC-A with each dash denoting a change in packing.

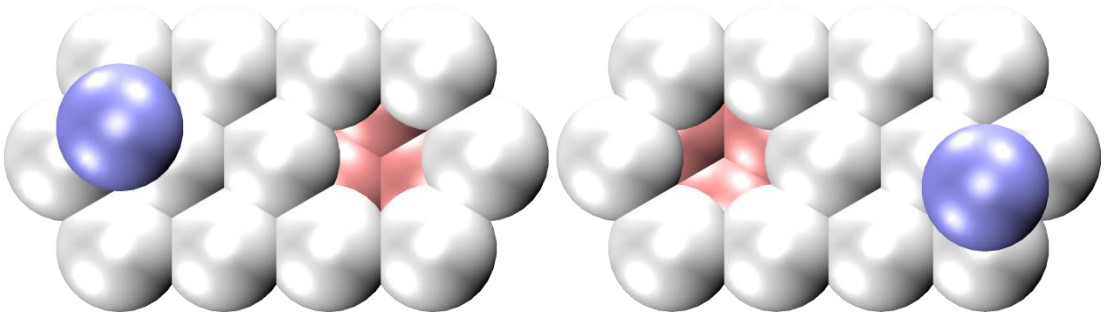


Figure 94: Representation of the two directions a (111) surface can grow

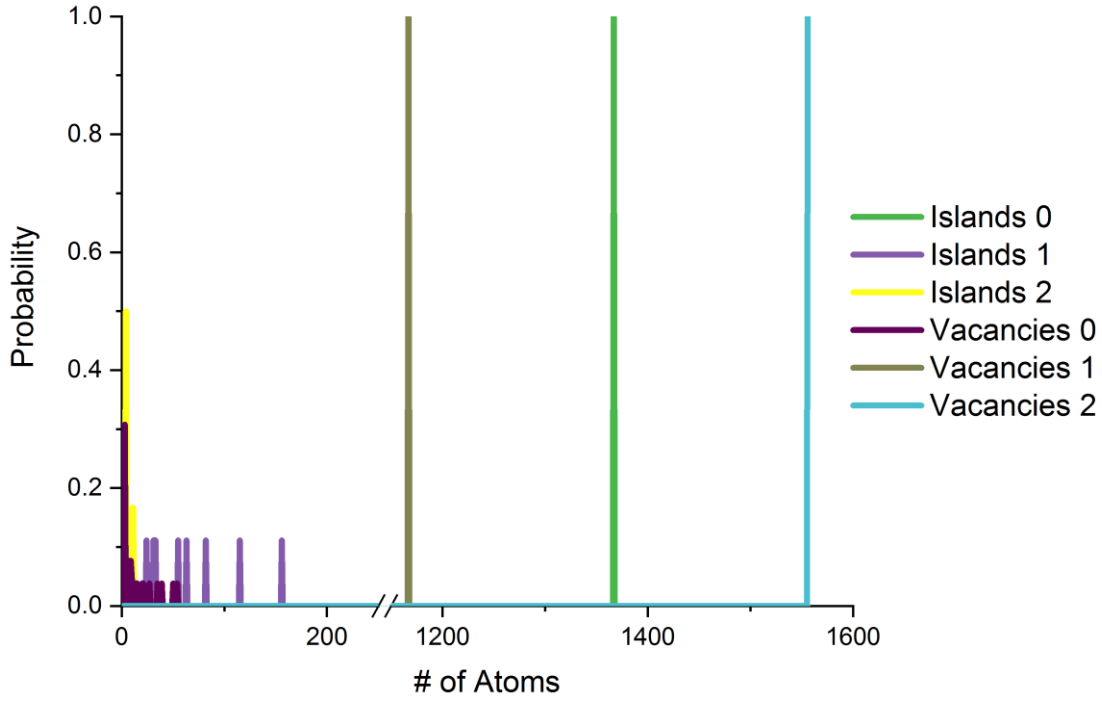


Figure 95: Size distribution graph of MD simulation showing probability of island or vacancy with a given number of atoms at the various layers. The numbers in the legend are the number of layers they are above the original surface layer (i.e. Islands 0 is the island size distribution of the original surface layer)

From Figure 95, it can seen that in MD simulation, the surface layer is made of one large island of ~1370 atoms with a large number of small vacancies ranging in size from 3 to ~50 atoms. The layer above is mostly vacant with one large vacancy of ~1160 and several islands ranging in size from ~20 to ~160 atoms. The top layer has significantly less islands with all being no greater than 15 atoms and a vacancy of ~1560 atoms. Note that the size of a vacancy is defined by the number of sites/atoms exposed by the vacancy so the combined size of the vacancies and islands in a layer will be larger than the size of the layer.

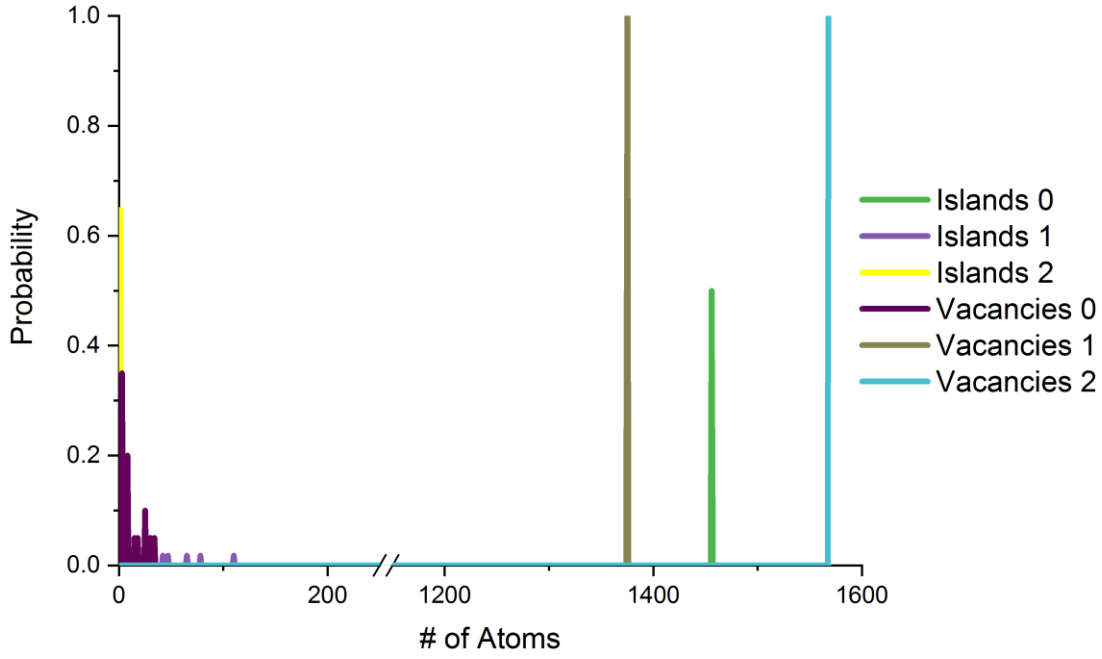


Figure 96: Size distribution graph of Dv3 kMC simulation showing probability of island or vacancy with a given number of atoms at the various layers

It can be seen in Figure 96 that the surface produced by a kMC simulation using the parameters for Dv3 have less vacancies in the original surface layer compared to the MD as the main island of the original surface layer is much larger than in the MD. On the first island layer, there are substantially less island atoms than in the MD but a much larger number of islands overall, though these islands are typically much smaller. Smaller islands are also seen on the second island layer though these differences are less noticeable.

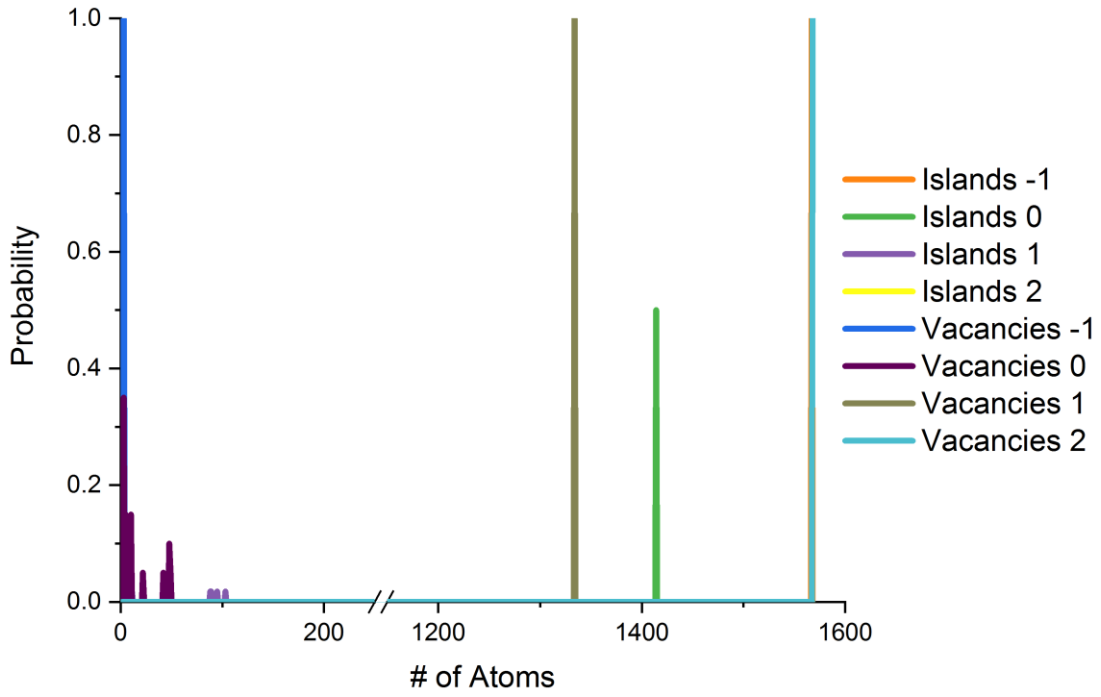


Figure 97: Size distribution graph of Jv3 kMC simulation showing probability of island or vacancy with a given number of atoms at the various layers

In Figure 97, an extra layer is present as a vacancy formed in the sub-surface layer. This extra layer appears to mirror the second island layer as the size of the island in the sub-surface is similar in size to the vacancy across the second island layer at ~1568 atoms. In the original surface layer, Jv3, like Dv3, has fewer vacancies in the layer than the MD but Jv3 has more vacancies than Dv3. Similarly, Jv3 has less island atoms than the MD but more than Dv3 in the first island layer. This suggests that Jv3 is a closer match than Dv3 but it is still significantly different from the MD.

The size distribution for Pv3 in Figure 98 seems to show the closest agreement to the MD in some ways but the worst agreement in other ways as the original surface layer has the smallest number of island atoms of the three kMC simulations. However, it also formed two smaller islands within the vacancies and although the total number of atoms exposed by vacancies is the largest, the vacancies appear to be larger but form less frequently than the MD. Conversely, for the first island layer, the total number of island atoms is closer to the MD than in the other kMC simulations but the islands are smaller and more frequent than the MD.

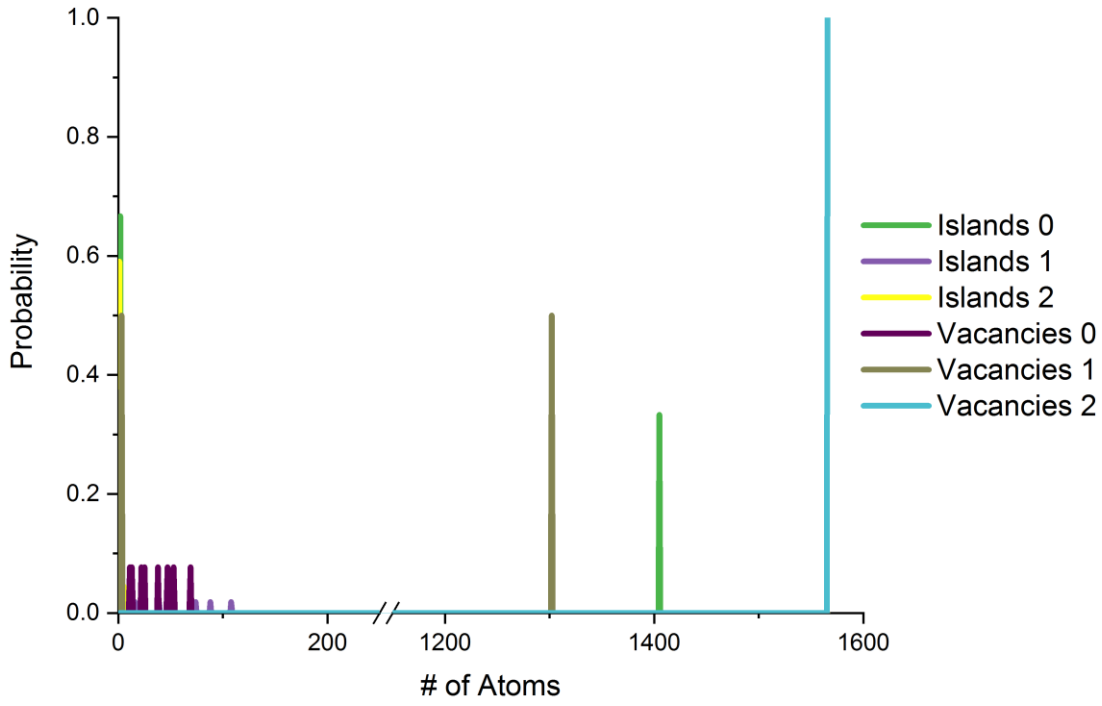


Figure 98: Size distribution graph of Pv3 kMC simulation showing probability of island or vacancy with a given number of atoms at the various layers

The size distributions were also calculated for the other kMC simulations that were a poorer visual resemblance to the MD. A sample of three simulations with good agreement is shown in Figures 99 to 101. The three simulations chosen were Bv3, Hv3 and Nv3 as they had what appeared to be the best agreement of the other kMC simulations and in some ways, appear to have better agreement than the three kMC simulations selected to be more thoroughly compared to the MD.

Looking at the island size distribution of Bv3 in Figure 99, it appears that it heavily favours island growth to vacancy formation as the number of sites exposed by the vacancy in the first island layer is much lower than the previous kMC simulations and is even lower than the MD while the island on the original surface layer is roughly the same size as the one seen in Dv3. It can also be seen that the islands in the first island layer are bigger than in the MD but less frequent. However, the second island layer only has a single island atom suggesting that growth in Bv3 is more layer by layer rather than the multi-layered growth seen in the MD.

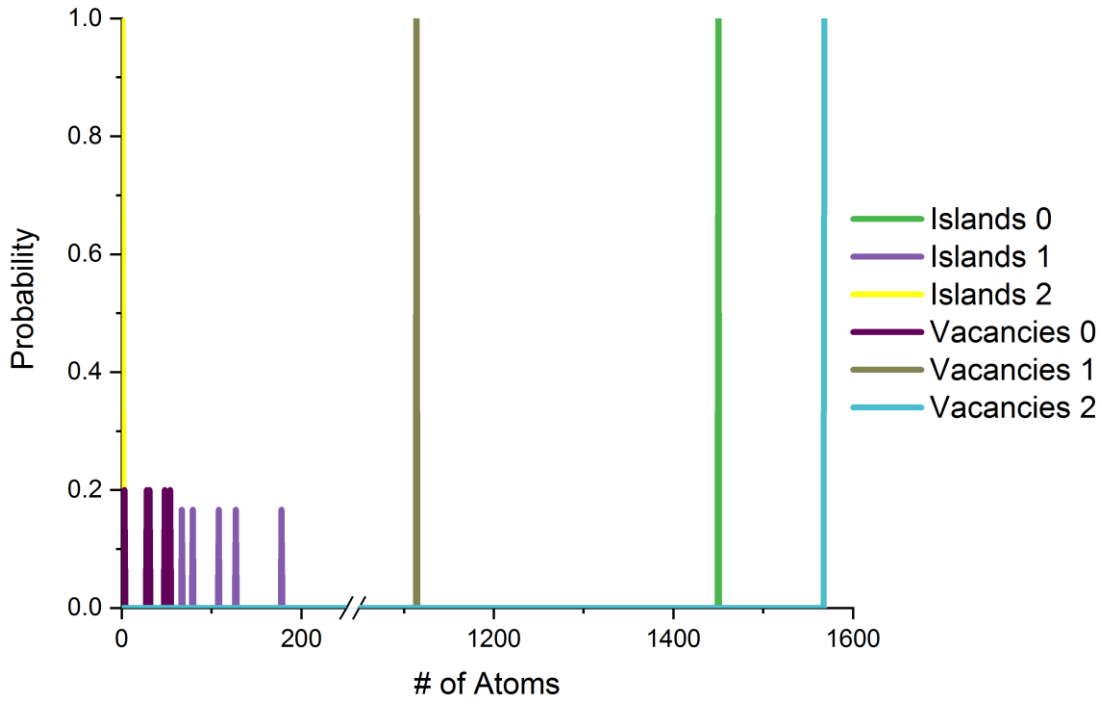


Figure 99: Size distribution graph of Bv3 kMC simulation showing probability of island or vacancy with a given number of atoms at the various layers

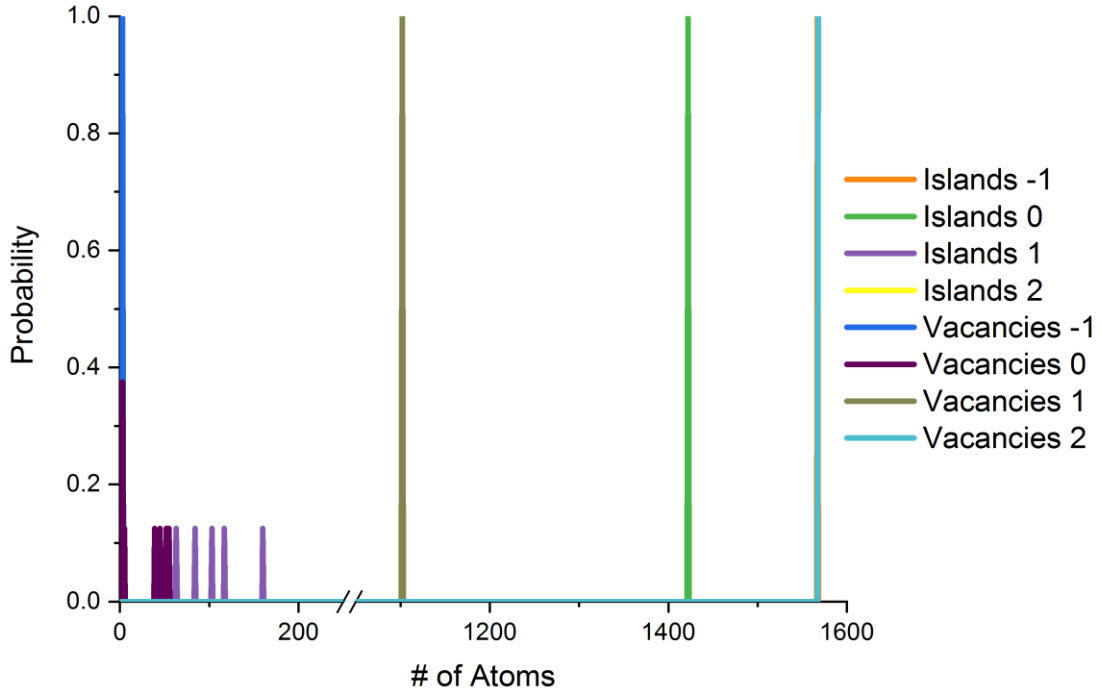


Figure 100: Size distribution graph of Hv3 kMC simulation showing probability of island or vacancy with a given number of atoms at the various layers

The Hv3 simulation, like the Jv3 simulation, produced a vacancy in the sub-surface layer. Compared to Bv3, the vacancy size distribution for the original surface layer in

Figure 100 is more varied and closer resembles the numerous small vacancies in the MD but in most other aspects, Hv3 has many of the same issues as Bv3 as well as the same strengths.

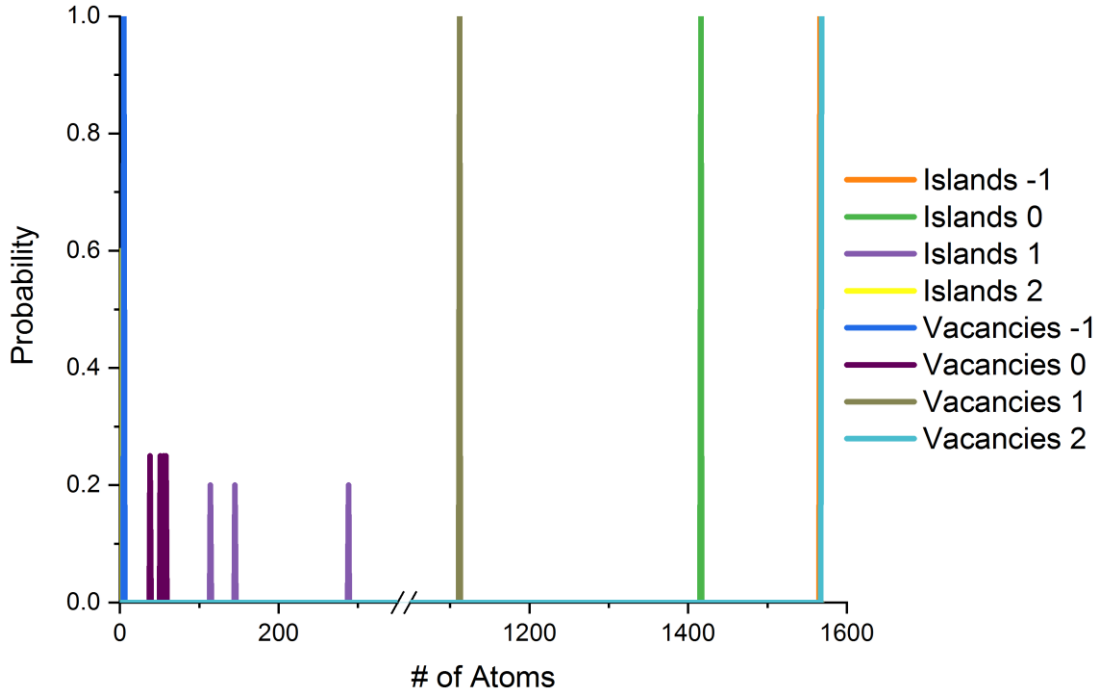


Figure 101: Size distribution graph of Nv3 kMC simulation showing probability of island or vacancy with a given number of atoms at the various layers

The Nv3 simulation, like the Hv3 and Jv3, formed a vacancy within the sub-surface layer. The distributions in Figure 101 seem to show that Nv3 favoured larger groups of atoms with only four vacancies formed in the original surface layer and only 5 islands in the first island layer, with one island being close to 300 atoms.

Although simulations Bv3, Hv3 and Nv3 matched the size distributions of the MD better in some aspects, all three of these simulations were worse with the second island layer and fail to produce a surface similar to the MD. While the size distribution quantifies parts of the surface, it does not provide enough clarity to conclusively select a kMC simulation to use further. However, it does provide an insight that allows a more objective comparison than relying on the visual representation of the surface so this could be a useful approach to use in future work. Currently, the approach needs to be developed further and was not used in any of the subsequent simulations.

5.2 Surface Growth Analysis

From Figures 90 to 93, it was determined that simulation Jv3 is most likely the best compromise. It produced a reasonably good match with the MD when looking at average surface height but had a poorer match when considering surface roughness.

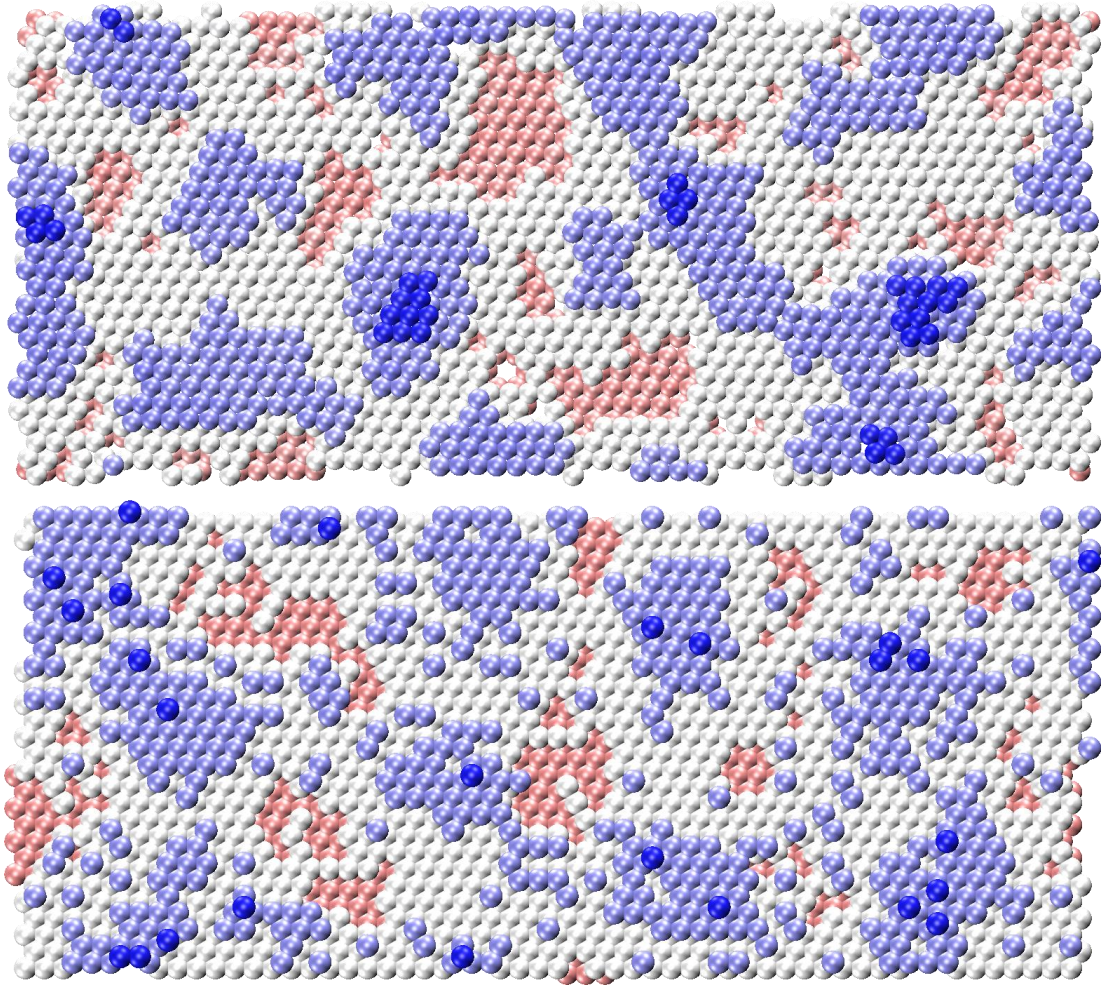


Figure 102: Comparison of the surface from an MD simulation (top) to a Jv3 kMC simulation (bottom)

In Figure 102, the surface of an MD simulation was compared to the surface of a Jv3 kMC simulation. Comparing the two, it can be seen that both have a substantial number of islands and vacancies of various sizes. The vacancies and islands of both simulations also have somewhat complex shapes. However, it can be seen that while both favour the production of islands, the MD has a larger proportion of adsurface atoms and a smaller proportion of subsurface atoms than the kMC. Island

merging is also much more common in MD while the kMC has islands that are much more scattered and sporadic. This can further be seen in the multi-layered island atoms as in the MD, they have all formed islands of more than one atom while in the kMC, most of the multi-layered islands atoms are lone atoms.

Figure 103: Average surface height during the Jv3 simulations of the deposition of the equivalent of 2.5 monolayers at various surface sizes

In Figure 103, how the average surface height changes during deposition at various surface sizes was analysed, starting with a surface of 56 by 28 lattice sites, which is the surface size used in the comparison with MD. The number of sites was then doubled in alternating directions until a surface with 224 by 112 lattice sites, 16 times larger than the initial surface, was produced. The surface sizes are summarized in Table 10.

Table 10: Surface size for each simulation

Simulation	Number of lattice sites
Size 1	56 x 28
Size 2	56 x 56
Size 4	112 x 56
Size 8	112 x 112
Size 16	224 x 112

Looking back at Figure 103, all 5 simulations follow roughly the same pattern of dipping for the first 0.1 monolayers deposited before the surface begins to grow linearly. The dip seen at the beginning of the simulations in this work was discussed in more detail in Figure 90. The Size 1 surface appears to grow the most erratically but this would be expected due to its small size as minor variations would have a much more pronounced effect on smaller surfaces.

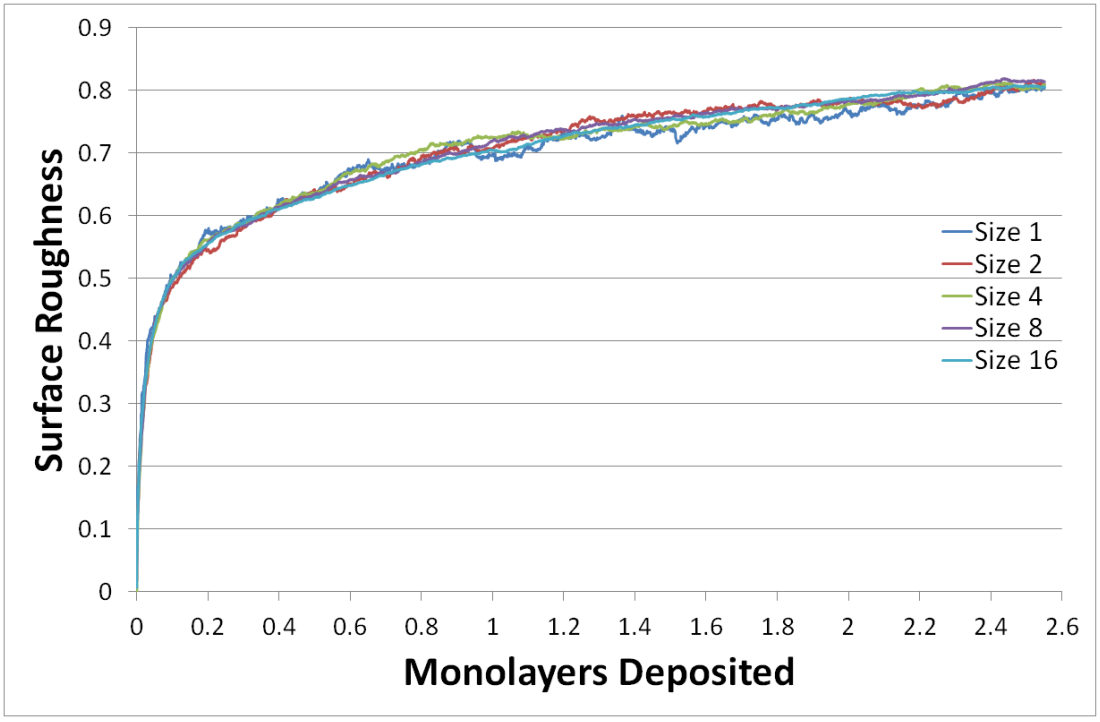


Figure 104: Surface roughness during the Jv3 simulations of the deposition of the equivalent of 2.5 monolayers at various surface sizes

How the surface roughness changes during deposition at various surface sizes was analysed in Figure 104. It can be seen that the surface roughness for all 5 simulations roughly follows the same pattern, rising sharply during the equilibration period before growing more slowly over time. However, it is unclear if the surface roughness is growing as a power law or if the surface roughness will eventually level off. To remove uncertainty, the data from the graph could be plotted on a logarithmic scale or the logarithm of the data could be plotted. If the surface roughness followed a power law, a linear trend would be observed on a logarithmic scale or with logarithmic data.

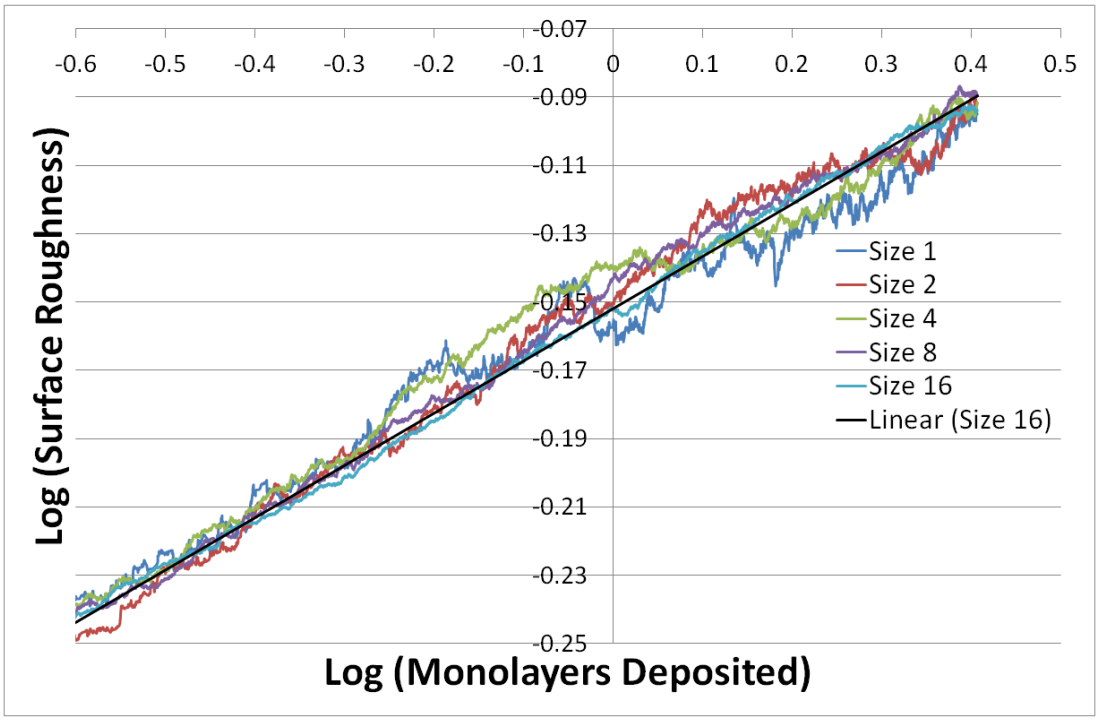


Figure 105: Log (Surface Roughness) against Log (Monolayers Deposited) at various surface sizes using a subset of the data displayed in Figure 104

In Figure 105, the log of the surface roughness was plotted against the log of the monolayers deposited for a section of the data displayed in Figure 104 to better analyse the trend of the surface roughness. In the figure, data from approximately 0.25 monolayers deposited onwards is plotted and values of the surface roughness from approximately 0.56 to 0.82 are shown. Linear trendlines were fit to the data and the gradient obtained from each trendline is shown in Table 11. The trendline for the Size 16 surface is shown in the figure.

Table 11: The gradient of the trendline fit to each simulation

Simulation	Trendline Gradient
Size 1	0.1326 ± 0.0056
Size 2	0.1515 ± 0.0025
Size 4	0.1403 ± 0.0014
Size 8	0.1521 ± 0.0004
Size 16	0.1533 ± 0.0004

The trendline gradients in Table 11 all show a confidence interval. This interval was obtained by taking 20 random subsets from each simulation of 1% of the data for the logarithm of the surface roughness. A trendline was then fit to each subset and the

gradient of that trendline was recorded. From the 20 gradients recorded for each simulation, the standard deviation of those gradients was calculated and used as the confidence interval.

