Simulation Primer

An Introduction to *Simulation Kit 2.4* and *Kalypso 1.0*

M.A. Karolewski

University of Alberta

Author e-mail:
mkarol@ualberta.ca
karolewski@alum.mit.edu

Version 4 of this document, written April 2001.

The *Simulation Kit* and *Kalypso* are distributed by anonymous ftp from:
http://www.ualberta.ca/~mkarol

CONTENTS

1. INTRODUCTION

1.1. Background

Two years have passed since I released version 2.3 of the *Simulation Kit* (SK). The pressure of earning a living, moving here and there, and doing my own research has delayed the process of producing the documentation which I felt was needed for its many-body successor, *Kalypso*.

Kalypso and the SK are packages for molecular dynamics simulations of atomic collisions in metallic and bimetallic crystals. Typically energy is imparted to the system by a primary projectile such as an inert gas atom or a metal atom. The range of particle energies which can be treated by these programs ranges from ~0.1 eV to 100 keV. The lower energy limit is imposed by quantum effects, while the upper limit is determined by the treatment used for modelling inelastic effects, as well as the practical difficulty of containing fast projectiles in small targets. This energy range typically covers secondary ion mass spectrometry (SIMS) and ion scattering spectroscopy (ISS) as well as a number of less familiar techniques and processes.

I hope that these programs will stimulate interest in experimental studies of ion-surface phenomena such as sputtering and ion scattering. All too often I observe that an experimental group lacks the means to model its experiments with up-to-date simulation methods. These programs will help such groups plan their experiments and analyse their data. Success will, of course, require great perseverence on the part of users. Students who are in a hurry to write up a PhD thesis should probably look elsewhere for salvation. I have chosen to develop software for the Windows operating system because I feel that its familiarity and ubiquity on desktop machines outweighs its other disadvantages.[*]

The original reason for writing this Primer was to cover the minimum of theory needed by those wishing to model the behaviour of atomic collision systems using my first simulation package, the SK, or a similar classical dynamics package. However, in the writing I found that all but the introductory chapters became quite specific to the *Simulation Kit*. This is not necessarily a bad thing, since the reader can find excellent discussions of simulation theory in the review literature (see below), and also nowadays on the World Wide Web. The present version of the Primer now covers both the *Simulation Kit* (version 2.4) and *Kalypso* (version 1.0).

Computer simulations allow us to predict the consequences of theoretical models. They are not particularly useful when we wish to understand an entirely new phenomenon, unless it depends on physics which is already inherent in the model. The most convincing atomistic simulations use parameters which are derived objectively, and which are independent of the phenomenon being modelled. Realistically, there are times when some parameters in the model will have to be chosen heuristically for optimum fit to the experimental data (e.g. screening lengths), but this should not be overdone if the purpose of the simulation is to demonstrate a connection with the accepted body of theory.

If you need to model absolute quantities, especially sputter yields, you will probably be disappointed by the performance of computer simulation programs. This is because absolute quantities are quite sensitive to the input parameters, which may be difficult to choose optimally.

[*] A Linux version may be released in 2002 based on the new Kylix compiler from Borland.

The most interesting use of simulations is to study the relative sensitivity of an experimental quantity to a particular system parameter, e.g. the variation of sputter yields with ion energy or target orientation, or the angular distribution of ejected particles.

The discussion in this primer begins with a review of the theory of the binary collision, and very quickly moves on to the task of computer implementation of many-particle simulations. The treatment is more superficial and less rigorous than what is found in the standard texts, since its objective is to get the reader going on the simulations as quickly as possible. This primer serves as a repository of background information on the *Simulation Kit* and *Kalypso*. To a large extent, the discussion throughout the text is built around the features offered by the programs. Ideally, you should browse through this primer from cover to cover before you attempt a serious evaluation of the packages.

Snook and *Kalypso* are instruments which permit the evolution of a model system (defined by certain initial conditions and an interaction model) to be monitored over a period of time. The output is a set of particle coordinates and momenta at one or more specified sampling intervals. The programs are not designed to simulate specific experiments, so it is up to the user to manipulate the output data in a way which can be compared with experimental data (if that is the purpose of the simulation).

The user will probably find that while it is easy to generate simulation data, it is usually more difficult to extract meaningful output in the form of scattering profiles, spectra and so forth. Given a body of simulation data, it is necessary to specify one's informational requirements both in terms of the particle records which should be ignored (i.e. the filtering operation), as well as the parameter which should form the basis of averaging or of spectral analysis. Normally these operations entail 2 distinct steps (experience with databases will prove invaluable here).

For example, before plotting a spectrum of reflected projectile energies, you first need to ensure that you filter output records that refer to (a) target particles; (b) projectiles that lodged in the target. The exact filtering procedure required will depend on what output you chose to write at simulation design time (in the Run file made by *Spider*). The subject of data processing is discussed in greater depth in the manual for *Winnow* (`winnow.pdf`) which you can find in the `docs` directory of the package installation.

The best way to start learning about the packages is to run the programs *Snook* or *Kalypso* (as applicable) using the tutorial and example projects provided. The projects can be run immediately by loading the input files into *Snook* or *Kalypso*. (With *Snook*'s `iciss` project, you first need to view the `readme.txt` file found in the `examples\iciss` project directory.). Afterwards, the projects can be modified by using *Spider* to edit the input files. Two example projects are discussed in this article. One is a sputtering simulation, the other is an ion scattering simulation.

1.2. Sources

A number of references to the primary and review literature are supplied in context and may be found at the end of the primer. There are several texts and review papers which cover the central subject matter of this article especially well, which should be mentioned at the outset. These are by: Smith *et al*.,[1] Eckstein,[2] Mashkova and Molchanov,[3] Harrison,[4] Smith and Harrison,[5] Robinson[6], Niehus *et al*.[7], Rabalais *et al*.,[8] and Nastasi *et al*.[9] The reader should get hold of as

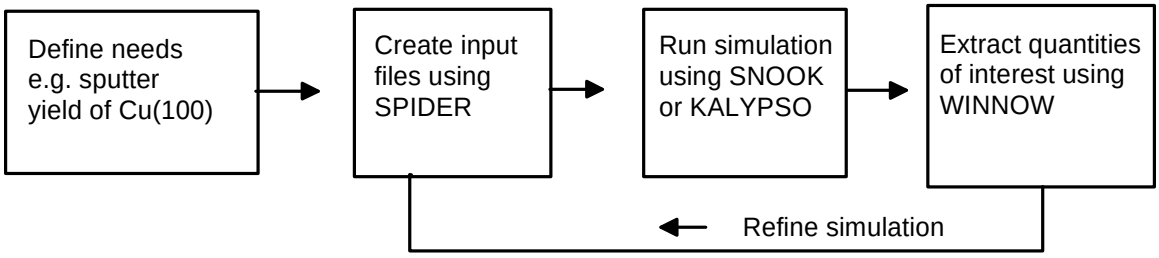
many of these key references as possible (the first two are especially valuable), since they cover practical and theoretical issues far better than this article does, and give leads to the early literature of this field, which remains remarkably relevant and valuable today. In this connection, I again wish to express my gratitude to those colleagues who took the trouble to send me reprints of their work, or to answer my questions on uncertain points.

1.3. Notational Conventions

Vector quantities are written in boldface (**r**) whereas scalar quantities are represented by an italic font (*r*), or sometimes (by mistake) plain font (r). I use *b* for the impact parameter (designated as *p* or *s* by some writers).

1.4. Package Components

The *Simulation Kit* (SK = '*Spider* for SK' + '*Snook*' + '*Winnow*') and *Kalypso* ('*Spider* for *Kalypso*' + '*Kalypso*' + '*Winnow*') packages consist of several integrated programs. The programs are very similar in terms of their interfaces and most of their internal details (except for the force calculations).



The SK is recommended for beginners to simulation, and for all ion-scattering simulations. The SK uses a Morse pair-potential to describe attractive interactions between atoms in the target, and its simulation engine, ***Snook***, is optimised for speed. The pair potential formalism used by the SK is most apt for the description of scattering processes in elemental targets. However, scattering processes that are dominated by the repulsive core potentials, such as ion scattering spectrometry, can be modelled by the SK for targets of arbitrary composition (e.g. ISS simulations).

The SK can also be applied for other purposes (with varying degrees of realism) to compound target systems that satisfy certain subsidiary conditions. For example, scattering processes in binary compounds composed of atoms with similar electronic shells (e.g. CuNi or GaAs) can be modelled on the assumption that the same long-range (attractive) interaction holds for all atoms. Another strategy is to include a planar surface potential which is applied to only one type of atom. A system of logical control flags is provided in the SK for this and other specialised applications, but the usefulness of such compromises needs to be evaluated on a case-by-case basis.

Kalypso is essentially a many-body development of the SK, which permits simulations based on the Sutton-Chen potentials or Tight-Binding potentials (the latter are also known as Gupta potentials). *Kalypso* is designed for simulations of binary compound (metallic) targets, as well as

elements. *Kalypso uses centrosymmetric many-body potentials which are not suitable for modelling the behaviour of semiconductors.*

Spider (which comes in a SK and a *Kalypso* version) is a utility program whose function is to design and generate the input data files used by the associated simulation engines (*Snook* or *Kalypso*). All simulation projects begin with *Spider*, and this is the most important program to master. ***Winnow*** is a program which processes the output data created by *Snook* or *Kalypso*, and turns this data into scientifically useful information (sputter yields, energy spectra etc.). The input files for *Snook* and *Kalypso* have different formats, so their respective *Spider* programs differ. The same version of *Winnow* is used with both packages since the output data have a common format.

It may be helpful to new users of *Winnow* to regard the *Snook* output files as database files; the purpose of *Winnow*, then, is to process user-defined queries to this database, and to present the information in an acceptable form. Users of the *Simulation Kit* and *Kalypso* will undoubtedly have to learn how to use both *Spider* and *Snook*. *Winnow* can be ignored (except as a file conversion utility) if the user is sophisticated enough to prefer his own tools (spreadsheet, statistical package, or database). The programs should first be studied by running them on the demonstration projects.

Another program supplied with both packages is `Cone`. This program, which was written by Mr Tan Hean Seng (Singapore), is extremely useful for quick calculations of shadow-cone radii in ion scattering experiments.

1.5. Documentation

Kalypso and the SK come with a large amount of documentation, which is found in the `docs` directory of the package installation:

- The Simulation Primer (this document) gives a general overview of simulation methodology as it is implemented in Kalypso and the SK
- The User Guide is a manual which explains the project development procedure in depth
- Short illustrated guides to the user interfaces of the programs (`Spider UI.pdf`, `Snook UI.pdf` or `Kalypso UI.pdf`, `Winnow UI.pdf`)
- Dumps of the online Help files for the programs in the package (`spider.pdf`, `snook.pdf` or `kalypso.pdf`, `winnow.pdf`)

1.6. Computation Time

Serious simulations of atomic collision phenomena make heavy demands on current-day personal computers. Although individual simulations run quite quickly, the time problem arises because of the need to gather adequate statistics. Sputtering simulations need several hundred runs, while ion scattering (ISS) simulations need many thousands.

Frequently, one runs a variety of similar simulations while varying one system parameter (e.g. the angle of projectile incidence). The total simulation time required for a parameter-variation 'experiment' of this kind is often on the order of days for publication quality data (i.e. good statistics). Even survey scans may consume a morning or afternoon (although here you do have the choice of concentrating only on the main features at the survey stage). The simulation engines, *Snook* and *Kalypso*, can coexist happily with other (well-behaved) Windows 95

programs;* nor do the programs need to be monitored as they run. The best time-management strategy, therefore, is to run long simulation projects as background batch jobs. You can even halt them for temporarily CPU-intensive work with other programs, or the program interfaces can be 'minimised' and put out of sight while the computer is used for other work. You can read about batch processing in the online Help for the simulation programs; *Spider*'s gadgets menu has a utility for generating 'batch definition file' templates.

The speed of execution of *Snook*/*Kalypso* can be optimised by a correct setting of various of the simulation options parameters. Again, the reader is referred to the online Help. In brief, to speed up *Snook*/*Kalypso* you should: (a) disable screen output options; (b) minimise the program window; (c) close the Graph window (which displays atomic trajectories).

The author doubts whether ion-surface simulations will ever execute quickly in a package of this nature. As computer speeds increase, one's ambitions grow in equal measure.

* Sadly, many common programs (notoriously Netscape Navigator) are not well-behaved and will crash or freeze your system if you run them repeatedly over several days (or even once!) without rebooting.

2. CLASSICAL SCATTERING THEORY

2.1. Introduction

The interatomic interaction is represented by a potential. The potential defines the system energy as a function of the particles' relative positions. Typically the strength of the interaction is assumed to depend only on the separation (r_{12}) of the two particles. This is termed a centrosymmetric (or radial) potential.

The potential of an *N*-body system may be equivalent to the sum of the *½N(N-1)* atom-atom potentials in the system, in which case the potential is described as a pair potential (the interactions are said to be pairwise). If the potentials do not add together in pairwise fashion, the potential is said to have a many-body or *N*-body character. What that often means in practice is that the potential of an *N*-body system is a non-linear function (e.g. a square root function) of *½N(N-1)* individual pairwise potentials.

Moving atoms are deflected from their straight-line trajectories by the forces associated with interatomic potentials This process is known as scattering. The classical scattering theory described in undergraduate textbooks deals predominantly with the binary collision, involving a two-body, radial potential, $V(r_{12})$, in which two interacting particles first approach each other, and later recede. Good graduate-level text-book treatments of scattering can be found in Goldstein,[10] or in Landau and Lifshitz.[11] A nice contemporary treatment can be found in ref. [8].

The scattering terminology applies to forces which have a finite range and tend to zero as the separation increases. However, the decline of the potential with separation need not be monotonic (interatomic potentials usually have a minimum). The classical scattering theory is most useful for understanding the scattering of fairly energetic particles (E > 10 eV) at small separations (< 2 Å), whose interaction can be described by via pairwise repulsive potentials. See refs. [3,8] for thorough reviews of the classical scattering theory, and its applications in atomic physics.

More complex interaction potentials can be envisaged. In metals, valence electrons are delocalised, so bonding forces cannot be treated realistically using pairwise, point-to-point, potentials. Empirically, it is observed that the chemical bonds weaken as coordination number increases, which defies explanation in terms of pairwise interactions. For reasons such as these, *Kalypso* employs a many-body potential.

For the most part, the collisions discussed in this chapter (and in this article) will be elastic collisions. An elastic collision is one in which the sum of the kinetic energies of the particles after the collision, $E_1 + E_2$, is the same as it was before the collision, E_0 (see Fig. 2.1). Inelastic collisions are those in which energy is transferred to some internal mode of excitation (e.g. electronic excitations) or dissipated (e.g. by friction). Chapter 8 will describe the simulation of inelastic processes in atomic collision systems. The purpose of the following review is not to work through the equations of classical scattering theory (which can be found in any mechanics textbook), but to summarise the objectives and achievements of the classical theory which are useful in the context of atomic collisions.

2.2. Kinematics of Binary Collision

The classical scattering problem can be analysed at several different levels. At the first level, the kinematic analysis involves the application of conservation laws (energy, momentum) to the problem, and establishes what relationships hold between respectively the collisional energy transfer, the scattering angles and the relative masses of the participating particles (see Fig. 2.1 for explanation of symbols). These kinematic results are obtained from a consideration of the asymptotic particle trajectories, so they are valid for any type of central scattering potential.

Scattering under the influence of a central potential $V(r)$ takes place in a plane. The geometry of the scattering process is depicted in Fig. 2.1, where the target is initially at rest. This coordinate system is known as the Laboratory coordinate system (Lab system or Lab frame), to distinguish it from the Centre-of-Mass reference frame (COM system or COM frame) introduced for analytical convenience in the following section.

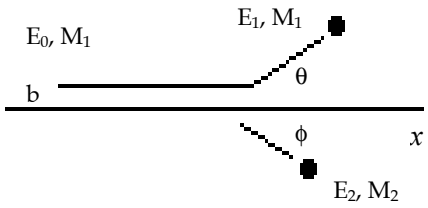


Figure 2.1. Scattering geometry for a binary collision. The projectile of mass M_1, and energy E_0, approaches a stationary target of mass M_2 located on the x-axis. After the interaction, the motion of the projectile is characterised by the scattering angle θ and its asymptotic energy E_1, while that of the target is characterised by the recoil angle ϕ and its asymptotic energy E_2 (= E_0 -E_1 in an elastic collision). b (the off-axis distance of the projectile trajectory) is the impact parameter.

The most useful equation produced by the kinematical theory is one which relates the collisional energy transfer to the projectile scattering angle in the Lab system:

$$E_1 = E_0((M_1 M_2 / (M_1 + M_2))^2 \{\cos\theta \pm [(M_2 / M_1)^2 - \sin^2\theta]^{1/2}\}^2 \qquad (2.1)$$

Both *Spider* and *Winnow* have gadgets which allow you to calculate relationships based on Eq. 2.1. Equation 2.1 simplifies considerably if q = 90° or 180°. This equation is the basis of ion scattering spectrometry (ISS), which involves the measurement of E_1 for fixed E_0, M_1 and q, and thereby allows the mass M_2 of the scatterer to be deduced. For $(M_2/M_1) < 1$, only the positive sign before the $[(M_2/M_1)^2 - \sin^2 q]^{1/2}$ term is applicable. Light projectiles are backscattered from heavy targets in small impact parameter collisions, whereas heavy projectiles push light targets before them, and only undergo small deflections.

The following remarks in this section apply to a potential which falls off monotonically with separation.

If $M_1 < M_2$, the projectile may be scattered over the entire range of q (0-180°). However, for a heavy projectile ($M_1 > M_2$), the deflection angle q increases as b increases, up to a maximum

value given by $\sin(q) = M_2/M_1$. This tendency is illustrated by the plot of trajectories shown in Fig. 2.2 (for argon scattering by a carbon atom target) which is taken from ref. [12].

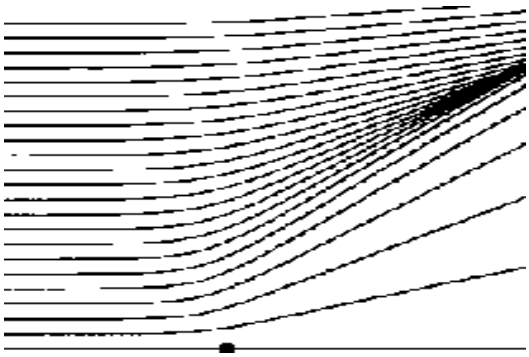


Figure 2.2. Trajectories followed by 1 keV Ar projectiles (mass 40) scattered from a lighter carbon target (mass 12, initial location shown as black dot). The impact parameters of the various trajectories are separated by 0.05 Å.

The energy of the recoiling target particle is given by the kinematic theory as:

$$E_2 = 4E_0M_1M_2\cos2\phi/(M_1+M_2)^2 \qquad (2.2)$$

The angles q and ϕ are related by a fairly complicated expression (it is easier to relate ϕ to the scattering angle in the COM system). According to equation (2.2), the maximum energy transferred to the target particle in an elastic collision, E_{max}, is:

$$E_{max}/E_0 = 4M_1M_2/(M_1+M_2)^2 \,, \qquad (2.3)$$

which occurs when the impact parameter is zero: this is called a direct impact (or centre-to-centre) collision. Equation 2.3 implies that energy transfers decline as the disparity in masses of the particles increases. For $M_1 = M_2$, $E_{max}/E_0 = 1$, whereas for $M_1 = 0.1M_2$ (or $M_2 = 0.1M_1$), $E_{max}/E_0 = 0.33$. In a proton-electron collision, $E_{max}/E_0 = 0.002$. Gryzinski's paper, which is something of a classic, gives a very thorough review of the results of the kinematic theory, including the cases involving an inelastic energy loss or a non-stationary target.[13]

2.3. Dynamics of Binary Collision

2.3.1. The Scattering Cross-Section

The dynamical theory of scattering is concerned with the determination of particle trajectories and scattering cross-sections for specific interaction potentials. To simplify the discussion we assume that $M_1 < M_2$, and that the potential falls off monotonically with separation. Consider an experiment in which we measure the fraction of a projectile beam of incident intensity I which is scattered into an angular range q to $q+dq$ by two different target potentials $V_1(r)$ and $V_2(r)$. In the first case, this scattering will be associated with impact parameters between b_1 and b_1+db_1, and in the second case with impact parameters between b_2 and b_2+db_2. In other words, there is a distinct mapping b ® q of a given impact parameter to a specific scattering angle for each type of potential. In three dimensions, the angular range q *to* $q+dq$ represents an annular-shaped solid angle $d\Omega$, as depicted in Fig.2.3.

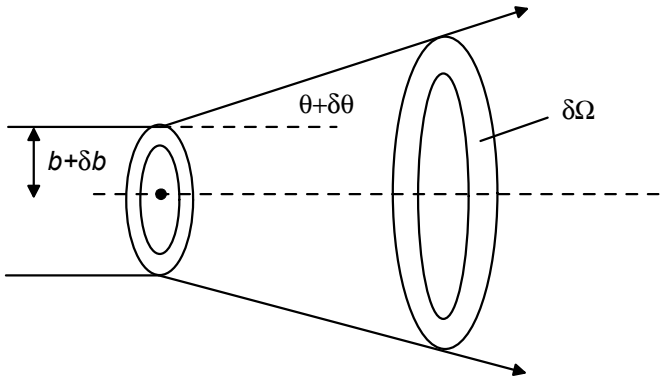


Figure 2.3. Depiction of a scattering process in three dimensions, showing how projectiles incident within an annulus defined by impact parameters in the range b to $b+db$ are scattered (deflected) into a solid angle $d\Omega$.

If the incident particle flux impinging on the target has a uniform cross-section, then the number of incident particles in a small impact parameter range would be proportional to the area of an annulus of radius b and width $b+db$, i.e $2\pi bdb$. Due to particle conservation, the number of particles $\delta N(\Omega)$ scattered into the specified solid angle $\delta\Omega$ (representing the area of a detector) would also be proportional to the area of this contributing annulus of impact parameters. The scattering cross-section $\sigma(\Omega)$, which can be expressed as a function of q (as well as E_0, M_1 and M_2), is the coefficient which relates the flux of incident particles (I) to the number of scattered particles detected at a particular deflection:

$$dN(\Omega) = I\sigma(\Omega)d\Omega \tag{2.4}$$

The concept of a scattering cross-section is useful when comparing the behaviour of a similar process in two different scattering systems: for example: 'the cross-section for 90° scattering is higher for Cu than for Al' (meaning, more scattered particles are detected for Cu at this angle).

If the potential is non-monotonic, or if $M_1 > M_2$, then scattering by an angle q may be associated with more than one value of b (see Fig. 2.2, for example). Effects of the first kind are important in determining the cross-sections of low-energy scattering processes, where they give rise to the rainbow scattering effect.

The concept of the scattering cross-section is closely related to that of a collision radius (or alternatively, the collision diameter). Some physical processes (e.g. certain kinds of electron emission) can only be initiated if the collision impact parameter is below some critical threshold, b_{min}, known as the collision radius. The cross-section for such a process is equal to the area of the circle defined by a radius b_{min}. The cross-section is often replaced by the idea of a *yield* in solid-state scattering processes (e.g. the sputter yield, the secondary electron yield and so on): the yield is a measure of the average detection/emission probability for a signature particle associated with the process, referenced to the flux of stimulating particles (electrons per incident ion, etc.).

2.3.2. The Coulomb Potential

The Coulomb potential is described by an inverse square force law ($1/r^2$), which gives rise to a potential energy function of the form:

$$V(r) = \frac{Z_1 Z_2 e^2}{4\pi\varepsilon_0 r}, \tag{2.5}$$

at a distance r from a point charge Z_2e, while Z_1e is the charge of the interacting projectile. (In a nuclear/atomic context, Z_1 and Z_2 represent atomic numbers, while e is the charge of the proton). $\boldsymbol{\varepsilon}_0$ is the vacuum permittivity constant (8.854×10^{-12} F m $^{-1}$). The equation of energy conservation for the Coulomb scattering problem in the Lab frame is:

$$\tfrac{1}{2}M_1 v_1^2 + \tfrac{1}{2}M_2 v_2^2 + \frac{Z_1 Z_2 e^2}{4\pi\varepsilon_0 |\mathbf{r}_1 - \mathbf{r}_2|} = E_0, \tag{2.6}$$

where v_i (i = 1,2) is the magnitude of the velocity of the ith particle (= $|\mathbf{i}v_x + \mathbf{j}v_y|$).

The goal of the analysis is to predict the scattering angle as a function of the impact parameter, b. The method of solution involves, as a first step, expressing the problem in a new system of coordinates ($\mathbf{R}$, $\mathbf{r_{12}}$) which are linear combinations of the coordinates ($\mathbf{r_1}$, $\mathbf{r_2}$) in the Lab frame; the first of these represents the position of the system centre-of-mass (COM):

$$\mathbf{R} = (M_1\mathbf{r_1} + M_2\mathbf{r_2})/(M_1+M_2), \tag{2.7}$$

while the relative particle position ($\mathbf{r_{12}}$) is given by:

$$\mathbf{r_{12}} = \mathbf{r_1} - \mathbf{r_2} \tag{2.8}$$

In terms of these variables, equation (2.6) can be expressed as:

$$\tfrac{1}{2}MV^2 + \tfrac{1}{2}m v_{12}^2 + \frac{Z_1 Z_2 e^2}{4\pi\varepsilon_0 r_{12}} = E_0 \tag{2.9}$$

where V = d$\mathbf{R}$/dt, v_{12} = d$\mathbf{r_{12}}$/dt, M (the total mass) = M_1+M_2, and μ (the reduced mass) = $M_1M_2/(M_1+M_2)$. Equation 2.9 implies the following equations of motion:

$$M\mathrm{d}^2\mathbf{R}/\mathrm{dt}^2 = 0 \tag{2.10}$$

$$\mu\frac{d^2\mathbf{r}_{12}}{dt^2} = \frac{Z_1 Z_2 e^2 \mathbf{r}_{12}}{4\pi\varepsilon_0 r_{12}^3} \tag{2.11}$$

According to equation 2.10, the COM will be in uniform motion throughout the collision, giving rise (equation 2.9) to a constant associated kinetic energy of $\frac{1}{2}MV^2$. At the start of the collision,

the interaction term is negligible (r_{12} is large), and the target has zero velocity (v_2 = 0), so equations 2.6 and 2.9 can be expressed as:

$$½MV^2(0) + ½mv_{12}^2(0) = E_{COM} + E_{12} = ½M_1v_1^2(0) = E_0\,, \tag{2.12}$$

where E_{COM} is the constant kinetic energy associated with the COM motion; E_{12} (the relative energy, or energy in the COM frame), which is also constant, is the energy available for the *pseudo* 1-body scattering problem expressed by Eq. 2.11, and initial condition values are indicated by $V^2(0)$ etc. The important point to appreciate is that the distance of closest approach (and other features of the collision) depends on the relative energy, E_{12}, rather than the Lab incident energy, E_0.