From the Table, it can be seen that while trendline gradients for the Size 2, 8 and 16 surfaces are approximately similar, the trendline gradients for the Size 1 and 4 surfaces are significantly lower. Looking back at Figures 104 and 105, this appears to be due to both the Size 1 and 4 surfaces being rougher than the other surfaces during the first half of the simulation but becoming smoother during the second half of the simulation.

Returning to Figure 105, the logarithm of the surface roughness appears to grow linearly with the logarithm of the monolayers deposited for all 5 simulations, suggesting that the surface roughness for all 5 follows a power law. It can also be seen that the smaller Size 1, 2 and 4 simulations tend to deviate from this linear growth significantly more than the Size 8 and 16 surfaces.

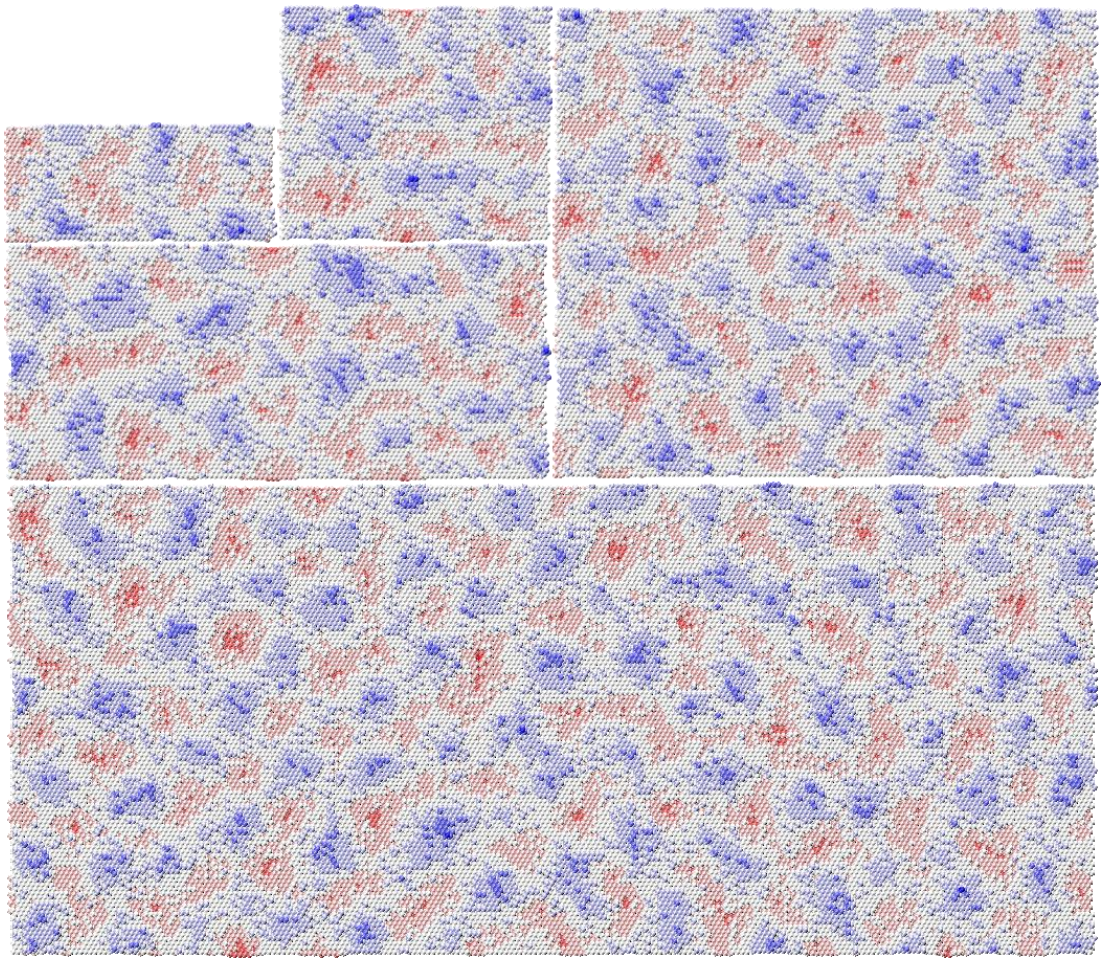


Figure 106: The final surface of Jv3 kMC simulations after deposition of 2.5 monolayers at surface sizes 56x28, 56x56, 112x56, 112x112 and 224x112, displayed to scale. This figure uses a similar colour gradient to Figure 43 but ranges from three layers below the original surface in red to three layers above the original surface in blue.

In Figure 106, the surfaces obtained at the end of the simulations analysed in Figures 103 and 104 are shown. The surfaces were produced by running Jv3 kMC simulations until an equivalent of 2.5 monolayers are deposited on surfaces starting with the Size 1 surface (56 by 28 lattice sites), which is the surface size used in the comparison with MD and doubling the number of sites in alternating directions until the Size 16 surface with 224 by 112 lattice sites was obtained. All surfaces are shown approximately to the same scale as the Size 16 surface. Comparing the Size 16 surfaces with the other surfaces, they seem to match very well such that they could be transplanted into the Size 16 surface without appearing incongruous with the rest of the surface. This would suggest that for 2.5 monolayers, the Size 1 surface is large enough to simulate how the surface evolves without being negatively affected by finite size effects.

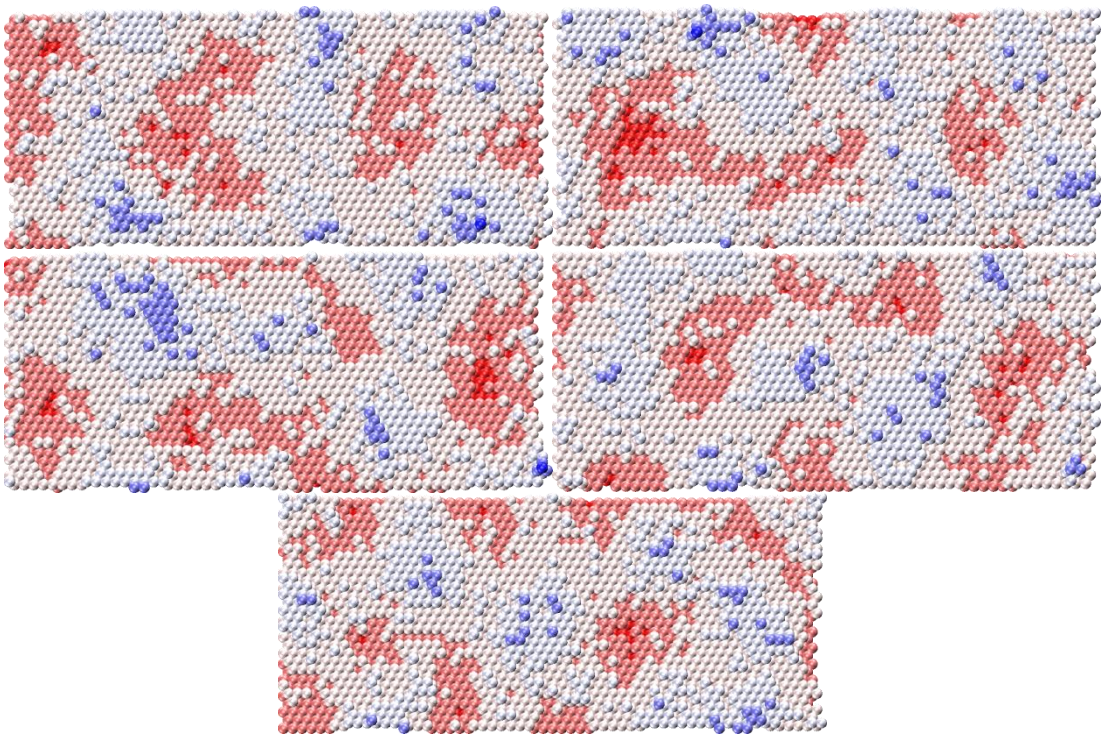


Figure 107: The final surface of the Size 1 simulation (top left) and the final surfaces of the Size 2 (top right), Size 4 (center left), Size 8 (center right) and Size 16 simulations (bottom), cropped to show a 56x28 lattice site section. This figure uses a similar colour gradient to Figure 43 but ranges from two layers below the original surface in red to three layers above the original surface in blue with the original surface layer in light pink as it is not the center of the range.

Figure 107 shows the same surfaces shown in Figure 106 after cropping the Size 2, 4, 8 and 16 surfaces down to show a section that is roughly similar in size to the Size 1 surface, 56 by 28 lattice sites. The Size 2 simulation uses the top half of its surface, and the Size 4, 8 and 16 simulations all use the section in the top right corner of the their respective surfaces.

Again no visual clues that the images come from different system sizes are seen and it can be concluded that Size 1 serves well as a model system for the deposition of 2.5 monolayers.

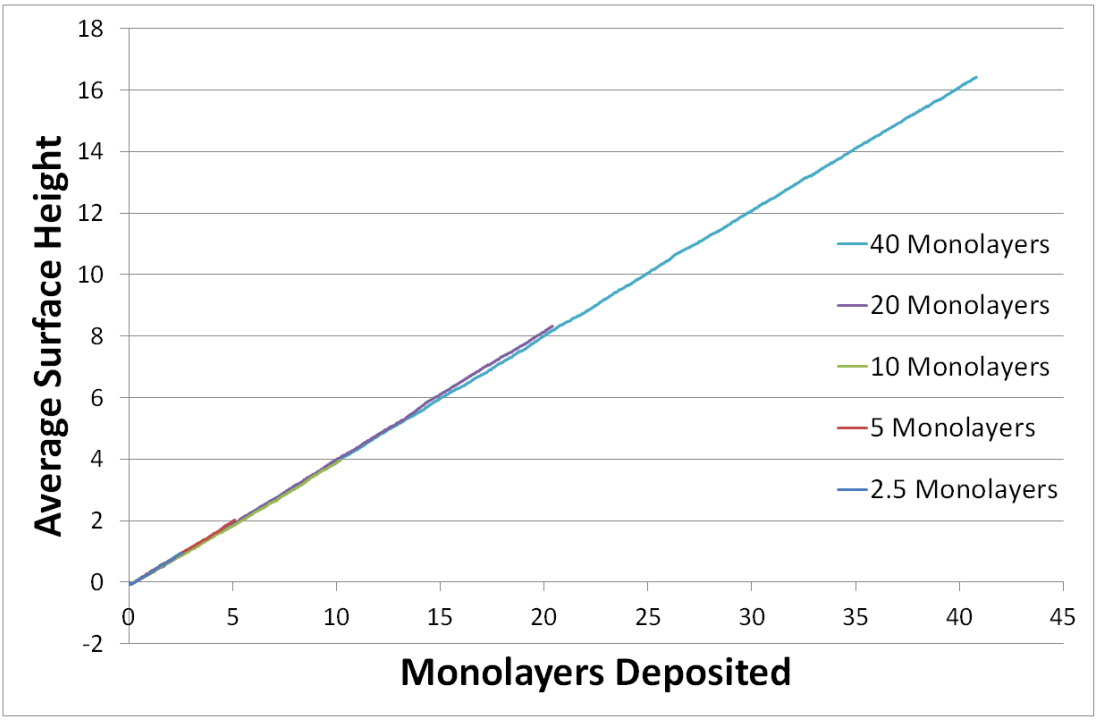


Figure 108: Average surface height of Jv3 simulation after depositing the equivalent of 2.5, 5, 10, 20 and 40 monolayers

Figure 108 presents the average surface height observed during the Jv3 simulation on a 56 by 28 atom surface when depositing the equivalent of 2.5, 5, 10, 20 and 40 monolayers.

Table 12: The gradient of the trendline fit to each simulation

Simulation	Trendline Gradient
2.5 Monolayers	0.4155 ± 0.0070
5 Monolayers	0.4106 ± 0.0016
10 Monolayers	0.3972 ± 0.0015
20 Monolayers	0.4165 ± 0.0003
40 Monolayers	0.4065 ± 0.0001

Table 12 details the gradients obtained from trendlines fit to the plot of the average surface height for each simulation. Like in Table 11, all of the trendline gradients have a confidence interval taken from the standard deviation of gradients from trendlines fit to random subsets of the data, this time of the average surface height.

The rate of surface growth for the 2.5, 5 and 20 monolayers simulations all overshoot the growth rate of the 40 monolayers simulation while the growth rate of

the 10 monolayers simulation undershoots the surface growth rate. The 5 monolayers simulation is closest to the 40 monolayers simulation, with a difference of 0.0051, while the others have close to double the error at 0.009-0.01.

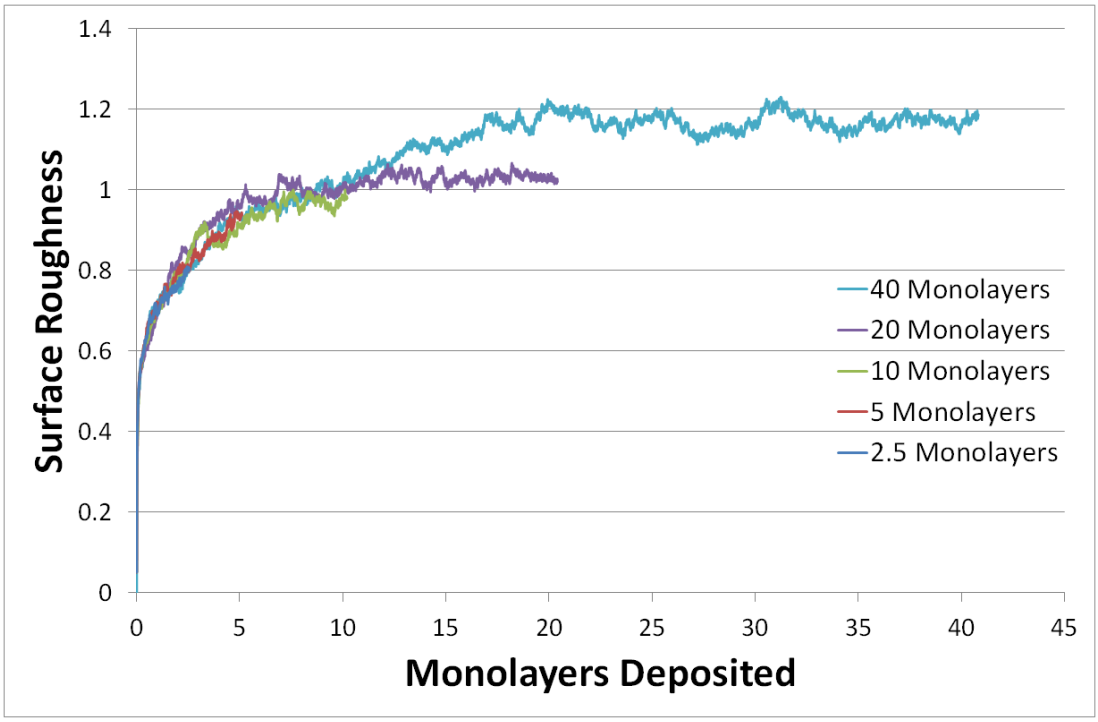


Figure 109: Surface roughness of Jv3 simulation after depositing the equivalent of 2.5, 5, 10, 20 and 40 monolayers

In Figure 109, how the surface roughness evolves was analysed during the Jv3 simulation on a 56 by 28 surface as the equivalent of 2.5, 5, 10, 20 and 40 monolayers was deposited. The 20 monolayers simulation roughened faster than the others up to the equivalent of ~8 monolayers deposited. The 2.5 and 5 monolayers simulations are clearly still roughening at the end of their simulations and it appears that the 10 monolayers simulation begins to roughen much more slowly but it still doesn't level off at the end of the simulation. The surface roughness in the simulations for 20 monolayers and 40 monolayers both appear to level off. However, they level off at different values of roughness after different amounts of deposition. The 20 monolayers simulation levels off much sooner, settling after the deposition of the equivalent of 9-10 monolayers, at a surface roughness equivalent to the height of 1 monolayer. The 40 monolayers simulation continues to roughen until the equivalent of about 20 monolayers had been deposited when the surface roughness levelled off at a roughness equivalent to the height of 1.2 monolayers. It is unclear why this large discrepancy is seen so the 20 and 40 monolayers

simulations were run an extra three times. The randomness inherent in the simulation would cause the evolution of the surface roughness during the simulations to change in the subsequent runs. Comparison of the simulations and their repeated runs highlight how the surface roughness is affected by the inherent randomness of the simulation.

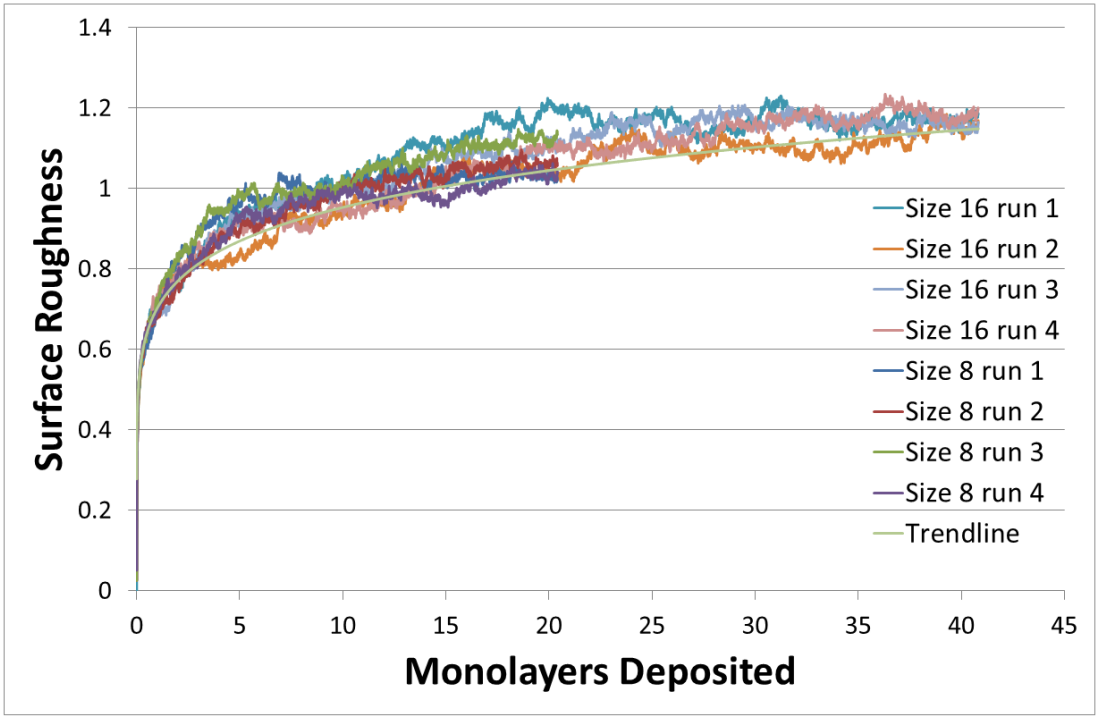


Figure 110: Surface roughness of repeated 20 and 40 monolayers simulations

In Figure 110, the surface roughness of the 20 and 40 monolayers simulations from Figure 109 are compared to an extra three simulations using the same conditions. All four 40 monolayers simulations settle at roughly the same final surface roughness. At 20 monolayers deposited, three of the 20 monolayers simulations and one of the 40 monolayers simulations have roughly the same roughness at 1.02-1.06. The remaining 20 monolayers simulation and two of the 40 monolayers simulations were also close to one another at 1.10-1.13. The original 40 monolayers simulation was significantly higher at ~1.20.

A trendline based on the trend fit to the Size 1 surface data in Figure 105 is also plotted. It can be seen that the second simulation of 40 monolayers deposition matches quite well with the trendline and has roughly similar values of surface roughness after the deposition of the equivalent of 10, 20, 30 and 40 monolayers. Most of the other simulations produce a relatively poor match and are rougher than

expected from the trendline, suggesting the trendline is not a good fit for the surface roughness over a longer period of deposition.

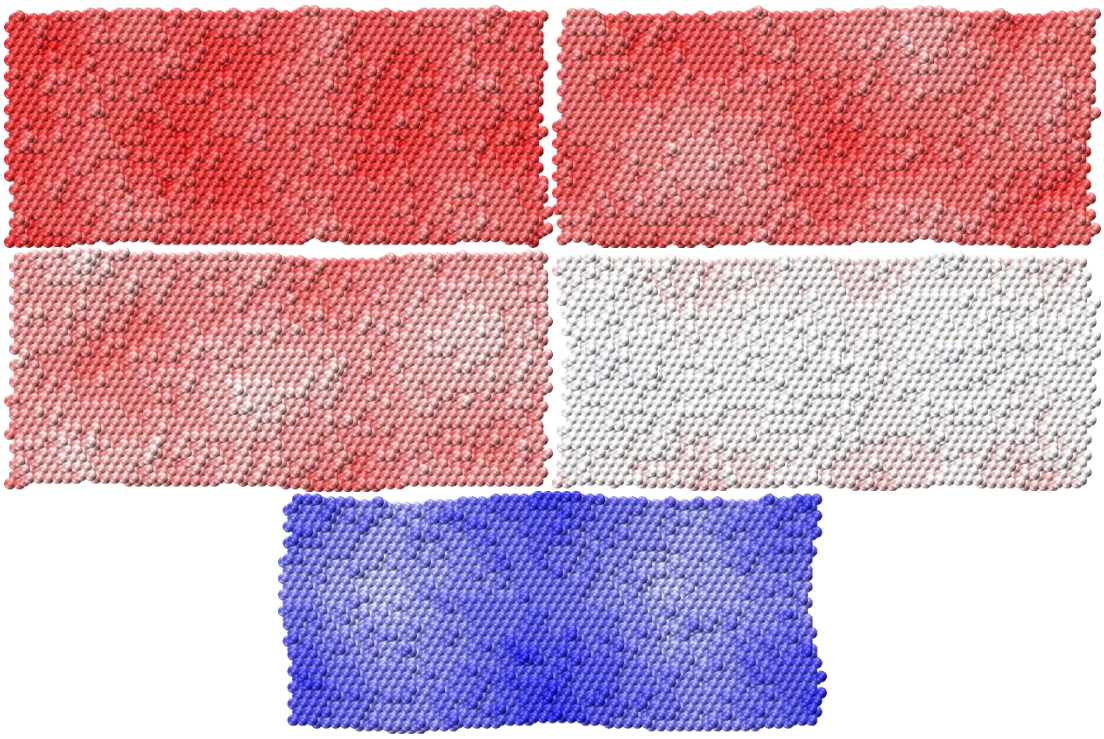


Figure 111: Final surface of Jv3 simulations after depositing the equivalent of 2.5 (top left), 5 (top right), 10 (middle left), 20 (middle right) and 40 (bottom) monolayers. All use the same colour gradient to denote height with blue denoting the highest atoms and red denoting the lowest and white denoting the middle of the range

In Figure 111, the surfaces obtained at the end of the simulations used in Figures 108 and 109 are shown. The surfaces were produced by running Jv3 kMC simulations on the 56 by 28 atom surface and depositing the equivalent of 2.5, 5, 10, 20 and 40 monolayers. All are shown on the same colour gradient with atoms from red to white to blue as their height increases.

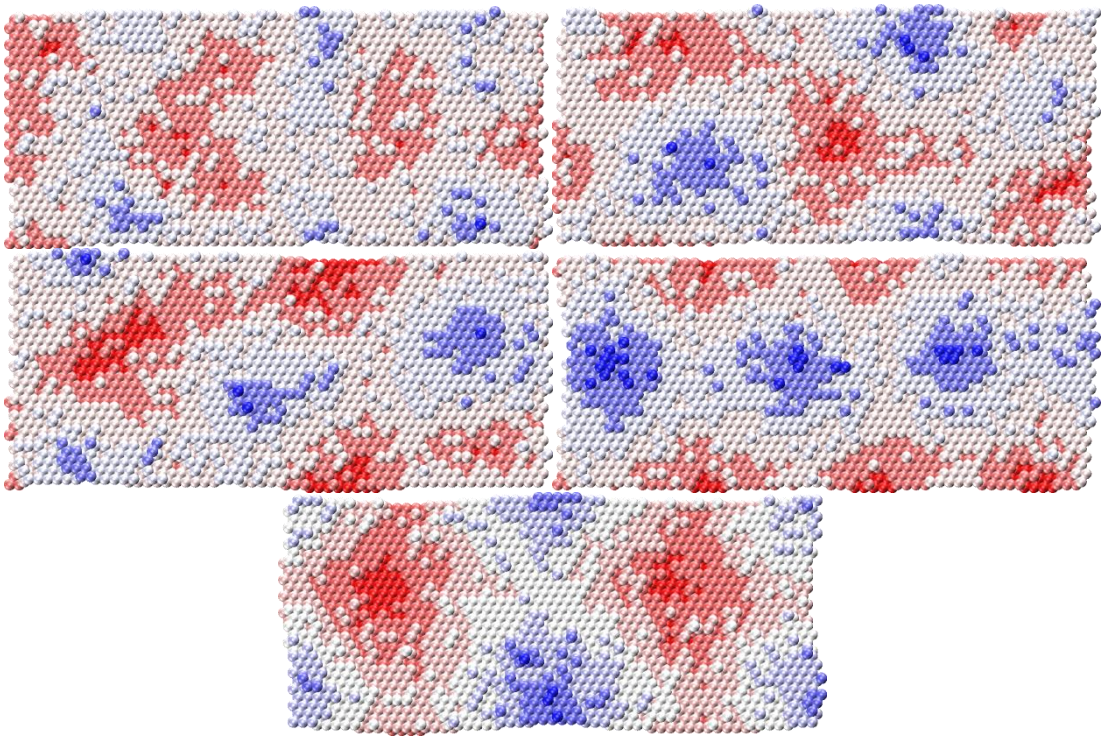


Figure 112: Final surface of Jv3 simulations after depositing the equivalent of 2.5 (top left), 5 (top right), 10 (middle left), 20 (middle right) and 40 (bottom) monolayers. Each uses a colour gradient for height that is relative to the range of height in each surface

In Figure 112, the same surfaces depicted in Figure 111 are shown but each surface has been given a colour gradient relative to the range of heights seen on that surface. The increasing surface roughness can be seen from the expansion of islands and vacancies. It can also be seen that the overall number of islands and vacancies decreases with the 2.5 monolayers simulation having numerous small islands and three large vacancies while the 40 monolayers simulation has two very large islands and two very large vacancies.

Figure 113 analyses the average surface height of the simulations at the various surface sizes detailed in Table 10 when run until the equivalent of 40 monolayers is deposited. It can be seen that the average surface height continued to follow the linear growth trend seen in Figure 103. All five surface sizes have very similar growth rates with only minor differences seen in the average surface heights even after the deposition of the equivalent of 40 monolayers. This is expected as the average rate of monolayer deposition over the course of the simulations should be the same and the statistics governing whether a deposition sticks to the surface or causes sputtering is also the same.

Figure 113: Average surface height during simulations of the deposition of the equivalent of 40 monolayers at various surface sizes

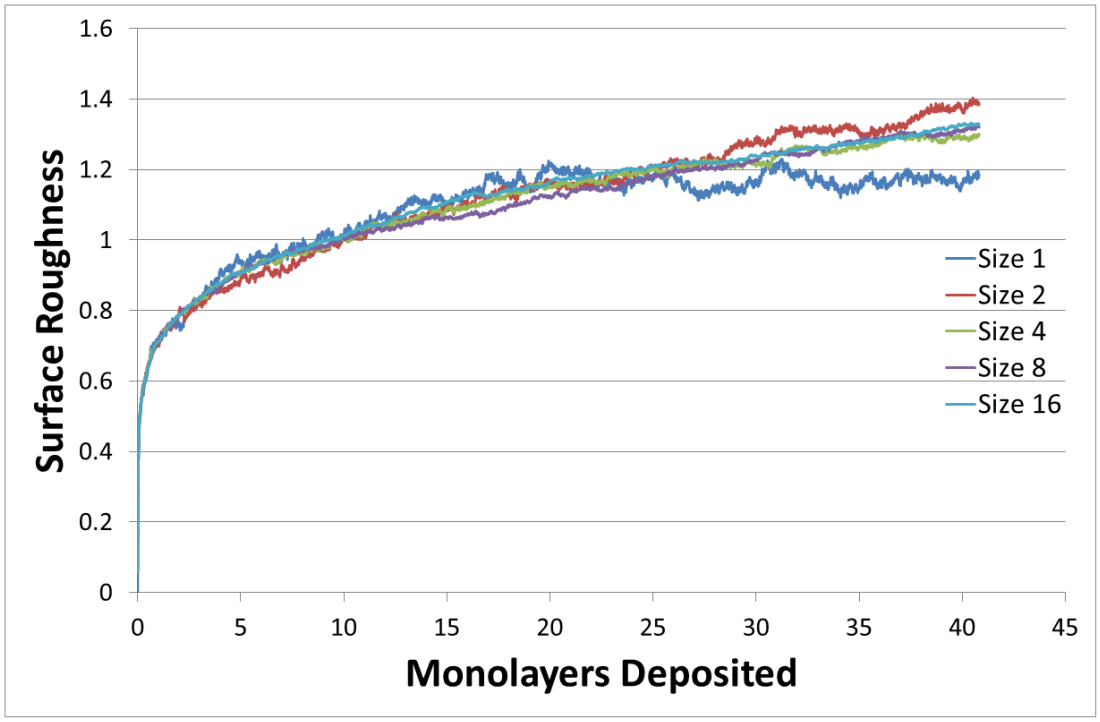


Figure 114: Surface roughness during simulations of the deposition of the equivalent of 40 monolayers at various surface sizes

Figure 114 shows the surface roughness throughout the deposition of the equivalent of 40 monolayers on the surface sizes detailed in Table 10. The Size 1 surface roughness is the same as the one analysed in Figures 109 and 110. Compared to the larger simulations, it appears to roughen quicker until it reaches the equivalent of 20 monolayers deposited when it levels off at a surface roughness equivalent to the height of 1.2 monolayers. The larger systems on the other hand, continue to roughen for the duration of the deposition. From this Figure alone, it is clear that the Size 1 surface (56 by 28 lattice sites) is not suitable for deposition simulations longer than 20 monolayers as it will suffer from finite size effects. However, when considering how drastically random fluctuations affected the surface roughness of the Size 1 surface in Figure 110, it is very likely that the Size 1 surface may suffer from finite size effects sooner with large differences seen in the surface roughness even at the equivalent of 5 monolayers deposited.

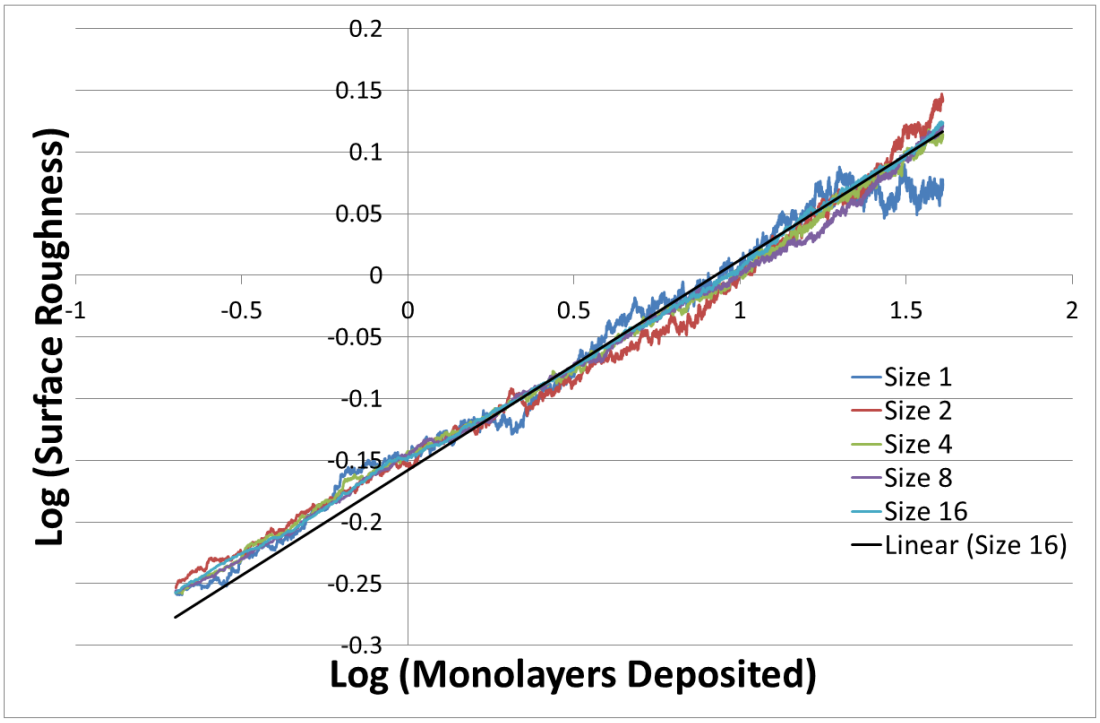


Figure 115: Log (Surface Roughness) against Log (Monolayers Deposited) at various surface sizes using a subset of the data displayed in Figure 114

In Figure 115, the log of the surface roughness in Figure 114 was plotted against the log of the monolayers deposited. The data shown begins from the equivalent of ~0.2 monolayers deposited as the data below 0.2 monolayers is sparse but would contribute to a large proportion of the plot, which means it would fluctuate rapidly and drastically skew any trendlines. As in Figure 105, linear trendlines were fit to the

logarithm of the surface roughness for each of the simulations. The gradients of the trendlines gradients are detailed in Table 13. Like in Table 11, all of the trendline gradients have a confidence interval taken from the standard deviation of gradients from trendlines fit to random subsets of the data.

Table 13: The gradient of the trendline fit to each simulation

Simulation	Trendline Gradient
Size 1	0.1405 ± 0.0050
Size 2	0.1852 ± 0.0045
Size 4	0.1672 ± 0.0025
Size 8	0.1677 ± 0.0032
Size 16	0.1706 ± 0.0017

The trendline gradient for the Size 1 surface is much lower than the other gradients. This is because the surface roughness levelled off, ending the apparent power law growth seen earlier in the simulation.

Looking back at Figure 115, the growth of the Size 1 surface appears similar to the Family-Vicsek scaling relation (81) for 2D ballistic deposition,

$$w(L,t) \sim L^{\alpha} f\left(\frac{t}{L^{\gamma}}\right) \tag{28}$$

where L is the system size, t is the time, w(L,t) is the surface thickness (equivalent to the surface roughness), α is the growth exponent, γ is the dynamic exponent and f(x) is a scaling function of the form

$$f(x) \sim \begin{cases} x^{\beta} & x \ll 1 \\ constant & x \gg 1 \end{cases} \tag{29}$$

where β is the roughness exponent. The dynamic exponent, γ, is related to the growth and roughness exponents as

$$\gamma = \frac{\alpha}{\beta} \tag{30}$$

From the equations, it can be seen that if the simulations in the current work follow this relation, the surface roughness will grow as a power law until it approaches a value that is dependent on the system size, at which point it will transition to a constant value, which is also dependent on the system size. This would suggest that given an infinite surface size, the surface roughness would grow infinitely following a power law. For finite system sizes, it would be possible to predict how many

monolayers need to be deposited before the surface roughness saturates and what value the surface roughness would be when it saturates.