By using the relation:

$$½MV^2(0) + ½mv_{12}^2(0) = ½M_1v_1{}^2(0)\,, \tag{2.13}$$

it is not difficult to show that:

$$E_{12} = M_2/(M_1+M_2)E_0. \tag{2.14}$$

Evidently, E_{12} approaches E_0 as the target mass increases relative to the projectile mass. Eq. 2.9 can be expressed as:

$$½\,\mu v_{12}{}^2 + Z_1Z_2e^2/(4\pi\varepsilon_0 r_{12}) = E_{12} = M_2/(M_1+M_2)E_0. \tag{2.15}$$

Example. A 1 MeV proton is projected at a stationary alpha particle (He nucleus). What is the distance of closest approach (r_{min}) in a direct impact collision? Would it be the same if the alpha particle were projected towards the proton with the same initial energy?

Solution. The distance of closest approach is reached when all of the relative kinetic energy E_{12} is converted to potential energy. Hence, the first term in equation 2.15 is zero, giving:
$Z_1Z_2e^2/(4\pi\varepsilon_0 r_{12}) = M_2/(M_1+M_2)E_0$. Using Z_1 = 1, Z_2 = 2, $e = 1.6\times10^{-19}$ C, $M_2/(M_1+M_2) = 0.8$, $\varepsilon_0 = 8.85\times10^{-12}$ F m^{-1} , $E_0 = 10^6 e$ J, we obtain $r_{12} = 3.6\times10^{-15}$ m. For the case of an alpha particle projectile, the $M_2/(M_1+M_2)$ term is now 0.2, and the distance of closest approach is accordingly four times larger (1.4×10^{-14} m).

Eq. (2.11) is formally equivalent to the equation of motion of a particle of mass μ moving under the influence of a Coulomb potential originating from a fixed centre. The equation of motion 2.11 can sometimes be solved analytically. The Cartesian coordinate system is converted to a circular system: (x_{12}, y_{12}) ® (r_{12}, q). Standard texts show that the scattering angle in this coordinate system (the COM system) can be expressed as:

$$\theta = \pi - 2\cdot\int_{r\,\mathrm{min}}^{\infty} \frac{b\cdot dr_{12}}{r_{12}\sqrt{(1-V(r_{12})/E_{12})\cdot \mathrm{r}_{12}{}^2 - b^2}} \tag{2.16}$$

where $V(r_{12}) = Z_1 Z_2 e^2 /(4\pi\varepsilon_0 r_{12})$. This integral can be evaluated for the Coulomb potential, as well as a few others such as the inverse square potential (the latter is quite useful for testing simulation programs because it represents a screened Coulomb potential). For the Coulomb potential, $V(r_{12}) = \mathrm{A}/r_{12}$, equation 2.16 evaluates as:[3]

$$q^* = p - 2\arctan(2bE_{12} / A) \tag{2.17}$$

where the asterisk (*) reminds us that the angle in question refers to the COM coordinate system. The corresponding result for the inverse square potential, $V(r_{12}) = A/r_{12}^2$, is:

$$q^* = p\left[1-(1+A/(b^2E_{12}))^{-\frac{1}{2}}\right] \tag{2.18}$$

The COM result, equation 2.17, can be expressed in the Lab frame by means of transformations shown in the standard texts on mechanics, and in a similar fashion, scattering cross-sections for the laboratory system can be derived. These are not particularly interesting for the present discussion, so they are passed over here. However, it is important to emphasise that knowledge of the Lab scattering angle is not enough to specify the particles' trajectories in the Lab frame, although it does specify their asymptotic Lab directions of motion. To know the trajectory in the interaction region, one would have to solve for the variation of r_{12} with time, and express this motion in the coordinates of the Lab frame. This is quite an involved problem, which also requires the evaluation of the so-called time integral, which gives the interval t between the initial and final limits of the motion (r_1, r_2 respectively):

$$t = \int_{r1}^{r2} \frac{dr}{\sqrt{(2/\mu)\cdot[E-U(r_{12})]-M^2/\mu^2 r_{12}^{\,2}}} \tag{2.19}$$

For the Coulomb potential, the time integral diverges as the initial radial separation increases, which means that it cannot be evaluated for the ideal case of a projectile 'approaching from infinity'. Computer simulations of atomic collisions based on the binary collision approximation (BCA) model do make use of the time integral (for the screened Coulomb interaction). The reader is referred to the research literature for further discussion of this topic.[14]

In BCA simulations of atomic scattering in solids, it is important to be able to track the actual motion of the projectile (taking the displacement of the target into account) because of the need to calculate the impact parameter for a possible subsequent collision with another target atom.

2.3.4. Other Potentials

The integral in equation 2.16 can only be evaluated analytically for a few potentials. A number of approximate analytical treatments have been developed in response to this. The small-angle (or impulse) approximation introduces a number of assumptions which simplify the scattering integral, but it is not applicable to hard (small impact parameter) collisions.[3] Another approach is to develop universal scattering formulae based on regression fits to parametric models.[15] Approximate techniques of this kind have an important role to play in simulation models based on the BCA, where a multiple scattering process is represented as a sequence of binary

collisions. The BCA model is the basis of some well-known computer simulation programs (TRIM, MARLOWE). The main differences between the CD and BCA programs are that (a) the BCA programs only consider a subset of the target atom interactions; (b) the BCA programs rely on algorithmic prescriptions for dealing with quasi-simultaneous scattering configurations; (c) BCA programs tend to neglect attractive interactions in the target. The BCA model will not be discussed further in this primer: see the books by Smith *et al*.[1] or by Eckstein[2] for compact reviews of the model, or the article by Robinson[6] for a comprehensive review.

2.3.5. Many-Body Systems

More interesting (from the viewpoint of the surface analyst) is a system in which a projectile is scattered by two fixed centres. The case of 2-centre scattering has been thoroughly investigated by mathematicians (Euler reported the integrability of the 2-centre Coulomb problem in 1760).[16] Where the projectile-target interaction is governed by screened Coulomb potentials no analytic solution of the 2-centre problem is possible, but the theoretical analysis of this system by approximate methods remains a subject of current research.[17]

In the normal conception of a many-particle system, the scattering centres (target particles) would undergo displacement as a result of interaction with each other as well as with the projectile. In the early simulation literature it was assumed that all interactions could be expressed using pairwise potentials. Thus, an N-body system accordingly gives rise to ½N(N-1) pairwise interactions which govern motion in the system. The motion of the ith particle in a many-particle system is then determined by the resultant of all of the forces acting upon it, which can be expressed in the following vector equation ($m\boldsymbol{a}_i = \boldsymbol{F}_i$):

$$M_i \frac{d^2 r_i}{dt^2} = -\sum_n \frac{\partial V(|\mathbf{r}_i - \mathbf{r}_n|)}{\partial \mathbf{r}_i} = -\sum_n \nabla_i V(|\mathbf{r}_i - \mathbf{r}_n|) \tag{2.20}$$

where the summation runs over the index n (n = 1...N, omitting the value i). The derivatives are evaluated by means of 'chain-rules' like the following:

$$\partial \mathrm{V}(r_{12})/\partial x_1 = \partial r_{12}/\partial x_1 * \partial \mathrm{V}(r_{12})/\partial r_{12} = (x_1\text{-}x_2)/r_{12} * \partial \mathrm{V}(r_{12})/\partial r_{12}\,, \tag{2.21}$$

where $r_{12} = [(x_1\text{-}x_2)^2 + (y_1\text{-}y_2)^2 + (z_1\text{-}z_2)^2]^{1/2}$. Terms like $(x_1\text{-}x_2)/r_{12}$ are known as 'direction cosines'.

Here is a concrete example of the evaluation of equation 2.20, for the case when particle 1 is interacting via Coulomb forces with two others (i = 1, n = 2,3):

$$M_1 d^2x_1/\mathrm{d}t^2 = Z_1e^2/(4\pi\varepsilon_0).\{\, Z_2(x_1\text{-}x_2)/r_{12}^3 + Z_3(x_1\text{-}x_3)/r_{13}^3 \,\} \tag{2.22a}$$
$$M_1 d^2y_1/\mathrm{d}t^2 = Z_1e^2/(4\pi\varepsilon_0).\{\, Z_2(y_1\text{-}y_2)/r_{12}^3 + Z_3(y_1\text{-}y_3)/r_{13}^3 \,\} \tag{2.22b}$$
$$M_1 d^2z_1/\mathrm{d}t^2 = Z_1e^2/(4\pi\varepsilon_0).\{\, Z_2(z_1\text{-}z_2)/r_{12}^3 + Z_3(z_1\text{-}z_3)/r_{13}^3 \,\} \tag{2.22c}$$

Unlike the case of the binary collision, there is no advantage in transforming Newton's equations to a non-Cartesian or COM coordinate system for many-particle systems. Equations analogous to 2.22 are the starting point for computer-based classical dynamics calculations which employ pairwise potentials. (See 4.3.4 for a discussion of force calculations based on many-body potentials.)

3. COMPUTER IMPLEMENTATION

3.1. Introduction

This chapter reviews some miscellaneous ideas about the development of programs for classical dynamics (CD) simulations of atomic collisions on personal computers (PCs). The chapter is intended to provide a few practical tips for research students who are commencing investigations in this area, and is not necessary reading for those who are purely users of the SK or *Kalypso*, who may omit this chapter entirely. When I first wrote this chapter, I had not seen the book by *Smith et al*.[1] which covers much the same ground, but far more competently and thoroughly. For that reason, I have resisted the temptation to enlarge or update this chapter.

It is worth remarking that the development of PC based simulation programs is only possible because of the rapid increase in CPU speeds which have been achieved since 1990 or so. Robinson mentions that a CDC 7600 mainframe in ~1980 required about 1 s for each integration timestep involving a ~1000 atom system, which was similar to the performance of *Snook* 2.*x* running on a 200 MHz Pentium PC in 1997.[6] However, it must be emphasised that a fast CPU cannot compensate for a poorly implemented simulation algorithm, which can increase computation time by several orders of magnitude. A first attempt at writing a simulation program will usually produce an inefficient result; but if it runs correctly, the programmer should be encouraged, rather than discouraged, by his creation.

3.2. Computer languages and tools

Traditionally, CD programs have mostly been developed in Fortran, or to a lesser extent in C. (Examples of such programs can be found at many Internet web sites.[18]) There are good reasons (e.g. portability) for continuing to work in these languages when command-line mainframe programs are being developed. For a PC based application, these languages can also be used, but they suffer the drawback that they are not particularly well-supported by the mainstream compiler manufacturers, and so lack the sophisticated ‘rapid application development’ (RAD) features for visual development associated with C++ and Pascal products.

Visual programming and user interface development is not a normal feature of most computer courses for scientists. It is probably wise to write early simulation routines as simple command-line Dos applications. Success in this task will increase motivation to tackle the visual programming hurdle. This is particularly important if you plan to pass on the fruits of your labours to others. Be warned that today’s PC users may simply ignore a scientific program that lacks a menu-driven user interface! Fortunately, with a compiler product like Delphi or C++ Builder it is almost child’s play to develop a minimal interface. The *Simulation Kit* and *Kalypso* were built and compiled using Borland Delphi. Rapaport has written an introduction to visual molecular dynamics programming using X-Windows under Unix.[19]

C++ and Object Pascal are object-oriented languages which take most of the labour out of designing windows, menus, dialog boxes and other visual objects. Although it is not difficult to link together object files built from different languages, most programmers would presumably prefer to develop and debug their programs from within a single, integrated environment. (For C, this can of course be achieved with a C++ compiler.) The prominent names in PC compiler technology today are Microsoft (C++) and Borland (Pascal, C++). Modern compiler products

feature a high degree of integration between the code editor, the compiler, and various supporting tools (including, hopefully, a debugger and profiler). For scientific applications, it is very important to have (a) the capability of inserting assembly language statements into the body of high-level language code; (b) a debugger which allows you to view the registers of the numeric coprocessor. (You may not use assembly language now, but you may eventually.)

If you plan to distribute your program, you may need to develop a context-sensitive, online Help system for it. For Windows environments, the use of a help-authoring tool is recommended for this task. You can find shareware products in the `/winhelp` directories of Simtel.Net's win95 and win31 collections (`http://www.simtel.net`). Another class of tools that is worth mention comes under the description of memory checkers for instance: BoundsChecker (`http://www.numega.com`). The latter are expensive tools, but you can often download and use them for a limited period to run over your final program, which they will check for various kinds of memory and resource allocation errors.

3.3. Performance issues

It is meaningless to argue along lines such as 'Fortran is faster than C++': the relative performances of two programs depend on how the compiler manufacturers have chosen to implement floating point operations at the assembly language level. The relative performances can also vary according to the hardware, operating environment and compilation options. Borland Pascal 7, for example, used a very slow implementation of the `exp()` function, but the same function in its 32 bit successor (Delphi) is as tight and fast as the coprocessor architecture allows. Essentially, performance has to be determined by empirical investigation. It should be pointed out that most so-called optimising compilers ignore floating point optimisations completely, which makes their optimisation features more-or-less irrelevant for scientific applications.

A code profiler (such as Turbo Profiler) may be shipped with your compiler, although good ones seem to be rare and expensive for the Windows 95 environment. This type of tool allows you to determine the 'hot-spots' or 'time-eaters' of your program, and thereby develop strategies for performance optimisation. For example, a CD simulation program spends most of its time either computing the forces (~90%), or building the neighbour lists (~10%). You can use a profiler to identify the critical statements which you need to (a) rewrite in a high level language or in assembly language, or (b) replace by a faster algorithm (the preferred method). Unfortunately, profilers do not always work correctly.

Program precision is also dependent on low-level language and hardware details, rather than the choice of high-level language. The double-precision IEEE floating point types used by Fortran, C, C++ and Pascal are identical. However, hardware (numeric processor) performance varies from manufacturer to manufacturer.

3.4. Programming tips

3.4.1. Programming style

From the outset, you should write your code in the expectation that it will have to be modified many times. This means choosing meaningful variable names, and documenting the logic of obscure algorithms (Fortran and C users take note!). As far as possible, you should compile and run the program as soon as each new function/subroutine is completed: this may involve writing

some temporary skeleton code which calls the new function for testing purposes. (The ability of PC compilers to integrate editing/compilation/debugging operations is a major advantage over mainframe development environments.) In this respect, the fast compilation time of Pascal is a major advantage over C++.