However, as the Family-Vicsek scaling relation was for 2D ballistic deposition, equation 28 is likely an oversimplification of the relation for the simulations in the current work, which use 3D kMC. For example, while the Family-Vicsek uses the system size as a variable, the surface roughness of the simulations in the current work could be dependent on the length of the system in the x and y directions rather than the overall system size. Also, the time in the Family-Vicsek determines the number of depositions that have taken place, while in the simulations in the current work the number of depositions is also dependent on the rate of impacts and the sticking probability. As such, the time variable in equation 28 could be a function that is dependent on other factors, such as sticking probability and rate of impact, as well as time. From the MD results in the current work, it was seen that the sputter yield also affected the surface roughness so that could also be a potential factor.

Further simulations using a longer range could be carried out to verify that the other surface sizes saturate and determine their point of saturation. Simulations at a similar scope to the Size 1 simulations could be used to determine how certain simulation conditions affect the point of saturation.

The trendline fit through the Size 16 surface data matches the data well after ~0.4 but prior to that point, the surface appears to grow slower. This is also shown in Table 11 where the trendlines fit through the data shown in Figure 105 have lower gradients than the gradients in Table 13. 0.4 corresponds to the equivalent of approximately 2.5 monolayers. It should be noted that this is roughly the point where the average surface height reaches the equivalent of one monolayer. This would imply that while the rate of surface growth remains roughly constant around that point, the dynamics of the surface growth changes as the surface approaches the equivalent of one monolayer added, causing the surface to roughen more quickly as it grows.

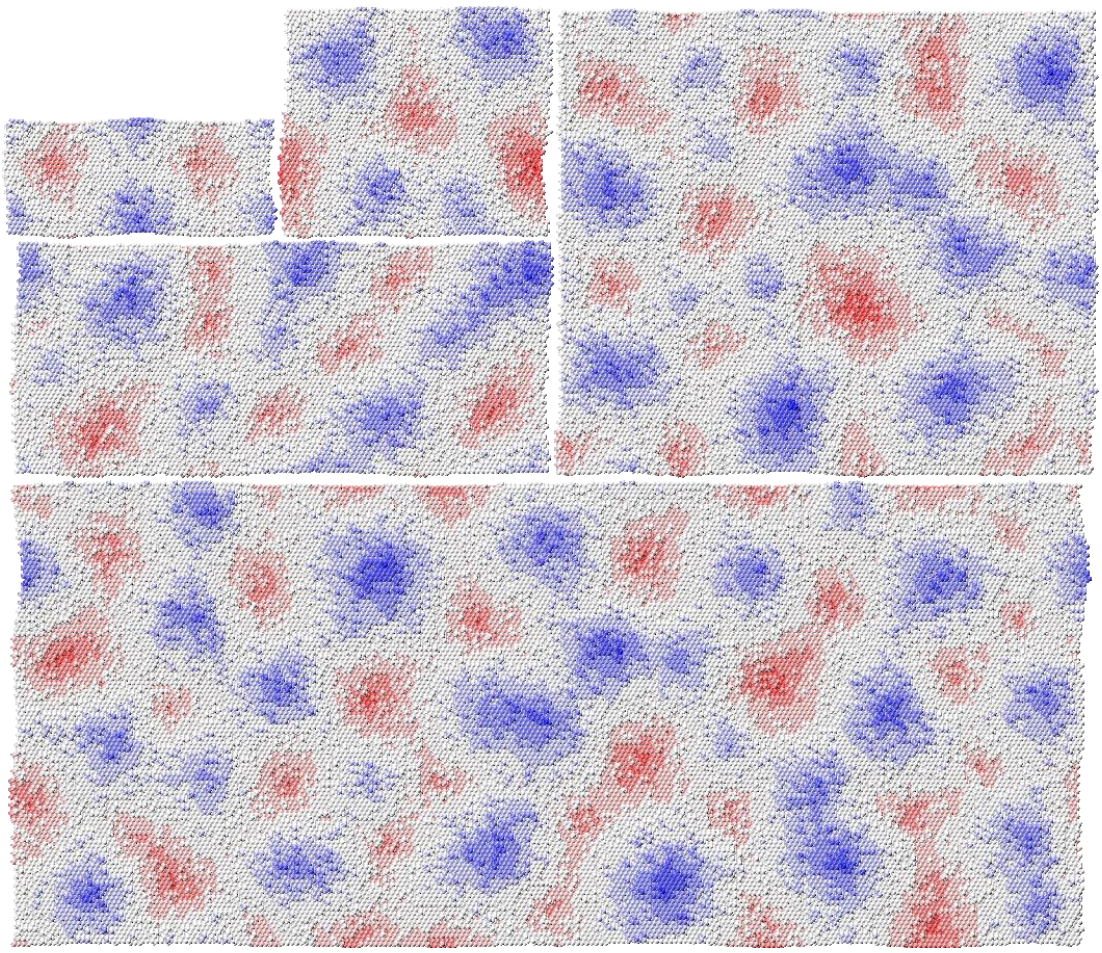


Figure 116: The final surface of Jv3 kMC simulations after deposition of 40 monolayers at surface sizes 56x28, 56x56, 112x56, 112x112 and 224x112, displayed to scale. A colour gradient denotes the height of atoms relative to the highest layer with that layer in blue. The gradient starts with the lowest layer in red and transitions from red to white and from white to blue at the layer in the middle of the range.

In Figure 116, the surfaces obtained at the end of the simulations analysed in Figures 113 and 114 are shown. The surfaces were produced by running Jv3 kMC simulations until the equivalent of 40 monolayers have been deposited. Comparing the Size 16 surface to the Size 1 surface, it is clear that, as concluded in Figure 114, the Size 1 surface is being heavily affected by finite size effects with very shallow vacancies and relatively small islands packed closer together than the islands and vacancies on the Size 16 surface. However, when comparing the Size 16 surface to the rest of the surfaces, it appears the Size 2 and Size 4 surfaces may also be affected by finite size effects, with the Size 2 surface appearing to have its islands and vacancies more packed than the Size 16 and Size 8 surfaces. The Size 4 surface also appears to be more packed, but only in the y-direction (up the page).

This could suggest that a surface used to simulate the evolution of the surface over the deposition of the equivalent of 40 monolayers must be larger than 56 lattice sites in both directions. The Size 8 surface, which uses 112 by 112 lattice sites, appears to be unaffected by finite size effects so, of the five surfaces used, this represents the minimum surface size that can simulate the deposition of the equivalent of 40 monolayers without being affected by finite size effects.

It was concluded that finite size effects constrain the minimum surface size required to accurately simulate deposition and the minimum required size is proportional to how much deposition occurs. This means that as larger amounts of deposition is required, so too are larger surfaces, as seen in Figure 114, where the Size 1 surface was shown to be unsuitable for simulations greater than 20 monolayers. This will cause the computational load required to grow nonlinearly ($\sim N^2$) with increasing number N of deposited monolayers.

Considering that with the constraints on processing power, a 40 monolayer deposition on a 14 by 14 surface took in the order of weeks in MD and that it was later determined that a 112 by 112 was the minimum size that could reliably simulate 40 monolayer deposition without finite size effects, it is clear that a MD simulation without finite size effects for the deposition of the equivalent of 40 monolayers is too computationally intensive to run, as it would have to run for several decades with the computational constraints this work had. However, the same simulation in kMC only took approximately an hour on a single core of an Intel Xeon E5420 @ 2.5 GHz, compared to the 12 cores of Intel Xeon X5650 @ 2.66 GHz used for MD.

5.3 Summary

In this chapter, 43 different parameter configurations were simulated between the three production versions of the kMC code and it was found that the third production version of the kMC code (Prdv3) was the best with the three best candidates for replicating the MD all using that code. This version coupled all surface relaxation steps to the previous impact, rather than allowing the surface relaxation steps to occur randomly on the surface. This implied that relaxation processes driven by the impact energy is more important than the surface diffusion that is thermally induced.

This is further reinforced by the three best simulations for replicating the MD (designated Dv3, Jv3 and Pv3) all being simulations that greatly restricted the distance that the surface relaxation steps occurred from the last impact site.

With the three best candidates selected, all three were compared to the MD to determine the best parameters to replicate the MD, looking at the average surface height and surface roughness for the simulations and then taking averages of multiple simulations and comparing the averages as well. From this, it was determined that the simulation designated Jv3 was the best set of parameters tested. The parameters for Jv3 had used 5 post-impact relaxation steps, which has fewer barriers impeding the movement of atoms than surface relaxation steps, instead of only 2 steps per impact like in simulation Dv3. Simulation Jv3 also used a less restrictive Schwoebel barrier for the surface relaxation steps, with the barrier to interlayer movement essentially being treated as equivalent to the energy barrier for moving away from 5 neighbouring atoms rather than the barrier for moving away from 10 neighbouring atoms like in simulation Pv3.

After choosing the best set of parameters, the surface growth was analysed using the kMC for surfaces from 56 by 28 sites to 224 by 112 sites and impacting them with the equivalent of 2.5-40 monolayers. It was observed that at 2.5 monolayers, the 56 by 28 surface worked well as a model system as it could be transplanted into larger surfaces without appearing incongruous and there were no finite size effects observed in any of the surfaces. However, finite size effects were observed at 40 monolayers for the simulations on the 56 by 28, 56 by 56 and 112 by 56 surfaces suggesting that the surface size required to adequately simulate growth on a surface grows with an increasing number of monolayers, causing the computational load required to grow nonlinearly.

Chapter 6 – Discussion

We set out to develop a model that can analyse surface growth during high energy deposition. In this work, a MD model was created, first using the Lennard-Jones potential then using the Sutton-Chen potential, and it has been used to analyse how surface growth is affected by various different conditions. A kMC model that tries to replicate the MD and model surface growth at more realistic sizes and timeframes was then developed.

6.1 Molecular Dynamics Review and suggestions for future work

In the simulations using the Sutton-Chen potential, it was seen that there were potentially still some finite sizes effects due to the propagation of the energy from an impact in the 28 by 28 surface if no thermostat was used. It would be worthwhile simulating impacts on 42 by 42 and 56 by 56 surfaces both with and without the thermostat. However, it is anticpated that these surfaces will still be affected by the finite size effects as the depth of the crystal being used (9 layers, 2 of which are fixed) is significantly smaller than the size of the surface. Without a thermostat, the small depth is unlikely to be sufficient to disspate the impact energy before it reaches the fixed layers and is reflected back towards the surface. If the finite size effects are caused by the depth, this would confirm that a thermostat would always be required when impacting crystals using that depth.

In the Berendsen thermostat, a time constant is used to determine the strength of the coupling of the thermostated layers to an external bath used to remove excess heat. Although the time constant used appears to be reasonable for the simulations in this work, the strength of the coupling required and consequently, the value of the time constant could be investigated. It could also be worthwhile to investigate other thermostats such as the Nose-Hoover thermostat to validate the suitability of the Berendsen thermostat for the simulations in this work.

During this work, the effects the impact delay had on a surface of 14 atoms by 14 atoms was analysed and it was concluded that after each impact, delaying the next

impact by 4 picoseconds with the thermostat being active was the shortest delay that would produce realistic results and this delay was then used for all surfaces. However, the effect the length of the impact delay has on larger surfaces could be investigated. It is expected that larger surfaces could still produce realistic results with shorter delays but it is unknown if the length of the delay needed is inversely proportional to the surface size, such that the 28 by 28 surface, a surface that is four times larger, could reasonably use a 1 ps delay, a delay that is four times shorter. The length of the surface's shortest side would likely also be a factor in how much the delay could be shortened. If the delay can be shortened significantly for increasing size, larger surfaces and more impacts on the current surfaces would become more feasible to simulate, allowing for a greater understanding of how the surface evolves as it is impacted in MD.

When looking at the polar and azimuthal angles, it was noted that at higher polar angles, random azimuthal angles produced more surface erosion and surface roughness than a fixed azimuthal angle. This effect could be further investigated by running more simulations that use the same azimuthal angle throughout but varying the angle used between simulations. It is expected that the erosion seen for random azimuthal angles is roughly equivalent to the average erosion seen across a range of different azimuthal angles. However, it is also possible, though believed to be unlikely, that the simulations using random azimuthal angles will cause more surface erosion than simulations using the same azimuthal angle, regardless of the azimuthal angle chosen.

The effects on the surface could also be further analysed by looking at surface structures other than (111) such as (100) and (110). As these surfaces are less tightly packed than (111), they could be more greatly affected by erosion after an impact as they are less strongly held to the surface by the surrounding atoms or they could instead be more able to absorb the energy of the impact as there is more room for the atoms to move around when impacted.

Throughout this work, the scale of the simulations using the Molecular Dynamics code was restricted by the number of computer processors that could be used and how effectively the processors were being used. An example of this can be seen in 3.2.3 when it was discovered that adding one extra flag when compiling the code optimised the program produced, making it almost twice as efficient during calculations and making it feasible to impact the 56 by 28 surface 1000 times in a

single simulation. It is likely that the production MD code can be further optimised. Another consideration would be how the code handles parallelisation as currently the code uses OpenMP to parallelise the processing of some calculations. The usage of parallelisation could likely be improved allowing faster processing with the same resources. OpenMP also restricts the code to running on one node (12 cores on ARCHIE-WeSt at the time the simulations in this work were carried out) so the scale of the simulations could be increased on a more powerful computing node or on a node with more cores available. If the scale of the simulations can be increased, that would allow larger surfaces and more impacts to occur, providing simulations that would have more detail. This extra detail could then be used to further improve the accuracy of the kMC code.

Another option for better parallelisation implementation would be rewriting the code to use MPi to parallelise sections of the code. MPi would allow the code to use more than one computer node during a simulation but it is designed for each parallel process to use its own private data and not share any data with the other processes. This may make MPi unsuitable for certain calculations that are currently parallelised with OpenMP. It may be possible rewrite the code to use a hybrid approach with OpenMP and MPi both being used to parallelise different sections of the code to make the most optimum use of resources.

As mentioned in sections 1.4, 3.1 and 3.2, previous simulation work has been done to analyse surface evolution such as the growth of a carbon film and the deposition of metal atoms upon a metal surface. As the work in this thesis is on a nickel surface with nickel atoms, the closest comparison was with the paper by Hanson et al. (35). However this paper only analysed the sticking probability and sputter yield after 50 depositions on a small surface and due to unknown factors of their work, their results could not be replicated by this work. The work in this thesis simulates deposition on a much larger scale, with depositions on surfaces up to 8 times larger and up to 4,000 depositions, and this work also analyses how the surface height and roughness evolve during the depositions. An advantage of the larger scale used in this work is that it aims to let the surface evolve sufficiently so that sputter yield and the sticking probability are more consistent. This allows them to be used reliably in kMC modelling.

Future work should also look into developing the model further with simulations using other materials for the surface. These materials could be metals, such as

titanium or aluminium, or alloys and oxides. Other ways to advance the model is to simulate depositions with air, which is more suited for hypersonic applications. Furthermore, simulations for spacecraft re-entry will need to simulate reactions between the surface and the ionized air particles that can be encountered.

6.2 Kinetic Monte Carlo Review and suggestions for future work

With the kinetic Monte Carlo code, we aimed to replicate the behaviour seen in the MD results and use the code to simulate larger sizes on longer timeframes. We were able to optimize our kMC code to replicate some of the behaviour seen in MD and analysed surface growth using the most optimal parameters simulated. We found that as more monolayers are deposited, larger surfaces are needed to avoid finite size effects.

Looking at the kMC code, there were some issues noted with unrealistic overhangs appearing despite the site optimisation algorithm being designed to avoid them. We believe the reason that the algorithm occasionally failed is that the site selected originally was occasionally at the boundary of consecutive step changes while the algorithm was only designed to optimize the site chosen when at the boundary of a single step change. This means that when selecting a site, the algorithm was selecting sites in between step changes rather than picking the atom at the opposite boundary of the change, creating unrealistic overhangs. To make the code capable of handling multiple step changes, the site optimisation algorithm could be tweaked to loop endlessly until the site being selected stops changing.

While we made the decision to stop optimising the parameters of the kMC code with the selection of simulation Jv3 due to time constraints, it is expected that further, more subtle optimisations could be made to the parameters used.

A change to the kMC code that could be worth investigating is changing the method of surface relaxations. Currently, the code determines whether to move the representation of an atom to a random nearby site using the total number of interactions at both sites. The interaction counter algorithm could be altered to apply directionality to the number of interactions, using the number of interactions in a

direction to influence the destination of the moving atom. This would introduce the directionality of the attractive forces experienced by the atoms in MD.

The kMC simulation with the deposition of the equivalent of 40 monolayers on a 224 by 112 lattice site surface used a small amount of computational power (~1 hour on 1 core of an Intel Xeon E5420 @ 2.5 GHz) when compared to the limit of the MD simulations (two weeks on 12 cores of Intel Xeon X5650 @ 2.66 GHz). Given this, it would be possible to simulate much larger surfaces with a greater number of monolayer depositions. This allows a more in-depth analysis of its scaling properties, which is important for applications to real-world applications.

Surface evolution during deposition using kMC has been studied for a number of different systems, including a large number that analyse homoepitaxy of metals on various crystallographic surfaces (82). They show that during multilayer growth the surface will form mounds. Two important features that affect the shape and roughness of these mounds and how the surface grows are the Schwoebel barrier and diffusion events. The work in this thesis uses kMC to model the surface evolution including surface erosion so it cannot be directly compared to homoepitaxy. However, the formation of mounds and the importance of diffusion and the Schwoebel barrier are also observed in our work. Further developments to the kMC model developed in this work could allow observations of how the model contributes to the understanding of surface evolution.

6.3 Conclusions

In this work, we:

- Developed an MD code to analyse surface growth
- Discovered finite size effects at small system sizes that could be resolved by adding a thermostat to sub-surface layers
- Found that the surface undergoes an equilibration period during the initial impacts before experiencing growth proportional to the number of monolayers deposited
- Showed that the surface began eroding at a polar angle of 50°

- Found substantial differences in the surface height and roughness when investigating the effect of the azimuthal angle at polar angles of 70° and 80°
- Developed a kMC code
- Optimised it to replicate the MD as much as possible
- Analysed surface growth on larger surfaces at longer timescales
- Found finite size effects proportional to the number of monolayers deposited

If this work were to be developed further, we suggest:

- Trying larger surfaces without a thermostat in MD
- Investigating the strength of the coupling in the thermostat in MD
- Investigating shorter delays for larger systems in MD
- Simulating a range of different azimuthal angles in MD
- Simulating different surface structures in MD
- Attempting to further optimize the MD code and parallelization used
- Making the kMC site optimization algorithm run recursively
- Further optimizing the kMC parameters
- Simulating even larger sizes over longer timeframes with the kMC

Works Cited

1. **Bird, Graeme A.** *Molecular Gas Dynamics and the Direct Simulation of Gas Flows.* 2nd. Oxford : Clarendon Press, 1994. ISBN-13: 978-0198561958.

2. *Open-Source Direct Simulation Monte Carlo Chemistry Modeling for Hypersonic Flows.* **Scanlon, Thomas J., et al., et al.** 6, s.l. : Aerospace Research Central, June 2015, AIAA Journal, Vol. 53, pp. 1670-1680. doi: 10.2514/1.J053370.

3. *An open source, parallel DSMC code for rarefied gas flows in arbitrary geometries.* **Scanlon, T. J., et al., et al.** 10, s.l. : Elsevier, December 2010, Computers & Fluids, Vol. 39, pp. 2078-2089. doi: 10.1016/j.compfluid.2010.07.014.

4. *The Q-K model for gas-phase chemical reaction rates.* **Bird, Graeme A.** 10, s.l. : American Institute of Physics, October 2011, Physics of Fluids, Vol. 23, p. 106101. doi: 10.1063/1.3650424.

5. *Modeling backward chemical rate processes in the direct simulation Monte Carlo method.* **Boyd, Iain D.** 12, s.l. : American Institute of Physics, December 2007, Physics of Fluids, Vol. 19, p. 126103. doi: 10.1063/1.2825038.

6. **Bird, Graeme A.** Simulation of Multi-Dimensional and Chemically Reacting Flows (Past Space Shuttle Orbiter). [ed.] R. Campargue. *Rarefied Gas Dynamics.* s.l. : Commissariat à l'énergie atomique, 1979, Vol. 1, pp. 365-388.

7. *Chemically reacting hypersonic flows over 3D cavities: Flowfield structure characterisation.* **Palharini, Rodrigo C., Scanlon, Thomas J. and White, Craig.** s.l. : Elsevier, March 2018, Computers and Fluids, Vol. 165, pp. 173-187. doi: 10.1016/j.compfluid.2018.01.029.

8. *dsmcFoam+: An OpenFOAM based direct simulation Monte Carlo solver.* **White, C., et al., et al.** s.l. : Elsevier, March 2018, Computer Physics Communications, Vol. 224, pp. 22-43. doi: 10.1016/j.cpc.2017.09.030.

9. **Orava, J., Kohoutek, T. and Wagner, T.** 9 - Deposition techniques for chalcogenide thin films. [ed.] Jean-Luc Adam and Xianghua Zhang. *Chalcogenide Glasses: Preparation, Properties and Applications.* s.l. : Woodhead Publishing, 2014, pp. 255-209.

10. *Study of mechanical behavior of BNNT-reinforced aluminum composites using molecular dynamics simulations.* **Cong, Ziyu and Lee, Seungjun.** s.l. : Elsevier, 15 June 2018, Composite Structures, Vol. 194, pp. 80-86. doi: 10.1016/j.compstruct.2018.03.103.

11. *New empirical approach for the structure and energy of covalent systems.* **Tersoff, J.** 12, s.l. : American Physical Society, April 1988, Physical Review B, Vol. 37, pp. 6991-7000. doi: 10.1103/PhysRevA.8.1504.

12. *Compressive behavior of CNT-reinforced aluminum composites using molecular dynamics.* **Silvestre, Nuno, Faria, Bruno and Canongia Lopes, José N.** s.l. : Elsevier, January 2014, Composites Science and Technology, Vol. 90, pp. 16-24. doi: 10.1016/j.compscitech.2013.09.027.

13. *Molecular dynamics simulation of size and strain rate dependent mechanical response of FCC metallic nanowires.* **Koh, S J A and Lee, H P.** 14, s.l. : Institute of Physics Publishing, July 2006, Nanotechnology, Vol. 17, pp. 3451-3467. doi: 10.1088/0957-4484/17/14/018.

14. *Molecular dynamics studies of CNT-reinforced aluminum composites under uniaxial tensile loading.* **Choi, Bong Kyu, Yoon, Gil Ho and Lee, Seungjin.** s.l. : Elsevier, April 2016, Composites Part B, Vol. 91, pp. 119-125. doi: 10.1016/j.compositesb.2015.12.031.

15. *Mechanical characteristics of CNT-reinforced metallic glass nanocomposites by molecular dynamics simulations.* **Rezaei, Reza, et al., et al.** s.l. : Elsevier, June 2016, Computational Materials Science, Vol. 119, pp. 19-26. doi: 10.1016/j.commatsci.2016.03.036.

16. *Investigation of mechanical properties of CNT reinforced epoxy nanocomposite: a molecular dynamic simulations.* **Dikshit, Mithilesh K. and Engle, Pravin E.** 1, s.l. : Peter the Great St. Petersburg Polytechnic University, 2018, Materials Physics and Mechanics, Vol. 137, pp. 7-15. doi: 10.18720/MPM.3712018_2.

17. *Mechanical Properties of Bamboo-like Boron Nitride Nanotubes by In Situ TEM andMDSimulations: Strengthening Effect of Interlocked Joint Interfaces.* **Tang, Dai-Ming, et al., et al.** 9, s.l. : American Chemical Society, September 2011, ACS Nano, Vol. 5, pp. 7362-7368. doi: 10.1021/nn202283a.

18. *Long-range Finnis-Sinclair potentials.* **Sutton, A. P. and Chen, J.** 3, s.l. : Taylor & Francis, 1990, Philosophical Magazine Letters, Vol. 61, pp. 139-146. doi: 10.1080/09500839008206493.

19. *Long-range Finnis-Sinclair potentials for f.c.c metallic alloys.* **Rafii-Tabar, H. and Sutton, A. P.** 4, s.l. : Taylor & Francis, 21 Janurary 1991, Philosophical Magazine Letters, Vol. 63, pp. 217-224. doi: 10.1080/09500839108205994.

20. *Temperature- and surface orientation-dependent calculated vacancy formation energy for Cu nanocubes.* **van der Walt, C., Terblans, J. J. and Swart, H. C.** 1, s.l. : Springer, January 2018, Journal of Materials Science, Vol. 53, pp. 814-823. doi: 10.1007/s10853-017-1502-y.

21. *Global minima for transition metal clusters described by Sutton-Chen potentials.* **Doye, Jonathan P. K. and Wales, David J.** 7, s.l. : Royal Society of Chemistry, 1998, New Journal of Chemistry, Vol. 22, pp. 733-744. doi: 10.1039/A709249K.

22. *Molecular dynamics simulation of crack propagation in fcc materials containing clusters of impurities.* **Rafii-Tabar, H., et al., et al.** 3, s.l. : Elsevier, March 2006, Mechanics of Materials, Vol. 38, pp. 243-252. doi: 10.1016/j.mechmat.2005.06.006.

23. *Planar Molecular Dynamics Simulation of Metallic Nanoparticles Manipulation.* **Mahboobi, S. H., et al., et al.** Arlington, TX, USA : IEEE, 2008. 2008 8th IEEE Conference on Nanotechnology. pp. 163-166. doi: 10.1109/NANO.2008.55.

24. *An investigation on the validity of Cauchy–Born hypothesis using Sutton-Chen many-body potential.* **Khoei, A. R., et al., et al.** 3, s.l. : Elsevier, January 2009, Computational Materials Science, Vol. 44, pp. 999-1006. doi: 10.1016/j.commatsci.2008.07.022.

25. *Calculation of Mechanical, Thermodynamic and Transport Properties of Metalic Glass formers.* **Çağin, Tahir , et al., et al.** s.l. : Cambridge University Press, 1998. MRS Proceedings. Vol. 554, pp. 43-48. doi: 10.1557/PROC-554-43.

26. *Melting and crystallization in Ni nanoclusters: The mesoscale regime.* **Qi, Yue, et al., et al.** 1, s.l. : American Institute of Physics, July 2001, The Journal of Chemical Physics, Vol. 115, pp. 385-394. doi: 10.1063/1.1373664.

27. *Size effects on the melting of nickel nanowires: a molecular dynamics study.* **Wen, Yu-Hua, et al., et al.** 1, s.l. : Elsevier, October 2004, Physica E, Vol. 25, pp. 47-54. doi: 10.1016/j.physe.2004.06.048.

28. *Molecular dynamics study of microscopic structures, phase transitions and dynamic crystallization in Ni nanoparticles.* **Nguyen, Trong Dung, Nguyen, Chinh Cuong and Tran, Vinh Hung.** 41, s.l. : Royal Society of Chemistry, May 2017, RSC Advances, Vol. 7, pp. 25406-25413. doi: 10.1039/C6RA27841H.

29. *The melting mechanism in binary Pd0.25Ni0.75 nanoparticles: molecular dynamics simulations.* **Domekeli, U., et al., et al.** 5, s.l. : Taylor & Francis, 2018, Philosophical Magazine, Vol. 98, pp. 371-387. doi: 10.1080/14786435.2017.1406671.

30. *Liquid properties of Pd–Ni alloys.* **Özdeimir Kart, S., et al., et al.** 1, s.l. : Elsevier, July 2004, Journal of Non-Crystalline Solids, Vol. 337, pp. 101-108. doi: 10.1016/j.jnoncrysol.2004.03.121.

31. *Molecular Dynamics of Free and Graphite-Supported Pt-Pd Nanoparticles.* **Fernández-Navarro, Carlos and Mejía-Rosales, Sergio.** 4, s.l. : Scientific Research Publishing, November 2013, Advances in Nanoparticles, Vol. 2, pp. 323-328. doi: 10.4236/anp.2013.24044 .

32. *Molecular dynamics simulation study of the melting of Pd-Pt nanoclusters.* **Sankaranarayanan, Subramanian K. R. S., Bhethanabotla, Venkat R. and Joseph, Babu.** 19, s.l. : American Physical Society, May 2005, Physical Review B, Vol. 71, p. 195415. doi: 10.1103/PhysRevB.71.195415.

33. *Molecular-dynamics simulations of glass formation and crystallization in binary liquid metals: Cu-Ag and Cu-Ni.* **Qi, Yue, et al., et al.** 5, s.l. : American Physical Society, February 1999, Physical Review B, Vol. 59, pp. 3527-3533. doi:10.1103/PhysRevB.59.3527.

34. *Molecular-dynamics simulations of binary Pd-Si metal alloys: Glass formation, crystallisation and cluster properties.* **Faruq, Muhammad, Villesuzanne, Antoine and Shao, Guosheng.** s.l. : Elsevier, May 2018, Journal of Non-Crystalline Solids, Vol. 487, pp. 72-86. doi: 10.1016/j.jnoncrysol.2018.02.016.

35. *Molecular dynamics simulations of ion self-sputtering of Ni and Al surfaces.* **Hanson, D. E., et al., et al.** 3, s.l. : AVS: Science & Technology of Materials, Interfaces, and Processing, May 2001, Journal of Vacuum Science and Technology A, Vol. 19, pp. 820-825. doi: 10.1116/1.1365134.

36. *Theoretical studies on an empirical formula for sputtering yield at normal incidence.* **Yamamura, Y., Matsunami, N. and Itoh, N.** 1-2, s.l. : Taylor & Francis, 1983, Radiation Effects, Vol. 71, pp. 65-86. doi: 10.1080/00337578308218604.

37. *Atomistic modeling of large-scale metal film growth fronts.* **Hansen, U., Vogl, P. and Fiorentini, V.** 12, s.l. : American Physical Society, 15 March 1999, Physical Review B, Vol. 59, pp. R7856-R7859. doi: 10.1103/PhysRevB.59.R7856.

38. *Sputtering Yield Measurements with Low-Energy Metal Ion Beams.* **Hayward, W. H. and Wolter, A. R.** 7, s.l. : American Institute of Physics, June 1969, Journal of Applied Physics, Vol. 40, pp. 2911-2916. doi: 10.1063/1.1658100.

39. *Low energy selfsputtering yields of nickel.* **Hechtl, E., Bay, H. L. and Bohdansky, J.** 2, s.l. : Springer-Verlag, June 1978, Applied Physics A, Vol. 16, pp. 147-150. doi: 10.1007/BF00930378.

40. *Reflection and self-sputtering of nickel at oblique angles of ion incidence.* **Hechtl, E., Eckstein, W. and Roth, J.** 1-4, s.l. : Elsevier, May 1994, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms, Vol. 90, pp. 505-508. doi: 10.1016/0168-583X(94)95603-0.

41. *Direct collection of some metal ions in an electromagnetic isotope separator and related surface effects.* **Fontell, A. and Arminen, E.** 21, s.l. : Canadian Science Publishing, 1969, Canadian Journal of Physics, Vol. 47, pp. 2405-2414. doi: 10.1139/p69-293.

42. *Modeling of metal thin film growth: Linking angstrom-scale molecular dynamics results to micron-scale film topographies.* **Hansen, U., Rodgers, S. and Jensen, K. F.** 4, s.l. : American Physical Society, 15 July 2000, Physical Review B, Vol. 62, pp. 2869-2878. doi: 10.1103/PhysRevB.62.2869.

43. *Multiscale modeling, simulations, and experiments of coating growth on nanofibers. Part I. Sputtering.* **Buldum, A., et al., et al.** 04, s.l. : American Institute

of Physics, August 2005, Journal of Applied Physics, Vol. 98, p. 044303. doi: 10.1063/1.2007848.

44. *Multiscale modeling, simulations, and experiments of coating growth on nanofibers. Part II. Deposition.* **Buldum, A., et al., et al.** 4, s.l. : American Institute of Physics, August 2005, Journal of Applied Physics, Vol. 98, p. 044304. doi: 10.1063/1.2007849.

45. *Reaction rates for ionized physical vapor deposition modeling from molecular-dynamics calculations: Effect of surface roughness.* **Hansen, U. and Kersch, A.** 20, s.l. : American Physical Society, November 1999, Physical Review B, Vol. 60, pp. 14 417-14 421. doi: 10.1103/PhysRevB.60.14417.

46. *Multiscale approaches for metal thin film growth.* **Vogl, P., Hansen, U. and Fiorentini, V.** 1-2, s.l. : Elsevier, May 2002, Computational Materials Science, Vol. 24, pp. 58-65. doi: 10.1016/S0927-0256(02)00164-7.

47. *Three-dimensional spatiokinetic distributions of sputtered and scattered products of Ar+ and Cu+ impacts onto the Cu surface: molecular dynamics simulations.* **Abrams, Cameron F. and Graves, David B.** 5, s.l. : IEEE, October 1999, IEEE Transactions on Plasma Science, Vol. 27, pp. 1426-1432. doi: 10.1109/27.799821.

48. *Cu sputtering and deposition by off-normal, near-threshold Cu+ bombardment: Molecular dynamics simulations.* **Abrams, Cameron F. and Graves, David B.** 4, s.l. : American Institute of Physics, 15 August 1999, Journal of Applied Physics, Vol. 86, pp. 2263-2267. doi: 10.1063/1.371040.

49. *Molecular dynamics-based ion-surface interaction models for ionized physical vapor deposition feature scale simulations.* **Coronell, Daniel G., et al., et al.** 26, s.l. : American Institute of Physics, 28 December 1998, Applied Physics Letters, Vol. 73, pp. 3860-3862. doi: 10.1063/1.122917.

50. *Molecular dynamics study on low-energy sputtering properties of MgO surfaces.* **Ahn, Hyo-Shin, et al., et al.** 7, s.l. : American Institute of Physics, April 2008, Journal of Applied Physics, Vol. 103, p. 073518. doi: 10.1063/1.2899182.

51. *Monte Carlo simulations of MgO and Mg(OH)2 thin films sputtering yields by noble-gas ion bombardment in plasma display panel PDP.* **El Marsi, M., et al., et al.** s.l. : Elsevier, September 2018, Nuclear Instruments and Methods in Physics

Research Section B: Beam Interactions with Materials and Atoms, Vol. 430, pp. 72-78. doi: 10.1016/j.nimb.2018.05.046.

52. *Net sputtering rate due to hot ions in a Ne-Xe discharge gas bombarding an MgO layer.* **Ho, S., et al., et al.** 8, s.l. : American Institute of Physics, April 2011, Journal of Applied Physics, Vol. 109, p. 084908. doi: 10.1063/1.3554687.

53. *Molecular dynamics simulation study of the growth of a rough amorphous carbon film by the grazing incidence of energetic carbon atoms.* **Joe, Minwoong, et al., et al.** 2, s.l. : Elsevier Ltd., February 2012, Carbon, Vol. 50, pp. 404-410. DOI: 10.1016/j.carbon.2011.08.053.

54. *Molecular dynamics with coupling to an external bath.* **Berendsen, H. J. C., et al., et al.** 8, s.l. : American Institute of Physics, 15 October 1984, Journal of Chemical Physics, Vol. 81, pp. 3684-3690. doi: 10.1063/1.448118.

55. *Structure evolution and stress transition in diamond-like carbon films by glancing angle deposition.* **Lei, Yu, et al., et al.** s.l. : Elsevier B.V., 15 June 2019, Applied Surface Science, Vol. 479, pp. 12-19. doi: 10.1016/j.apsusc.2019.02.063.

56. *Molecular-Dynamics Simulation of the Growth of Diamondlike Films by Energetic Carbon-Atom Beams.* **Kaukonen, H.-P. and Nieminen, R. M.** 5, s.l. : American Physical Society, 3 February 1992, Physical Review Letters, Vol. 68, pp. 620-623. doi: 10.1103/PhysRevLett.68.620.

57. *Molecular-Dynamics Study of the Fundamental Processes Involved in Subplantation of Diamondlike Carbon.* **Uhlmann, S., Frauenheim, Th. and Lifshitz, Y.** 3, s.l. : American Physical Society, 20 July 1998, Physical Review Letters, Vol. 81, pp. 641-644. doi: 10.1103/PhysRevLett.81.641.

58. *Molecular-dynamics simulations of steady-state growth of ion-deposited tetrahedral amorphous carbon films.* **Jäger, H. U. and Albe, K.** 2, s.l. : American Institute of Physics, 15 July 2000, Journal of Applied Physics, Vol. 88, pp. 1129-1135. doi: 10.1063/1.373787.

59. *Lattice kinetic Monte Carlo simulation study of the early stages of epitaxial GaN(0001) growth.* **Chugh, Manjusha and Ranganathan, Madhav.** s.l. : Elsevier, 15 November 2017, Applied Surface Science, Vol. 422, pp. 1120-1128. doi: 10.1016/j.apsusc.2017.06.067.

60. *A kinetic Monte Carlo method for the atomic-scale simulation of chemical vapor deposition: Application to diamond.* **Battaile, C. C., Srolovitz, D. J. and Butler, J. E.** 12, s.l. : American Institute of Physics, December 1997, Journal of Applied Physics, Vol. 82, pp. 6293-6300. doi: 10.1063/1.366532.

61. *Etching effects during the chemical vapor deposition of (100) diamond.* **Battaile, C. C., et al., et al.** 9, s.l. : American Institute of Physics, September 1999, Journal of Chemical Physics, Vol. 111, pp. 4291-4299. doi: 10.1063/1.479727.

62. *Simulations of chemical vapor deposition diamond film growth using a kinetic Monte Carlo model.* **May, P. W., et al., et al.** 1, s.l. : American Institute of Physics, July 2010, Journal of Applied Physics, Vol. 108, p. 014905. doi: 10.1063/1.3437647.

63. *Kinetic Monte Carlo simulations of CVD diamond growth—Interlay among growth, etching, and migration.* **Netto, Armando and Frenklach, Michael.** 10, s.l. : Elsevier, October 2005, Diamond & Related Materials, Vol. 14, pp. 1630-1646. doi: 10.1016/j.diamond.2005.05.009.

64. *Kinetic Monte Carlo simulations of water ice porosity: extrapolations of deposition parameters from the laboratory to interstellar space.* **Clements, Aspen R., et al., et al.** 8, s.l. : Royal Society of Chemistry, February 2018, Physical Chemistry Chemical Physics, Vol. 20, pp. 5553-5568. doi:10.1039/C7CP05966C.

65. *Three-dimensional, Off-lattice Monte Carlo Kinetics Simulations of Interstellar Grain Chemistry and Ice Structure.* **Garrod, Robin T.** 2, s.l. : The American Astronomical Society, December 2013, The Astrophysical Journal, Vol. 778, p. 158. doi: 10.1088/0004-637X/778/2/158.

66. *H2O Condensation Coefficient and Refractive Index for Vapor-Deposited Ice from Molecular Beam and Optical Interference Measurements.* **Brown, D. E., et al., et al.** 12, s.l. : American Chemical Society, 21 March 1996, Journal of Physical Chemistry, Vol. 100. doi: 10.1021/jp952547j .

67. *Long time scale kinetic Monte Carlo simulations without lattice approximation and predefined event table.* **Henkelman, Graeme and Jónsson, Hannes.** 21, s.l. : American Institute of Physics, December 2001, Journal of Chemical Physics, Vol. 115, pp. 9657-9666. doi: 10.1063/1.1415500.

68. *A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives.* **Henkelman, Graeme and Jónsson, Hannes.** 15, s.l. : American Institute of Physics, October 1999, Journal of Chemical Physics, Vol. 111, pp. 7010-7022. doi:10.1063/1.480097.

69. *Following atomistic kinetics on experimental timescales with the kinetic Activation–Relaxation Technique.* **Mousseau, Normand, et al., et al.** s.l. : Elsevier B.V., 1 April 2015, Computational Materials Science, Vol. 100 Part B, pp. 111-123. doi: 10.1016/j.commatsci.2014.11.047.

70. *A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters.* **Swope, William C., et al., et al.** 1, s.l. : American Institute of Physics, 1 January 1982, Journal of Chemical Physics, Vol. 76, pp. 637-649. doi: 10.1063/1.442716.

71. *A simple empirical N-body potential for transition metals.* **Finnis, M. W. and Sinclair, J. E.** 1, s.l. : Taylor & Francis, 1984, Philosophical Magazine A, Vol. 50, pp. 45-55. doi: 10.1080/01418618408244210.

72. *Comyuter "Exyeriments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules.* **Verlet, Loup.** 1, s.l. : American Physical Society, July 1697, Physical Review, Vol. 159, pp. 98-103. doi: 10.1103/PhysRev.159.98.

73. **Hockney, R. W. and Eastwood, J. W.** *Computer Simulation Using Particles.* New York : McGraw-Hill, 1981. ISBN-13: 978-0070291089.

74. *New method for searching for neighbors in molecular dynamics computations.* **Quentrec, B. and Brot, C.** 3, s.l. : Elsevier, November 1973, Journal of Computational Physics, Vol. 13, pp. 430-432. doi: 10.1016/0021-9991(73)90046-6.

75. *Communication: Shifted forces in molecular dynamics.* **Toxvaerd, Søren and Dyre, Jeppe C.** 8, s.l. : American Institute of Physics, February 2011, Journal of Chemical Physics, Vol. 134, p. 081102. doi: 10.1063/1.3558787.

76. *Numerical Experiments on the Stochastic Behavior of a Lennard-Jones Gas System.* **Stoddard, Spotswood D. and Ford, Joseph.** 3, s.l. : American Physical Society, September 1973, Physical Review A, Vol. 8. doi: 10.1103/PhysRevA.8.1504.

77. *Dynamics of resonant charge transfer in low-energy alkali-metal-ion scattering.* **Kimmel, G. A. and Cooper, B. H.** 16, s.l. : American Physical Society, October 1993, Physical Review B, Vol. 48, pp. 12164-12177. doi: 10.1103/PhysRevB.48.12164.

78. *Calculation of the Lennard-Jones n-m Potential Energy Parameters for Metals.* **Zhen, Shu and Davies, G. J.** 2, s.l. : Wiley, August 1983, physica status solidi (a), Vol. 78, pp. 595-605. doi: 10.1002/pssa.2210780226.

79. *ATOMISTIC SIMULATIONS OF PLANAR DEFECTS IN ORDERED NI-AL ALLOYS.* **Srolovitz, D. J., Chen, S. P. and Voter, A. F.** s.l. : Springer-Verlag, 1987, Journal of Metals, Vol. 39.

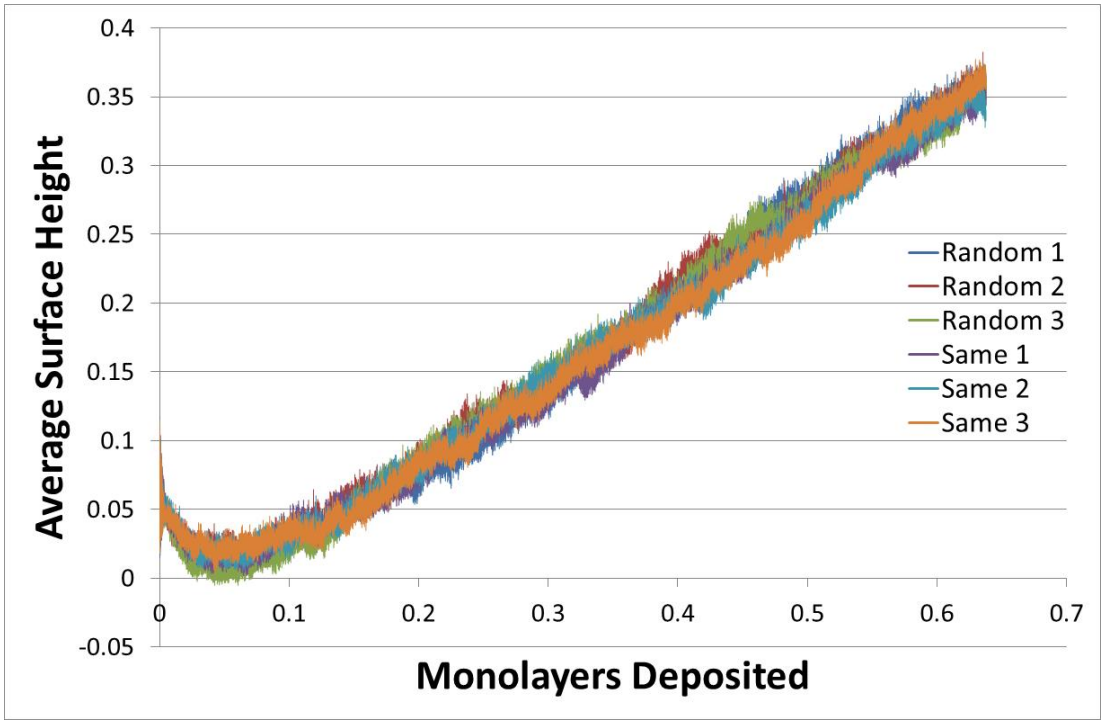
80. **Lide, David R., [ed.].** *CRC Handbook of Chemistry and Physics, 85th Edition.* s.l. : CRC Press, 2004. p. 2712. ISBN-13: 978-0849304859.

81. *Scaling of the active zone in the Eden process on percolation networks and the ballistic deposition model.* **Family, Fereydoon and Vicsek, Tamás.** 2, s.l. : Institute of Physics, February 1985, Journal of Physics A: Mathematical and General, Vol. 18, pp. L75-L81. doi: 10.1088/0305-4470/18/2/005.

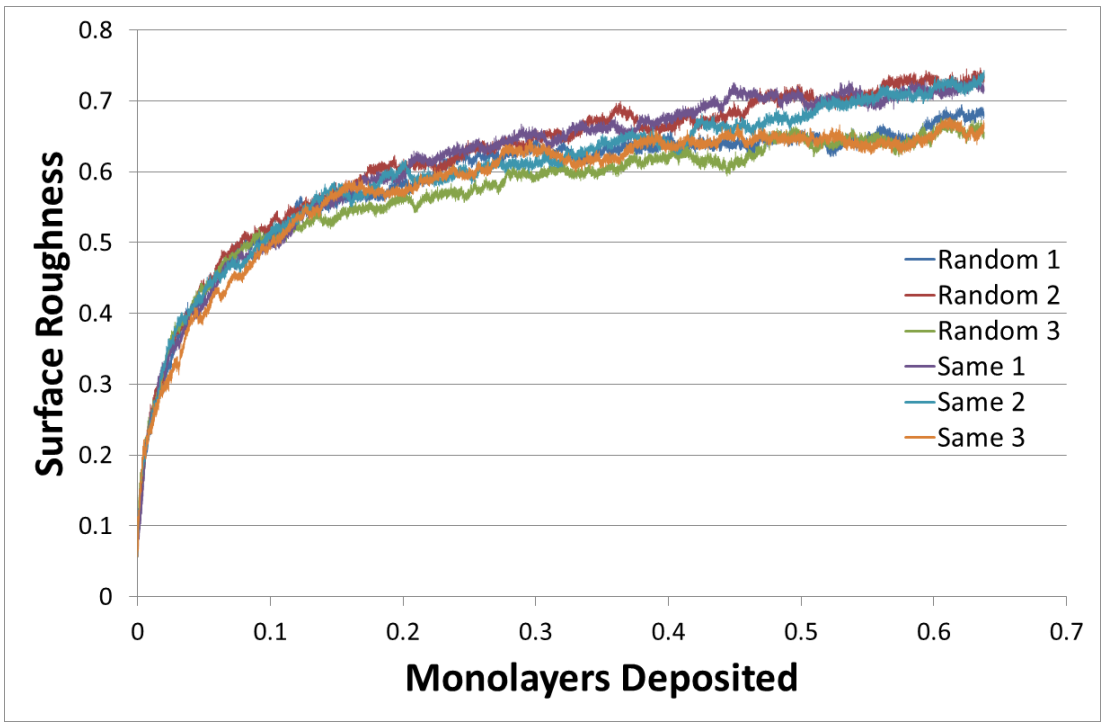
82. *Morphological evolution during epitaxial thin film growth: Formation of 2D islands and 3D mounds.* **Evans, J. W., Thiel, P. A. and Bartelt, M. C.** 1-2, s.l. : Elsevier B.V., April 2006, Surface Science Reports, Vol. 61, pp. 1-128. doi: 10.1016/j.surfrep.2005.08.004.

Appendices

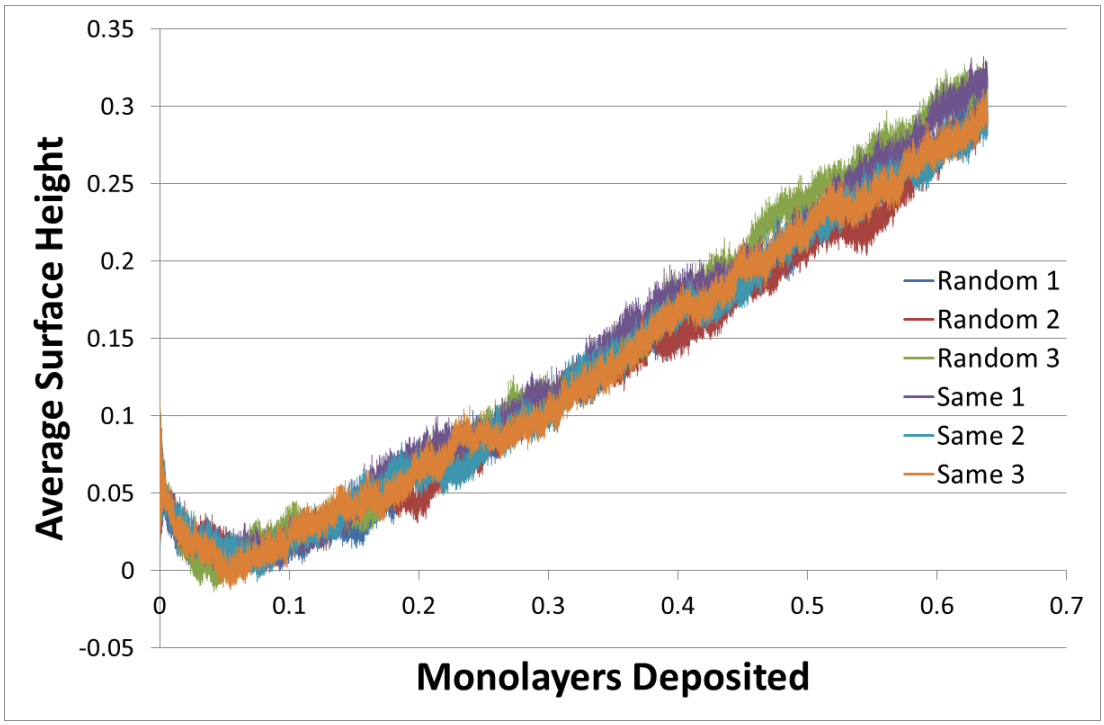
A – Impact Angle graphs



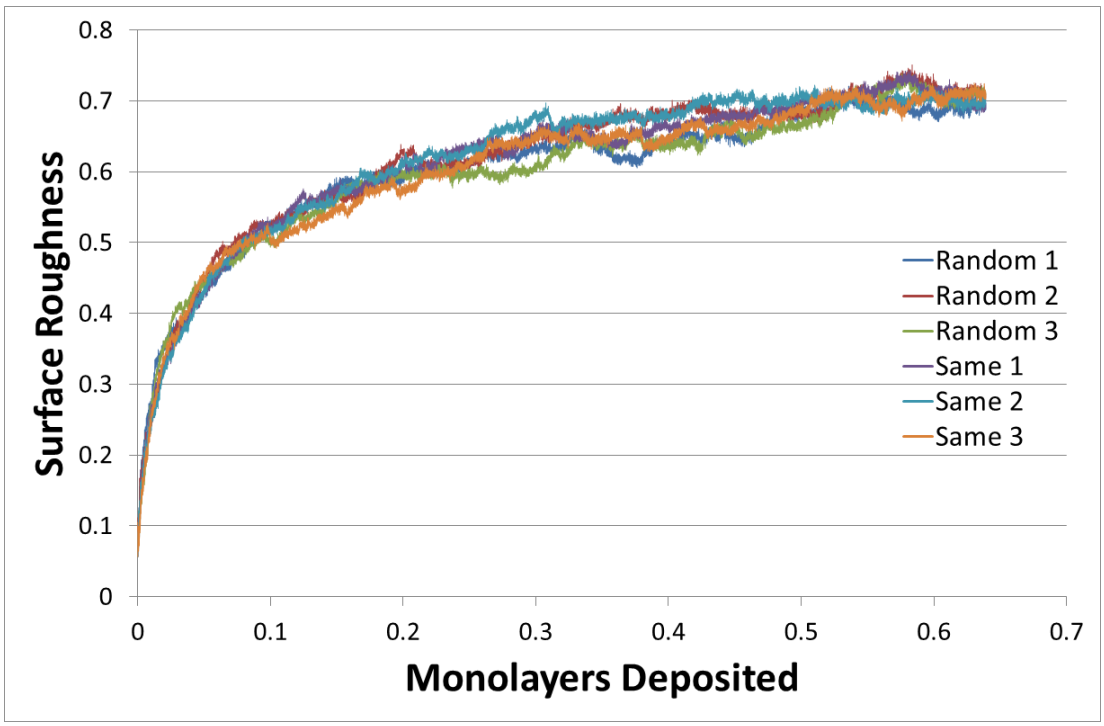
Comparison of average surface height for simulations of a polar angle of 10° with 3 using random azimuthal angles and 3 using the same azimuthal angle



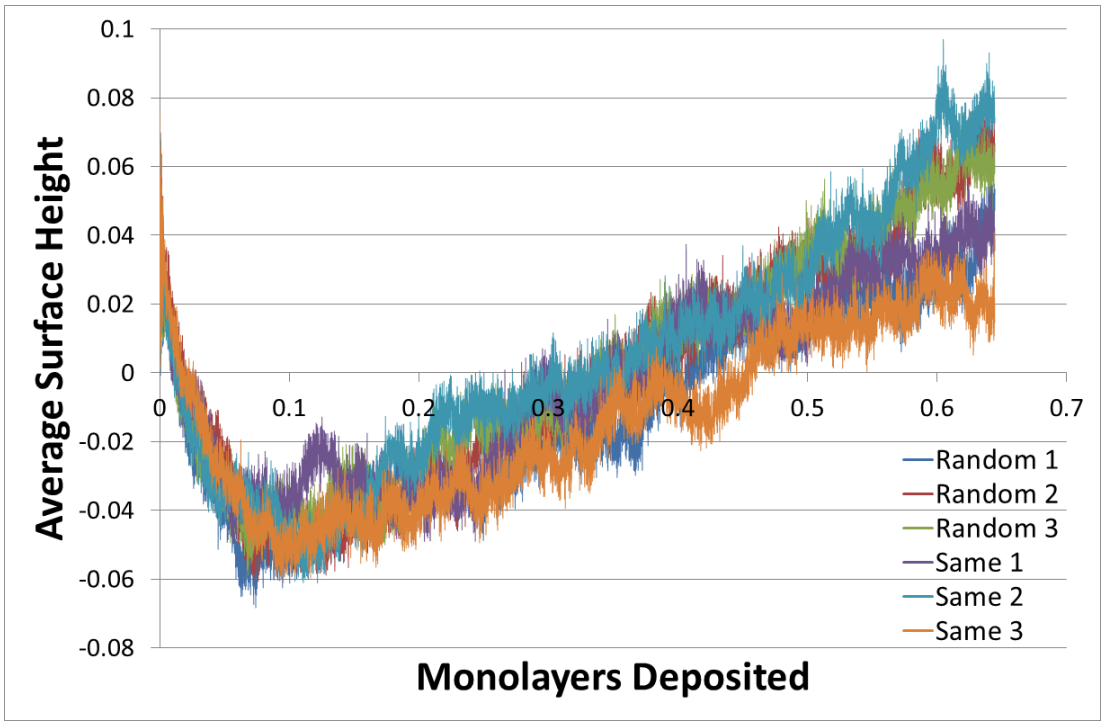
Comparison of surface roughness for simulations of a polar angle of 10° with 3 using random azimuthal angles and 3 using the same azimuthal angle



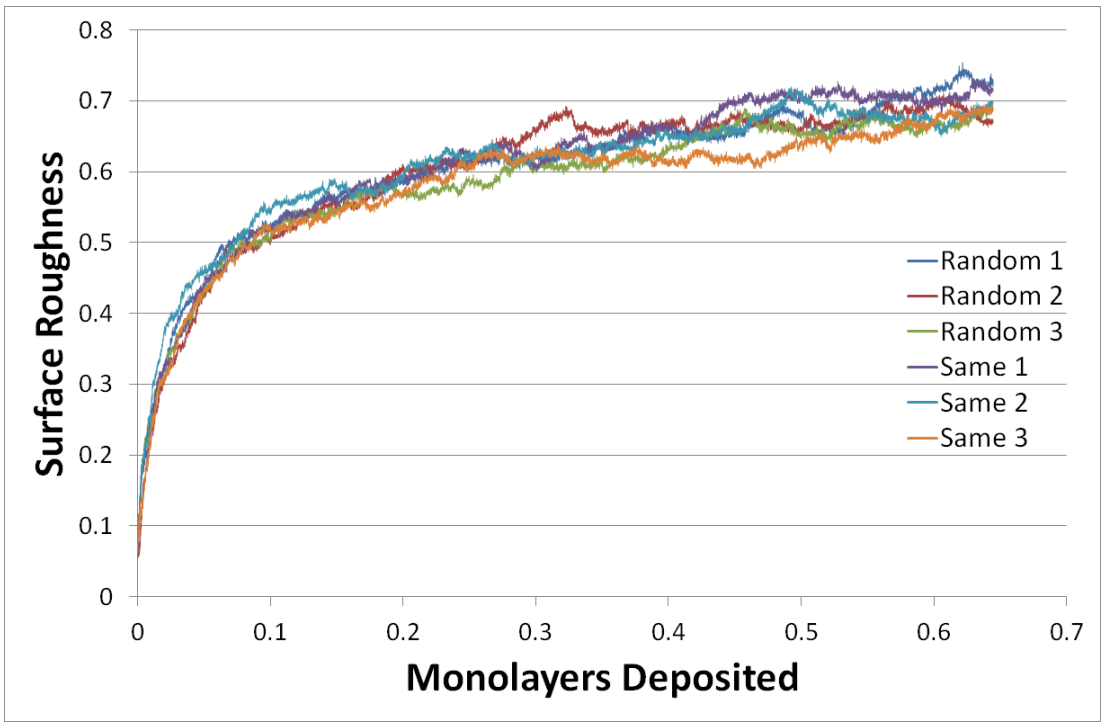
Comparison of average surface height for simulations of a polar angle of 20° with 3 using random azimuthal angles and 3 using the same azimuthal angle



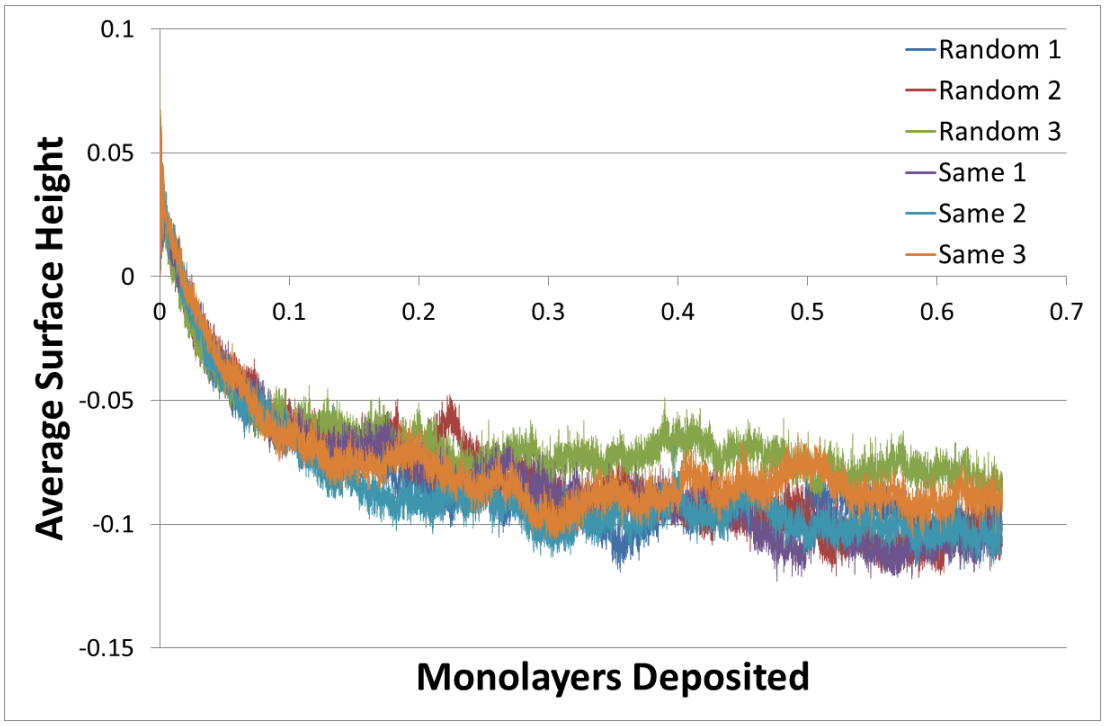
Comparison of surface roughness for simulations of a polar angle of 20° with 3 using random azimuthal angles and 3 using the same azimuthal angle



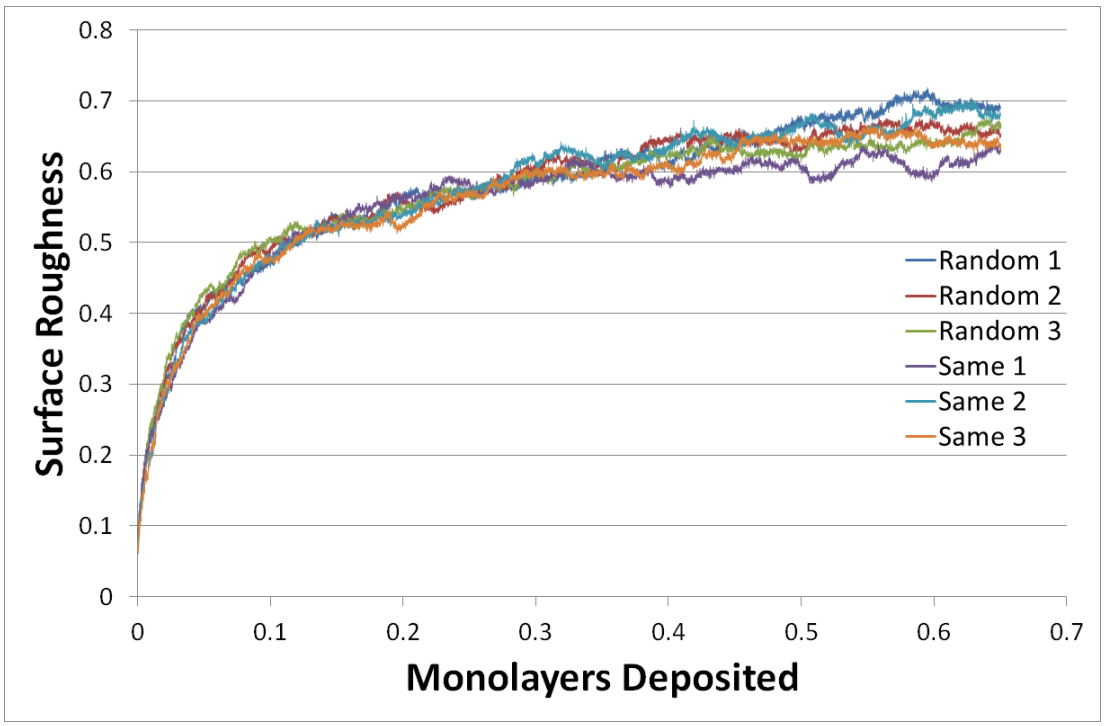
Comparison of average surface height for simulations of a polar angle of 40° with 3 using random azimuthal angles and 3 using the same azimuthal angle



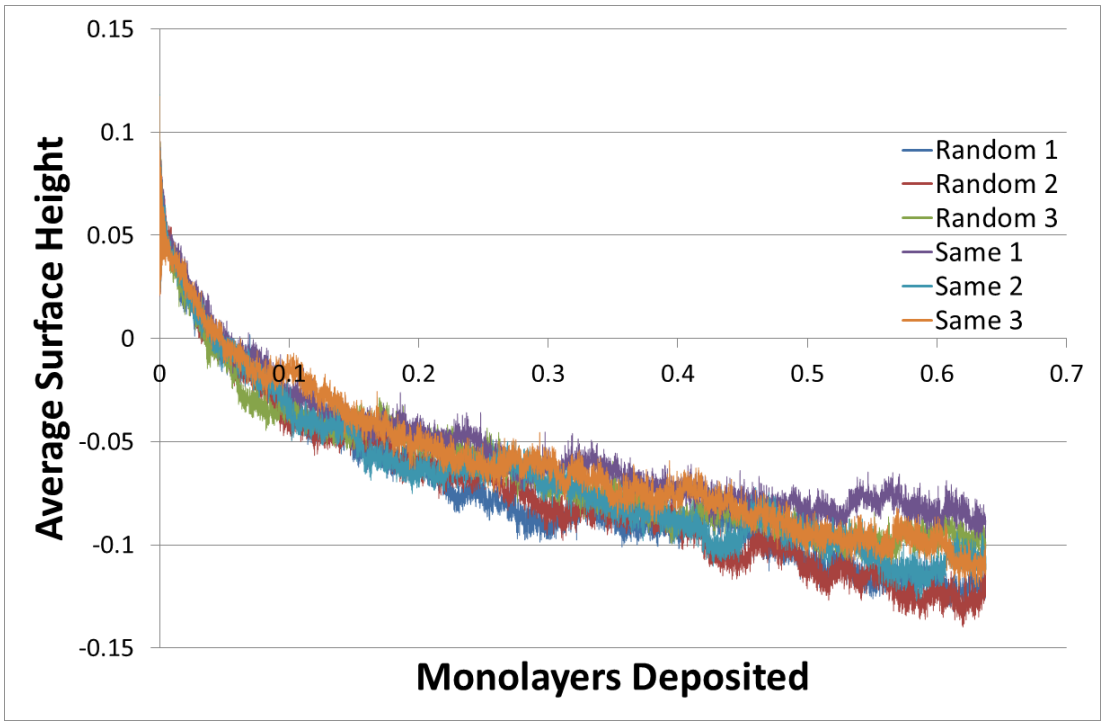
Comparison of surface roughness for simulations of a polar angle of 40° with 3 using random azimuthal angles and 3 using the same azimuthal angle



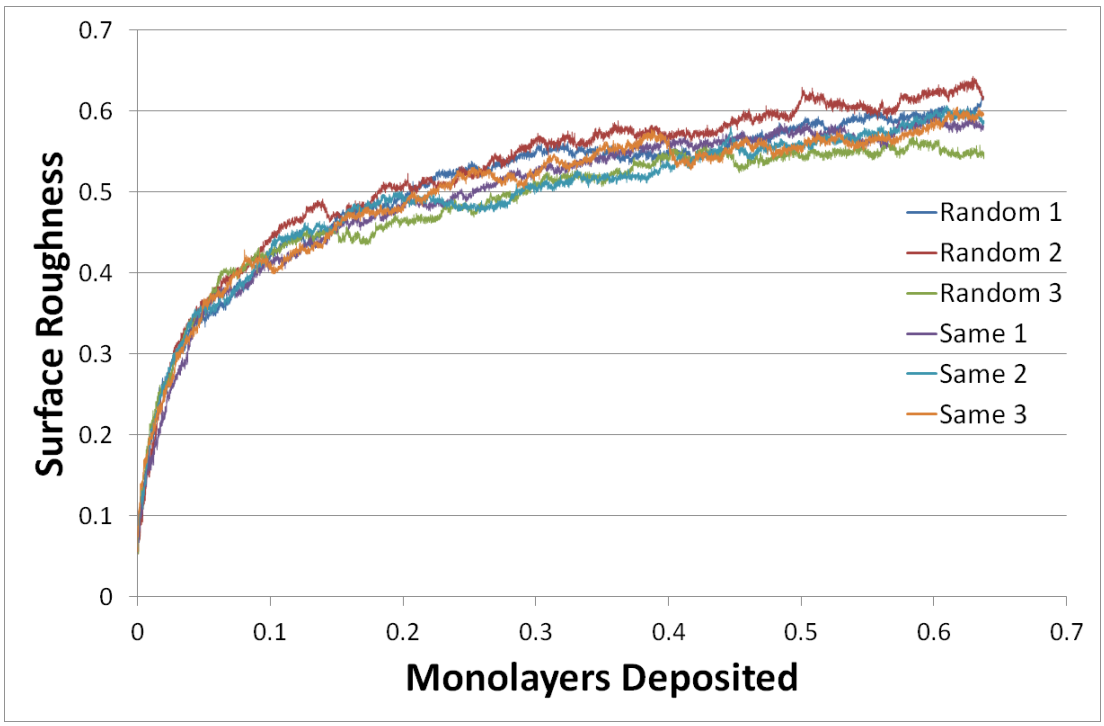
Comparison of average surface height for simulations of a polar angle of 50° with 3 using random azimuthal angles and 3 using the same azimuthal angle



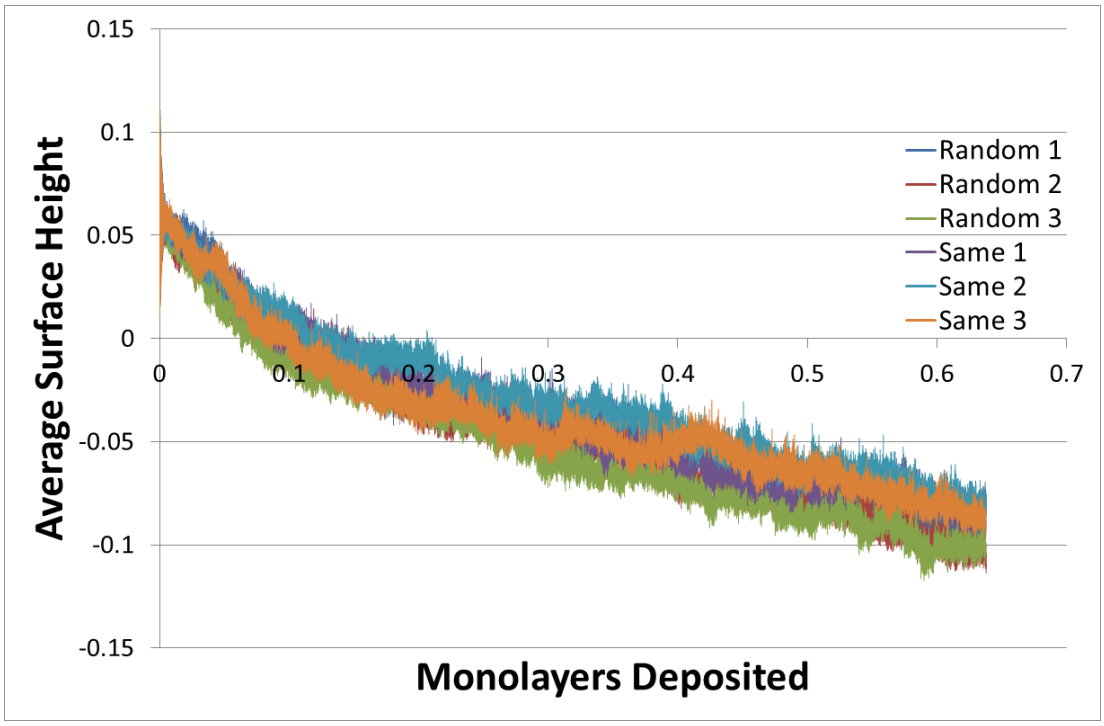
Comparison of surface roughness for simulations of a polar angle of 50° with 3 using random azimuthal angles and 3 using the same azimuthal angle