3.4.2. Operating systems

Since PC operating systems and compilers change so quickly nowadays, it is essential to separate the code for your user interface (‘front end’) from that for your simulation routines (‘back end’ or ‘engine’). For example, the simulation engine used by *Snook* consists of one big function (subroutine) contained in a module of its own which is called by the main program:

```
...
Simulate(TrgFile, PrjFile,...);
...
```

The same function can be used with minor modifications by a Dos program or by a Windows 95 program.

Once the simulation is running, the main difference between these two platforms is in the way user inputs (e.g. Ctrl-Break) and screen outputs (e.g. messages) are handled. Instead of executing your chosen input/output method directly, you should create a procedure for the purpose:

```
...
Msg := ‘Output information’;
WriteScreen(Msg)
...
```

Now when you change OS platforms, only the `WriteScreen()` routine has to be changed.

3.4.3. User input

Input data should be validated as much as possible at the time of entry (in the *Simulation Kit* and *Kalypso*, most data validation is handled by *Spider*). Modern visual development environments provide dialog box ‘objects’ which can automate a wide variety of data validation tasks (e.g. validating the range and format of floating point and integer numbers). Avoid ‘hard-coding’ data into your simulation routines as much as possible. It is tempting to do this when you are the sole user of the program, because you can always change the data and recompile the program as you wish. However, in the author’s experience, this procedure tends to introduce errors.

If you find yourself changing hard coded constants, it is better to provide some method of inputting them into the program from an external source. Ideally, input should be in the form of data stored in files, rather than data entered via the keyboard. File-based data mean less work for the user, and are self-archiving. However, for small programs (such as the *Cone* utility) it may be too pedantic to insist on file-based inputs.

3.4.4. Dynamic memory issues

You can write small simulation programs without using pointers and dynamic memory allocation. But eventually you will wish to do so for serious simulations. In a Dos/Win 3 environment, one quickly exhausts the 64 k array size limitation, which applies regardless of your machine's physical RAM. Under Windows 95/NT, more memory is available, but it has to be shared with other programs.

In general, dynamic memory allocation and deallocation is an area of programming which always requires special attention.

- Failure to deallocate memory will cause memory 'leaks'. The easiest way to check for these is to continuously monitor memory usage. You can see that both *Snook* and *Kalypso* offer options of this kind, in the form of visual memory meters. These are used by the author for debugging purposes. After running a simulation, the memory allocation should return to its original value. For a variety of reasons, such as pointer management overheads, this does not usually occur precisely. However, apparent leaks due to legitimate overheads will saturate after the routine has been run 2 or 3 times, unlike true leaks which will continue to deplete the memory pool ad infinitum.
- Failure to allocate sufficient memory (e.g. for arrays whose length is set at run-time) is another source of dangerous bugs, particularly the well-known 'out-by-one' error that plagues C programmers. Such bugs (memory overwrites) will not necessarily be indicated (by a general protection fault) even in a protected mode environment, so it is good practice to examine the contents of arrays (or at least, their first and last elements) in the 'Watch' window of a debugger as you step through the program. Once things are running, it is a good idea to temporarily modify the program so that that you are deliberately under-allocating memory, in order to see what behaviour this produces.
- Most non-trivial program errors are associated with illegal pointer operations. It is a good idea to assign pointers to nil (or null) both on program start-up, and after deallocating dynamic memory; and to allocate or deallocate memory for a given pointer in only one place in the program. The definition of a pointer error depends in part on the operating system. Under Windows 95 an attempt to read unallocated array elements may generate an 'exception' (an error flagged by the operating system), whereas this is legal under Dos.

3.4.5. Compiler bugs

Know your compiler well: most libraries and compilers released by compiler vendors contain dozens of bugs. (Lists of these can be found on Internet sites and in Usenet groups `comp.lang.xxx` devoted to that compiler.) For this reason, you need to ascertain that your compiler package contains the source code for all libraries, so that you can debug them if necessary.

3.4.6. Data structures and algorithms

For classical dynamics simulations, the main data structures you will use are the 1- and 2-dimensional arrays. 1-dimensional arrays are used to hold vector components (x[n], y[n], z[n]), while 2-dimensional arrays are used for holding neighbour lists and possibly force components (depending on which integration algorithm you use). Do not use container classes (objects) to store vector quantities, because access to their elements is needlessly slow.

CD simulations do not usually require an in-depth knowledge of computer science algorithms beyond what can be gleaned from numerical analysis texts like Numerical Recipes.[20] A possible exception is range-searching algorithms (for improving the efficiency of neighbour list construction).[21]

3.5. Program structure

In this section I want to highlight some fragments of code extracted from *Snook* for two purposes. First, the code fragments will shed light on the internal structure of the simulation routine. Also, they may be of interest for those who plan to develop their own routines. I have simplified the appearance of the code by replacing dereferenced pointers by static variables (in practice, memory for all array variables is allocated dynamically), and by removing any details of minor interest.

It should be appreciated that the bulk of the code in *Snook*'s simulation routine (i.e. excluding the user interface and event handling system) is devoted to initialisation processes, namely (a) memory allocation, (b) data input/validation, and (c) assignments to variables. The initialisations performed in (b) and (c) can be divided into two types: those that must be initialised once only (such as the potential parameters), and those that have to be initialised for every new run, i.e. every new projectile trajectory (for example, the target atom coordinates). Although the bulk of the programming effort is devoted to the initialisation routines, the bulk of execution time is spent in computing the forces and in updating the neighbour lists. Most optimisation efforts will be focussed on these routines, and it is these sections which have to be constructed most carefully.

3.5.1. The integration loop

The code fragment shown below implements the 'velocity form' of the Verlet integration algorithm:

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{v}_n \Delta t + \tfrac{1}{2}\mathbf{F}_n \Delta t^2 / m \tag{3.1}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \tfrac{1}{2}[\mathbf{F}_{n+1} + \mathbf{F}_n]\Delta t / m \tag{3.2}$$

On entry to the procedure, some useful composite constants are prepared, and the arrays, which on exit hold the force components (from the current and preceding timesteps), are initialised. The forces are summed by calling the `GetForce()` procedure, and finally the particle velocities are updated. The following code fragment is only applicable to a pair potential. (Many-body potentials require an extra (i, j) loop in order to compute and store the local 'density' terms.)

```
Procedure Get_rv_Verlet(NoOfAtoms:longint;ti:float);
var
  i,j,k: longint;
  Fx, Fy, Fz: float; // used to hold forces
  ti2div2,tidiv2,Cut-off2: float; // composites
  kr,kv: float; // Composites
  ArraySize: longint;
begin // Get_RV_Verlet
  ti2div2 := 0.5*sqr(ti); // ti is the timestep
  tidiv2 := 0.5*ti;
```

```
  Cut-off2 := sqr(Cut-offDistance);
  for i := 0 to NoOfAtoms do // FNew gets value of Force from prev. step
  begin
    FNewX[i] := FOldX[i];
    FNewY[i] := FOldY[i];
    FNewZ[i] := FOldZ[i];
  end;
  ArraySize := (NoOfAtoms+1)*SizeOf(float);
  FillChar(FOldX,ArraySize,0); // zero out all elements
  FillChar(FOldY,ArraySize,0);
  FillChar(FOldZ,ArraySize,0);
  // Calculate predictor for r
  for i := 0 to NoOfAtoms do // Calculate drift during dt
  begin
    kr := ti2div2/mass[i]; // = ti^2/2m
    x[i] := x[i] + vx[i]*ti + FNewX[i]*kr;
    y[i] := y[i] + vy[i]*ti + FNewY[i]*kr;
    z[i] := z[i] + vz[i]*ti + FNewZ[i]*kr;
  end;
// Evaluate forces at new r
for i := 0 to NoOfAtoms do
begin
  k := 0;
  j := partner[i][k]; // j is partner #k of i
  while (j <> EndOfList) do // EndOfList is end of partner list
  begin // j is always > i in the partner array
    if (sqr(y[i]-y[j])+sqr(x[i]-x[j])+sqr(z[i]-z[j]) <= Cut-off2) then
    begin
    Get_Force // at r[n+1]
    (Fx,Fy,Fz, // <-- outputs these force components
     x[i],x[j], // interaction of #i with its k_th partner
     y[i],y[j], // which is particle #j
     z[i],z[j],i,j);
    FOldX[i] := FOldX[i] + Fx;
    FOldY[i] := FOldY[i] + Fy;
    FOldZ[i] := FOldZ[i] + Fz;
    FOldX[j] := FOldX[j] - Fx;
    FOldY[j] := FOldY[j] - Fy;
    FOldZ[j] := FOldZ[j] - Fz;
    end;
  inc(k);
    j := partner[i][k] // Break when j = EndOfList
  end; // while j
end; // i
for i := 0 to NoOfAtoms do // update velocity
begin
  kv := tidiv2/mass[i];
  vx[i] := vx[i] + (FNewX[i] + FOldX[i])*kv;
  vy[i] := vy[i] + (FNewY[i] + FOldY[i])*kv;
  vz[i] := vz[i] + (FNewZ[i] + FOldZ[i])*kv;
end; // for i
end; // Get_rv_Verlet
```

Each atom, designated by an index *N*, has a list of potential neighbours or collision partners which is stored in the partner array (the 'neighbour lists'). Thus, the partners of atom #*N* are stored as integer indexes as the array elements partner[*N*][0], partner[*N*][1], partner[*N*][2] and so on, until a negative index signals the end of the list for atom #*N*.* This arrangement means that time is not wasted in checking the location of all atoms in the lattice. The neighbour lists are updated periodically. To speed things further, the only atoms included in the neighbour list of an atom of index #*N* are those whose indices are greater than *N*. This means, for example, that F_{31}

* The maximum number of partners that can be stored must be specified by the user in the Run file of the simulation project, so that memory can be allocated correctly.

(the force acting on particle #3 due to interaction with #1) is computed at the time that F_{13} is computed, exploiting the fact that $F_{13} = -F_{31}$.

The GetForce routine returns the force components calculated from the particle vectors. Indices (i, j) identifying the particles are also passed to the routine. Based on these indices, the routine can decide whether the force should be calculated from the projectile-target or target-target potential parameters respectively.

3.5.2. The timestepping loop

The main loop of the program has a very simple structure. As shown in the simplified fragment below, this loop repeatedly calls the integration algorithm discussed in the preceding section. The loop also has responsibility for initiating rebuilds of the neighbour lists (see next section), and for updating the timer (time elapsed), the timestep and the interaction sheath (‘test range’) respectively.

```
// Perform various initialisations of variables (deleted)
// Start main loop
Repeat
  if (NeighbourListTimer = UpdateTime) then
  begin
    CorrectTestRange; // Shld be called before GetNeighbours
    GetNeighbours(NoOfAtoms); // build neighbour list
    NeighbourListTimer := 1; // reset counter
    if UserAborted then break; // break from Repeat..until
  end
  else inc(NeighbourListTimer); // increase counter
  Get_rv_Verlet(Lattice,NoOfAtoms,ti)
  TimeElapsed := TimeElapsed + ti; // update elapsed time counter
  inc(TimeStepsExecuted); // update timesteps counter
  CorrectTimestep; // Update ti, V0 (fastest atom velocity)
Until TerminationConditionsMet; // time & energy conditions
```

3.5.3. The neighbour lists

The routine below shows how neighbour lists are built up in *Snook* by the ‘brute force’ method. For a given atom *i*, the algorithm prepares a list of those atoms *j*, *k*, *l*... which lie within a certain range. Some work is saved by using a projective technique to filter out particles which are found to be out of range along a particular axis. However, for large lattices, this is not the fastest way to build up the lists. *Snook* offers a second (default) option for building the neighbour lists (the ‘cell index' or 'box' method), which will not be presented here, as its algorithmic details are rather tedious.

```
Procedure GetNeighbours(NoOfAtoms:longint);
var
  i,j,k: integer;
  range:float;
begin
// User key-presses/mouse clicks are processed here (not shown)
range := sqrt(TestRange2); // range = width of interaction sheath
for i := 0 to NoOfAtoms do // i must run over ALL particles
begin
  k := 0;
```

```
  Partner[i][0] := EndOfList; // Partner = i,j,k,..EndOfList)
  for j := i+1 to NoOfAtoms do // Ignore j <= i
  if (abs(x[i]-x[j]) > range) // This condition filters out
  or (abs(y[i]-y[j]) > range) // the majority of atoms in the lattice
  or (abs(z[i]-z[j]) > range) then // do nothing
  else if (sqr(x[i]-x[j])+sqr(y[i]-y[j])+sqr(z[i]-z[j]) <
TestRange2)then
   begin
     Partner[i][k] := j;
     inc(k);
     if (k = NoOfPartners) then // Stop! (not shown)
   end; // for j
  Partner[i][k] := EndOfList;
end; // for
end; // GetNeighbours
```

3.6. Program validation

It is difficult to formally prove the correctness of a classical dynamics simulation program. The program should, of course, conserve energy, and linear and angular momentum (provided inelastic energy loss effects are not included in the simulation). Program validation should begin with a study of the kinematics and dynamics of the binary collision, for which it is easy to calculate (for example) the correct apsidal distance by hand (see the 'Gadgets' menu in *Spider*). Three-body systems in symmetric or linear configurations can also be analysed in a similar fashion. The `cone` program, which ships with the *Simulation Kit*, is a useful benchmark for the correctness of binary collision simulations (`cone` uses the slow but accurate Runge-Kutta algorithm, and was programmed quite independently of the *Simulation Kit*).

There are relatively few benchmarks available in the literature which serve to test program performance and accuracy under realistic running conditions. Gärtner et al.[22] recently reported the results of a round robin comparison of 6 established classical dynamics (CD) programs. The various programs were used to calculate the elastic energy loss and angular deflection suffered by a projectile passing through a thin crystalline target. (The reader is referred to the original source for specific details of the simulation problems.)