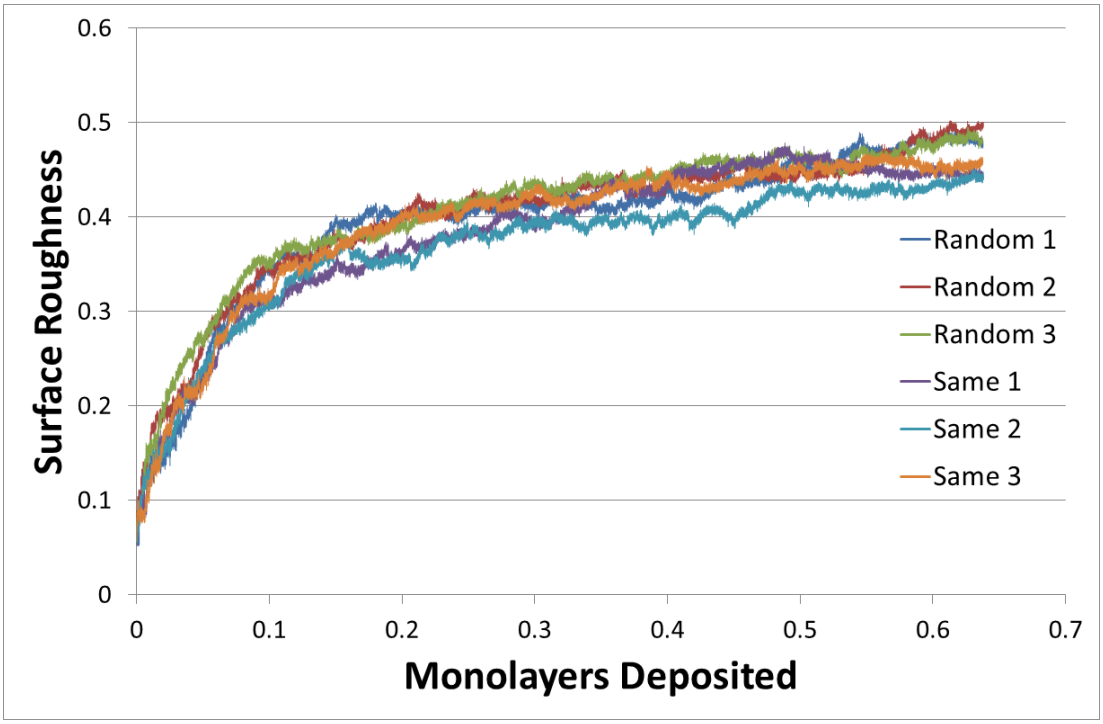
Comparison of average surface height for simulations of a polar angle of 60° with 3 using random azimuthal angles and 3 using the same azimuthal angle



Comparison of surface roughness for simulations of a polar angle of 60° with 3 using random azimuthal angles and 3 using the same azimuthal angle



Comparison of average surface roughness for simulations of a polar angle of 70° with 3 using random azimuthal angles and 3 using the same azimuthal angle



Comparison of surface roughness for simulations of a polar angle of 70° with 3 using random azimuthal angles and 3 using the same azimuthal angle

B – Lattice Crystal Generator Code

```
! Slab Generator designed to a variety of fcc crystals

PROGRAM slab
  IMPLICIT NONE

  INTEGER                       :: na,nb,nc,N,pl(3,3)
  INTEGER, PARAMETER            :: R=selected_real_kind(15,300)
  INTEGER, ALLOCATABLE          :: iontype(:)   ! 22=titanium
  REAL(KIND=R)                  :: a            ! Lattice parameters
  REAL(KIND=R), ALLOCATABLE     :: rx(:), ry(:), rz(:), q(:), rx2(:), &
       ry2(:), rz2(:)
  REAL(KIND=R), ALLOCATABLE     :: rx1(:), ry1(:), rz1(:)
  REAL(KIND=R)                  :: Bx0, By0, Bz0, rx0(4), ry0(4), &
       rz0(4)
  REAL(KIND=R)                  :: dot, aor2, zero, fa, Bx, By, Bz, &
       rc(3), rr(3)
  INTEGER                       :: ni, i, j, k, it0(4), m, mi
  CHARACTER(len=75)             :: ofile

  CALL SETUP(a, na, nb, nc, pl, rc, rr)

  N=32*na*nb*nc ! 4*2na*2nb*2nc
  ALLOCATE(rx(N))
  ALLOCATE(ry(N))
```

```
ALLOCATE(rz(N))
ALLOCATE(q(N))
ALLOCATE(rx2(N))
ALLOCATE(ry2(N))
ALLOCATE(rz2(N))
ALLOCATE(iontype(N))

WRITE(ofile,'(a4, 3i2.2, 3i1, a4)')'slab', na, nb, nc, pl(3,:), &
      '.xyz'
OPEN(7,file=ofile)
Bx0=a
By0=a
Bz0=a
!
! Box vectors
!
aor2=a/2.0
zero=0.0
!
! Base Coordinates
!
rx0 = (/  zero, zero, aor2, aor2 /)
ry0 = (/  zero, aor2, zero, aor2 /)
rz0 = (/  zero, aor2, aor2, zero /)
!
! Charges and ion types
!
it0 = (/ 22,22,22,22 /)
!
! Create super slab to be cut down into new repeating unit
! Need at least 2x2x2 to produce the full repeating unit
!
ni=0
DO k=nc-1,1-nc,-1
   DO i=1-na,na-1
      DO j=1-nb,nb-1
         rx(ni+1:ni+4) = rx0 + i*Bx0
         ry(ni+1:ni+4) = ry0 + j*By0
         rz(ni+1:ni+4) = rz0 + k*Bz0
         iontype(ni+1:ni+4) = it0
         ni=ni+4
      END DO
   END DO
END DO
!
! Calculate dot product of atom coordinate and unit vectors of new
! axis planes. Atoms with a dot product within a set range become
! new repeating unit
!
mi=0
DO m=1,ni
   dot=(rx(m)*pl(1,1)+ry(m)*pl(1,2)+rz(m)*pl(1,3)) &
        /SQRT(REAL(pl(1,1)**2+pl(1,2)**2+pl(1,3)**2,R))
   IF (dot<a*rc(1).AND.dot>=0) THEN
      dot=(rx(m)*pl(2,1)+ry(m)*pl(2,2)+rz(m)*pl(2,3)) &
           /SQRT(REAL(pl(2,1)**2+pl(2,2)**2+pl(2,3)**2,R))
      IF (dot<a*rc(2).AND.dot>=0) THEN
         dot=(rx(m)*pl(3,1)+ry(m)*pl(3,2)+rz(m)*pl(3,3)) &
              /SQRT(REAL(pl(3,1)**2+pl(3,2)**2+pl(3,3)**2,R))
```

```
            IF (dot>-a*rc(3).AND.dot<=0) THEN
               rx2(mi+1)=rx(m)
               ry2(mi+1)=ry(m)
               rz2(mi+1)=rz(m)
               mi=mi+1
            END IF
         END IF
      END IF
   END DO
   !
   ! Express Coordinates in terms of the new axis planes
   !
   ALLOCATE(rx1(mi))
   ALLOCATE(ry1(mi))
   ALLOCATE(rz1(mi))
   rx1 = (rx2(1:mi)*pl(1,1)+ry2(1:mi)*pl(1,2)+rz2(1:mi)*pl(1,3)) &
        /SQRT(REAL(pl(1,1)**2+pl(1,2)**2+pl(1,3)**2))
   ry1 = (rx2(1:mi)*pl(2,1)+ry2(1:mi)*pl(2,2)+rz2(1:mi)*pl(2,3)) &
        /SQRT(REAL(pl(2,1)**2+pl(2,2)**2+pl(2,3)**2))
   rz1 = (rx2(1:mi)*pl(3,1)+ry2(1:mi)*pl(3,2)+rz2(1:mi)*pl(3,3)) &
        /SQRT(REAL(pl(3,1)**2+pl(3,2)**2+pl(3,3)**2))
   !
   ! Scale number of layers
   ! Accounts for repeating unit size differences
   !
   fa = (REAL(mi)/4.0)**(1.0/3.0)
   na = nint(na/fa)
   nb = nint(nb/fa)
   nc = nint(nc/fa)
   !
   ! Create repeats of the unit to produce slab
   !
   ni=0
   iontype=22
   DO k=0,1-nc,-1
      DO i=0,na-1
         DO j=0,nb-1
            rx(ni+1:ni+mi) = rx1 + i*Bx0*rr(1)
            ry(ni+1:ni+mi) = ry1 + j*By0*rr(2)
            rz(ni+1:ni+mi) = rz1 + k*Bz0*rr(3)
            !iontype(nions+1:nions+4) = it0
            ni=ni+mi
         END DO
      END DO
   END DO
   !
   ! Write out data
   !
   Bx = na*Bx0*rr(1)
   By = nb*By0*rr(2)
   Bz = nc*Bz0*rr(3)
   WRITE(7,'(3i6)') ni,ni,ni
   WRITE(7,'(3i1,a17,3f11.6)') pl(3,:),' fcc crystal slab',Bx, By, Bz
   DO m = 1,ni
      WRITE(7,'(i3, 4f14.6)') iontype(m), rx(m), ry(m), rz(m)
   END DO

CONTAINS
```

```
  SUBROUTINE setup (a, na, nb, nc, pl, rc, rr)
    INTEGER,INTENT(OUT)       :: na, nb, nc, pl(:,:)
    REAL(KIND=R),INTENT(OUT)  :: a, rc(:), rr(:)
    CHARACTER(LEN=99)         :: buffer
    OPEN(5,file='setup.dat')
    DO
       READ(5,*)buffer
       IF (buffer=='LatPara') READ(5,*)a
       IF (buffer=='Layers') READ(5,*)na, nb, nc
       IF (buffer=='Planes') READ(5,*)pl(1,:), pl(2,:), pl(3,:)
       IF (buffer=='dpcut') READ(5,*)rc(:)
       IF (buffer=='rdist') READ(5,*)rr(:)
       IF (buffer=='RUN') EXIT
    END DO
    RETURN
  END SUBROUTINE setup
END PROGRAM slab
```