The results from the round robin are compared with the corresponding results calculated using *Snook* in Table 3.1. The agreement between *Snook* and the round robin results is generally satisfactory, although two of the angular deflections fall just outside the range reported by the contributors to the round robin study. Nevertheless, it is still puzzling to observe the relatively large discrepancies which exist between different MD programs, e.g., the variation of 9eV in the ΔE values reported for 0.5 keV B-Si(100) (Table 3.1). Such discrepancies are far greater than the typical average energy conservation errors expected from the integration algorithm (~ 1 eV for a 0.5 keV system). Probably the only way to understand discrepancies at this level is for the authors of different programs to compare the results of integrating individual trajectories.

Table 3.1. Calculation by *Snook* of the average elastic energy losses (ΔE) and root mean square angular deflections (Δq) suffered by projectiles passing through thin crystalline targets, compared with the results of a round robin study.[22]

System	ΔE (eV)	ΔE (eV), ref. [22]	Δθ (°)	Δθ (°), ref. [22]
0.2 keV B-Si(100)	66.1	63.1-68.4	35.2	32.6-34.5
0.5 keV B-Si(100)	64.6	55.9-65.1	25.7	22.1-25.1
1.0 keV Ar-Cu(100)	429.4	412.1-428.4	19.6	19.1-19.8

4. THE PHYSICAL MODEL

4.1 Introduction

This chapter will review the physical model and assumptions underlying the simulations performed by the programs *Snook* and *Kalypso*. Except for the attractive part of the target-target potential, the two programs have similar physical foundations. Unless stated otherwise, references to *Snook* in this chapter can be taken to also apply to *Kalypso*.

All simulation models are founded upon a multitude of assumptions. To use and interpret the results from a simulation, one must understand the limitations of the simulation as well as its capabilities. It is extremely difficult to prove the physical (as opposed to computational) correctness of any particular simulation model.

As it executes a simulation, *Snook* writes a real-time commentary to the screen in a manner which reflects the user's preferences (i.e. degree of verbosity). The meaning of this output should be broadly clear, but it is explained in the tutorial file, which ships with the *Simulation Kit* (`...\tutorial\tutorial.pdf`). The collision dynamics can also be viewed graphically in real-time. Verbose output and graphical views are extremely useful when fine-tuning a simulation project. During a production run, however, you should switch these options off, and 'minimise' the *Snook* main window, in order to obtain maximum performance.

In the following brief review, I have tried to address the issues which I think will be most important for an understanding of the design of the *Simulation Kit* and *Kalypso*. I hope that readers will point out omissions to me.

4.2. Initial conditions

4.2.1. Lattice structure

The lattice structure used by *Snook* is defined in the Target file (*.TRG) which is a listing of coordinates and other information about the particles in the target. The *xy* plane of the coordinate system is parallel to the surface, while the *z*-axis corresponds to the surface normal. The negative *z*-direction corresponds to movement into the lattice. The projectile approaches the surface in the *xz* plane if the azimuthal angle is set to zero (in the Run file), starting from the +*x* direction, and moves in the negative *x*-direction towards the location of the anchor atom (see below for explanation of this term, and also Fig. 4.1).

If thermal vibrational displacements are selected (in the Model file), then at run-time *Snook* will add appropriate (random, Gaussian, Debeye-Waller) thermal displacements, including any surface effects specified by the user. In this case, the user also has an option (under *Snook*'s Options|Simulation menu) to add an average kinetic energy of 3/2kT to the lattice atoms (which is derived from a Maxwellian distribution of velocities). If the latter option is not selected (normally it is not), the lattice atoms are stationary at the start of the simulation. Chapter 8 discusses the implementation of thermal vibration effects in more detail.

4.2.2. Projectile coordinates

The starting position of the projectile is referenced to the coordinates of an *anchor atom* in the target (see Fig. 4.1): these are the coordinates listed in the first line of the Target file (i.e. the coordinates prior to application of thermal displacements). For small impact parameters, the anchor atom is the projectile's first collision partner. By default, in all lattices generated by *Spider*, the anchor atom is located at the origin (0,0,0) of the coordinate system, but any other value is acceptable.[*] The vertical (z) projectile position *relative to the anchor atom* is determined by the z_0 parameter (by default, 3 Å) in the Impact file. The initial (x, y) coordinates of the projectile are determined by (a) the impact parameters (b_x and b_y, along the x- and y-directions respectively) for the current run (trajectory), which are listed in the Impact file; and (b) the projectile altitudinal and azimuthal angles of incidence.[†] The angles in (b) also determine the relative magnitudes of the projectile starting velocity components, (v_x, v_y v_z). Typically, $v_y = 0$.

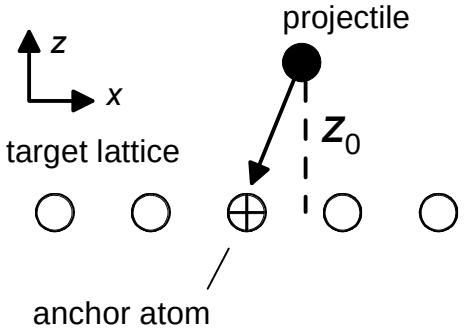


Figure 4.1. Starting geometry for a simulation run. The uppermost particle (black) is the projectile, which is located at a vertical distance z_0 above the anchor atom (crossed circle) in the target lattice. The lateral (x, y) displacement of the projectile relative to the anchor atom is determined by (a) the impact parameter for that run, (b) the projectile angles of incidence. See the *Spider* on-line Help for more information on this subject.

The arrangement based on relative coordinates just described has the advantage that it allows the same set of input files to be used for similar scattering systems with different altitudinal directions of projectile incidence (with the sole difference that the particular altitudinal angle involved has to be specified in the Run file). For example, the same input files can be used for a simulation of Cu(100) sputtering by Ar^+ at $\varphi = 45°$ and 30°, except for the change to the angular variable in the Run file.

4.2.3. Particle masses

The isotopic distributions of the target atoms are not known by the program (*Spider*) which generates the Target file (the target particles are assigned the user-specified average mass), but it

[*] If you rotate the lattice in the *yz* or *zx* planes you will normally have to edit (cut/paste) the resulting Target file to ensure that your desired anchor atom coordinates are found in the first line of the new file.

[†] The projectile altitudinal angle of incidence, φ, is a "polar" angle which takes values between +90° (normal to surface) and 0° (parallel to surface). The projectile azimuthal angle of incidence is the angle subtended around the z-axis in the xy plane; it is zero for incidence in the $-x$ direction (as shown in Fig. 4.1), and 90° for incidence in the $-y$ direction. The angular convention used by Winnow (and Spider's "User-Defined" Run file output option) uses (unlike here) a strict sign convention based on the direction (sign) of vector components. See section 6.3 for further remarks.

is possible (although tedious) to modify the atomic mass data in the Target file by manual editing of each entry. A better way to achieve the same result is to write a computer program that will modify the mass field of your target file according to your specifications. The projectile mass is likewise fixed at the value stipulated in the Projectile file.

4.2.4. Initial conditions data

If you want to examine the state variables (position, momentum and others) which were actually applied at the start of the simulation, there is an option in *Snook* which causes them to be dumped to a file `inivars.snk`, from which they can be regenerated by *Winnow*'s Process|Convert command. (There are not many reasons why you would want to do this.)

4.2.5. Reduced impact zone

A simulation project typically consists of a large number (10^2 - 10^5) of simulations of independent projectile trajectories, usually referred to as 'runs' in the *Simulation Kit* documentation. The trajectories are directed towards a representative region of the surface of the target lattice which the author calls the *reduced impact zone* or RIZ. (Other terms are in use in the literature, e.g. *zone of irreducible symmetry*.) The reduced impact zone is a region of the periodic surface which is sized in such a way that every possible incident trajectory on the surface can be mapped to a symmetrically equivalent trajectory directed into that zone. This idea is discussed in refs. [4,23]. The dimensions of the reduced impact zone depend on the surface orientation, and the symmetry inherent in the collision problem (i.e. the direction of projectile incidence). Fig. 4.2 illustrates the reduced impact zone for a FCC (111) crystal surface. The reduced impact zone is always smallest at normal projectile incidence.

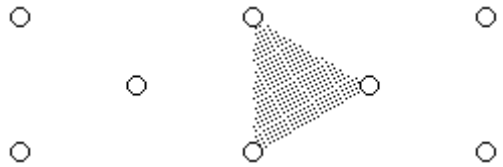


Figure 4.2. Projection of the "representative region" or "reduced impact zone" for projectiles normally incident onto a FCC (111) surface. The atoms at the corners of the zone lie in 3 different layers of the target.

The Impact file of a *Snook* simulation project lists the impact coordinates which collectively define the dimensions of the reduced impact zone. The *Spider* online Help includes templates for the most common surfaces. The impact coordinates are listed in the form (b_x, b_y, z_0), where b_x and b_y represent the *x*- and *y*-resolved components of the impact parameter, *b*, while z_0 is the vertical position of the projectile (see Fig. 4.1).

For an ideal lattice, there is no error introduced by using a reduced impact zone which is, for example twice or four times as large in area as the true zone. Although this can be inefficient in computational terms, true redundancies do not generally occur in the Impact files generated by *Spider*, because of the character of the underlying generating algorithm. Here is a simple numeric illustration: for 5 impacts along an axis of length *a*, the impact points generated by *Spider*

(starting from 0.0) will be 0, $a/5$, $2a/5$, $3a/5$, $4a/5$ while for 5 impacts along an zone axis of length $2a$ the impact points will be: 0, $2a/5$, $4a/5$, $6a/5$ ($\equiv a/5$), $8a/5$ ($\equiv 3a/5$).

The introduction of randomly-applied thermal vibration effects has two consequences in connection with the impact zone; (a) each target configuration is unique when vibrational effects are included in the simulation, so every simulation will yield different results; (b) the random thermal displacements blur the notion of symmetrically equivalent trajectories (indeed, the same projectile impact coordinates will generally give rise to different collision cascades on repeat simulations, because of the different displacements of the target atoms).[*]

It is sometimes convenient to use an impact zone that is larger (by an integral factor) than the reduced impact zone. A typical example occurs during the simulation of processes as a function of projectile altitudinal angle, where one may prefer (for the sake of simplicity) to use the same impact zone (i.e. Impact file) at normal incidence as was used for the simulations at oblique angles of incidence.

4.3. Interaction potentials

4.3.1. Interaction Model used by Snook

Atomic interactions modelled by *Snook* in the *Simulation Kit* are based on analytic Morse potential functions as specified in the Model file. The *Simulation Kit* user is required to select the type of potential employed by the simulations. There are two broad choices: (a) screened Coulombic potential; (b) composite potential.

Potential (a) normally applies to the projectile-target interaction.[†] The user may select either (a) or (b) for target-target interactions. In both cases the potential is cut off at a finite distance specified by the user (normally between the 1st and 2nd nearest neighbours, or between the 2nd and 3rd nearest neighbours). Optionally, a polynomial switching function (recommended) may be applied to bring the potential and forces smoothly to zero at the cut-off distance. The switching function takes the form $S(x) = 1 - 6x^5 + 15x^4 - 10x^3$ for separations $r_{sw} < r < r_{cut}$, where $x = (r - r_{sw})/(r_{cut} - r_{sw})$.[24] The parameters are calculated automatically at run-time by *Snook*.

The Morse potential, although well established by tradition, does not provide a particularly realistic description of forces in a metal. Its main advantage lies in its speed. Pair potentials offer a 'first-order' description of the properties of virtually any non-molecular solid. Most pair potentials are fitted to equilibrium state properties, so they presumably are most realistic for the undisturbed lattice. For simulations involving metals and semiconductors, the tendency today is to use many-body potentials when the results are sensitive to the attractive interaction region (see 4.3.4).

---

[*] A deterministic (reproducible) series of random numbers can be generated for the thermal vibration corrections by setting the 'random seed' simulation option to any value except zero (0) in Snook/Kalypso. A zero value sets the seed irreproducibly using the system clock.

[†] Normally Snook treats the projectile-target interaction differently from the target-target interaction, in that the former only uses a screened Coulombic potential (not a Morse potential). You can override that behaviour by checking the 'Self-Bombardment' simulation option availablr in both *Snook* and *Kalypso*. This makes physical sense if your projectile species (atomic number) is the same as that of the target e.g. Cu projectiles bombarding a Cu target, or (for a multi-element target) a projectile which is the same as one of the elements in the target. The potential parameters specified in the Model file will then be ignored. Instead, the target-target composite potential will be used.

The composite pair potential used by *Snook* consists of a screened Coulombic potential at small internuclear separations, a Morse potential at large internuclear separations, and a cubic spline function which connects the two in the intermediate region. The spline function is computed automatically, based on the user's specification of the spline region limits. The user should check that the spline function is realistic, i.e. that it does not introduce maxima or minima into the force function. The optimum spline limits are found by trial and error. The upper spline limit should be below the first neighbour separation.