C – Molecular Dynamics Code

```
! Program for the collision of particles with a slab using MD and
the Sutton-Chen potential

PROGRAM MolecularDynamics
  USE iso_fortran_env
  IMPLICIT NONE

  !=======================================INITIALISER==============
==========================================
  ! This part of the program specifies all global variables. It then
uses initaliser subroutines and defines the variables needed
  ! to run the later parts of the program. Many of the variables are
set to initial values unless the program is resuming from a
  ! backup (which is defined as when the variable lc is set to 2).
At the end of the initialiser, a logical array is used to
  ! determine where the program advances to.
  !================================================================
  INTEGER                       ::
n,s,c,nlayers,nc,id,m,o,p,cell,it,lc,lrs,ss,m2,o2,p2,ks
  INTEGER (int64)               :: time,pid
  INTEGER, PARAMETER            :: R=SELECTED_REAL_KIND(15,300)
  INTEGER                       ::
i,ii,ij,ik,k,j,klimx,loopi,loopj,loopk,nl(3),numt,ion,q,u,tc,ts,coun
t
  INTEGER                       ::
n1,n2,n3,nsa,maxbin,bin,bin2,maxbinz,nsp,nst
  INTEGER, ALLOCATABLE          ::
mass(:),imass(:),ll(:),head(:),map(:),var(:),seed(:),imp(:),ll2(:),h
ead2(:),map2(:)
  INTEGER, ALLOCATABLE          ::
adj(:,:),sa(:),hist(:),histz(:),histl(:,:),layer(:)
  REAL (KIND=R)                 ::
xr,yr,zr,den,dv,gamma,vi,vo,con,lat,eta,ttau,limh
  REAL (KIND=R), PARAMETER      :: PI=3.14159265358979323846_R
  REAL (KIND=R)                 ::
e0,e1,e2,eTotk,eTot,eads,ekav,ru,eu,sh,sr
  REAL (KIND=R), ALLOCATABLE    :: r1(:,:),z(:)       ! Holds current
config
  REAL (KIND=R), ALLOCATABLE    :: r0(:,:)            ! Copy of start
config
  REAL (KIND=R), ALLOCATABLE    :: v(:,:)             ! velocities at
t
  REAL (KIND=R), ALLOCATABLE    :: a(:,:)             ! acceleration
at t
  REAL (KIND=R), ALLOCATABLE    :: a1(:,:)            ! acceleration
at t+dt
  REAL (KIND=R), ALLOCATABLE    :: f(:,:)             ! forces at t
  REAL (KIND=R), ALLOCATABLE    :: fo(:,:)            ! Output forces
for minimizer
  REAL (KIND=R), ALLOCATABLE    :: fi(:,:)            ! Input forces
for minimizer
  REAL (KIND=R), ALLOCATABLE    :: h(:,:)             ! Scaled forces
for minimizer
  REAL (KIND=R), ALLOCATABLE    :: rmass(:),amass(:)  ! mass as
```

```
  REAL (KIND=R), ALLOCATABLE  ::
gr(:),gr2(:),Sk(:),Sk2(:),gz(:),gz2(:),lgr(:,:),lgr2(:,:)
  REAL (KIND=R), PARAMETER    :: kb=1.380648813D-23 ! Boltzmann in
J/K
  !REAL (KIND=R), parameter    :: Ti_mass=1 , Oxy_mass =2.6567625D-
26 !7.94850136D-26
  REAL (KIND=R)               ::
T,dT,dT2,tT,vint,ek,rint(3),ang(3),vr(3),rc(3),pol,azi,aa,azi2,ri(3)
  REAL (KIND=R)               ::
B1(3),B2(3),B3(3),rcut,sigma,epsil,intek,intsv,rho,const,dr,rhi,rlow
,nid,dz,dr2
  REAL (KIND=R)               ::
bsum,smsum,mav,cmass,xi,yi,tauav,mf(3),scale,w1,w(3),theta,E100!,E10
1,E102,SE,SO,kk
  CHARACTER (len=100)         ::
args(3),ofile,fofile,efile,sfile,tfile,emfile,message,surfile,gfile,
rofile,refile,rsfile,rsurfile
  LOGICAL                     :: jump(2),cda

  maxbin=1000
  maxbinz=700
  CALL RANDOM_SEED(size=ss)
  ALLOCATE(seed(ss))
  CALL SYSTEM_CLOCK(time)
  pid = GETPID()
  time=IEOR(time,pid)
  DO i=1,ss
     IF (time==0) THEN
        time=104729
     ELSE
        time=MOD(time,4294967296_int64)
     END IF
     time=MOD(time*279470273_int64,4294967291_int64)
     lrs=INT(MOD(time,INT(HUGE(0),int64)),KIND(0))
     seed(i)=lrs
  END DO
  CALL RANDOM_SEED(put=seed)
  WRITE(*,*) seed
  CALL CPU_TIME(bsum)                                  ! Start timer
for CPU time
  CALL SYSTEM_CLOCK(n1,n2,n3)                          ! Start timer
for Wall-clock time in ms
  CALL
setup(dT,T,tT,ek,rint,ang,loopi,loopj,loopk,nl,numt,rcut,sigma,epsil
,pol,azi,ru,eu,ion,imass,amass,nc,id,jump,lc, &
       cda,lat,eta,con,q,u)
  CALL GET_COMMAND_ARGUMENT(1,args(1))                 ! Obtain
first argument specified at run-time, used as input slab file
  CALL GET_COMMAND_ARGUMENT(2,args(2))                 ! Obtain
second argument specified at run-time, used to name outputs
  CALL GET_COMMAND_ARGUMENT(3,args(3))                 ! Obtain
third argument specified at run-time, used as input cluster file
  WRITE(ofile,'(a7,a4)')args(2),'.xyz'                 ! Generates
name for trajectory file
  WRITE(rofile,'(a7,a7)')args(2),'_rs.xyz'             ! Generates
name for trajectory file
  WRITE(efile,'(a7,a4)')args(2),'.csv'                 ! Generates
name for energy output file
```

```
  WRITE(refile,'(a7,a7)')args(2),'_rs.csv'             ! Generates
name for energy output file
  WRITE(fofile,'(a5,a7,a4)')'final',args(2),'.xyz'     ! Generates
name for final output file
  WRITE(sfile,'(a3,a7,a4)')'sum',args(2),'.xyz'        ! Generates
name for summary output file
  WRITE(rsfile,'(a3,a7,a7)')'sum',args(2),'_rs.xyz'    ! Generates
name for summary output file
  WRITE(tfile,'(i0,a1,a17)')INT(T*eta),'K',args(1)     ! Generates
name for thermal slab file
  WRITE(surfile,'(a4,a7,a4)')'surf',args(2),'.xyz'     ! Generates
name for surface file
  WRITE(rsurfile,'(a4,a7,a7)')'surf',args(2),'_rs.xyz' ! Generates
name for surface file
  WRITE(gfile,'(a5,a7,a4)')'hists',args(2),'.csv'      ! Generates
name for distribution file
  eta=eta*kb/eu                                        ! Converts
epsilon to desired energy units from Temperature-equivalent
  CALL Init(args(1),args(3),n,s,c,nc,lc)

  ! Allocate size of most allocateable arrays based on number of
atoms
  ALLOCATE(rmass(n))        ; ALLOCATE(mass(n))       ;
ALLOCATE(r1(3,n))        ; ALLOCATE(z(n))       ; ALLOCATE(r0(3,n))
  ALLOCATE(v(3,n))          ; ALLOCATE(a(3,n))        ;
ALLOCATE(a1(3,n))        ; ALLOCATE(f(3,n))     ; ALLOCATE(fo(3,n))
  ALLOCATE(fi(3,n))         ; ALLOCATE(h(3,n))        ;
ALLOCATE(adj(3,n))       ; ALLOCATE(ll(n))      ; ALLOCATE(var(n))
  ALLOCATE(imp(nc*c))       ; ALLOCATE(sa(n))         ;
ALLOCATE(ll2(n))
  ALLOCATE(hist(maxbin))    ; ALLOCATE(gr(maxbin))    ;
ALLOCATE(gr2(maxbin))    ; ALLOCATE(Sk(maxbin)) ;
ALLOCATE(Sk2(maxbin))
  ALLOCATE(histz(maxbinz))  ; ALLOCATE(gz(maxbinz))   ;
ALLOCATE(gz2(maxbinz))
  ALLOCATE(histl(8,maxbin)) ; ALLOCATE(lgr(8,maxbin)) ;
ALLOCATE(lgr2(8,maxbin)) ; ALLOCATE(layer(8))


  scale=1.0_R
  v=0                                                  ! Zero atom
velocity array
  adj=0                                                ! Zero atom
adjusment array
  CALL
LoadConfig(args(1),args(3),r1,z,v,a,n,s,c,B1,B2,B3,nc,eta,lat,amass(
:),eu,ru,lc,adj,ks,tc)
  nlayers=SUM(nl)
  rho=3.88515365E-25/amass(1)
  dr=B2(2)/(2.0*maxbin)
  dz=(B3(3)/REAL(nlayers)*REAL(nlayers-
nl(3))+10.0_R/lat)/REAL(maxbinz)
  m=FLOOR(B1(1)/rcut)                                  ! Calculate
number of cells in the x-direction
  o=FLOOR(B2(2)/rcut)                                  ! Calculate
number of cells in the y-direction
  p=FLOOR(B3(3)/rcut)                                  ! Calculate
number of cells in the z-direction
```

```
  cell=m*o*p                                              ! Calculate
total number of cells

  m2=FLOOR(B1(1)/2.0_R)
  o2=FLOOR(B2(2)/2.0_R)
  p2=FLOOR(B3(3)/2.0_R)

  limh=30.0_R/lat

  ! Allocate some of remaining allocateable arrays based on total
number of cells
  ALLOCATE(head(cell)) ; ALLOCATE(map(cell*26)) ;
ALLOCATE(head2(m2*o2*p2)) ; ALLOCATE(map2(m2*o2*p2*26))

  CALL Celllink(m,o,p,map)
  CALL Celllink(m2,o2,p2,map2)

  IF (lc==2)THEN
     OPEN(10,file=sfile,position='APPEND')
     BACKSPACE(10)
     READ(10,*)nc,nsp,nst,theta,e100
     CLOSE(10)
     OPEN(10,file=rsfile)
     OPEN(17,file=refile)
     OPEN(20,file=rsurfile)
     OPEN(12,file=rofile)
     WRITE(12,'(i6)')n
     WRITE(12,*)'Restarting from time
',dt*(tc)*lat*ru*SQRT(amass(1)/(eta*eu))
     DO j=1,n
        WRITE(12,*)mass(j),r1(:,j)*lat,z(j)
     END DO
     ks=ks+1
     IF(NINT(rint(1))==-100) rint(1)=MINVAL(r1(1,1:s))+B1(1)/2 ;
IF(NINT(rint(2))==-100) rint(2)=MINVAL(r1(2,1:s))+B2(2)/2
     imp=0
     ri=rint
     DO i = 1, n
        DO j=1,ion                                        ! Assign mass of
atoms as ratio of mass of first atom species
           IF (mass(i)==imass(j)) THEN                    ! Assign mass by
comparing atom's atomic number to species atomic number
              rmass(i) = amass(j)/amass(1)
           END IF
        END DO
     END DO
     cmass=0                                              ! Zero the
cluster mass variable
     DO i=1+n-c,n                                         ! Calculate total
mass of cluster
        cmass=cmass+rmass(i)
     END DO
     mav=SUM(rmass)/n                                     ! Calculate
average mass of an atom
     dr2=(B3(3)/REAL(nlayers))+(sh+sr*2.5)/5.0
     vr(3)=1                                              ! Set ratio of z
velocity with respect to z velocity
     vint=-SQRT(ek*2/cmass)                               ! Calculate
initial overall velocity of cluster atoms
```

```
     pol=pol*pi/180.0_R
     azi2=azi

     DO j=1,n                                    ! Save the trajectory
of any atom that leaves the surface
        IF(r1(3,j)>=rcut*1.5.AND.v(3,j)>0) THEN
           IF (var(j)==0)THEN
              var(j)=tc
           END IF
        END IF
     END DO

     GOTO 1000
  END IF

  ks=1
  DO i=1,3                                     ! Find center of the
cluster(s) on each of the 3 axes
     rc(i)=(MAXVAL(r1(i,1+s:n))+MINVAL(r1(i,1+s:n)))/2
  END DO

  DO j=0,nc-1
     DO i=1+s+c*j,s+c*(j+1)                    ! Duplicate and
centralise the cluster locations
        r0(:,i)=r1(:,i)-rc(:)
     END DO
  END DO

  ! Generate random rotations when required
  IF(NINT(ang(1))==-180) THEN
     CALL RANDOM_NUMBER(xr)
     ang(1)=15*INT(24*xr)
  END IF
  IF(NINT(ang(2))==-180) THEN
     CALL RANDOM_NUMBER(yr)
     ang(2)=15*INT(24*yr)
  END IF
  IF(NINT(ang(3))==-180) THEN
     CALL RANDOM_NUMBER(zr)
     ang(3)=15*INT(24*zr)
  END IF

  ang=ang*pi/180.0_R

  DO j=0,nc-1
     DO i=1+s+c*j,s+c*(j+1)                    ! Rotate the
cluster(s)
        r1(1,i)=(COS(ang(2))*COS(ang(3)))*r0(1,i)-
(COS(ang(2))*SIN(ang(3)))*r0(2,i)+SIN(ang(2))*r0(3,i)

r1(2,i)=(SIN(ang(1))*SIN(ang(2))*COS(ang(3))+COS(ang(1))*SIN(ang(3))
)*r0(1,i) &
             +(COS(ang(1))*COS(ang(3))-
SIN(ang(1))*SIN(ang(2))*SIN(ang(3)))*r0(2,i) &
             -(SIN(ang(1))*COS(ang(2)))*r0(3,i)
        r1(3,i)=(-
COS(ang(1))*SIN(ang(2))*COS(ang(3))+SIN(ang(1))*SIN(ang(3)))*r0(1,i)
&
```

```
+(SIN(ang(1))*COS(ang(3))+COS(ang(1))*SIN(ang(2))*SIN(ang(3)))*r0(2,
i) &
             +(COS(ang(1))*COS(ang(2)))*r0(3,i)
     END DO
  END DO

  ! Generate initial cluster position on x and y axes as center of
the slab when required
  IF(NINT(rint(1))==-100) rint(1)=MINVAL(r1(1,1:s))+B1(1)/2 ;
IF(NINT(rint(2))==-100) rint(2)=MINVAL(r1(2,1:s))+B2(2)/2

  rint(3)=rint(3)+100
  DO i=1+s,n
     r1(3,i)=r1(3,i)+ rint(3)                ! move the rotated
cluster(s)
  END DO
  rint(3)=rint(3)-100
  ri=rint
  !     END IF

  DO i = 1, n
     DO j=1,ion                                    ! Assign mass of
atoms as ratio of mass of first atom species
        IF (mass(i)==imass(j)) THEN                ! Assign mass by
comparing atom's atomic number to species atomic number
           rmass(i) = amass(j)/amass(1)
        END IF
     END DO
  END DO

  cmass=0                                          ! Zero the cluster
mass variable
  DO i=1+n-c,n                                     ! Calculate total
mass of cluster
     cmass=cmass+rmass(i)
  END DO
  mav=SUM(rmass)/n                                 ! Calculate average
mass of an atom


  vr(3)=1                                          ! Set ratio of z
velocity with respect to z velocity
  eTotk=ek*nc                                      ! Calculate initial
kinetic energy of system
  vint=-SQRT(ek*2/cmass)                           ! Calculate initial
overall velocity of cluster atoms
  pol=pol*pi/180.0_R
  azi2=azi
  IF (NINT(azi2)==-180) THEN
     DO i=0,nc-1
        CALL RANDOM_NUMBER(aa)
        azi=360*aa*pi/180.0_R
        vr(1)=TAN(pi/2.0)*COS(azi) ; vr(2)=TAN(pi/2.0)*SIN(azi)
        DO j=1+s+c*i,s+c*(i+1)                   ! Calculate initial
velocity of cluster atoms in each of the 3 directions
           v(:,j)=vr(:)*vint/SQRT(vr(1)**2+vr(2)**2+vr(3)**2)
        END DO
     END DO
```

```
  ELSE
     azi=azi*pi/180.0_R
     vr(1)=TAN(pi/2.0)*COS(azi) ; vr(2)=TAN(pi/2.0)*SIN(azi)
     DO i=1+s,n                                    ! Calculate initial
velocity of cluster atoms in each of the 3 directions
        v(:,i)=vr(:)*vint/SQRT(vr(1)**2+vr(2)**2+vr(3)**2)
     END DO
  END IF

!!$     IF (nc>1) THEN                                ! Move
additional clusters to control time of impact
!!$        DO j=1,nc-1
  DO i=1,s
!!$              r1(:,i)=r1(:,i)-id*dT*loopj*loopk*v(:,i)*j
!!$              IF (cda) THEN
!!$                 r1(3,i)=r1(3,i)+rint(3)*j
!!$              END IF
     DO
        IF (r1(1,i)<-0.0000000001_R)THEN  ! Adjust cluster atoms
above images in the x-direction
           r1(1,i)=r1(1,i)+B1(1) ; adj(1,i)=adj(1,i)-1
        ELSEIF (r1(1,i)>=B1(1)) THEN
           r1(1,i)=r1(1,i)-B1(1) ; adj(1,i)=adj(1,i)+1
        ELSE
           EXIT
        END IF
     END DO
     DO
        IF (r1(2,i)<-0.0000000001_R)THEN  ! Adjust cluster atoms
above images in the y-direction
           r1(2,i)=r1(2,i)+B2(2) ; adj(2,i)=adj(2,i)-1
        ELSEIF (r1(2,i)>=B2(2)) THEN
           r1(2,i)=r1(2,i)-B2(2) ; adj(2,i)=adj(2,i)+1
        ELSE
           EXIT
        END IF
     END DO
  END DO
!!$        END DO
!!$     END IF

  OPEN(12,file=ofile)
  WRITE(12,'(i6)') n
  WRITE(12,*)'Titanium Dioxide Slab and Cluster'
  DO i=1,n                                      ! Save initial
configuation of system to trajectory file
     WRITE(12,'(i3,3f10.4,f4.1)') mass(i),r1(:,i)*lat,z(i)
  END DO

  CALL Celllist(m,o,p,ll,head,r1,B1,B2,B3,limh)
  CALL
Forces(rcut,r1,B1,B2,e0,f,ll,head,cell,con,q,u,scale,w1,w,.TRUE.)
  !
  ! Force and Energy verification
  !
!!$WRITE(*,*)e0,f(:,1)
!!$r1(1,1)=r1(1,1)+0.00001_R
!!$     CALL Forces(rcut,r1,B1,B2,e1,f,ll,head,cell,con,q,u,scale)
!!$WRITE(*,*)e1
```

```
!!$r1(1,1)=r1(1,1)-0.00002_R
!!$     CALL Forces(rcut,r1,B1,B2,e2,f,ll,head,cell,con,q,u,scale)
!!$WRITE(*,*)e2,(e2-e1)/0.00002_R
!!$r1(1,1)=r1(1,1)+0.00001_R
!!$r1(2,1)=r1(2,1)+0.00001_R
!!$     CALL Forces(rcut,r1,B1,B2,e1,f,ll,head,cell,con,q,u,scale)
!!$WRITE(*,*)e1
!!$r1(2,1)=r1(2,1)-0.00002_R
!!$     CALL Forces(rcut,r1,B1,B2,e2,f,ll,head,cell,con,q,u,scale)
!!$WRITE(*,*)e2,(e2-e1)/0.00002_R
!!$r1(2,1)=r1(2,1)+0.00001_R
!!$r1(3,1)=r1(3,1)+0.00001_R
!!$     CALL Forces(rcut,r1,B1,B2,e1,f,ll,head,cell,con,q,u,scale)
!!$WRITE(*,*)e1
!!$r1(3,1)=r1(3,1)-0.00002_R
!!$     CALL Forces(rcut,r1,B1,B2,e2,f,ll,head,cell,con,q,u,scale)
!!$WRITE(*,*)e2,(e2-e1)/0.00002_R
  !
  ! Potential and Force grapher
  !
!!$  OPEN(1104,file='forceSCCO.csv')
!!$  CALL
Forces(rcut,r1,B1,B2,e1,f,ll,head,cell,con,q,u,scale,w1,w,.TRUE.)
!!$  Write(1104,*)r1(3,1+s),',',e1,',',f(3,1+s)
!!$
!!$  DO WHILE(r1(3,1+s)>0.11_R)
!!$     r1(3,1+s)=r1(3,1+s)-0.001_R
!!$     CALL
Forces(rcut,r1,B1,B2,e1,f,ll,head,cell,con,q,u,scale,w1,w,.TRUE.)
!!$     Write(1104,*)r1(3,1+s),',',e1,',',f(3,1+s)
!!$  END DO

!!$  WRITE(*,*)e0
!!$  DO ii=0,200
!!$     scale=0.99_R+ii*0.0001_R
!!$     rcut=rcut*scale
!!$     CALL Celllist(m,o,p,ll,head,r1,B1,B2,B3,limh)
!!$     CALL
Forces(rcut,r1,B1,B2,e1,f,ll,head,cell,con,q,u,scale,w1,w,.TRUE.)
!!$     WRITE(*,*)e1,scale
!!$     rcut=rcut/scale
!!$  END DO
!!$  STOP

  ! Calculate accelerations of atoms in each of the 3 directions
  a(1,:)=f(1,:)/rmass ; a(2,:)=f(2,:)/rmass ; a(3,:)=f(3,:)/rmass

  DO j=1, s
     IF(r1(3,j)<=-B3(3)/nlayers*(nlayers-nl(3))+0.01_R)THEN
        a(:,j)=0                                          ! Zero
acceleration of Fixed-layer atoms
     END IF
  END DO

  OPEN(10,file=sfile)
  WRITE(10,*)'Initial Cluster Trajectory'
  DO j=1+s,n
     WRITE(10,'(i7,3f11.6,3f13.6)')j,r1(:,j),v(:,j)
  END DO
```

```
  WRITE(10,*)'Sputtered atoms'
  WRITE(10,*)'Atom no.   Cluster atom?   Time   Atom Postition
Atom Velocity   Atom Acceleration'

  CALL CPU_TIME(smsum)                              ! Stop timer for
CPU Time
  !================================================END
INITIALISER=============================================================
========

  !================================================SLAB
MINIMIZER===============================================================
=======
  IF (jump(1)) THEN
     fo=f
     Vo=e0
     it=0
     h=fo
     DO
        it=it+1
        fi=h
        Vi=Vo
        CALL minp(fi,Vi,r1,s,B1,B2)
        CALL Celllist(m,o,p,ll,head,r1,B1,B2,B3,limh)
        CALL
Forces(rcut,r1,B1,B2,Vo,f,ll,head,cell,con,q,u,scale,w1,w,.TRUE.)
        fo=f
        dv=Vo-Vi
        WRITE(*,*) 'Iteration, Energy, dE ',it,Vo,dv
        mf(1)=MAXVAL(ABS(fo(1,1:(s/nlayers*(nlayers-nl(3))))))
        mf(2)=MAXVAL(ABS(fo(2,1:(s/nlayers*(nlayers-nl(3))))))
        mf(3)=MAXVAL(ABS(fo(3,1:(s/nlayers*(nlayers-nl(3))))))
        WRITE(*,*)'Maximum x,y,z force = ',mf(:)
        IF(-dv<0.02)EXIT
        IF(MAXVAL(mf)<0.001_R) EXIT
        gamma=0.0_R
        den=0.0_R
        DO i=1,n
           gamma=gamma+SUM((fo(:,i)-fi(:,i))*fo(:,i))
           den=den+SUM(fi(:,i)**2)
        END DO
        gamma=gamma/den
        h=gamma*h+fo
     END DO
     WRITE(*,*) 'Total iterations = ',it

     WRITE(message,'(a40,f0.5)')'Relaxed Slab with a potential
energy of ',Vo*eta
     CALL
SaveConfig(fofile,r1,z,v,a,s,s,0,eta,lat,amass,eu,ru,message,adj)
     e0=Vo

     ! Calculate accelerations of atoms in each of the 3 directions
     a(1,:)=f(1,:)/rmass ; a(2,:)=f(2,:)/rmass ; a(3,:)=f(3,:)/rmass

     DO j=1,s
        IF (r1(3,j)<=-B3(3)/nlayers*(nlayers-nl(3))+0.01_R)THEN
           a(:,j)=0                                  ! Zero
acceleration of fixed-layer atoms
```

```
        END IF                                              ! Fixed-layer
atoms
     END DO

     CALL CPU_TIME(smsum)                                   ! Stop timer
for CPU Time
  END IF
  !================================================END SLAB
MINIMIZER=============================================================
===

  !================================================EQUILIBRATOR=====
==============================================================
  IF (jump(2)) THEN
     WRITE(*,*)'Equilibriating to ',T*(eta*eu/kb),'K'
     intek=(3*s/nlayers*(nlayers-nl(3)))*T/2
     intsv=SQRT(intek*2/SUM(rmass(1:s/nlayers*(nlayers-nl(3)))))
     WRITE(*,*) intek,intsv

     DO i=1,s
        IF(r1(3,i)>-B3(3)/nlayers*(nlayers-nl(3))+0.01_R)THEN
           DO j=1,3
              CALL RANDOM_NUMBER(aa)
              v(j,i)=intsv*(4.0/3.0*aa-2.0/3.0)
           END DO
        END IF
     END DO

     nl(1)=0
     ii=0
     DO WHILE (ii<300)
        ii=ii+1
        CALL CPU_TIME(bsum)
        ekav=0                                              ! Zero average
kinetic energy variable
        Tauav=0

        DO ij=1,5
           WRITE(12,*)
           WRITE(12,*)'Time = ',dt*(ij*10+(ii-
1)*100)*lat*ru*SQRT(amass(1)/(eta*eu))

           DO k=1,100

              CALL timestep(s,r1,v,a,adj,a1,f,eTot,eTotk,e0,ekav)

              Tauav=Tauav+etotk*2/(3*s/nlayers*(nlayers-nl(3)))
           END DO

           DO j=1,n                                         ! Save
configuration at current timestep to trajectory file
              WRITE(12,'(i3,3f10.4,f4.1)')mass(j),r1(:,j)*lat,z(j)
           END DO                                           ! j

        END DO                                              ! ij

        Tauav=Tauav/500                                     ! Calculate
the average slab Temperature
        CALL CPU_TIME(smsum)
```

```
WRITE(*,'(i9,f14.5,f24.8,f12.6)')ii*500,eTotk*eta,Tauav,(smsum-bsum)
        CALL FLUSH()                                    ! Force all
files to update
     END DO                                             ! ii
     nl(1)=nlayers-nl(2)-nl(3)
     WRITE (message,'(a21,f0.2,a1)')'Slab equilibrated to
',T*eta*eu/kb,'K'
     CALL
SaveConfig(tfile,r1,z,v,a,s,s,0,eta,lat,amass,eu,ru,message,adj)
  END IF
  !================================================END
EQUILIBRATOR============================================================
========

  !================================================TIME-STEP
INTEGRATOR==============================================================
==
  e1=e0                                                 ! Saves initial
potential energy for later use
  eTotk=0                                               ! Resets kinetic
energy
  DO i=1,n                                              ! Calculates
kinetic energy
     eTotk=eTotk+0.5_R*rmass(i)*(v(1,i)**2+v(2,i)**2+v(3,i)**2)
  END DO
  eTot=eTotk+e0                                         ! Calculates
initial Total energy
  imp=0
  WRITE(*,'(a,4es14.5)')' Initial velocities = ',vint, v(:,1+s)
  WRITE(*,*)'Initial Potential Energy = ',e0
  OPEN(17,file=efile)

WRITE(17,'(a61,es8.1e2,a,f8.3,a,es8.1e2,a,f8.3,a,f8.3,a,f8.3,a,f8.3,
a,f6.2,a,f6.2,a,f6.2,a,i2,a,i2,a,i2)') &
       'Iteration,Total Energy,Potential Energy,Kinetic
Energy,,,,,,,', dT,',',T*(eta*eu/kb),',',tT,',',ek,',', &

rint(1),';',rint(2),';',rint(3),',',ang(1),';',ang(2),';',ang(3),','
,nl(1),';',nl(2),';',nl(3)

WRITE(17,'(i9,a,f17.8,a,f17.8,a,f17.8,a7,f8.3,a,es10.3e2,a,es10.3e2,
a,f7.3,a,f7.3,a,f10.6,a,es10.3e2,a,es10.3e2)') &

0,',',eTot*eta,',',e0*eta,',',eTotk*eta,',,,,,,,',rcut,',',ru,',',eu
,',', &

pol*180.0_R/pi,';',azi*180.0_R/pi,',',lat,',',eta,',',amass(1)
  OPEN(20,file=surfile)

  WRITE(*,'(a59)')'Iteration  Total Energy  Average Kinetic Energy
Time Taken'
  WRITE(*,'(i9,f14.5,f24.8,f12.6)')0,eTot*eta,eTotk*eta,(smsum-bsum)
  WRITE(emfile,'(i0,a7,a6)')0,args(2),'bk.xyz'
  WRITE(message,'(a50,i0)')'Backup of slab and cluster trajectory at
timestep ',0
  CALL
SaveConfig(emfile,r1,z,v,a,n,s,c,eta,lat,amass,eu,ru,message,adj)
```

```
  CALL Celllist(m2,o2,p2,ll2,head2,r1,B1,B2,B3,limh)
  CALL surface(r1,b1,b2,sh,sr,ll2,head2,m2*o2,sa,nsa)

  nsp=0
  nst=nc*c
  tc=0

  WRITE (20,*)INT(n/5.0/100)*100
  WRITE (20,*)nsa,tc
  DO i=1,INT(n/5.0/100)*100
     IF(i<=nsa)THEN
        WRITE (20,'(i3,3f10.4,i5)')mass(sa(i)),r1(:,sa(i))*lat,sa(i)
     ELSE
        WRITE (20,'(i3,3f10.4)')imass(1),-10.0,-10.0,sh*lat
     END IF
  END DO

  hist=0
  histz=0
  histl=0
  layer=0
  gr=0
  gz=0
  lgr=0
  dr2=(B3(3)/REAL(nlayers))+(sh+sr*2.5)/5.0
1000 DO k=ks,nc
     IF(NINT(ri(1))==-200) THEN              ! Generate random
initial cluster position on x axis for each cluster
        IF(NINT(ri(2))==-200) THEN           ! Generate random
initial cluster position on y axis for each cluster
           CALL RANDOM_NUMBER(xi)             ; CALL
RANDOM_NUMBER(yi)
           rint(1)=MINVAL(r1(1,1:s))+B1(1)*xi ;
rint(2)=MINVAL(r1(2,1:s))+B2(2)*yi
        ELSE
           CALL RANDOM_NUMBER(xi)
           rint(1)=MINVAL(r1(1,1:s))+B1(1)*xi
        END IF
     ELSEIF(NINT(ri(2))==-200) THEN
        CALL RANDOM_NUMBER(yi)
        rint(2)=MINVAL(r1(2,1:s))+B2(2)*yi
     END IF

     DO j=1+s+c*(k-1),s+c*k
        r1(:,j)=rint                         ! move the rotated
cluster(s)
     END DO

     IF (NINT(azi2)==-180) THEN
        CALL RANDOM_NUMBER(aa)
        azi=360*aa*pi/180.0_R
        vr(1)=TAN(pol)*COS(azi) ; vr(2)=TAN(pol)*SIN(azi)
        DO j=1+s+c*(k-1),s+c*k               ! Calculate initial
velocity of cluster atoms in each of the 3 directions
           v(:,j)=vr(:)*vint/SQRT(vr(1)**2+vr(2)**2+vr(3)**2)
        END DO
     ELSE
        azi=azi*pi/180.0_R
        vr(1)=TAN(pol)*COS(azi) ; vr(2)=TAN(pol)*SIN(azi)
```

```
         DO j=1+s+c*(k-1),s+c*k
            v(:,j)=vr(:)*vint/SQRT(vr(1)**2+vr(2)**2+vr(3)**2)
         END DO
      END IF

      DO WHILE(imp(k)==0.AND.var(k+s)==0)
         CALL CPU_TIME(bsum)
         ekav=0                                  ! Zero average kinetic
energy variable
         ts=tc

         nl(1)=1
         DO ij=1,loopj
            DO ik=1,loopk
               tc=tc+1
               CALL timestep(n,r1,v,a,adj,a1,f,eTot,eTotk,e0,ekav)

               IF (tc<=100) CALL
Histograms(maxbin,maxbinz,dr,dr2,dz,hist,histz,histl,layer,r1,B1,B2,
B3,sh,sr)

               DO j=1,n                                  ! Save the
trajectory of any atom that leaves the surface
                  IF(INT(j/(1+s))==1)THEN
                     IF(imp(j-
s)==0.AND.r1(3,j)<(rcut+sh).AND.a(3,j)>=0.0_R)THEN
                        imp(j-s)=tc
                        IF (j-s>=ks)THEN
                           WRITE(10,'(i7,a18,i7,es13.6e2)') j,'
impacted surface ',imp(j-s), &
                                 imp(j-
s)*dt*lat*ru*SQRT(amass(1)/(eta*eu))

WRITE(10,*)nc*c,nsp,nst,(nsp*1.0_R)/(1.0_R*nc*c),(nst*1.0_R)/(nc*c*1
.0_R)
                           BACKSPACE(10)
                        END IF
                     END IF
                  END IF
                  IF(r1(3,j)>=(rcut+sh)*1.5.AND.v(3,j)>0) THEN
                     IF (var(j)==0)THEN
                        var(j)=tc
                        IF(j-s>=k-1)THEN
                           nst=nst-1
                        ELSE
                           nsp=nsp+1
                        END IF
                        WRITE(10,'(i7,es13.6e2,3f10.5,6es14.6e2)')
j,var(j)*dt*lat*ru*SQRT(amass(1)/(eta*eu)), &

r1(:,j)*lat,v(:,j)*SQRT(eta*eu/amass(1))/ru,a(:,j)*eta*eu/(amass(1)*
lat*ru**2)

WRITE(10,*)nc*c,nsp,nst,(nsp*1.0_R)/(1.0_R*nc*c),(nst*1.0_R)/(nc*c*1
.0_R)
                        BACKSPACE(10)
                     END IF
                  END IF
```

```
IF(var(j)>0.AND.r1(3,j)<(rcut+sh).AND.a(3,j)>=0.0_R)THEN
                    var(j)=tc
                    WRITE(10,'(i7,a20,i7,es13.6e2)') j, ' reimpacted
surface ',var(j),var(j)*dt*lat*ru*SQRT(amass(1)/(eta*eu))
                    var(j)=0
                    nsp=nsp-1

WRITE(10,*)nc*c,nsp,nst,(nsp*1.0_R)/(1.0_R*nc*c),(nst*1.0_R)/(nc*c*1
.0_R)
                    BACKSPACE(10)
                 END IF
              END DO
              IF (imp(k)/=0) EXIT
           END DO                                          ! ik

           IF (tc==100) THEN
              const=4.0/3.0*PI*rho
              OPEN (14,file=gfile)
              WRITE(14,*)'rij,g(r),,,layer 1 g(r),layer 2 g(r),layer
3 g(r),layer 4 g(r),layer 5 g(r),layer 6 g(r),layer 7 g(r)' &
                   ,',layer 8 g(r),,,,,,,,,,z,g(z),,,Total number of
atoms in each layer after 100 timesteps:,'&
                   ,(layer(bin2),',',bin2=1,8)
              DO bin=1,maxbin
                 rlow=REAL(bin-1)*dr
                 rhi=rlow+dr
                 nid=const*(rhi**3-rlow**3)
                 gr(bin)=REAL(hist(bin))/REAL(s/nlayers*(nlayers-
nl(3)))/nid/100.0
                 lgr(:,bin)=REAL(histl(:,bin))/nid/layer(:)
                 IF (bin<=maxbinz)THEN

gz(bin)=REAL(histz(bin))/(B1(1)*B2(2)*rho*dz)/100.0

WRITE(14,*)rhi*lat,',',gr(bin),',,,',(lgr(bin2,bin),',',bin2=1,8),',
,,,,,,,,'&
                         ,lat*(bin*dz-(B3(3)/nlayers*(nlayers-
nl(3)))),',',gz(bin)!,',,,',kk,',',Sk(bin)
                 ELSE

WRITE(14,*)rhi*lat,',',gr(bin),',,,',(lgr(bin2,bin),',',bin2=1,8)
                 END IF
              END DO
              CLOSE (14)
           END IF

           IF (imp(k)/=0) EXIT
        END DO                                             ! ij

        CALL Celllist(m2,o2,p2,ll2,head2,r1,B1,B2,B3,limh)
        CALL surface(r1,b1,b2,sh,sr,ll2,head2,m2*o2,sa,nsa)
        WRITE (20,*)INT(n/5/100)*100
        WRITE (20,*)nsa,tc
        DO i=1,INT(n/5/100)*100
           IF(i<=nsa)THEN
              WRITE
(20,'(i3,3f10.4,i5)')mass(sa(i)),r1(:,sa(i))*lat,sa(i)
           ELSE
```

```
              WRITE (20,'(i3,3f10.4)')imass(1),-10.0,-10.0,sh*lat
           END IF
        END DO

        ttau=0
        count=0
        DO i=1,n
           IF(r1(3,i)<=sh+2.5*sr.AND.r1(3,i)>=-
B3(3)/nlayers*(nlayers-nl(3))+0.01_R)THEN
              count=count+1

ttau=ttau+0.5_R*rmass(i)*(v(1,i)**2+v(2,i)**2+v(3,i)**2)
           END IF
        END DO
        ttau=ttau/count*2.0_R/3.0_R

        WRITE(12,*)
        WRITE(12,*)'Time = ',dt*(tc)*lat*ru*SQRT(amass(1)/(eta*eu))

WRITE(17,'(i9,a,f17.8,a,f17.8,a,f17.8,a,i4,a,f11.7,a,f11.7,a,f8.2)')
tc,',',eTot*eta,',',e0*eta,',',eTotk*eta,',',nsa&
             ,',',sh*lat,',',sr*lat,',',ttau*(eta*eu/kb)
        DO j=1,n                                    ! Save
configuration at current timestep to trajectory file
           WRITE(12,'(i3,3f10.4,f4.1)')mass(j),r1(:,j)*lat,z(j)
        END DO                                      ! j

        ekav=ekav/(tc-ts)            ! Calculate the average kinetic
energy
        CALL CPU_TIME(smsum)

WRITE(*,'(i9,f14.5,f24.8,f12.6)')tc,eTot*eta,ekav*eta,(smsum-bsum)
        CALL FLUSH()                            ! Force all files to
update

     END DO
     nl(1)=nlayers-nl(2)-nl(3)
     hist=0
     histz=0
     histl=0
     layer=0
     DO ii=1,id
        CALL CPU_TIME(bsum)
        ekav=0                                  ! Zero average kinetic
energy variable

        IF(ii>=loopi) nl(1)=1
        DO ij=1,loopj
           DO ik=1,loopk
              tc=tc+1
              CALL timestep(n,r1,v,a,adj,a1,f,eTot,eTotk,e0,ekav)

              IF (ii==id.AND.ij==loopj)CALL
Histograms(maxbin,maxbinz,dr,dr2,dz,hist,histz,histl,layer,r1,B1,B2,
B3,sh,sr)

              DO j=1,n                          ! Save the trajectory
of any atom that leaves the surface
                 IF(r1(3,j)>=(rcut+sh)*1.5.AND.v(3,j)>0) THEN
```

```
                    IF (var(j)==0)THEN
                       var(j)=tc
                       IF(j-s>=k)THEN
                          nst=nst-1
                       ELSE
                          nsp=nsp+1
                       END IF
                       WRITE(10,'(i7,es13.6e2,3f10.5,6es14.6e2)')
j,var(j)*dt*lat*ru*SQRT(amass(1)/(eta*eu)), &

r1(:,j)*lat,v(:,j)*SQRT(eta*eu/amass(1))/ru,a(:,j)*eta*eu/(amass(1)*
lat*ru**2)

WRITE(10,*)nc*c,nsp,nst,(nsp*1.0_R)/(1.0_R*nc*c),(nst*1.0_R)/(nc*c*1
.0_R)
                       BACKSPACE(10)
                    END IF
                 END IF

IF(var(j)>0.AND.r1(3,j)<(rcut+sh).AND.a(3,j)>=0.0_R)THEN
                    var(j)=tc
                    nsp=nsp-1
                    WRITE(10,'(i7,a20,i7,es13.6e2)') j, ' reimpacted
surface ',var(j),var(j)*dt*lat*ru*SQRT(amass(1)/(eta*eu))
                    var(j)=0

WRITE(10,*)nc*c,nsp,nst,(nsp*1.0_R)/(1.0_R*nc*c),(nst*1.0_R)/(nc*c*1
.0_R)
                    BACKSPACE(10)
                 END IF
              END DO
           END DO                                        ! ik

           IF (ii==id.AND.ij==loopj)THEN
              const=4.0/3.0*PI*rho
              OPEN (14,file=gfile)
              WRITE(14,*)'rij,g(r)o,g(r)c,,layer 1 g(r)o,layer 2
g(r)o,layer 3 g(r)o,layer 4 g(r)o,layer 5 g(r)o,layer 6 g(r)o,',&
                   'layer 7 g(r)o,layer 8 g(r)o,layer 1 g(r)c,layer
2 g(r)c,layer 3 g(r)c,layer 4 g(r)c,layer 5 g(r)c,',&
                   'layer 6 g(r)c,layer 7 g(r)c,layer 8
g(r)c,,z,g(z)o,g(z)c,,',&
                   'Total number of atoms in each layer after 100
timesteps:,',(layer(bin2),',',bin2=1,8)
              DO bin=1,maxbin
                 rlow=REAL(bin-1)*dr
                 rhi=rlow+dr
                 nid=const*(rhi**3-rlow**3)
                 gr2(bin)=REAL(hist(bin))/REAL(s/nlayers*(nlayers-
nl(3))+k)/nid/loopk
                 lgr2(:,bin)=REAL(histl(:,bin))/nid/layer(:)
                 IF (bin<=maxbinz)THEN

gz2(bin)=REAL(histz(bin))/(B1(1)*B2(2)*rho*dz)/loopk

WRITE(14,*)rhi*lat,',',gr(bin),',',gr2(bin),',,',(lgr(bin2,bin),',',
bin2=1,8),(lgr2(bin2,bin),',',bin2=1,8),&
```

```
                          ',',lat*(bin*dz-(B3(3)/nlayers*(nlayers-
nl(3)))),',',gz(bin),',',gz2(bin)!',,',kk,',',Sk(bin),',',Sk2(bin),'
,',SE,',',SO,',',theta
                  ELSE

WRITE(14,*)rhi*lat,',',gr(bin),',',gr2(bin),',,',(lgr(bin2,bin),',',
bin2=1,8),(lgr2(bin2,bin),',',bin2=1,8)
                  END IF
               END DO
               CLOSE (14)
            END IF
         END DO                                          ! ij

         CALL Celllist(m2,o2,p2,ll2,head2,r1,B1,B2,B3,limh)
         CALL surface(r1,b1,b2,sh,sr,ll2,head2,m2*o2,sa,nsa)
         WRITE (20,*)INT(n/5.0/100)*100
         WRITE (20,*)nsa,tc
         DO i=1,INT(n/5.0/100)*100
            IF(i<=nsa)THEN
               WRITE
(20,'(i3,3f10.4,i5)')mass(sa(i)),r1(:,sa(i))*lat,sa(i)
            ELSE
               WRITE (20,'(i3,3f10.4)')imass(1),-10.0,-10.0,sh*lat
            END IF
         END DO

         ttau=0
         count=0
         DO i=1,n
            IF(r1(3,i)<=sh+2.5*sr.AND.r1(3,i)>=-
B3(3)/nlayers*(nlayers-nl(3))+0.01_R)THEN
               count=count+1

ttau=ttau+0.5_R*rmass(i)*(v(1,i)**2+v(2,i)**2+v(3,i)**2)
            END IF
         END DO
         ttau=ttau/count*2.0_R/3.0_R

         WRITE(12,*)
         WRITE(12,*)'Time = ',dt*(tc)*lat*ru*SQRT(amass(1)/(eta*eu))

WRITE(17,'(i9,a,f17.8,a,f17.8,a,f17.8,a,i4,a,f11.7,a,f11.7,a,f8.2)')
tc,',',eTot*eta,',',e0*eta,',',eTotk*eta,',',nsa&
              ,',',sh*lat,',',sr*lat,',',ttau*(eta*eu/kb)
         DO j=1,n                                 ! Save configuration at
current timestep to trajectory file
            WRITE(12,'(i3,3f10.4,f4.1)')mass(j),r1(:,j)*lat,z(j)
         END DO                                   ! j

         IF (MOD(k,s/SUM(nl))==0) limh=30.0/lat+sh

         ekav=ekav/(loopk*loopj)                  ! Calculate the average
kinetic energy
         CALL CPU_TIME(smsum)

WRITE(*,'(i9,f14.5,f24.8,f12.6)')tc,eTot*eta,ekav*eta,(smsum-bsum)
         CALL FLUSH()                             ! Force all files to
update
      END DO                                      ! ii
```

```
     OPEN(1234,file=emfile)
     CLOSE(1234,status='DELETE')
     WRITE(emfile,'(i0,a7,a6)')tc,args(2),'bk.xyz'
     WRITE(message,'(a67,i4,a13,i0)')'Backup of slab and cluster
trajectory after the impact of cluster ',k,' at timestep ',tc
     CALL
SaveConfig(emfile,r1,z,v,a,n,s,c,eta,lat,amass,eu,ru,message,adj)

  END DO


WRITE(10,*)nc*c,nsp,nst,(nsp*1.0_R)/(nc*c*1.0_R),(1.0_R*nst)/(nc*c*1
.0_R)
  CLOSE(10)

  WRITE(*,*)"Saved to: ",ofile
  CLOSE(12)

  e2=e0                                        ! Save final potential
energy for later use
  eads=e2-e1                                   ! Calculate adsorption
energy of cluster

  i=1
  j=0
  DO WHILE (i<=n)
     DO
        IF (r1(3,i+j)>=30.0_R/lat.AND.v(3,i+j)>0)THEN
           IF(i+j<=s)THEN
              s=s-1
           END IF
           j=j+1
           n=n-1
           IF (i>n) EXIT
        ELSE
           mass(i)=mass(i+j)
           r1(:,i)=r1(:,i+j)
           z(i)=z(i+j)
           v(:,i)=v(:,i+j)
           a(:,i)=a(:,i+j)
           adj(:,i)=adj(:,i+j)
           EXIT
        END IF
     END DO
     i=i+1
  END DO

  WRITE (message,'(a26,i0,a25,es13.6e2,a1)')'Slab and cluster(s)
after ',loopi*loopj*loopk,' timesteps or at time t= ', &
       dt*(loopi*loopj*loopk)*lat*ru*SQRT(amass(1)/(eta*eu)),'s'
  CALL
SaveConfig(fofile,r1,z,v,a,n,s,c,eta,lat,amass,eu,ru,message,adj)
  OPEN(1234,file=emfile)
  CLOSE(1234,status='DELETE')
  CALL SYSTEM_CLOCK(klimx)                          ! Stop timer for
Wall-clock time in ms
  CALL CPU_TIME(smsum)
  WRITE(*,*) (smsum-bsum), smsum, (klimx-n1)
```

```
  STOP

CONTAINS

  !================================================INITIALISER
SUBROUTINES=============================================================
  ! Andrew's subroutine to load variables to offset the need to
recompile
  SUBROUTINE
setup(dT,T,tT,ek,rint,ang,loopi,loopj,loopk,nl,numt,rcut,sigma,epsil
,pol,azi,ru,eu,ion,imass,amass,nc, &
       id,jump,lc,cda,lat,eta,con,q,u)
    REAL(KIND=R),INTENT(OUT)             ::
dT,T,tT,ek,rint(3),ang(3),rcut,
sigma,epsil,pol,azi,ru,eu,lat,eta,con
    INTEGER, INTENT(OUT)                 ::
loopi,loopj,loopk,nl(3),numt,ion,nc,id,lc,q,u
    REAL(KIND=R),INTENT(OUT),ALLOCATABLE :: amass(:)
    INTEGER, INTENT(OUT),ALLOCATABLE     :: imass(:)
    LOGICAL, INTENT(OUT)                 :: jump(2)
    CHARACTER(len=99)                    :: buffer
    LOGICAL, INTENT(OUT)                 :: cda
    OPEN(18,file='conditions.dat')
    DO         ! Read lines of the file looking for Preset phrases
and read in the value of variables
       READ(18,*) buffer
       IF(buffer=="dtime") READ(18,*) dT
       IF(buffer=="loops") READ(18,*) loopi,loopj,loopk
       IF(buffer=="numt") READ(18,*) numt
       IF(buffer=="lengths") READ(18,*) rcut, ru
       IF(buffer=="lenjon") READ(18,*) epsil, sigma
       IF(buffer=="sutchen") READ (18,*)lat,eta,con,q,u
       IF(buffer=="tT") READ(18,*) tT
       IF(buffer=="Temperature") READ(18,*) T
       IF(buffer=="nlayers") READ(18,*) nl(:)
       IF(buffer=="energy") READ(18,*) ek,eu
       IF(buffer=="rint") READ(18,*) rint(:)
       IF(buffer=="clusters") READ(18,*) nc,id
       IF(buffer=="angles") READ(18,*) ang(:),pol,azi
       IF(buffer=="mass")THEN
          READ(18,*)ion
          ALLOCATE(imass(ion)) ; ALLOCATE(amass(ion))
          READ(18,*)imass(:),amass(:)
       END IF
       IF(buffer=="state") READ(18,*)jump(:),lc
       IF(buffer=="RUN") EXIT
    END DO

    T=T/eta  ! Set thermal layer temperature with respect to epsilon
    dT2=dT**2
    IF (id<0) THEN
       id=-id
       cda=.TRUE.
    END IF
  END SUBROUTINE setup

  ! Andrew's routine to get sizes for allocatable arrays
  SUBROUTINE Init(filename,filename2,n,s,c,nc,lc)
    IMPLICIT NONE
```

```
    INTEGER,  INTENT(IN OUT)        :: n,s,c
    CHARACTER(LEN=75), INTENT(IN)   :: filename, filename2
    INTEGER,  INTENT(IN)            :: nc,lc
    CHARACTER(LEN=75)               :: tttt

    OPEN(11,file=filename2)
    READ(11,*)c,tttt
    OPEN(5,file=filename)
    IF (lc/=2) THEN
       READ(5,*)s,tttt
       n=s+c*nc
    ELSE
       READ(5,*)n,s,c
    END IF
    WRITE(*,*)n,s,c
    CLOSE(11)
    CLOSE(5)
  END SUBROUTINE Init

  ! Andrew's routine for input of slab and cluster co-ordinates
  SUBROUTINE
LoadConfig(filename,filename2,r1,z,v,a,n,s,c,B1,B2,B3,nc,eta,lat,ama
ss,eu,ru,lc,adj,ks,tc)
    IMPLICIT NONE
    REAL (KIND=R), INTENT(OUT)    ::
r1(:,:),z(:),v(:,:),a(:,:),eta,lat,amass(:),eu,ru
    REAL (KIND=R), INTENT(OUT)    :: B1(3),B2(3),B3(3)
    INTEGER, INTENT(IN)           :: c,nc,lc
    INTEGER, INTENT(IN OUT)       :: s,n,adj(:,:),ks,tc
    INTEGER                       :: N1,N2,N3,i
    CHARACTER(len=75), INTENT(IN) :: filename,filename2
    CHARACTER(len=99)             :: tttt

    ! Zero variables before use
    B1 = 0.0_R ; B2 = 0.0_R ; B3 = 0.0_R

    OPEN(12,file=filename)
    READ(12,*) N1,N2,N3
    IF (lc/=2)THEN
       READ(12,'(a99,3f11.6)') tttt,B1(1),B2(2),B3(3)
    ELSE
       READ(12,'(a67,i4,a13,i15,3f11.6)')
tttt,ks,tttt,tc,B1(1),B2(2),B3(3)
    END IF

    IF (lc==0) THEN
       DO i=1,s
          READ(12,*) mass(i),r1(:,i)!,z(i)
       END DO
    ELSE IF (lc==1) THEN
       i=1
       DO WHILE (i<=s)
          READ(12,'(i3,3f10.4,f4.1,6es13.5e2,3i6)')
mass(i),r1(:,i),z(i),v(:,i),a(:,i)
          DO
             IF (r1(3,i)>=30.0_R.AND.v(3,i)>0)THEN
                READ(12,'(i3,3f10.4,f4.1,6es13.5e2,3i6)')
mass(i),r1(:,i),z(i),v(:,i),a(:,i),adj(:,i)
                s=s-1
```

```
            ELSE
               EXIT
            END IF
         END DO
         r1(:,i)=r1(:,i)/lat
         v(:,i)=v(:,i)/SQRT(eta*eu/amass(1))*ru
         a(:,i)=a(:,i)*amass(1)*lat*ru**2/(eta*eu)
         i=i+1
      END DO
   ELSE
      DO i=1,n
         READ(12,'(i3,3f10.4,f4.1,6es13.5e2,3i6)')
mass(i),r1(:,i),z(i),v(:,i),a(:,i),adj(:,i)
      END DO
      r1=r1/lat
      v=v/SQRT(eta*eu/amass(1))*ru
      a=a*amass(1)*lat*ru**2/(eta*eu)
   END IF
   CLOSE(12)

   n=s+c*nc

   IF (lc/=2) THEN
      OPEN(10,file=filename2)
      READ(10,*)
      READ(10,*)
      READ(10,*)
      DO i=1,c
         READ(10,*) mass(i+s),r1(:,i+s),z(i+s)
      END DO
      CLOSE (10)

      DO j=1,nc-1  ! Move cluster atom data to the end of the
arrays
         DO i=1,c
            mass(i+s+c*j)=mass(i+s)
            r1(:,i+s+c*j)=r1(:,i+s)
            z(i+s+c*j)=z(i+s)
         END DO
      END DO
   END IF

   WRITE(*,*)'LoadConfig n = ',n
 END SUBROUTINE LoadConfig

 ! Andrew's subroutine for outputting coordinate files
 SUBROUTINE
SaveConfig(filename,r1,z,v,a,n,s,c,eta,lat,amass,eu,ru,message,adj)
   IMPLICIT NONE
   INTEGER, INTENT(IN)            :: n,s,c,adj(:,:)
   REAL (KIND=R), INTENT(IN)      ::
r1(:,:),z(:),v(:,:),a(:,:),eta,lat,amass(:),eu,ru
   INTEGER                        :: i
   CHARACTER(len=100), INTENT(IN) :: filename,message

   OPEN(13,file=filename)
   WRITE(13,'(3i6)') n,s,c
   WRITE(13,'(a99,3f11.6)') message,B1(1),B2(2),B3(3)
```

```
    DO i=1,n
       WRITE(13,'(i3,3f10.4,f4.1,6es13.5e2,3i6)')
mass(i),r1(:,i)*lat,z(i), &

v(:,i)*SQRT(eta*eu/amass(1))/ru,a(:,i)*eta*eu/(amass(1)*lat*ru**2),a
dj(:,i)
    END DO

    !WRITE(*,*)"Saved to: ",filename
    CLOSE(13)
  END SUBROUTINE SaveConfig

  !================================================MINIMIZER
SUBROUTINES============================================================
==
  ! One-dimensional minimiser from Numerical Recipes
  SUBROUTINE minp(fi,Vi,r1,s,B1,B2)
    IMPLICIT NONE
    REAL(KIND=R), INTENT(IN)     :: fi(:,:),Vi
    REAL(KIND=R), INTENT(IN OUT) :: r1(:,:)
    INTEGER, INTENT(IN)          :: s
    REAL(KIND=R), INTENT(IN)     :: B1(3),B2(3)
    REAL(KIND=R)                 :: r3(3,n)
    REAL(KIND=R)                 :: norm,dir(3,n)
    REAL(KIND=R)                 :: stepsize,xa,Va,xb,Vb,xc,Vc
    REAL(KIND=R),PARAMETER       :: CGOLD=.3819660_R,ZEPS=1.0E-10
    REAL(KIND=R)                 ::
A,B,V,Ww,w(3),XBRENT,E,FX,FV,FW,XM,TOL1,TOL2,RRR,Qq,Pp,ETEMP,D,FU,Uu
,BTOL
    INTEGER                      :: i,ITER, ineed
    r3=r1
    norm = 0.0_R
    DO i=1,s
       norm = norm + SQRT(SUM(fi(:,i)**2))           ! Magnitude of
Force
    END DO
    dir=fi/norm                                      ! Direction of
force: fi(:,:)=norm*dir(:,:)
    ! The strategy is to get the minimum bracketed between step
lengths x: xa(=0?), xb, xc
    ! When this is true, Va(=Vtotin?) > Vb < Vc
    stepsize = norm/50.0_R                           ! Trial
steplength - will be halved straight off
    xa=0.0_R
    Va=Vi
    ineed=1
    DO WHILE(ineed==1)
       stepsize=stepsize/2.0_R                       ! Make stepsize
smaller when repeating first step.
       WRITE(*,*)'Bracketing stepsize = ',stepsize,vb
       xb = xa + stepsize
       DO i=1,s
          IF(r1(3,i)>-B3(3)/nlayers*(nlayers-nl(3))+0.01_R)THEN
             r1(:,i) = r3(:,i) + xb*dir(:,i)
             IF (r1(1,i)<-0.0000000001_R)THEN     ! Adjust atoms
above images in the x-direction
                r1(1,i)=r1(1,i)+B1(1) ; adj(1,i)=adj(1,i)-1
             ELSE IF (r1(1,i)>=B1(1)) THEN
                r1(1,i)=r1(1,i)-B1(1) ; adj(1,i)=adj(1,i)+1
```

```
            END IF

            IF (r1(2,i)<-0.0000000001_R)THEN     ! Adjust atoms in
images in the y-direction
               r1(2,i)=r1(2,i)+B2(2) ; adj(2,i)=adj(2,i)-1
            ELSE IF (r1(2,i)>=B2(2)) THEN
               r1(2,i)=r1(2,i)-B2(2) ; adj(2,i)=adj(2,i)+1
            END IF
         END IF
      END DO
      CALL Celllist(m,o,p,ll,head,r1,B1,B2,B3,limh)
      CALL
Forces(rcut,r1,B1,B2,Vb,f,ll,head,cell,con,q,u,scale,w1,w,.FALSE.)

      IF(Vb<Va) THEN                             ! First step
downhill OK - find bracket....
         ineed=0
         DO
            xc = xb + stepsize
            DO i=1,s
               IF(r1(3,i)>-B3(3)/nlayers*(nlayers-
nl(3))+0.01_R)THEN
                  r1(:,i) = r3(:,i) + xc*dir(:,i)
                  IF (r1(1,i)<-0.0000000001_R)THEN     ! Adjust
atoms above images in the x-direction
                     r1(1,i)=r1(1,i)+B1(1) ; adj(1,i)=adj(1,i)-1
                  ELSE IF (r1(1,i)>=B1(1)) THEN
                     r1(1,i)=r1(1,i)-B1(1) ; adj(1,i)=adj(1,i)+1
                  END IF

                  IF (r1(2,i)<-0.0000000001_R)THEN     ! Adjust
atoms in images in the y-direction
                     r1(2,i)=r1(2,i)+B2(2) ; adj(2,i)=adj(2,i)-1
                  ELSE IF (r1(2,i)>=B2(2)) THEN
                     r1(2,i)=r1(2,i)-B2(2) ; adj(2,i)=adj(2,i)+1
                  END IF
               END IF
            END DO
            CALL Celllist(m,o,p,ll,head,r1,B1,B2,B3,limh)
            CALL
Forces(rcut,r1,B1,B2,Vc,f,ll,head,cell,con,q,u,scale,w1,w,.FALSE.)

            IF(Vc>Vb) EXIT                       ! Found far
bracket
            xa=xb                                   ! Reset a and b
points and try again to find c
            Va=Vb
            xb=xc
            Vb=Vc
         END DO
      END IF                                        ! First step
past minimum - half stepsize and try again
   END DO                                           ! While ineed=1
keep trying first step
   ! At this stage know step lengths and potentials that bracket
minimum
   WRITE(*,*)'============================================='
   WRITE(*,*)'Bracketing xa,xb,xc = '
   WRITE(*,'(3e15.5)') xa,xb,xc
```

```
    WRITE(*,*)'Potentials dVb,dVc = '
    WRITE(*,'(3f15.5)') Va-Vi,Vb-Vi,Vc-Vi
    WRITE(*,*)'============================================='
    ! Lets import BRENT here - only one function evaluation needed
per iteration!
    ! Start up with A=xa, B=xc, V=xb, FX=Fb
    A=xa
    B=xc
    V=xb
    Ww=V
    XBRENT=V
    D=0.0_R
    E=0.0_R
    FX=Vb
    FV=FX
    FW=FX
    BTOL=1.0e-2
    ITER=0
    DO
       ITER=ITER+1
       XM=0.5_R*(A+B)
       TOL1=BTOL*ABS(XBRENT)+ZEPS
       TOL2=2.0_R*TOL1
       IF(ABS(XBRENT-XM)<=(TOL2-0.5_R*(B-A))) EXIT ! Leave main
Brent iteration with best x
       IF(ABS(E)>TOL1) THEN
          RRR=(XBRENT-Ww)*(FX-FV)
          Qq=(XBRENT-V)*(FX-FW)
          Pp=(XBRENT-V)*Qq-(XBRENT-Ww)*RRR
          Qq=2.*(Qq-R)
          IF(Qq>0.0) Pp=-Pp
          Qq=ABS(Qq)
          ETEMP=E
          E=D
          IF(ABS(Pp)>=ABS(.5*Q*ETEMP).OR.Pp<=Qq*(A-
XBRENT).OR.Pp>=Qq*(B-XBRENT)) GOTO 1
          D=Pp/Qq
          Uu=XBRENT+D
          IF(Uu-A<TOL2.OR.B-Uu<TOL2) D=SIGN(TOL1,XM-XBRENT)
          GOTO 2
       ENDIF
1      IF(XBRENT>=XM) THEN
          E=A-XBRENT
       ELSE
          E=B-XBRENT
       END IF
       D=CGOLD*E
2      IF(ABS(D)>=TOL1) THEN !2
          Uu=XBRENT+D
       ELSE
          Uu=XBRENT+SIGN(TOL1,D)
       ENDIF
       ! Evaluated V here with stepsize U
       DO i=1,s
          IF(r1(3,i)>-B3(3)/nlayers*(nlayers-nl(3))+0.01_R)THEN
             r1(:,i) = r3(:,i) + Uu*dir(:,i)
             IF (r1(1,i)<-0.0000000001_R)THEN     ! Adjust atoms
above images in the x-direction
                r1(1,i)=r1(1,i)+B1(1) ; adj(1,i)=adj(1,i)-1
```

```
            ELSE IF (r1(1,i)>=B1(1)) THEN
               r1(1,i)=r1(1,i)-B1(1) ; adj(1,i)=adj(1,i)+1
            END IF

            IF (r1(2,i)<-0.0000000001_R)THEN     ! Adjust atoms in
images in the y-direction
               r1(2,i)=r1(2,i)+B2(2) ; adj(2,i)=adj(2,i)-1
            ELSE IF (r1(2,i)>=B2(2)) THEN
               r1(2,i)=r1(2,i)-B2(2) ; adj(2,i)=adj(2,i)+1
            END IF
         END IF
      END DO
      CALL Celllist(m,o,p,ll,head,r1,B1,B2,B3,limh)
      CALL
Forces(rcut,r1,B1,B2,FU,f,ll,head,cell,con,q,u,scale,w1,w,.FALSE.)
      IF(FU<=FX) THEN
         IF(Uu>=XBRENT) THEN
            A=XBRENT
         ELSE
            B=XBRENT
         ENDIF
         V=Ww
         FV=FW
         W=XBRENT
         FW=FX
         XBRENT=Uu
         FX=FU
      ELSE
         IF(Uu.LT.XBRENT) THEN
            A=Uu
         ELSE
            B=Uu
         ENDIF
         IF(FU<=FW.OR.Ww==XBRENT) THEN
            V=Ww
            FV=FW
            Ww=Uu
            FW=FU
         ELSE IF(FU<=FV.OR.V==XBRENT.OR.V==Ww) THEN
            V=Uu
            FV=FU
         ENDIF
      ENDIF
   END DO !11
   ! XBRENT is best step size found!
   ! Set positions
   DO i=1,s
      IF(r1(3,i)>-B3(3)/nlayers*(nlayers-nl(3))+0.01_R)THEN
         r1(:,i) = r3(:,i) + XBRENT*dir(:,i)
         IF (r1(1,i)<-0.0000000001_R)THEN     ! Adjust atoms above
images in the x-direction
            r1(1,i)=r1(1,i)+B1(1) ; adj(1,i)=adj(1,i)-1
         ELSE IF (r1(1,i)>=B1(1)) THEN
            r1(1,i)=r1(1,i)-B1(1) ; adj(1,i)=adj(1,i)+1
         END IF

         IF (r1(2,i)<-0.0000000001_R)THEN     ! Adjust atoms in
images in the y-direction
            r1(2,i)=r1(2,i)+B2(2) ; adj(2,i)=adj(2,i)-1
```

```
          ELSE IF (r1(2,i)>=B2(2)) THEN
             r1(2,i)=r1(2,i)-B2(2) ; adj(2,i)=adj(2,i)+1
          END IF
       END IF
    END DO
  END SUBROUTINE minp

  !=============================================LINKED LIST
SUBROUTINES============================================================
  FUNCTION ic(x,y,z,m,o,p)
    INTEGER :: ic,x,y,z,m,o,p
    ic=1+MOD(x-1+m,m)+MOD(y-1+o,o)*m+MOD(z-1+p,p)*m*o
    RETURN
  END FUNCTION ic

  ! Andrew's subroutine to generate the map of the cells
  SUBROUTINE Celllink(m,o,p,map)

    INTEGER, INTENT (IN)     :: m,o,p
    INTEGER                  :: x,y,z,im
    INTEGER, INTENT (IN OUT) :: map(:)

    DO z=1,p
       DO y=1,o
          DO x=1,m
             im=(ic(x,y,z,m,o,p)-1)*26
             map(im+1)=ic(x+1,y,z,m,o,p)  ;
map(im+2)=ic(x+1,y+1,z,m,o,p)  ; map(im+15)=ic(x+1,y-1,z,m,o,p)
             map(im+3)=ic(x,y+1,z,m,o,p)  ; map(im+14)=ic(x,y-
1,z,m,o,p)   ; map(im+4)=ic(x-1,y+1,z,m,o,p)
             map(im+16)=ic(x-1,y,z,m,o,p) ; map(im+17)=ic(x-1,y-
1,z,m,o,p)

             IF (z>1) THEN   ! Prevents Z-direction periodicity in
top layer
                map(im+5)=ic(x+1,y+1,z-1,m,o,p) ;
map(im+6)=ic(x+1,y,z-1,m,o,p)   ; map(im+20)=ic(x+1,y-1,z-1,m,o,p)
                map(im+8)=ic(x,y+1,z-1,m,o,p)   ;
map(im+18)=ic(x,y,z-1,m,o,p)    ; map(im+19)=ic(x,y-1,z-1,m,o,p)
                map(im+7)=ic(x-1,y+1,z-1,m,o,p) ; map(im+21)=ic(x-
1,y,z-1,m,o,p)  ; map(im+22)=ic(x-1,y-1,z-1,m,o,p)
             ELSE
                map(im+5:im+8)=-1 ; map(im+18:im+22)=-1
             END IF

             IF (z<p) THEN   ! Prevents Z-direction periodicity in
bottom layer
                map(im+9)=ic(x+1,y+1,z+1,m,o,p)  ;
map(im+10)=ic(x+1,y,z+1,m,o,p) ; map(im+24)=ic(x+1,y-1,z+1,m,o,p)
                map(im+12)=ic(x,y+1,z+1,m,o,p)   ;
map(im+13)=ic(x,y,z+1,m,o,p)   ; map(im+23)=ic(x,y-1,z+1,m,o,p)
                map(im+11)=ic(x-1,y+1,z+1,m,o,p) ; map(im+25)=ic(x-
1,y,z+1,m,o,p) ; map(im+26)=ic(x-1,y-1,z+1,m,o,p)
             ELSE
                map(im+9:im+13)=-1 ; map(im+23:im+26)=-1
             END IF
          END DO
       END DO
    END DO
```

```
  END SUBROUTINE Celllink

  ! Andrew's subroutine to assign all atoms to cells and generate
the linked list
  SUBROUTINE Celllist(m,o,p,ll,head,r1,B1,B2,B3,limh)

    INTEGER                    :: ic,i
    INTEGER, INTENT (IN)       :: m,o,p
    INTEGER, INTENT (IN OUT)   :: ll(:), head(:)
    REAL (kind=R), INTENT(IN) :: r1(:,:),B1(3),B2(3),B3(3),limh

    head=0                ! Resets head atom of all cells

    DO i=1,n              ! Assign every atom to a cell
       IF (r1(3,i)>limh) CYCLE
       IF (r1(3,i)>0)THEN  ! Assign atoms above slab to top layer of
cells (would get negative cell number otherwise)
          ic=1+INT((r1(1,i)/B1(1))*m)+INT((r1(2,i)/B2(2))*o)*m
       ELSE

ic=1+INT((r1(1,i)/B1(1))*m)+INT((r1(2,i)/B2(2))*o)*m+INT((r1(3,i)/(-
B3(3)))*p)*m*o
       END IF

       IF(ic<0)WRITE(*,*)i,r1(:,i),m,o,p
       ll(i)=head(ic)      ! Link current atom to head atom of the
cell
       head(ic)=i          ! Make current atom the head atom of the
cell
    END DO
  END SUBROUTINE Celllist

  !================================================TIME-STEP
INTEGRATOR
SUBROUTINES================================================

  SUBROUTINE timestep(n,r1,v,a,adj,a1,f,eTot,eTotk,e0,ekav)

    INTEGER                         :: j, CHUNK, count
    INTEGER, INTENT (IN)            :: n
    INTEGER, INTENT (IN OUT)        :: adj(:,:)
    REAL (kind=R), INTENT(IN OUT) :: r1(:,:),v(:,:),a(:,:)
    REAL (kind=R)                   ::
ektherm,chi,tau,ek(3),w1,w(3),pt,p1(3)
    REAL (kind=R), INTENT(OUT)    ::
a1(:,:),f(:,:),eTot,eTotk,e0,ekav

    CHUNK=1
    !$OMP PARALLEL PRIVATE(j) NUM_THREADS(numt)
    !$OMP DO SCHEDULE(GUIDED, CHUNK)
    DO
j=1,n                                                              !
Advance atom positions by one time-step
       r1(:,j)=r1(:,j)+dt*v(:,j)+ 0.5_R*dt2*a(:,j)

       IF (r1(1,j)/=r1(1,j).OR.r1(2,j)/=r1(2,j).OR.r1(3,j)/=r1(3,j))
THEN
          WRITE (*,*) 'Atom ',j,' has non-number positions.'
          WRITE (*,*) j, r1(:,j), v(:,j), a(:,j)
```

```
          CALL FLUSH()
          STOP
       END IF

       IF (r1(1,j)<-
0.0000000001_R)THEN                                    ! Adjust atoms
above images in the x-direction
          r1(1,j)=r1(1,j)+B1(1)
          IF(r1(3,j)<=100.0)THEN
             adj(1,j)=adj(1,j)-1
          END IF
       ELSE IF (r1(1,j)>=B1(1)) THEN
          r1(1,j)=r1(1,j)-B1(1)
          IF(r1(3,j)<=100.0)THEN
             adj(1,j)=adj(1,j)+1
          END IF
       END IF

       IF (r1(2,j)<-
0.0000000001_R)THEN                                    ! Adjust atoms
in images in the y-direction
          r1(2,j)=r1(2,j)+B2(2)
          IF(r1(3,j)<=100.0)THEN
             adj(2,j)=adj(2,j)-1
          END IF
       ELSE IF (r1(2,j)>=B2(2)) THEN
          r1(2,j)=r1(2,j)-B2(2)
          IF(r1(3,j)<=100.0)THEN
             adj(2,j)=adj(2,j)+1
          END IF
       END IF

       IF (r1(3,j)>=10000_R)THEN
          r1(3,j)=r1(3,j)-500_R
          adj(3,j)=adj(3,j)+1
       END IF

    END
DO                                                                  !
j
    !$OMP END DO
    !$OMP END PARALLEL

    CALL Celllist(m,o,p,ll,head,r1,B1,B2,B3,limh)
    CALL
Forces(rcut,r1,B1,B2,e0,f,ll,head,cell,con,q,u,scale,w1,w,.TRUE.)

    ! Calculate accelerations of atoms in each of the 3 directions
at new time-step
    a1(1,:)=f(1,:)/rmass ; a1(2,:)=f(2,:)/rmass ;
a1(3,:)=f(3,:)/rmass


count=0
  ! Zero counter of thrmostated atoms

ektherm=0
  ! Zero kinetic energy of thermostated atoms
```

```
etotk=0
  ! Zero total kinetic energy

    DO j=1,n
       IF (a1(1,j)/=a1(1,j).OR.a1(2,j)/=a1(2,j).OR.a1(3,j)/=a1(3,j))
THEN
          WRITE (*,*) 'Atom ',j,' has non-number accelerations. (a1
check)'
          WRITE (*,*) j, r1(:,j), v(:,j), a(:,j), a1(:,j)
          CALL FLUSH()
          STOP
       END IF

       IF (r1(3,j)<=-B3(3)/nlayers*(nlayers-
nl(3))+0.01_R)THEN            ! Fixed-layer atoms

a1(:,j)=0                                                          !
Zero acceleration of fixed-layer atoms

v(:,j)=0                                                           !
Zero velocity of any atom that becomes fixed
       ELSE IF (r1(3,j)>=-B3(3)/nlayers*nl(1)+0.015_R)
THEN               ! Free-layer atoms
          v(:,j)=(v(:,j)+ 0.5_R*dt*(a(:,j)+a1(:,j)))
          a(:,j)=a1(:,j)
          eTotk=eTotk+0.5_R*rmass(j)*(v(1,j)**2+v(2,j)**2+v(3,j)**2)
          ek(:)=ek(:)+0.5*rmass(j)*v(:,j)**2

ELSE                                                                !
Thermal-layer atoms
          count=count+1
          v(:,j)=(v(:,j)+
0.5_R*dt*(a(:,j)+a1(:,j)))                        ! Update velocity of
thermostated atoms

a(:,j)=a1(:,j)                                                     !
Update acceleration of thermostated atoms

ektherm=ektherm+0.5_R*rmass(j)*(v(1,j)**2+v(2,j)**2+v(3,j)**2)  !
Calculate the Kinetic energy of thermostated atoms
       END IF

       IF (a(1,j)/=a(1,j).OR.a(2,j)/=a(2,j).OR.a(3,j)/=a(3,j)) THEN
          WRITE (*,*) 'Atom ',j,' has non-number accelerations. (a
check)'
          WRITE (*,*) j, r1(:,j), v(:,j), a(:,j), a1(:,j)
          CALL FLUSH()
          STOP
       END IF

       IF (v(1,j)/=v(1,j).OR.v(2,j)/=v(2,j).OR.v(3,j)/=v(3,j)) THEN
          WRITE (*,*) 'Atom ',j,' has non-number velocities. (check
1)'
          WRITE (*,*) j, r1(:,j), v(:,j), a(:,j), a1(:,j)
          CALL FLUSH()
          STOP
       END IF
    END DO
```

```
Tau=ektherm*2/(3*count)
  ! Calculate temperature from average Kinetic energy
    Chi=(1+(dt/tT)*((T/(Tau))-
1))**0.5_R                                    ! Calculate scaling
factor

    DO j=1,n
       IF(r1(3,j)>-B3(3)/nlayers*(nlayers-
nl(3))+0.01_R.AND.r1(3,j)<-B3(3)/nlayers*nl(1)+0.015_R)THEN

v(:,j)=Chi*v(:,j)                                                    !
Update velocity of thermal atoms using scaling factor

eTotk=eTotk+0.5_R*rmass(j)*(v(1,j)**2+v(2,j)**2+v(3,j)**2)      !
Add Kinetic energy of thermostated atoms to total
          ek(:)=ek(:)+0.5*rmass(j)*v(:,j)**2
       END IF

       IF (v(1,j)/=v(1,j).OR.v(2,j)/=v(2,j).OR.v(3,j)/=v(3,j)) THEN
          WRITE (*,*) 'Atom ',j,' has non-number velocities. (check
2)'
          WRITE (*,*) j, r1(:,j), v(:,j), a(:,j), a1(:,j)
          CALL FLUSH()
          STOP
       END IF
    END DO

    pt=(2*eTotk-w1)/(3*(B1(1)*B2(2)*B3(3)))
    p1(:)=(2*ek(:)-w(:))/(B1(1)*B2(2)*B3(3))


eTot=eTotk+e0
  ! Calculate Total Energy

ekav=ekav+eTotk
  ! Add Total Kinetic energy to the average variable
  END SUBROUTINE timestep

  !================================================SUTTON-CHEN
SUBROUTINES=============================================================
  ! Andrew's subroutine to calculate the forces (Smoothed-Force
Sutton-Chen Equation with Cell index and Linked-list structure)
  SUBROUTINE
Forces(rcut,r1,B1,B2,e0,f,ll,head,cell,con,q,u,scale,w1,w,force)

    REAL (kind=R), INTENT(IN)     ::
rcut,r1(:,:),B1(3),B2(3),con,scale
    REAL (kind=R), INTENT(OUT)    :: e0,f(:,:),w(3),w1
    REAL (kind=R)                 :: rij,e,f1,den,rho(n),ct(6)
    INTEGER                       :: i,j,CHUNK,adjc,jc0,jc,ic
    INTEGER, INTENT (IN)          :: ll(:),head(:),cell,q,u
    LOGICAL, INTENT (IN)          :: force

    ! Zero energy and force before calculation
    e0=0 ; f(:,:)=0 ; rho(:)=0.0_R ; w=0 ; w1=0

    ct=0
```

```
    ct(1)=(1/rcut)**q ; ct(2)=q*1**q/rcut**(q+1) ;
ct(3)=q*(q+1)*1**q/rcut**(q+2)
    ct(4)=(1/rcut)**u ; ct(5)=u*1**u/rcut**(u+1) ;
ct(6)=u*(u+1)*1**u/rcut**(u+2)

    CHUNK=1
    !$OMP PARALLEL PRIVATE(rij,i,j,f1,e,ic,jc0,jc,adjc,den)
REDUCTION(+:e0,f) NUM_THREADS(numt)
    !$OMP DO SCHEDULE(GUIDED, CHUNK)
    DO ic=1,cell
       i=head(ic)

       DO WHILE (i>0)              ! Looks at all atoms in current
cell
          j=head(ic)

          DO WHILE (j>0)           ! Looks at all atoms in current
cell after atom i
             IF(i==j)THEN
                j=ll(j)
                CYCLE
             END IF
             rij=scale*SQRT(((r1(1,i)-r1(1,j))-(ANINT((r1(1,i)-
r1(1,j))/B1(1))*B1(1)))**2&
                  +((r1(2,i)-r1(2,j))-(ANINT((r1(2,i)-
r1(2,j))/B2(2))*B2(2)))**2&
                  +(r1(3,i)-r1(3,j))**2)

             IF (rij<=rcut)THEN    ! only considers energy and
forces within rcut
                den=(1/rij)**q-ct(1)-(rcut-rij)*ct(2)+(-
rcut**2/2+rij*rcut-rij**2/2)*ct(3)
                rho(i)=rho(i)+den
             END IF

             j=ll(j)
          END DO                   ! j

          jc0=26*(ic-1)

          DO adjc=1,26             ! Progress through the adjacent
cells
             jc=map(jc0+adjc)

             IF (jc==-1.OR.jc==ic) CYCLE     ! Skip current cell
index since no adjacent cell present
             j=head(jc)

             DO WHILE (j>0)        ! Looks at all atoms in adjacent
cell
                rij=scale*SQRT(((r1(1,i)-r1(1,j))-(ANINT((r1(1,i)-
r1(1,j))/B1(1))*B1(1)))**2&
                     +((r1(2,i)-r1(2,j))-(ANINT((r1(2,i)-
r1(2,j))/B2(2))*B2(2)))**2&
                     +(r1(3,i)-r1(3,j))**2)

                IF (rij<=rcut)THEN ! only considers energy and
forces within rcut
```

```
                  den=(1/rij)**q-ct(1)-(rcut-rij)*ct(2)+(-
rcut**2/2+rij*rcut-rij**2/2)*ct(3)
                  rho(i)=rho(i)+den
               END IF

               j=ll(j)
            END DO                ! j
         END DO                   ! adjc
         rho(i)=SQRT(rho(i))
         i=ll(i)
      END DO                      ! i
   END DO                         ! ic
   !$OMP END DO

   !$OMP DO SCHEDULE(GUIDED, CHUNK)
   DO ic=1,cell
      i=head(ic)

      DO WHILE (i>0)              ! Looks at all atoms in current
cell
         e0=e0-con*rho(i)
         j=ll(i)

         DO WHILE (j>0)           ! Looks at all atoms in current
cell after atom i
            rij=scale*SQRT(((r1(1,i)-r1(1,j))-(ANINT((r1(1,i)-
r1(1,j))/B1(1))*B1(1)))**2&
              +((r1(2,i)-r1(2,j))-(ANINT((r1(2,i)-
r1(2,j))/B2(2))*B2(2)))**2&
               +(r1(3,i)-r1(3,j))**2)

            IF (rij<=rcut)THEN    ! only considers energy and
forces within rcut
               e=(1/rij)**u-ct(4)-(rcut-rij)*ct(5)+(-
rcut**2/2+rij*rcut-rij**2/2)*ct(6)
               e0=e0+e
               IF (force) THEN
                  f1=u*1**u/rij**(u+2)-
0.5_R*con*(q*1**q/rij**(q+2)-ct(2)/rij-(rcut-rij)*ct(3)/rij)*&
                        (1./rho(i)+1./rho(j))-ct(5)/rij-(rcut-
rij)*ct(6)/rij
                  f(1,i)=f(1,i)+f1*((r1(1,i)-r1(1,j))-
(ANINT((r1(1,i)-r1(1,j))/B1(1))*B1(1)))
                  f(1,j)=f(1,j)-f1*((r1(1,i)-r1(1,j))-
(ANINT((r1(1,i)-r1(1,j))/B1(1))*B1(1)))
                  f(2,i)=f(2,i)+f1*((r1(2,i)-r1(2,j))-
(ANINT((r1(2,i)-r1(2,j))/B2(2))*B2(2)))
                  f(2,j)=f(2,j)-f1*((r1(2,i)-r1(2,j))-
(ANINT((r1(2,i)-r1(2,j))/B2(2))*B2(2)))
                  f(3,i)=f(3,i)+f1*(r1(3,i)-r1(3,j))!-
(ANINT((r1(3,i)-r1(3,j))/B3(3))*B3(3)))
                  f(3,j)=f(3,j)-f1*(r1(3,i)-r1(3,j))!-
(ANINT((r1(3,i)-r1(3,j))/B3(3))*B3(3)))
                  !                  w1=w1-f1*rij**2
                  !                  w(1)=w(1)-f1*((r1(1,i)-
r1(1,j))-(ANINT((r1(1,i)-r1(1,j))/B1(1))*B1(1)))**2
                  !                  w(2)=w(2)-f1*((r1(2,i)-
r1(2,j))-(ANINT((r1(2,i)-r1(2,j))/B2(2))*B2(2)))**2
```

```
                  !                    w(3)=w(3)-f1*(r1(3,i)-
r1(3,j))**2
               END IF
            END IF

            j=ll(j)
         END DO                    ! j

         jc0=26*(ic-1)

         DO adjc=1,13              ! Progress through the adjacent
cells
            jc=map(jc0+adjc)

            IF (jc==-1.OR.jc==ic) CYCLE     ! Skip current cell
index since no adjacent cell present
            j=head(jc)

            DO WHILE (j>0)         ! Looks at all atoms in adjacent
cell
               rij=scale*SQRT(((r1(1,i)-r1(1,j))-(ANINT((r1(1,i)-
r1(1,j))/B1(1))*B1(1)))**2&
                     +((r1(2,i)-r1(2,j))-(ANINT((r1(2,i)-
r1(2,j))/B2(2))*B2(2)))**2&
                     +(r1(3,i)-r1(3,j))**2)!-(ANINT((r1(3,i)-
r1(3,j))/B3(3))*B3(3)))

               IF (rij<=rcut)THEN ! only considers energy and
forces within rcut
                  e=(1/rij)**u-ct(4)-(rcut-rij)*ct(5)+(-
rcut**2/2+rij*rcut-rij**2/2)*ct(6)
                  e0=e0+e
                  IF (force) THEN
                     f1=u*1**u/rij**(u+2)-
0.5_R*con*(q*1**q/rij**(q+2)-ct(2)/rij-(rcut-rij)*ct(3)/rij)*&
                          (1./rho(i)+1./rho(j))-ct(5)/rij-(rcut-
rij)*ct(6)/rij
                     f(1,i)=f(1,i)+f1*((r1(1,i)-r1(1,j))-
(ANINT((r1(1,i)-r1(1,j))/B1(1))*B1(1)))
                     f(1,j)=f(1,j)-f1*((r1(1,i)-r1(1,j))-
(ANINT((r1(1,i)-r1(1,j))/B1(1))*B1(1)))
                     f(2,i)=f(2,i)+f1*((r1(2,i)-r1(2,j))-
(ANINT((r1(2,i)-r1(2,j))/B2(2))*B2(2)))
                     f(2,j)=f(2,j)-f1*((r1(2,i)-r1(2,j))-
(ANINT((r1(2,i)-r1(2,j))/B2(2))*B2(2)))
                     f(3,i)=f(3,i)+f1*(r1(3,i)-r1(3,j))!-
(ANINT((r1(3,i)-r1(3,j))/B3(3))*B3(3)))
                     f(3,j)=f(3,j)-f1*(r1(3,i)-r1(3,j))!-
(ANINT((r1(3,i)-r1(3,j))/B3(3))*B3(3)))
                     !                    w1=w1-f1*rij**2
                     !                    w(1)=w(1)-f1*((r1(1,i)-
r1(1,j))-(ANINT((r1(1,i)-r1(1,j))/B1(1))*B1(1)))**2
                     !                    w(2)=w(2)-f1*((r1(2,i)-
r1(2,j))-(ANINT((r1(2,i)-r1(2,j))/B2(2))*B2(2)))**2
                     !                    w(3)=w(3)-f1*(r1(3,i)-
r1(3,j))**2
                  END IF
               END IF
```

```
               j=ll(j)
            END DO                 ! j
          END DO                   ! adjc
          i=ll(i)
       END DO                      ! i
    END DO                         ! ic
    !$OMP END DO
    !$OMP END PARALLEL
  END SUBROUTINE Forces

  SUBROUTINE surface(r1,B1,B2,sh,sr,ll,head,cell,sa,nsa)

    REAL (kind=R), INTENT(IN)     :: r1(:,:),B1(3),B2(3)
    REAL (kind=R), INTENT(OUT)    :: sh,sr
    REAL (kind=R)                 :: rij
    INTEGER                       ::
i,j,CHUNK,adjc,jc0,jc,ic,obs(n),nnn(n)
    INTEGER, INTENT (IN)          :: ll(:),head(:),cell
    INTEGER, INTENT (OUT)         :: sa(:),nsa

    nnn=0
    obs=0
    sh=0
    sr=0
    sa=0
    nsa=0

    CHUNK=1
    !$OMP PARALLEL PRIVATE(rij,i,j,ic,jc0,jc,adjc)
REDUCTION(+:obs,nnn) NUM_THREADS(numt)
    !$OMP DO SCHEDULE(GUIDED, CHUNK)
    DO ic=1,cell*2
       i=head(ic)

       DO WHILE (i>0)                     ! Looks at all atoms in
current cell
          j=ll(i)

          DO WHILE (j>0)           ! Looks at all atoms in current
cell after atom i
             rij=SQRT(((r1(1,i)-r1(1,j))-(ANINT((r1(1,i)-
r1(1,j))/B1(1))*B1(1)))**2&
                  +((r1(2,i)-r1(2,j))-(ANINT((r1(2,i)-
r1(2,j))/B2(2))*B2(2)))**2&
                  +(r1(3,i)-r1(3,j))**2)

             IF (rij<=1.8_R)THEN    ! only considers energy and
forces within rcut
                IF(ABS(r1(3,i)-r1(3,j))/rij>0.9)THEN
                   IF(r1(3,i)>r1(3,j))THEN
                      obs(j)=obs(j)+1
                   ELSE
                      obs(i)=obs(i)+1
                   END IF
                END IF
                IF (rij<=0.9_R)THEN    ! only considers energy and
forces within rcut
                   nnn(j)=nnn(j)+1
                   nnn(i)=nnn(i)+1
```

```
               END IF
            END IF

            j=ll(j)
         END DO                    ! j

         jc0=26*(ic-1)

         DO adjc=1,13              ! Progress through the adjacent
cells
            jc=map2(jc0+adjc)

            IF (jc==-1.OR.jc==ic) CYCLE     ! Skip current cell
index since no adjacent cell present
            j=head(jc)

            DO WHILE (j>0)         ! Looks at all atoms in adjacent
cell
               rij=SQRT(((r1(1,i)-r1(1,j))-(ANINT((r1(1,i)-
r1(1,j))/B1(1))*B1(1)))**2&
                     +((r1(2,i)-r1(2,j))-(ANINT((r1(2,i)-
r1(2,j))/B2(2))*B2(2)))**2&
                     +(r1(3,i)-r1(3,j))**2)

               IF (rij<=1.8_R)THEN    ! only considers energy and
forces within rcut
                  IF(ABS(r1(3,i)-r1(3,j))/rij>0.9)THEN
                     IF(r1(3,i)>r1(3,j))THEN
                        obs(j)=obs(j)+1
                     ELSE
                        obs(i)=obs(i)+1
                     END IF
                  END IF
                  IF (rij<=0.9_R)THEN    ! only considers energy
and forces within rcut
                     nnn(j)=nnn(j)+1
                     nnn(i)=nnn(i)+1
                  END IF
               END IF

               j=ll(j)
            END DO                        ! j
         END DO                           ! adjc

         i=ll(i)
      END DO                              ! i
   END DO                                 ! ic
   !$OMP END DO
   !$OMP END PARALLEL


   DO ic=1,cell
      i=head(ic)
      DO WHILE (i>0)
         IF(nnn(i)>3.AND.nnn(i)<12)THEN
            IF(obs(i)<1)THEN
               nsa=nsa+1
               sa(nsa)=i
               sh=sh+r1(3,i)
```

```
            END IF
          END IF
          i=ll(i)
       END DO
    END DO
    sh=sh/nsa
    DO i=1,nsa
       sr=sr+(r1(3,sa(i))-sh)**2
    END DO
    sr=SQRT(sr/nsa)
  END SUBROUTINE surface

  SUBROUTINE
Histograms(maxbin,maxbinz,dr,dr2,dz,hist,histz,histl,layer,r1,B1,B2,
B3,sh,sr)

    INTEGER, INTENT(IN)        :: maxbin,maxbinz
    INTEGER, INTENT(IN OUT)    ::
hist(:),histz(:),histl(:,:),layer(:)
    INTEGER                    :: bin,bin2,i,j,chunk
    REAL (kind=R)              :: rij
    REAL (kind=R), INTENT(IN)  ::
r1(:,:),B1(3),B2(3),B3(3),sh,sr,dr,dr2,dz

    chunk=1
    !$OMP PARALLEL PRIVATE(rij,i,j,bin,bin2)
REDUCTION(+:hist,histz,layer,histl) NUM_THREADS(numt)
    !$OMP DO SCHEDULE(GUIDED, CHUNK)
    DO i=1,n-1
       DO j=i+1,n
          IF(r1(3,j)>=sh+2.5*sr.OR.r1(3,j)<-B3(3)/nlayers*(nlayers-
nl(3))+0.01_R)CYCLE
          rij=SQRT(((r1(1,i)-r1(1,j))-(ANINT((r1(1,i)-
r1(1,j))/B1(1))*B1(1)))**2&
               +((r1(2,i)-r1(2,j))-(ANINT((r1(2,i)-
r1(2,j))/B2(2))*B2(2)))**2&
               +(r1(3,i)-r1(3,j))**2)
          bin=INT(rij/dr)+1
          IF (bin<=maxbin) hist(bin)=hist(bin)+2
       END DO
    END DO
    !$OMP END DO

    !$OMP DO SCHEDULE(GUIDED, CHUNK)
    DO i=1,n
       IF(r1(3,i)>=sh+3*sr.OR.r1(3,i)<-B3(3)/nlayers*(nlayers-
nl(3))+0.01_R)CYCLE
       bin=NINT(r1(3,i)/dz)+1+NINT(B3(3)/nlayers*(nlayers-nl(3))/dz)
       IF (bin<=maxbinz) histz(bin)=histz(bin)+1
       IF(NINT((r1(3,i)+B3(3)/nlayers*(nlayers-nl(3)))/dr2)>7)THEN
          bin2=8
       ELSE
          bin2=NINT((r1(3,i)+B3(3)/nlayers*(nlayers-nl(3)))/dr2)
       END IF
       layer(bin2)=layer(bin2)+1
       DO j=1,n
          IF(r1(3,j)>=sh+2.5*sr.OR.r1(3,j)<-B3(3)/nlayers*(nlayers-
nl(3))+0.01_R)CYCLE
```

```
          rij=SQRT(((r1(1,i)-r1(1,j))-(ANINT((r1(1,i)-
r1(1,j))/B1(1))*B1(1)))**2&
               +((r1(2,i)-r1(2,j))-(ANINT((r1(2,i)-
r1(2,j))/B2(2))*B2(2)))**2&
               +(r1(3,i)-r1(3,j))**2)
          IF(rij>0.1_R)THEN
             bin=INT(rij/dr)+1
             IF (bin<=maxbin) THEN
                IF (NINT((r1(3,i)+B3(3)/nlayers*(nlayers-
nl(3)))/dr2)==NINT((r1(3,j)+B3(3)/nlayers*(nlayers-nl(3)))/dr2))
THEN
                   histl(bin2,bin)=histl(bin2,bin)+1
                ELSE
                   histl(bin2,bin)=histl(bin2,bin)+2
                END IF
             END IF
          END IF
       END DO
    END DO
    !$OMP END DO
    !$OMP END PARALLEL

  END SUBROUTINE Histograms

END PROGRAM MolecularDynamics
```