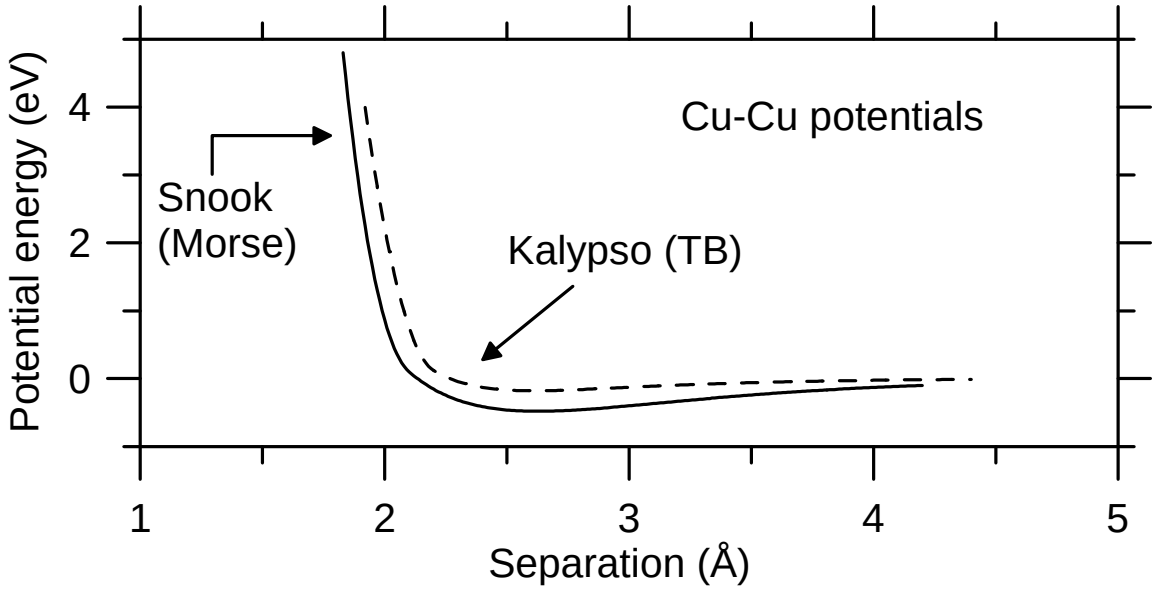


Figure 4.3. Composite potential, V(*R*), for Cu-Cu interaction as a function of separation *R*. In the region shown, the potential consists of a cubic spline function (1.5-2.1 Å for *Snook*, 1.6-2.2 Å for *Kalypso*) joined either to a Morse potential (*Snook*, *R* > 2.1 Å), or to a TB effective potential (*Kalypso*, *R* > 2.2 Å),

Fig. 4.3 shows the attractive part of the default composite Cu-Cu potential generated by *Spider*'s Model option. The corresponding screened Coulombic potential for Cu-Cu (not shown) decreases far less slowly towards zero in the region shown, and lacks any attractive potential well. The form of the attractive potential is only significant in collisions in which the apsidal distance falls outside the core region described by the screened Coulombic potential. (For Cu-Cu, for example, this corresponds to Lab collision energies below ~100 eV.) The explanation for this is that the scattering integral function (equation 2.16) is singular at the collision apsis, and so is largely determined by the force function in the vicinity of the apsidal distance. The scattering of projectiles of a few hundred eV energy is fairly insensitive to the character of the attractive potential.

4.3.2. Interaction Model used by Kalypso

The interaction model used by *Kalypso* is more complex than *Snook*'s. The short-range interactions are, however, similarly based on a screened Coulombic potential which is splined to the attractive region.

Kalypso offers two choices of attractive potential: the Sutton-Chen (SC) potential,[25] and a 'tight-binding' (TB) potential (also known as a Gupta potential),[26] both of which are many-body

potentials, and also supports binary compound targets. These many-body potentials, which can be regarded as particular examples of Finnis-Sinclair (FS) potentials,[27] are described below in section 4.3.4 and in chapter 10.

The potentials are many-body in the sense that the energy of a system cannot be decomposed into pairwise contributions. Like the Morse potential, the potentials are used with a cut-off distance, and (recommended) an optional switching function (see 4.3.1). Joining any FS potential to a pairwise repulsive core potential via a cubic spline is a non-trivial operation, which can only be done approximately. You should examine the User Guide to see how this is achieved for *Kalypso*. (The method used is to spline to an effective pair potential, based on a reference lattice density specified by the user. This seems to be the normal practice used in the literature.[28])

The properties (e.g. elastic constants, cohesive energy) predicted by pair and many-body potentials are most similar for the bulk, undisturbed lattice. The properties of the potentials start to diverge once the atomic coordination and density deviate from the bulk value. The rationale for using many-body potentials in atomistic simulations is to achieve greater accuracy in modelling the dynamics and energetics of atoms in situations of reduced coordination (at the surface, and in clusters). Many-body potentials also give better predictions of the static properties of bulk atoms.

Functionally, the SC and TB potentials are closely related to the Lennard-Jones (LJ) and Morse pair potentials respectively. The attractive part of the SC potential is a non-linear function (square root) of a sum of pairwise LJ terms, while the attractive part of the TB potential is a non-linear (square root) function of a sum of pairwise Morse terms. What this means is that cohesive energy is a non-linear function of the coordination number for a many-body potential.

The SC and TB potentials are expected to work best for fcc *d*-band metals. However, they have also been used for other kinds of metals, although they may not predict the correct relative stabilities of different crystal structures. This is not a critical problem for a sputtering simulation, since the different phases (e.g. fcc versus bcc) are typically of similar energy, and in any event, there are considerable energy barriers hindering spontaneous conversion on the simulation time scale. However, you may need to increase the size of your target, since it will invariably 'reconstruct' from its edges inwards as the simulation progresses.

4.3.2. Screened Coulombic potential

The short range potential used by *Snook* (and also *Kalypso*) is classified as a screened Coulombic potential. There are 3 variants in common use, which are named after their developers: the (Ziegler-Biersack-Littmark) ZBL potential, the Moliere-Firsov potential and the Moliere-Lindhard potential. The analytic form of these screened Coulombic potentials for interacting atoms of atomic number Z1, Z2 respectively, is as follows:

$$V(r_{ij}) = \frac{Z_1 Z_2 e^2}{4\pi\varepsilon_0 r_{ij}} \sum_{k=1}^{N} c_k \exp(-b_k r_{ij} / a) , \qquad (4.1)$$

where the summation index k runs from 1-3 for Moliere potentials, and 1-4 for the ZBL potential, and c_k, b_k and a (the screening length) are coefficients defined differently for the various potentials.

The user of the *Simulation Kit* has to select: (a) one form of screened Coulombic potential to represent the projectile atom-target atom interaction; (b) one form of screened Coulombic potential to represent the target atom-target atom interaction; (c) two 'screening length adjustment' factors (projectile-target, target-target) which typically fall in the range 0.7-1.0 (1.0 is normal for the ZBL potential). The target-target and projectile-target interactions do not have to be of the same form or have the same screening length correction.

The Moliere potential has two variants, according to the way in which the screening length (a, in Å) is chosen:

$$a = 0.4685/(Z_1^{0.5} + Z_2^{0.5})^{2/3} \text{ (Firsov form)} \quad (4.2a)$$

$$a = 0.4685/(Z_1^{2/3} + Z_2^{2/3})^{0.5} \text{ (Lindhard form)} \quad (4.2b)$$

In practice this distinction is irrelevant since (judging from the literature) every user of this potential seemingly adds his or her own screening length correction. The screening length correction is a 'correction' factor (< 1.0) which is used to scale the screening length parameter in screened Coulombic potentials, particularly the Moliere potential. A typical value for this correction for the Moliere potential would be 0.8, usually chosen by fits to experimental data, e.g. impact collision ion scattering spectrometry. The screening length (*a*, in A) for the ZBL potential is defined as:

$$a = 0.4685/(Z_1^{0.23} + Z_2^{0.23}) \quad (4.2c)$$

For the ZBL potential, you would normally choose a value of 1.0 for the screening length correction, unless you have reason to do otherwise (see below). You should be wary of changing the screening length correction parameter arbitrarily, as it has enormous effects on the potential and unrealistic values could conceivably undermine the credibility of the simulations.

Which screened Coulombic potential is best? This question cannot be answered with rigour. However, if you are not going to search for an optimum screening length correction, the ZBL potential is probably the easiest choice because it is normally used in 'unadjusted form', i.e. with a screening length correction of 1.0. Correction factors have an enormous effect on the potential, because they involve exponentiation. In fact, the effect of the corrections may be more significant than some of the terms in the original potential! The ZBL potential is generally used without any screening length correction, and this fixed form may be regarded as an advantage or a disadvantage, depending on how highly you rate the potential.

For the Ar-Cu system at least, this author notes that the ZBL potential fits closely the *ab initio* potential calculated by Broomfield et al.[29] Nordlund et al.[30] compared the ZBL potential for C-C, Si-Si, N-Si and H-Si with Hartree-Fock (HF) potentials; they found agreement typically to within ~ 3% for $V(r) < 5$ keV, and ~5% for $V(r) < 10$ keV. The worst agreement was for the C-C system (~5% at 3 keV).

Gamma-ray induced Doppler broadening data obtained for Ni-Ni, Fe-Fe and Cr-Cr collisions have recently been used to fit the exponent in the screening length definition for the ZBL potential (keeping other parameters fixed at the standard values).[31] This exponent is normally assigned the value 0.23, as in Eq. 4.2(c). The fitted values were 0.26 for Fe and Ni, and 0.31 for

Cr. These figures imply screening length correction factors of about 0.907 (Fe-Fe), 0.905 (Ni-Ni) and 0.776 (Cr-Cr), which represent quite large corrections to the ZBL potential.

The conclusion must be that in any serious study of ion-surface collisions one should experiment a little with the potential, to what effect this has on the simulation averages.

4.3.3. Morse and spline potential (Snook only)

The Morse potential V(r) is defined by:

$$V(r) = D\{ \exp{-2\alpha (r-r_0)} - 2\exp{-\alpha (r-r_0)} \} \tag{4.3}$$

The parameters D, α and r_0 are derived from fits to bulk solid state properties, and may vary somewhat from author to author, depending on the fitting method, and the potential cut-off distance (see Chapter 10). Typically, the potential is fitted to the cohesive energy, lattice constant and bulk modulus of the element. The Morse potential has a minimum when $r = r_0$, at which point $V(r_0) = -D$. The parameter r_0 is of the same order, but greater than, the lattice nearest neighbour distance (r_{NN}), unless the interaction is cut off at r_{NN}, in which case $r_0 = r_{NN}$. Quite considerable errors in cohesive energy can arise if you apply an arbitrary cut-off to a potential originally fitted to another cut-off distance.

The Morse potential, or any pairwise potential, is not a particularly good description of solids whose bonds are strongly non-central in character: e.g., delocalised bonding (as in metals), or directional bonding such as the tetrahedral bonds in diamond or silicon. This does not mean that such solids cannot be modelled via a Morse potential, only that effects which depend on directional bonding (e.g. the relative stabilities of different structures) cannot be included in the model. In general, it is expected that the potential would become more inaccurate as the structural environment is removed from the bulk reference environment to which the potential was originally fitted. Thus the Morse potential tends to underestimate surface binding energies, and Morse parameters derived for a crystalline element are usually very different from those applicable to a diatomic molecule of the same element. (There are a few exceptions to this, e.g. Ca.) These are recognised failings of the pair potential model.

For any particular case, it is difficult to assess both the magnitude and the consequences of the error introduced by the pair potential approximation. Its validity can be assessed by (among other things) consideration of the elastic constants of the solid of interest. If the pair potential model holds, then it can be shown for cubic crystals that the elastic constants C_{12} and C_{44} should be identical (the Cauchy relation). Here are C_{12}/C_{44} ratios for some elemental solids: Fe (1.15); Cu (1.61); Ni (1.12); Ag (2.02); Al (2.18); diamond (0.22); Si (0.8); Ge (0.72). According to this criterion, the pair potential model gives a better description for Si and Ge than it does for Cu, but it breaks down badly in the case of diamond. Ercolessi et al. list several other criteria which can be used to assess the need for a many-body potential description.[60]

If you choose to include a Morse potential, *Snook* will also calculate the cubic spline function required to join the Morse potential to the screened Coulombic potential. You can preview the composite potential (screened Coulombic + spline + Morse) in *Spider*.

The spline potential is recalculated at run-time: no explicit information about it is written to the Model file (this is not true for *Kalypso*). If you choose not to use a Morse potential, the simulations will use the screened Coulombic potential up to the range specified by the potential cut-off. This is adequate for ISS simulations, for example, because (a) the interaction time is short; (b) the process being modelled involves short-range interactions. Robinson gives a good discussion on this subject.[6] The projectile-target potential used by *Snook* is always a screened Coulombic potential, unless you have specifically selected the 'Self-Bombardment' option in *Snook*'s Simulation Options dialog box.

If the Morse potential option is selected, a cubic spline function will be used to join it to the screened Coulombic potential between the limits you specify in the Model file. The objective is to choose spline limits that give a smooth transition between the two potentials (no unphysical minima or maxima). The limits are chosen by trial and error. The composite potential will always show a smooth fit at the spline function limits. However, the force function will normally have a discontinuity in gradient at the upper and lower limits of the range covered by the spline function. The type of discontinuity to avoid is one which introduces a spurious maximum or minimum at one or both of the spline limits.

If, for some reason, you wish to employ a cubic (spline) potential all the way up to the cut-off distance, you can do so effectively by specifying a tiny separation between the high spline limit and the cut-off distance (e.g. cut-off distance = 3.8 Å, high spline limit = 3.7999 Å). A restriction you need to be aware of is that both spline limits should be greater than or equal to 1.0 Å. This restriction relates to the way in which the Morse look-up tables are implemented. The condition is enforced by *Spider* and checked by *Snook* at run-time.

4.3.4. Many-body and spline potentials (Kalypso only)

Readers new to the subject of many-body potentials should consult some of the research literature on these potentials (see section 10.1). This section should be read in conjunction with the material in chapter 10.

The total energy for an N atom system is defined for a Finnis-Sinclair (FS) potential (see 4.3.2) by the expression:

$$U_S = V_{pair} + V_{mb} = V_{pair} + \sum_{i=1}^{N} G_i \tag{4.4}$$

where the many-body part of the potential V_{mb} is a sum of N 'pair-functional' terms G_{i}, which will be defined later (Eqs. 4.10, 4.11). A number of potentials of this general form have developed since the mid-1980s, including the embedded-atom method (EAM) potentials, the glue potentials, and the Finnis-Sinclair (FS) potentials. Although each type of potential has a different theoretical justification, their properties are broadly similar. Typically, the pairwise part of the potential is repulsive, while the many-body part is attractive.