D – Kinetic Monte Carlo Code

```
PROGRAM KineticMonteCarlo
  IMPLICIT NONE
  INTEGER,PARAMETER       :: R=selected_real_kind(15,300),
Int=selected_int_kind(16)
  INTEGER,ALLOCATABLE     :: seed(:), x(:), y(:), latt(:,:), Bin(:),
xe(:), ye(:), latte(:)
  INTEGER                 :: xlat, ylat, nmon, nimp,timp
  INTEGER                 :: i, j, xdrop, ydrop, z, e,xint, yint
  INTEGER (Kind=Int)      :: c
  INTEGER                 :: xhere, yhere, dx, dy, n, d1, d2, d3,
d4, d5, sts, lts
  INTEGER                 :: nbonds1, nbonds2, nrelclus, schwoeb,
woebel
  REAL (KIND=R)           :: pdrop, prelax, nrelax, chance1,
chance2, xr, yr, T, sh, sr, stick, sputter
  REAL (KIND=R), PARAMETER:: sq3d6=SQRT(3.0_R)/6.0_R,
sq23=SQRT(2.0_R/3.0_R)
  CHARACTER (len=100)     :: args(1), lfile, rhofile, dfile, hfile,
cfile, gfile
  LOGICAL                 :: relax

  CALL RANDOM_SEED(size=n)
  ALLOCATE(seed(n))

OPEN(10,file='/dev/urandom',access='stream',form='unformatted',actio
n='read',status='old')
  READ(10)seed
  CLOSE(10)
  CALL RANDOM_SEED(put=seed)
  WRITE(*,*)seed

  CALL GETARG(1,args(1))                                ! Obtain
argument specified at run-time, used to name outputs
  WRITE(lfile,'(a4,a3,a4)')'latt',args(1),'.xyz'      ! Generates
name for lattice file
  WRITE(rhofile,'(a7,a3,a4)')'density',args(1),'.csv' ! Generates
name for density file
  WRITE(dfile,'(a7,a3,a4)')'deposit',args(1),'.csv'   ! Generates
name for density file
  WRITE(gfile,'(a4,a3,a4)')'imph',args(1),'.csv'      ! Generates
name for impact history file
  WRITE(cfile,'(a9,a3,a4)')'condition',args(1),'.dat' ! Generates
name for input condition file

  CALL
setup(cfile,xlat,ylat,stick,sputter,T,pdrop,prelax,timp,d1,d2,d3,d4,
d5,sts,lts,nrelax,woebel)

  ALLOCATE(x(xlat*ylat))       ; ALLOCATE(y(xlat*ylat))       ;
ALLOCATE(latt(xlat,ylat))      ; ALLOCATE(Bin(xlat*ylat))
  ALLOCATE(xe(xlat*ylat*3/2))  ; ALLOCATE(ye(xlat*ylat*3/2))  ;
ALLOCATE(latte(xlat*ylat*3/2))

  OPEN(7,file=lfile)
  OPEN(8,file=rhofile)
```

```
  OPEN(9,file=dfile)
  OPEN(20,file=gfile)

  latt(:,:)=0
  x=0
  y=0
  z=0
  nimp=0
  Bin=0
  CALL RANDOM_NUMBER(xr)
  CALL RANDOM_NUMBER(yr)
  xdrop=CEILING(xlat*xr)
  ydrop=CEILING(ylat*yr)

  nmon=xlat*ylat
  DO i=1,xlat
     DO j=1,ylat
        z=ylat*(i-1)+j
        x(z)=i
        y(z)=j
        latt(x(z),y(z))=0
     END DO
  END DO
  c=1
  WRITE(7,*)xlat*ylat*3/2
  WRITE(7,*)c
  DO i=1,xlat
     DO j=1,ylat
        WRITE(7,*)28,i-1+(MOD(j-1,2)+latt(i,j))*0.5_R,((j-
1)*3.0_R+latt(i,j))*sq3d6,latt(i,j)*sq23
     END DO
  END DO
  DO i=(xlat*ylat)+1,xlat*ylat*3/2
     WRITE(7,*)28,-1,-1,sh*sq23
  END DO

  DO WHILE(nimp<timp)
     c=c+1
     relax=.TRUE.
     CALL RANDOM_NUMBER(chance1)
     CALL RANDOM_NUMBER(chance2)

  !=====================================================================
  !                              IMPACT
  !=====================================================================

     IF(chance1<pdrop) THEN        !Drop particle
        IF(MOD(nimp,1000)==0)THEN

WRITE(hfile,'(a4,i5.5,a3,a4)')'hist',nimp,args(1),'.csv'      !
Generates name for histogram file
           CALL histograms(xlat,ylat,latt)
        END IF
        prelax=nrelax
        relax=.FALSE.
        CALL RANDOM_NUMBER(chance1)
        CALL RANDOM_NUMBER(chance2)
        CALL RANDOM_NUMBER(xr)
        CALL RANDOM_NUMBER(yr)
```

```
         xdrop=CEILING(xlat*xr)
         ydrop=CEILING(ylat*yr)
         IF(chance1<stick)THEN
            CALL lattopt(xdrop,ydrop,latt,xlat,ylat,1)
            z=ylat*(xdrop-1)+ydrop
            nmon=nmon+1
            x(z)=xdrop
            y(z)=ydrop
            latt(x(z),y(z))=latt(x(z),y(z))+1
         END IF
         IF(chance2<sputter)THEN
            DO
               CALL RANDOM_NUMBER(chance1)
               CALL RANDOM_NUMBER(chance2)
               dx=FLOOR((2*d1+3)*chance1)-(d1+1)
               dy=FLOOR((2*d1+3)*chance2)-(d1+1)
               xint=xdrop+dx
               yint=ydrop+dy
               IF(xint<1) xint=xint+xlat
               IF(yint<1) yint=yint+ylat
               IF(xint>xlat) xint=xint-xlat
               IF(yint>ylat) yint=yint-ylat
               CALL lattopt(xint,yint,latt,xlat,ylat,-1)
               dx=xint-xdrop
               dy=yint-ydrop
               IF(dx<-(d1+2)) dx=dx+xlat
               IF(dy<-(d1+2)) dy=dy+ylat
               IF(dx>d1+1) dx=dx-xlat
               IF(dy>d1+1) dy=dy-ylat
               IF(distance(ydrop,dx,dy,0)<=d1) EXIT ! loops until
atom chosen is within D1 interparticle spacings
            END DO
            xdrop=xdrop+dx
            ydrop=ydrop+dy
            IF(xdrop<1) xdrop=xdrop+xlat
            IF(ydrop<1) ydrop=ydrop+ylat
            IF(xdrop>xlat) xdrop=xdrop-xlat
            IF(ydrop>ylat) ydrop=ydrop-ylat
            z=ylat*(xdrop-1)+ydrop
            nmon=nmon-1
            x(z)=xdrop
            y(z)=ydrop
            latt(x(z),y(z))=latt(x(z),y(z))-1
         END IF
         Bin(z)=Bin(z)+1
         WRITE(9,'(4i8)') nimp,x(z),y(z),Bin(z)
         nimp=nimp+1
         CALL surface(xlat,ylat,latt,e,xe,ye,latte)
         sh=REAL(SUM(latte(1:e)),R)/REAL(e,R)
         sr=SQRT(SUM((latte(1:e)-sh)**2)/REAL(e,R))
         IF(MOD(nimp,sts*100)==0)
WRITE(*,'(a16,i9,2f12.6,i7)')'nmon,sh,sr,nsa =',nmon,sh,sr,e
         IF(MOD(nimp,sts)==0) WRITE(8,'(i11,2f12.6,i7)') c,sh,sr,e
         DO WHILE (prelax>0.0_R)
            prelax=prelax-1.0_R
            CALL RANDOM_NUMBER(chance1)
            CALL RANDOM_NUMBER(chance2)
            dx=FLOOR((2*d2+3)*chance1)-(d2+1)
            dy=FLOOR((2*d2+3)*chance2)-(d2+1)
```

```
            xint=xdrop+dx
            yint=ydrop+dy
            IF(xint<1) xint=xint+xlat
            IF(yint<1) yint=yint+ylat
            IF(xint>xlat) xint=xint-xlat
            IF(yint>ylat) yint=yint-ylat
            CALL lattopt(xint,yint,latt,xlat,ylat,-1)
            dx=xint-xdrop
            dy=yint-ydrop
            IF(dx<-(d2+2)) dx=dx+xlat
            IF(dy<-(d2+2)) dy=dy+ylat
            IF(dx>d2+1) dx=dx-xlat
            IF(dy>d2+1) dy=dy-ylat
            IF(distance(ydrop,dx,dy,0)<=d2)THEN
               xint=xdrop+dx
               yint=ydrop+dy
               IF(xint<1) xint=xint+xlat     ! Apply periodic BC's
               IF(yint<1) yint=yint+ylat
               IF(xint>xlat) xint=xint-xlat
               IF(yint>ylat) yint=yint-ylat
               CALL RANDOM_NUMBER(xr)
               CALL RANDOM_NUMBER(yr)
               dx=FLOOR((2*d3+3)*xr)-(d3+1)
               dy=FLOOR((2*d3+3)*yr)-(d3+1)
               xhere=xdrop+dx
               yhere=ydrop+dy
               IF(xhere<1) xhere=xhere+xlat
               IF(yhere<1) yhere=yhere+ylat
               IF(xhere>xlat) xhere=xhere-xlat
               IF(yhere>ylat) yhere=yhere-ylat
               CALL lattopt(xhere,yhere,latt,xlat,ylat,1)
               dx=xhere-xdrop
               dy=yhere-ydrop
               IF(dx<-(d3+1)) dx=dx+xlat
               IF(dy<-(d3+1)) dy=dy+ylat
               IF(dx>d3+2) dx=dx-xlat
               IF(dy>d3+2) dy=dy-ylat
               IF(distance(ydrop,dx,dy,0)<=d3)THEN
                  xhere = xdrop+dx
                  yhere = ydrop+dy
                  IF(xhere<1) xhere=xhere+xlat  ! Apply periodic BC's
                  IF(yhere<1) yhere=yhere+ylat
                  IF(xhere>xlat) xhere=xhere-xlat
                  IF(yhere>ylat) yhere=yhere-ylat
                  IF(latt(xint,yint)>=latt(xhere,yhere)) THEN
                     latt(xint,yint)=latt(xint,yint)-1
                     latt(xhere,yhere)=latt(xhere,yhere)+1
                  ELSE
                     latt(xint,yint)=latt(xint,yint)+1
                     latt(xhere,yhere)=latt(xhere,yhere)-1
                  END IF
               END IF
            END IF
         END DO
      END IF

!=====================================================================
!                       SURFACE RELAXATION
!=====================================================================
```

```
      IF (relax) THEN
         CALL RANDOM_NUMBER(chance1)      ! Choose atom to relax
         CALL RANDOM_NUMBER(chance2)
         dx = FLOOR((2*d4+3)*chance1)-(d4+1)
         dy = FLOOR((2*d4+3)*chance2)-(d4+1)
         xint=xdrop+dx
         yint=ydrop+dy
         IF(xint<1) xint=xint+xlat
         IF(yint<1) yint=yint+ylat
         IF(xint>xlat) xint=xint-xlat
         IF(yint>ylat) yint=yint-ylat
         CALL lattopt(xint,yint,latt,xlat,ylat,-1)
         dx=xint-xdrop
         dy=yint-ydrop
         IF(dx<-(d4+2)) dx=dx+xlat
         IF(dy<-(d4+2)) dy=dy+ylat
         IF(dx>d4+1) dx=dx-xlat
         IF(dy>d4+1) dy=dy-ylat
         IF(distance(ydrop,dx,dy,0)<=d4)THEN
            xint=xdrop+dx
            yint=ydrop+dy
            IF(xint<1) xint=xint+xlat     ! Apply periodic BC's
            IF(yint<1) yint=yint+ylat
            IF(xint>xlat) xint=xint-xlat
            IF(yint>ylat) yint=yint-ylat
            nrelclus=latt(xint,yint)
            CALL nncount(nrelclus,xint,yint,latt,xlat,ylat,nbonds1)
            CALL RANDOM_NUMBER(xr)
            CALL RANDOM_NUMBER(yr)
            dx = FLOOR((2*d5+3)*xr)-(d5+1)
            dy = FLOOR((2*d5+3)*yr)-(d5+1)
            xhere=xint+dx
            yhere=yint+dy
            IF(xhere<1) xhere=xhere+xlat
            IF(yhere<1) yhere=yhere+ylat
            IF(xhere>xlat) xhere=xhere-xlat
            IF(yhere>ylat) yhere=yhere-ylat
            CALL lattopt(xhere,yhere,latt,xlat,ylat,1)
            dx=xhere-xint
            dy=yhere-yint
            IF(dx<-(d5+1)) dx=dx+xlat
            IF(dy<-(d5+1)) dy=dy+ylat
            IF(dx>d5+2) dx=dx-xlat
            IF(dy>d5+2) dy=dy-ylat
            IF(distance(yint,dx,dy,0)<=d5)THEN
               xhere = xint+dx
               yhere = yint+dy
               IF(xhere<1) xhere=xhere+xlat     ! Apply periodic BC's
               IF(yhere<1) yhere=yhere+ylat
               IF(xhere>xlat) xhere=xhere-xlat
               IF(yhere>ylat) yhere=yhere-ylat
               schwoeb=(1+latt(xhere,yhere)-nrelclus)*woebel
               IF(latt(xhere,yhere)<nrelclus) THEN  ! Move possible
                  CALL
nncount(latt(xhere,yhere)+1,xhere,yhere,latt,xlat,ylat,nbonds2)
                  IF(nbonds2-schwoeb>nbonds1)THEN
                     latt(xint,yint)=nrelclus-1
                     xint=xhere
```

```
                  yint=yhere
                  latt(xint,yint)=latt(xint,yint)+1
               ELSEIF(RAND()<exp(-
(1.0_R/(0.00086_R*T))*REAL(nbonds1-(nbonds2-schwoeb))) THEN
                  latt(xint,yint)=nrelclus-1
                  xint=xhere
                  yint=yhere
                  latt(xint,yint)=latt(xint,yint)+1
               END IF
            END IF
         END IF
      END IF
   END IF

   IF (MOD(c,lts)==0) THEN
      CALL surface(xlat,ylat,latt,e,xe,ye,latte)
      WRITE(7,*)xlat*ylat*3/2
      WRITE(7,*)c
      DO i=1,e
         WRITE(7,*)28,xe(i)-1+(MOD(ye(i)-
1,2)+latte(i))*0.5_R,((ye(i)-1)*3.0_R+latte(i))*sq3d6,&
             latte(i)*sq23
      END DO
      DO i=e+1,xlat*ylat*3/2
         WRITE(7,*)28,-1,-1,sh*sq23
      END DO
   END IF
   CONTINUE

 END DO

 CALL surface(xlat,ylat,latt,e,xe,ye,latte)
 DO j=1,ylat
    WRITE(20,*)j,(Bin(ylat*(i-1)+j),i=1,xlat)
 END DO
 WRITE(7,*)xlat*ylat*3/2
 WRITE(7,*)c
 DO i=1,e
    WRITE(7,*)28,xe(i)-1+(MOD((ye(i)-1),2)+latte(i))*0.5_R,((ye(i)-
1)*3.0_R+latte(i))*sq3d6,&
       latte(i)*sq23
 END DO
 DO i=e+1,xlat*ylat*3/2
    WRITE(7,*)28,-1,-1,sh*sq23
 END DO
 STOP

 !=====================================================================
 !                            SUBROUTINES
 !=====================================================================

CONTAINS

 ! Calculates the distance between two sites
 FUNCTION distance(yin,dx,dy,dz)
   IMPLICIT NONE
   INTEGER        :: yin,dx,dy,dz
   REAL (KIND=R)  :: distance
```

```
    IF (MOD(yin,2)==0)THEN
       distance=SQRT((dx+0.5_R*(dz-
MOD(ABS(dy),2)))**2+(sq3d6*(3*dy+dz))**2+(dz*sq23)**2)
    ELSE

distance=SQRT((dx+0.5_R*(MOD(ABS(dy),2)+dz))**2+(sq3d6*(3*dy+dz))**2
+(dz*sq23)**2)
    END IF
    RETURN
  END FUNCTION distance

  ! Simple subroutine to load variables to offset the need to
recompile
  SUBROUTINE
setup(cfile,xlat,ylat,stick,sputter,T,pdrop,prelax,timp,d1,d2,d3,d4,
d5,sts,lts,nrelax,woebel)
    REAL(KIND=R), INTENT(OUT)            ::
stick,sputter,T,pdrop,prelax,nrelax
    INTEGER, INTENT(OUT)                 ::
xlat,ylat,timp,d1,d2,d3,d4,d5,sts,lts,woebel
    CHARACTER(len=100)                   :: buffer
    CHARACTER(len=100), INTENT(IN)       :: cfile
    OPEN(18,file=cfile)
    DO         ! Read lines of the file looking for Preset phrases
and read in the value of variables
       READ(18,*) buffer
       IF(buffer=="Lattice") READ(18,*)xlat,ylat
       IF(buffer=="MD") READ(18,*)stick,sputter
       IF(buffer=="Temperature") READ(18,*)T
       IF(buffer=="Probability") READ(18,*)pdrop,prelax,nrelax
       IF(buffer=="Impact") READ(18,*)timp
       IF(buffer=="Range") READ(18,*)d1,d2,d3,d4,d5
       IF(buffer=="Timestep") READ(18,*)sts,lts
       IF(buffer=="Schwoebel") READ(18,*)woebel
       IF(buffer=="RUN") EXIT
    END DO
  END SUBROUTINE setup

  SUBROUTINE surface(xlat,ylat,latt,e,xe,ye,latte)
    INTEGER, INTENT(IN)  :: xlat, ylat, latt(:,:)
    INTEGER, INTENT(OUT) :: e, xe(:), ye(:), latte(:)
    INTEGER              :: i, j, k, x2, y2, x3

    e=0
    latte=0
    xe=0
    ye=0
    DO j=1,ylat
       DO i=1,xlat
          e=e+1
          xe(e)=i
          ye(e)=j
          latte(e)=latt(i,j)
          IF(i==1)THEN
             x3=xlat
          ELSE
             x3=i-1
          END IF
          IF(j==1) THEN
```

```
            y2=ylat
         ELSE
            y2=j-1
         END IF
         IF(MOD(j,2)==1) THEN
            x2=x3
         ELSE
            x2=i
         END IF
         IF(latt(x2,y2)<latt(i,j).OR.latt(x3,j)<latt(i,j)) THEN
            DO k=1,(latt(i,j)-MIN(latt(x2,y2),latt(x3,j)))
               e=e+1
               xe(e)=i
               ye(e)=j
               latte(e)=latt(i,j)-k
            END DO
         END IF
      END DO
   END DO

  END SUBROUTINE surface

  ! Bond counter
  SUBROUTINE nncount(nrelclus,xin,yin,latt,xlat,ylat,nbonds)
    IMPLICIT NONE
    INTEGER, INTENT(IN)  :: nrelclus, xlat, ylat, latt(xlat,ylat),
xin, yin
    INTEGER, INTENT(OUT) :: nbonds
    INTEGER              :: xhere, yhere, dx, dy, i

    nbonds=0
    DO dx=-4,4                  ! Allow nnn sites in sum
       xhere = xin + dx
       IF(xhere>xlat) xhere=xhere-xlat
       IF(xhere<1)    xhere=xhere+xlat
       DO dy=-4,4
          yhere = yin + dy
          IF(yhere>ylat) yhere=yhere-ylat
          IF(yhere<1)    yhere=yhere+ylat
          DO i=0,(latt(xhere,yhere)-nrelclus)
             IF (distance(yin,dx,dy,i)>4.0_R) CYCLE
             nbonds=nbonds+1
          END DO
       END DO
    END DO
    RETURN
  END SUBROUTINE nncount

  ! Determines optimum lattice site when altering height
  SUBROUTINE lattopt(xin,yin,latt,xlat,ylat,dir)
    IMPLICIT NONE
    INTEGER, INTENT(IN)     :: xlat, ylat, latt(xlat,ylat), dir
    INTEGER                 :: xin2, xin3, yin3
    INTEGER, INTENT(IN OUT) :: xin, yin
    REAL (KIND=R)           :: rng

    xin2=xin+dir
    IF (dir>0) THEN
       xin3=xin+dir*MOD(yin-1,2)
```

```
    ELSE
       xin3=xin+dir*MOD(yin,2)
    END IF
    yin3=yin+dir
    IF (dir>0) THEN
       IF(xin2>xlat) xin2=xin2-xlat
       IF(xin3>xlat) xin3=xin3-xlat
       IF(yin3>ylat) yin3=yin3-ylat
    ELSE
       IF(xin2<1) xin2=xin2+xlat
       IF(xin3<1) xin3=xin3+xlat
       IF(yin3<1) yin3=yin3+ylat
    END IF

IF(latt(xin2,yin)*dir<latt(xin,yin)*dir.OR.latt(xin3,yin3)*dir<latt(
xin,yin)*dir) THEN
       CALL RANDOM_NUMBER(rng)
       IF((REAL(latt(xin2,yin)-latt(xin3,yin3))-0.5_R+rng)*dir>0_R)
THEN
          xin=xin3
          yin=yin3
       ELSE
          xin=xin2
       END IF
    END IF
    RETURN
  END SUBROUTINE lattopt

  SUBROUTINE histograms(xlat,ylat,latt)
    IMPLICIT NONE
    INTEGER, INTENT(IN)       :: xlat, ylat, latt(xlat,ylat)
    INTEGER, ALLOCATABLE      :: isle(:,:,:), vac(:,:,:), nisle(:),
nvac(:), is(:,:), vs(:,:), isc(:,:), vsc(:,:)
    INTEGER, ALLOCATABLE      :: ll(:,:,:,:,:), top(:,:,:,:)
    INTEGER                   :: i, j, k, x2, x3, x4, y4, x5, y5,
x6, y6, x7, y7, isles, vacs, bin, di, dv
    REAL (KIND=R),ALLOCATABLE :: isd(:,:), vsd(:,:)

    ALLOCATE (isle(xlat,ylat,MINVAL(latt(:,:)):MAXVAL(latt(:,:))))
    ALLOCATE (vac(xlat,ylat,MINVAL(latt(:,:)):MAXVAL(latt(:,:))))
    ALLOCATE (nisle(MINVAL(latt(:,:)):MAXVAL(latt(:,:))))
    ALLOCATE (nvac(MINVAL(latt(:,:)):MAXVAL(latt(:,:))))
    ALLOCATE (is(xlat*ylat,MINVAL(latt(:,:)):MAXVAL(latt(:,:))))
    ALLOCATE (vs(xlat*ylat,MINVAL(latt(:,:)):MAXVAL(latt(:,:))))
    ALLOCATE (isc(xlat*ylat,MINVAL(latt(:,:)):MAXVAL(latt(:,:))))
    ALLOCATE (vsc(xlat*ylat,MINVAL(latt(:,:)):MAXVAL(latt(:,:))))
    ALLOCATE (isd(xlat*ylat,MINVAL(latt(:,:)):MAXVAL(latt(:,:))))
    ALLOCATE (vsd(xlat*ylat,MINVAL(latt(:,:)):MAXVAL(latt(:,:))))
    ALLOCATE (ll(2,xlat,ylat,MINVAL(latt(:,:)):MAXVAL(latt(:,:)),2))
    ALLOCATE
(top(2,xlat*ylat,MINVAL(latt(:,:)):MAXVAL(latt(:,:)),2))

    is=0
    vs=0
    isc=0
    vsc=0
    isd=0
    vsd=0
    top=0
```

```
    ll=0
    isle=0
    vac=0
    DO k=MINVAL(latt(:,:))+1, MAXVAL(latt(:,:))
       isles=0
       vacs=0
       DO j=1,ylat
          DO i=1,xlat
             IF(latt(i,j)<k) CYCLE
             IF(isle(i,j,k)==0)THEN
                isle(i,j,k)=isles+1
                ll(:,i,j,k,1)=top(:,isle(i,j,k),k,1)
                top(:,isle(i,j,k),k,1)=(/i,j/)
                isles=isles+1
                is(isle(i,j,k),k)=is(isle(i,j,k),k)+1
             END IF

             IF(i==1)THEN
                x2=i+1
                x3=xlat
             ELSEIF(i==xlat)THEN
                x2=1
                x3=i-1
             ELSE
                x2=i+1
                x3=i-1
             END IF
             IF(j==1) THEN
                y4=j+1
                y5=j+1
                y6=ylat
                y7=ylat
             ELSEIF(j==ylat)THEN
                y4=1
                y5=1
                y6=j-1
                y7=j-1
             ELSE
                y4=j+1
                y5=j+1
                y6=j-1
                y7=j-1
             END IF
             IF(MOD(j,2)==1) THEN
                x4=i
                x5=x3
                x6=i
                x7=x3
             ELSE
                x4=x2
                x5=i
                x6=x2
                x7=i
             END IF
             CALL
isleassign(i,j,k,x2,j,top,isle,ll,1,is,xlat,ylat,latt,3)
             CALL
isleassign(i,j,k,x3,j,top,isle,ll,1,is,xlat,ylat,latt,3)
```

```
              CALL
isleassign(i,j,k,x4,y4,top,isle,ll,1,is,xlat,ylat,latt,3)
              CALL
isleassign(i,j,k,x5,y5,top,isle,ll,1,is,xlat,ylat,latt,3)
              CALL
isleassign(i,j,k,x6,y6,top,isle,ll,1,is,xlat,ylat,latt,3)
              CALL
isleassign(i,j,k,x7,y7,top,isle,ll,1,is,xlat,ylat,latt,3)
           END DO
        END DO
        nisle(k)=isles
        DO j=1,ylat
           DO i=1,xlat
              IF(latt(i,j)>=k) CYCLE
              IF(vac(i,j,k)==0)THEN
                 vac(i,j,k)=vacs+1
                 ll(:,i,j,k,2)=top(:,vac(i,j,k),k,2)
                 top(:,vac(i,j,k),k,2)=(/i,j/)
                 vacs=vacs+1
                 vs(vac(i,j,k),k)=vs(vac(i,j,k),k)+1
              END IF

              IF(i==1)THEN
                 x2=i+1
                 x3=xlat
              ELSEIF(i==xlat)THEN
                 x2=1
                 x3=i-1
              ELSE
                 x2=i+1
                 x3=i-1
              END IF
              IF(j==1) THEN
                 y4=j+1
                 y5=j+1
                 y6=ylat
                 y7=ylat
              ELSEIF(j==ylat)THEN
                 y4=1
                 y5=1
                 y6=j-1
                 y7=j-1
              ELSE
                 y4=j+1
                 y5=j+1
                 y6=j-1
                 y7=j-1
              END IF
              IF(MOD(j,2)==1) THEN
                 x4=i
                 x5=x3
                 x6=i
                 x7=x3
              ELSE
                 x4=x2
                 x5=i
                 x6=x2
                 x7=i
              END IF
```

```
              CALL
isleassign(i,j,k,x2,j,top,vac,ll,2,vs,xlat,ylat,latt,2)
              CALL
isleassign(i,j,k,x3,j,top,vac,ll,2,vs,xlat,ylat,latt,1)
              CALL
isleassign(i,j,k,x4,y4,top,vac,ll,2,vs,xlat,ylat,latt,2)
              CALL
isleassign(i,j,k,x5,y5,top,vac,ll,2,vs,xlat,ylat,latt,1)
              CALL
isleassign(i,j,k,x6,y6,top,vac,ll,2,vs,xlat,ylat,latt,1)
              CALL
isleassign(i,j,k,x7,y7,top,vac,ll,2,vs,xlat,ylat,latt,1)
           END DO
        END DO
        nvac(k)=vacs
     END DO
     DO k=MINVAL(latt(:,:))+1, MAXVAL(latt(:,:))
        di=0
        DO bin=1,nisle(k)
           IF(is(bin,k)==0)THEN
              di=di+1
              CYCLE
           END IF
           isc(is(bin,k),k)=isc(is(bin,k),k)+1
        END DO
        nisle(k)=nisle(k)-di
        dv=0
        DO bin=1,nvac(k)
           IF(vs(bin,k)==0)THEN
              dv=dv+1
              CYCLE
           END IF
           vsc(vs(bin,k),k)=vsc(vs(bin,k),k)+1
        END DO
        nvac(k)=nvac(k)-dv
     END DO
     OPEN(14,file=hfile)
     WRITE(14,*)'# of
Atoms',(',Islands',k,k=MINVAL(latt(:,:))+1,MAXVAL(latt(:,:))),',', &

(',Vacancies',k,k=MINVAL(latt(:,:))+1,MAXVAL(latt(:,:))),',,', &

(nisle(k),',',k=MINVAL(latt(:,:))+1,MAXVAL(latt(:,:))),',',(nvac(k),
',',k=MINVAL(latt(:,:))+1,MAXVAL(latt(:,:)))
     DO bin=1,MAX(MAXVAL(is(:,:)),MAXVAL(vs(:,:)))
        isd(bin,:)=REAL(isc(bin,:),R)/REAL(nisle(:),R)
        vsd(bin,:)=REAL(vsc(bin,:),R)/REAL(nvac(:),R)

WRITE(14,*)bin,',',(isd(bin,k),',',k=MINVAL(latt(:,:))+1,MAXVAL(latt
(:,:))),',', &
            (vsd(bin,k),',',k=MINVAL(latt(:,:))+1,MAXVAL(latt(:,:)))
     END DO
     CLOSE(14)
  END SUBROUTINE histograms

  SUBROUTINE isleassign
(i,j,k,d,e,top,temp,ll,iov,is,xlat,ylat,latt,temp3)
     IMPLICIT NONE
```

```
    INTEGER, INTENT(IN)       :: i, j, k, d, e, xlat, ylat,
latt(xlat,ylat), temp3, iov
    INTEGER, INTENT(IN OUT)   ::
temp(xlat,ylat,MINVAL(latt(:,:)):MAXVAL(latt(:,:)))
    INTEGER, INTENT(IN OUT)   ::
is(xlat*ylat,MINVAL(latt(:,:)):MAXVAL(latt(:,:)))
    INTEGER, INTENT(IN OUT)   ::
ll(2,xlat,ylat,MINVAL(latt(:,:)):MAXVAL(latt(:,:)),2)
    INTEGER, INTENT(IN OUT)   ::
top(2,xlat*ylat,MINVAL(latt(:,:)):MAXVAL(latt(:,:)),2)
    INTEGER                   :: l(2), m(2)

    IF(temp(d,e,k)/=0.AND.temp(d,e,k)/=temp(i,j,k))THEN
       l=top(:,temp(i,j,k),k,iov)
       m=ll(:,l(1),l(2),k,iov)
       DO WHILE (MAXVAL(m)/=0)

IF(temp(l(1),l(2),k)==0)WRITE(*,*)l,k,iov,temp(i,j,k),d,e,temp(d,e,k
)
          is(temp(l(1),l(2),k),k)=is(temp(l(1),l(2),k),k)-1
          temp(l(1),l(2),k)=temp(d,e,k)
          m=ll(:,l(1),l(2),k,iov)
          is(temp(l(1),l(2),k),k)=is(temp(l(1),l(2),k),k)+1
          ll(:,l(1),l(2),k,iov)=top(:,temp(l(1),l(2),k),k,iov)
          top(:,temp(l(1),l(2),k),k,iov)=l
          l=m
       END DO
    ELSE
       SELECT CASE (temp3)
       CASE(1)
          IF(latt(d,e)<k.AND.temp(d,e,k)==0)THEN
             temp(d,e,k)=temp(i,j,k)
             ll(:,d,e,k,iov)=top(:,temp(i,j,k),k,iov)
             top(:,temp(i,j,k),k,iov)=(/d,e/)
             is(temp(i,j,k),k)=is(temp(i,j,k),k)+1
          END IF
       CASE(2)
          IF(temp(d,e,k)==0)THEN
             temp(d,e,k)=temp(i,j,k)
             ll(:,d,e,k,iov)=top(:,temp(i,j,k),k,iov)
             top(:,temp(i,j,k),k,iov)=(/d,e/)
             is(temp(i,j,k),k)=is(temp(i,j,k),k)+1
          END IF
       CASE DEFAULT
          IF(latt(d,e)>=k.AND.temp(d,e,k)==0)THEN
             temp(d,e,k)=temp(i,j,k)
             ll(:,d,e,k,iov)=top(:,temp(i,j,k),k,iov)
             top(:,temp(i,j,k),k,iov)=(/d,e/)
             is(temp(i,j,k),k)=is(temp(i,j,k),k)+1
          END IF
       END SELECT
    END IF
  END SUBROUTINE isleassign

END PROGRAM KineticMonteCarlo
```

E – Example Input Files

E.1 Lattice Crystal Generator

```
!Slab setup file
LatPara
1
! Length of 100 Unit Cell
Layers
10       10       4
! Number of layers in the x,y,z direction
Planes
  1 -1  0 -1 -1  2  1  1  1
! Direction of x,y,z axis Planes (Z most important)
dpcut
1.41421356      1.22474487      1.41421356
! Dot Product cutoff distance (Will be multiplied by LatPara)
rdist
1.41421356      1.22474487      1.73205080
! Distance between Unit repetitions
RUN
```

E.2 Molecular Dynamics

```
! Input file for Molecular Dynamics code
! SYSTEM PARAMETERS
dtime
4.56499004E-4
!     timestep
loops
1     5    100
! number of updates and iterations between updates
numt
12
! number of threads used in multiprocessing
lengths
2      1e-10
! particle interaction cutoff distance and length units
lenjon
6022.3    2.214641376
! epsilon (as a temperature) and sigma parameters for Lennard-Jones
equation
sutchen
3.52      182.3418099       39.755    6       9
! Sutton-Chen parameters
mass
1
22      9.74627415e-26
! number of ions, atomic number of ions and mass of ions in kg
state
FALSE    FALSE    1
! Statements to determine if the slab needs minimised or
equilibrated and what data is loaded (0=default, 1=full, 2=no
cluster)
! SLAB PARAMETERS
tT
```

```
4.56499004E-2
! relaxation time (for thermal layers)
Temperature
300
! desired temperature of the thermal layers
nlayers
7      0       2
! number of free, thermal and fixed layers
! CLUSTER PARAMETERS
clusters
1000       8
! number of clusters and how many updates needed for the next
cluster to reach the initial point for the first cluster
energy
6366.588       1.60217656535e-19
! inital kinetic energy of the cluster and energy units
rint
-200       -200      9
! inital position of the cluster in the x, y and z direction (set x
or y to -100 to move it to the center of the slab)
angles
0       0       0       50      180
! rotation of cluster on x, y and z axis and polar and azimuthal
angle of trajectory(degrees) (set to -180 to get random angles)
RUN
```

E.3 Kinetic Monte Carlo

```
! Input file for kinetic Monte Carlo code
Lattice
224              112
! Number of x and y atoms in lattice
MD
0.835           0.427
! MD Sticking probability and sputter yield
Temperature
300
! Temperature (barely used)
Probability
0.00024         1               5
! Probability of impact and relaxation
Impact
1024000
! Total number of impacts
Range
4               4               4               1               2
! Maximum distance between two sites at various points in simulation
Timestep
16               25600000
! Number of impacts before capturing surface data and number of
timesteps before capturing trajectory data
Schwoebel
5
! Sets a number of bonds that the Schwoebel barrier is treated to be
equal to
RUN
```