Eq. (4.4) gives rise to a two-term expression for the components of the forces, e.g. for the x-component of the force on atom i:

$$F_i^x = -\left(\frac{\partial V_{pair}}{\partial x_i} + \frac{\partial V_{mb}}{\partial x_i} \right) \tag{4.5}$$

The pairwise part of the potential:

$$V_{pair} = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} V(r_{ij}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} V(r_{ij}) \tag{4.6}$$

can be differentiated in the usual manner to give the forces:

$$\frac{\partial V_{pair}}{\partial x_i} = \sum_{j \neq i} \frac{\partial V_{pair}}{\partial r_{ij}} \cdot \frac{\partial r_{ij}}{\partial x_i} = \sum_{j \neq 1} \frac{\partial V_{pair}}{\partial r_{ij}} \cdot \frac{(x_i - x_j)}{r_{ij}} \tag{4.7}$$

It is convenient to think of the many-body term G_i for each atom i as being determined by a local electron density r_i at the site of atom i:

$$r_i = \sum_{j=1, j \neq i}^{N} f(r_{ij}) \tag{4.8}$$

so that:

$$G_i = G_i(r_i) = G_i \left(\sum_{j=1, j \neq i}^{N} f(r_{ij}) \right) \tag{4.9}$$

This type of expression is known as a *pair functional* (meaning a function of a sum of pairwise terms). The 'density' defined by Eq. (4.8) evidently increases as coordination increases, but it is not strictly necessary to attach any precise physical meaning to this term. Eq. (4.9) can be regarded as an empirical form to be fitted optimally to experimental data.

In Finnis-Sinclair potentials, G is simply defined as the square root of the 'density':

$$G_i = \sqrt{\rho_i} \tag{4.10}$$

The Sutton-Chen (SC) and Tight-Binding (TB) potentials can be defined, and their relationship can be brought out, by writing them in the following form:

$$U_S = \sum_i \left(\sum_{j>i} A_{ij} \exp(-p_{ij} x_{ij}) - \sqrt{\sum_{j<>i} \xi_{ij}^2 \exp(-q_{ij} x_{ij})} \right) \tag{4.11}$$

For the SC potential, $x_{ij} = \ln(r_{ij} / r_0)$, while for the tight-binding potential $x_{ij} = (r_{ij} / r_0 - 1)$; the A_{ij}, p_{ij} and q_{ij}, r_0, ξ_{ij} are coefficients which depend on the types of the interacting atoms i and j. *Spider* uses different symbols for the coefficients, in order to conform more closely to the conventions used in the literature. The SC potential is normally written in the form:

$$U_S = \varepsilon \sum_i \left(\sum_{j>i} (\alpha_{ij} / r_{ij})^N - \sqrt{\sum_{j<>i} c_{ij}^2 (\alpha_{ij} / r_{ij})^M} \right) \tag{4.12}$$

(where ε_{ij}, α_{ij}, N_{ij}, M_{ij} and c_{ij} are coefficients) while the TB potential is written as:*

$$U_S = \sum_i \left(\sum_{j>i} A_{ij} \exp(-p_{ij}(r_{ij} / r_{0,ij} - 1)) - \sqrt{\sum_{j<>i} \xi_{ij}^2 \exp(-q_{ij}(r_{ij} / r_{0,ij} - 1))} \right) \tag{4.13}$$

Inspection demonstrates that Eqs. (4.12) and (4.13) are equivalent to Eq. (4.11) with the appropriate substitutions for x_{ij}. The length scales α_{ij} in Eq. 4.12 and $r_{0,ij}$ in Eq. 4.13 can be specified arbitrarily without loss of generality, and are conventionally set equal to the lattice constant and nearest neighbour distance of the target, respectively.

Kalypso's composite potentials consist of a repulsive short-range screened Coulomb potential which is splined to the attractive potential at internuclear separations somewhat below the first neighbour distance. This entails joining one of the nodes of the spline function to an effective pair potential, V_{ij}, which is derived from a series expansion of the many-body potential in the bulk environment.[32],[58] For TB potentials of the form of Eqs. 1-5, the effective pair potential acting between atoms *i* and *j* is given by:

$$V_{ij}(r_{ij}) = 2V_{pair}(r_{ij}) - \frac{f(r_{ij})}{G} + \frac{[f(r_{ij})]^2}{4G^3}. \tag{4.13a}$$

The functions $V_{pair}(r_{ij})$ and $f(r_{ij})$ are defined above. This equation assumes the usual pair potential evaluation convention, viz. that interaction terms are only counted once for each distinct pair of atoms. G is a lattice sum defined in Eq. 4.10 which represents the value of the total band energy associated with any atom k in the reference environment (normally a site in the bulk of the ideal lattice) .

The force components are calculated for the FS potential via the following type of expression:

$$\frac{\partial V_{mb}}{\partial x_i} = \sum_{j \neq i} \left(\frac{\partial G_i}{\partial \rho_i} + \frac{\partial G_j}{\partial \rho_j} \right) \frac{\partial \Phi(r_{ij})}{\partial r_{ij}} \cdot \frac{(x_i - x_j)}{r_{ij}} \tag{4.14}$$

---

* The TB potential will probably be encountered in the literature in the form:

$$U_S = \sum_i \left(\sum_{j<>i} A_{ij} \exp(-p_{ij}(r_{ij} / r_{0,ij} - 1)) - \sqrt{\sum_{j<>i} \xi_{ij}^2 \exp(-2q_{ij}(r_{ij} / r_{0,ij} - 1))} \right),$$

which implies that A_{ij} and q_{ij} have values half (0.5) of those needed for the FS-like formalism used by Spider and Kalypso. See Chapter 10 for further discussion.

Note that in Eq. (4.12) the interaction terms $\frac{\partial\Phi(r_{ij})}{\partial r_{ij}}$ are symmetric with respect to the interchange of *i* and *j*. This property distinguishes the formalism of FS potentials for compounds from that of the EAM potentials.

Unless you have reason to do otherwise, I suggest using the TB potentials, which have a more reasonable behaviour at distances below the equilibrium internuclear separation. They also have a shorter range than the SC potentials.

The estimation of energy conservation requires special care in a many-body simulation model. There are two problems. First, the force is discontinuous at the spline node, which results in a potential which is inherently non-conservative: a small energy leakage (~1-5 eV per run) from this source has to be accepted as an irremovable feature of the interaction model. Second, energy leakage also arises when atoms cross and re-cross the cut-off distance boundary of neighbouring atoms. Generally energy errors from this source are positive ($dE > 0$), and tend to mask integration errors ($dE < 0$). The boundary problem can be removed (or detected) by increasing the cut-off distance to a value beyond the effective range of the potential and observing the effect on energy conservation, or (recommended) by making use of the switching function option provided by *Spider*. The switching function causes the many-body potential and forces to vanish smoothly at the cut-off, so that boundary problems do not arise. Energy leakage effects need to be controlled because they complicate the task of selecting an appropriate integration timestep for the simulation, and may also compromise its accuracy.

4.3.5. Surface and bulk binding energies

A surface binding energy correction would not be needed in a classical dynamics simulation if the pair interaction potential completely described the system dynamics. Some simulation models have included a surface binding energy term (planar surface potential or barrier) to describe the non-local, long-range interaction between an escaping atom and the free electrons of the lattice. This kind of model views the atom-lattice interaction as a superposition of (a) pair potential effects, and (b) volume effects caused by a 3-dimensional “potential well.” Trajectories are only influenced by the continuum potential when particles cross the potential barrier, e.g. in sputtering.

It is apparent that simulation models describing "structureless media," or those which neglect interactions between target atoms can benefit from the introduction of a surface binding energy term, since there are no point sources of potentials. By contrast, it has been argued that in a classical dynamics simulation the surface binding energy correction is unnecessary, because its influence is implicit in the Morse or many-body potential deduced from solid state properties. However, as mentioned above, Morse pair potentials do tend to underestimate surface binding effects, which offers a possible justification for introducing a correction factor via a plane potential. (I do not recommend using such corrections to the many-body potentials used by *Kalypso*.)

Nevertheless, it is far from easy to decide on an appropriate value for the surface binding energy term in a classical dynamics simulation which already incorporates a Morse potential. There are grounds for expecting the total surface binding energy (Morse + planar potential) to be roughly the same as the bulk cohesive energy, E_c. The reasoning is as follows: (a) The energy required to

break N bonds and remove a single atom from the bulk into the gas phase should be roughly twice the cohesive energy, since atom formation by bulk vaporisation creates on average two gas phase atoms for every N broken bonds, where N is the coordination; (b) because of the lower coordination, the surface binding energy should be roughly half of the bulk binding energy. Both approximations are questionable for real solids. Another way to estimate the total surface binding energy is to calculate it using a suitably fitted many-body potential. Tight-binding potentials (see chapter 10) predict surface binding energies that are about 10% higher than the cohesive energy.[65]

Another point of view is to note that sputter yield measurements are normally carried out under dynamic sputtering condions. The instantaneous surface binding energy varies due to sputter-induce roughness effects. However, on average, the energy required for sputtering is the same as that required to decompose the solid (= E_c). Clearly this is not a good way to proceed if you are interested in the sputtering properties of a specific crystallographic surface.

If you wish to include a surface binding energy term (E_s) in your simulation, you can specify its magnitude in the Model file of your *Simulation Kit* project. In the current implementation of *Snook*, any required corrections to ejected particle trajectories will be made at the moment when the particle leaves the surface (defined by the initial z position of the anchor atom). Additionally you can specify (in the Model file) a 'bulk' binding energy* term which is applied in a similar fashion to particles which attempt to exit the target by any of its side faces. [Tip for advanced users: surface and bulk binding energy corrections are only applied to the motion of a particle if its `ofEmitted` and `ofNotContained` option flags are not yet set (see Chapter 11).]

The 'surface' binding energy correction implemented by *Snook* takes the form of reducing the perpendicular velocity (v_z) component of any particle (including the projectile) which is found to be emitted from the lattice, such that its kinetic energy falls by E_s. In cases where $\frac{1}{2}mv_z^2 < E_s$, the energy deduction is not carried out: instead, the sign of the z velocity component is reversed, and the particle suffers a reflection at the internal surface of the target. The 'bulk' (or edge) binding energy correction (if selected) is applied in a similar manner to particles exiting throught the side faces of the lattice.

The definition of 'emitted' or 'not contained' in this context is that (a) the particle separation from the bounding surface is greater than the cut-off distance of the potential, and (b) the velocity component is in a direction that moves the particle away from the surface (or a side face). A binding energy correction is applied only once (if at all) to any given escaping particle. But a non-escaping particle may be reflected internally at the surface many times. Thus, to sum up, the net effect of the correction is to apply either a 'reflection' or a 'refraction' to the trajectories of particles that cross the surface or side faces.

Note that the implementation of this feature does not conserve linear momentum for those particles that are reflected or refracted at the surface/edge. In the Model file, you should specify a value of 0.0 for the Surface (Bulk) Binding Energy (the default values) if you don't want to include a surface (bulk) energy correction. However, if you choose to include a non-zero binding energy, you will have to select a suitable value based on literature sources or models external to the *Simulation Kit*. None of the example projects discussed in this article incorporated such binding energy effects.

---

* The name is misleading: a better term might be 'surface energy for side faces of target crystallite'.

4.3.6. Image potential

A charged particle approaching the surface of a conductor experiences a classical image potential $V(r) = e^2/16\pi\varepsilon_0 r$ (approximately), where r is the particle-surface separation. If r is expressed in Å, then $V(r) = 3.6/r$ eV in this model. Since most primary projectiles are charged particles, there is clearly some possibility that image potential effects can influence the trajectories of slow projectiles. This would take the form of an acceleration towards the surface along the z-direction, with a corresponding increase in projectile kinetic energy (on the order of a few eV). The image potential would tend to retard emitted charged particles, in a fashion similar to the planar potential discussed in the preceding section. If you wish, you can fake the effects of a retarding image potential (on emitted particles only) by using this method.

The public releases of *Snook* and *Kalypso* do not yet implement the handling of image potential effects.[*] This subject is rarely discussed in the literature of scattering simulation, although it may be important in connection with the angular distribution of secondary ions, for example. If you plan to run simulations involving projectiles of less than ~100 eV energy, you should be aware of the possibility of image potential effects, which will tend to increase the projectile velocity in the z-direction, and thereby increase the effective impact energy.

4.4. Integration methods

The classical equations of motion are integrated by *Snook* using any of several finite difference integration algorithms.[5] The Verlet algorithm, which is used by default, has already been presented in eqns. 3.1 and 3.2 of chapter 3. Another algorithm offered in *Snook*'s simulation options (not available in *Kalypso*) is the HGE-A algorithm, which is a so-called predictor-corrector method:

Predictors:

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{v}_n \Delta t + 0.5\mathbf{F}_n \Delta t^2 / m \tag{4.15a}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + 0.5[\mathbf{F}_{n+1} + \mathbf{F}_n]\Delta t / m \tag{4.15b}$$

Corrector for $\mathbf{r}_{n+1}$:

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{v}_n \Delta t + 0.25[\mathbf{F}_{n+1} + \mathbf{F}_n]\Delta t^2 / m \tag{4.15c}$$

The meanings of the symbols are: $\mathbf{r}_n$, $\mathbf{v}_n$, $\mathbf{F}_n$: position, velocity, total force vectors at nth timestep; Δt: size of the current timestep.

These integration algorithms are uncontroversial. The best guarantee of their accuracy at run-time is the accuracy of energy conservation, and this is how the user should evaluate them.[†] The accuracy of the integration depends on the timestep specified by the user in the Run file of a

[*] Privately, I have made provision for inclusion of image forces, which explains why you will see a message 'Ignoring image potential effects' on startup. However, this feature is not yet properly tested or documented.

[†] You need to study a few dozen runs or more before you can be sure that the timestep is suitable for accurate simulations. Set the timestep too small rather than too large.

simulation project. One integration algorithm offered by *Snook*, the Beeman algorithm, should be used with a fixed timestep, and is therefore considerably more inefficient than the other choices. In general, the Verlet algorithm should prove adequate for all simulations, but it may be interesting to check one of the others for consistent behaviour from time to time.

The total system energy, as the sum of potential energy (PE) and kinetic energy (KE), can be calculated at any instant from the current particle positions (using the analytic potentials) and velocities. This can be compared with the energy calculated at the start of the simulation, which yields the energy error ΔE. *Snook* uses the formula $\Delta E/(\mathrm{KE} + |\mathrm{PE}|)$ to report energy conservation. Total energies reported by the energy conservation routine are not exactly the same as the total lattice energy, because of the special book-keeping required to take account of potential cut-off effects. However, *Kalypso* also offers the option of calculating the true lattice energy. Apart from integration errors, small energy discrepancies can arise because *Snook* or *Kalypso* may fail to track inelastic energy losses with sufficient precision. (To characterise these, try running the simulation with all inelastic options switched off.)

The forces appearing in Eqs. 4.15 are not necessarily calculated directly from the analytic potential functions. In *Snook*, Morse forces are calculated from a look-up table (essentially by indexing the inter-particle separation, *r*, into an array of pre-calculated values, with interpolation as necessary). Other forces (spline, screened Coulomb) are calculated directly from the respective analytic functions. In *Kalypso*, both Finnis-Sinclair and spline forces are calculated from look-up tables. The majority of particles in the system interact at any instant via long-range forces, so the use of a look-up table speeds the force calculations significantly.[*]

Another device which improves the speed of calculations is the use of neighbour lists. A particle's neighbour list is a list of those other particles in the system with which it may interact in the period before the next list update (normally carried out every ~10 timesteps). The neighbour lists are built up by taking into account the potential cut-off distance and the velocity of the fastest particle in the system (which determines the width of the sheath that has to be examined for potential collision partners). The 'Number of Partners' parameter which is specified in the Run file defines the maximum allowed size of the neighbour list (i.e. the memory allocation for the list). This parameter is typically set at 50-60, although larger values might be needed on occasions (e.g. if the cut-off is beyond the third neighbour distance).

Snook does not employ the 'moving atom approximation' described in refs. [1,5], which ignores the effect of small forces (in effect introducing a displacement threshold). The only special treatment that *Snook* (not available in *Kalypso*) accords to small forces is in connection with the HGE-A algorithm, eqns. 4.15: small pairwise forces can be treated as if they were constant during the integration timestep, the corrector force being approximated to its value at the start of the timestep. The definition of small forces is selected by the user of *Snook* (in the Options|Simulation dialog).

4.5. Inelastic processes

The physical models of inelastic energy losses used by *Snook* and *Kalypso* are described in Chapter 8. Inelastic energy loss processes are not well understood for keV particles, and all

[*] Floating point calculations, especially those involving the exponential or other functions, are extremely time-consuming.

extant models have something of a heuristic character. Experimentally, it is not always easy to identify the effects of inelastic energy losses. The influence of discrete inelastic events on simulated processes (e.g. sputtering) has been examined in detail recently by Shapiro and Tombrello.[33] They discuss several methods of incorporating inelastic effects into simulation models, and find that in some cases the inclusion of the inelastic effects leads to substantial differences (~30%) in the predicted sputtering yields compared with the elastic model. At first sight, this is an unexpected result, since for 5 keV Ar bombardment of Cu(100), the average inelastic energy losses were computed to be 50-100 eV, or 1-2% of the projectile energy. These inelastic losses are of the same order of magnitude as the energy losses associated with integration errors in the simulation (~0.5%). However, the explanation probably lies in the fact that inelastic events are correlated with the same hard collisions that result in efficient sputter yields. This is an area of research which requires further investigation.

4.6. Binary compound targets

4.6.1. Snook

Snook is principally designed for use with elemental targets. However, it can be used with binary (or arbitrarily complex) targets provided that (a) each of the various long-range target-target interactions (X-X, X-Y and Y-Y respectively) can reasonably be described by an identical Morse function, or (b) the experiment is relatively insensitive to the long-range attractive interaction..

The approximation (a) may well hold good for some common binary alloys or compounds, XY, such as CuNi or GaAs, in which the constituent atoms have similar electronic shells and occupy similar atomic sites. Small differences in surface binding energy can be accommodated by applying the surface binding energy to one component but not to the other. Simulations which depend on correct modelling of the attractive interactions should use *Kalypso* rather than *Snook*.

Snook runs noticeably more slowly with binary targets (which it detects by checking the atomic numbers in the Target file during initialisation), because these involve significantly more computations. It should be noted that in binary targets, the calculation of the several screened Coulombic interactions (both target-target and target-projectile) will always be handled correctly, regardless of the target composition. Spline functions joining the screened Coulombic potential to the (mutual) Morse potential will also be appropriate for each target-target interaction permutation (X-X, X-Y and Y-Y respectively).

The imposition of a single Morse function on all long-range target-target interactions by *Snook* may disqualify the simulation model from use in some applications: for example, preferential sputtering effects in alloys presumably cannot be accurately reproduced by such a model. However, this restriction would be irrelevant for the simulation of impact collision ion scattering spectra, which are mainly influenced by the short-range potential. Chapter 9 offers some suggestions on how to simulate collisions involving binary systems (e.g. adsorbed overlayers), by using options flags associated with each particle to switch off or modify the form of the short-range potential and surface energy respectively. This is a partial solution which addresses the problem of target stability in such a system. However, it does not solve the problem of modelling the attractive interactions.

4.6.2. Kalypso

Kalypso is designed to be used with both elemental and binary compound targets. The major difficulty for compound systems lies in choosing good values for the interaction cross-terms. It is possible to fit the cross-terms using thermodynamic data, but in the absence of better information, a heuristic method is to choose the cross-terms (interactions where atom *i* type ≠ atom *j* type) such that, with reference to Eqs. 4.4 and 4.8, the heteronuclear interaction is described by the geometric mean of the corresponding elemental terms:

$$V_{pair}(r_{ij}) = \sqrt{V_{pair}(r_{ii})V_{pair}(r_{jj})}\,, \tag{4.16a}$$

$$\xi_{ij}\Phi(r_{ij}) = \sqrt{\xi_{ii}\xi_{jj}\Phi(r_{ii})\Phi(r_{jj})}\,. \tag{4.16b}$$

The reader should consult the literature for a justification of this approximation, which would can be expected to work best for atoms of similar size and electronic properties.[34,63] One of the example projects which ships with the *Kalypso* package can be used to simulate sputtering from a binary target consisting of 0.5 monolayer of Ni deposited on a Cu(100) crystallite substrate. The potential parameters of this project were chosen using the geometric means method. The heteronuclear r_0 parameter is an average of the two (nearly-identical) estimates which result from application of Eqs. 4.16. See chapter 10 for further details.

Many of the common bimetallic systems have been studied in atomistic simulations previously: a search through the journals listed in section 10.1 for the period 1984 onwards, would probably turn up articles with useful data and tips for any system you might be interested in.

Table 4.1. Tight-binding potential parameters for Cu-Ni binary compound systems (see sections 10.4 and 10.5).

	A (eV)	ξ (eV)	p	q	r_0 (Å)	S_n
Cu-Cu	0.15653	1.2355	11.183	4.6394	2.5560	12.881
Ni-Ni	0.11300	1.4005	14.087	3.5874	2.4918	13.359
Cu-Ni	0.13299	1.3154	12.635	4.1134	2.5237	13.118

5. SPUTTERING SIMULATIONS

5.1. Introduction

The potential energy function is an important aspect of a sputtering simulation. A key property of the potential is the binding energy which it predicts for surface atoms. The sputter yield is typically inversely proportional to the surface atom binding energy. Many-body potentials give better estimates of this quantity than do pair potentials, but the exact behaviour depends on various factors such as the potential range and the surface structure of the sputtered target.

The failure to incorporate many-body effects in the target potential energy function may lead to significant differences in the properties of the sputtered atom flux for a strongly binding metal like Rh,[35] but nevertheless, pairwise potentials (and even planar potentials) can reproduce many characteristics of the sputter yield behaviour (e.g. dependence on primary ion energy and angle of incidence). Ref. [36] compares sputter yields predicted by a pair-potential MD model, and a planar potential BCA model, with the experimental data. Section 3.5 compares sputter yields predicted for Ag by Morse and TB potentials. Ref. [40] gives a similar comparison for Ca. In all cases, the differences between the models are quite subtle.

In any kind of simulation, the relative sputter yields tend to be more reliable than the absolute sputter yields. Furthermore, it is important to remember than experimental yields always refer to high ion fluences, whereas simulations refer to the zero fluence limit.[*]

Sputtering simulations at a fixed incident angle are one of the easiest simulations to set up and perform, and they do not make unreasonable demands on computing resources. Most keV sputtering simulations reported in the contemporary literature use targets of 2000-5000 atoms, and 300-1000 projectile trajectories (runs). Simulations are terminated after ~1000 fs, or when lattice particle energies fall below 1-2 eV. A thorough *Kalypso* simulation involving ~1000 runs of 1000 fs duration on a 5000 atom binary compound target takes about 3 days on a 200 MHz Pentium computer. For ~300 similar runs on a 2000 atom target the time required is about 0.5 days. For a given simulation problem, *Snook* is roughly twice as fast as *Kalypso*.

Some difficulties arise in the interpretation of results (see below), but general agreement with experimental sputtering coefficients is not difficult to achieve. Predicted sputter yields tend to be higher than those observed experimentally, possibly due to failure to dissipate energy in the small crystallite. For example, in some materials, e.g. Ni, electron-phonon coupling may quench the collision cascade, thereby reducing experimental sputter yields relative to a purely elastic model. This area is a subject of active research. See ref. [37] for further discussion.

A good literature source for sputter coefficients of polycrystalline materials is the article by Andersen and Bay,[38] while Roosendaal reviews single crystal yields in the same book.[39] More computational effort is required for simulations of sputtering involving various projectile incident angles (see chapter 6), but no new principles are involved. When setting up a sputtering simulation, you will need to watch for problems of lattice containment and lattice stability.

---

[*] Experimentally, one could measure sputter yields down to quite low fluences (~0.1 ML removal) by means of a quartz crystal microbalance. However, nobody seems to have done this.

5.1.1. Lattice stability and lattice containment

Lattice stability becomes an issue in sputtering simulations because of their long time span (~1000 fs). Over this time scale, you may see a tendency for the surface layer and edges of the lattice to "relax" towards a more stable configuration. A test for lattice stability is to use a projectile of negligible kinetic energy (0.1 eV, for example) and to start it far away from the surface (the starting point is specified as the z_0 parameter in the Impact file): this will in effect isolate the lattice from interactions with the projectile. This type of simulation must be carried out with a fixed timestep (which can be selected in *Snook*'s simulation options).

Excessive surface relaxation might indicate an error in the parameters of the attractive potential. For example, the cut-off potential specified in the simulation may not be the same as the cut-off used in the potential fitting procedure. However, most potentials give rise to surface relaxation to some extent (~0.1 Å typically). The Morse potential results in outward relaxation (except for Morse potentials cut off above the first neighbour, which do not produce surface relaxation), whereas inward relaxation results from the Finnis-Sinclair potentials. Physically, the latter is more realistic.

To check the surface relaxation associated with the target/potential combination, you need to plot variations in the lattice energy as a function of the surface interlayer distance. For Morse potentials use the Target|Neighbour Potential feature in *Spider*; for FS potentials (*Kalypso*), use the Lattice Energy simulation option in *Kalypso* itself. Use a text editor with search/replace capabilities to modify the Target file *z* coordinates for the surface layer atoms. If you want to eliminate relaxation effects, you can choose the target configuration which has the lowest energy. Note, however that relaxation of the lattice will have little, if any, effect on sputter yield predictions.

Like surface atoms, edge atoms can gradually reconstruct or diffuse during the course of a simulation. The best way to deal with this is to use a large target, so that any reconstruction takes place at a distance from the region of interest for the simulation. BCC lattices have a tendency to convert to FCC structures, due to the centrosymmetric character of the potential (which favours FCC and HCP structures). You will have to evaluate this problem on a case by case basis. Reconstruction may impose some restrictions on the simulation methodology. Long simulations (> 1 fs) will be most prone to problems of this type.

Lattice containment refers to the tendency for the effects of the collision to propagate beyond the boundaries of the finite sized crystallite used for the simulation. This problem can be identified by visual inspection, or by comparing the estimates for sputtering coefficients run on targets of different sizes. To avoid influencing the simulation predictions, your target should be large enough to avoid significant sputtering of edge atoms. Even with quite large lattices, 1-5% of the sputtered atoms can originate from the edge layers of the target. These sputtered edge atoms should be regarded as an artefact of the simulation method. The simulation output for these edge atoms can be separated from other data by careful application of the Merge/Remerge commands provided by *Winnow* (see section 5.2.2; alternatively you can write your own routines, or like most people ignore the problem altogether).

Fig. 5.1 shows the lateral distribution of sputtered atom sites for the 4 keV Ar-Ca(100) system.[40] In Fig. 5.1, most sputtered atoms originate within 30 Å of the impact zone. For sputtering simulations involving keV projectiles you may have to tolerate some lattice containment

violations, since to remove them completely may require an unrealistically large lattice. Focussed collision sequences along the *x*- and *y*-axes are responsible for pushing out some edge atoms, for example. It may be a good idea to ignore these in the final sputter count.

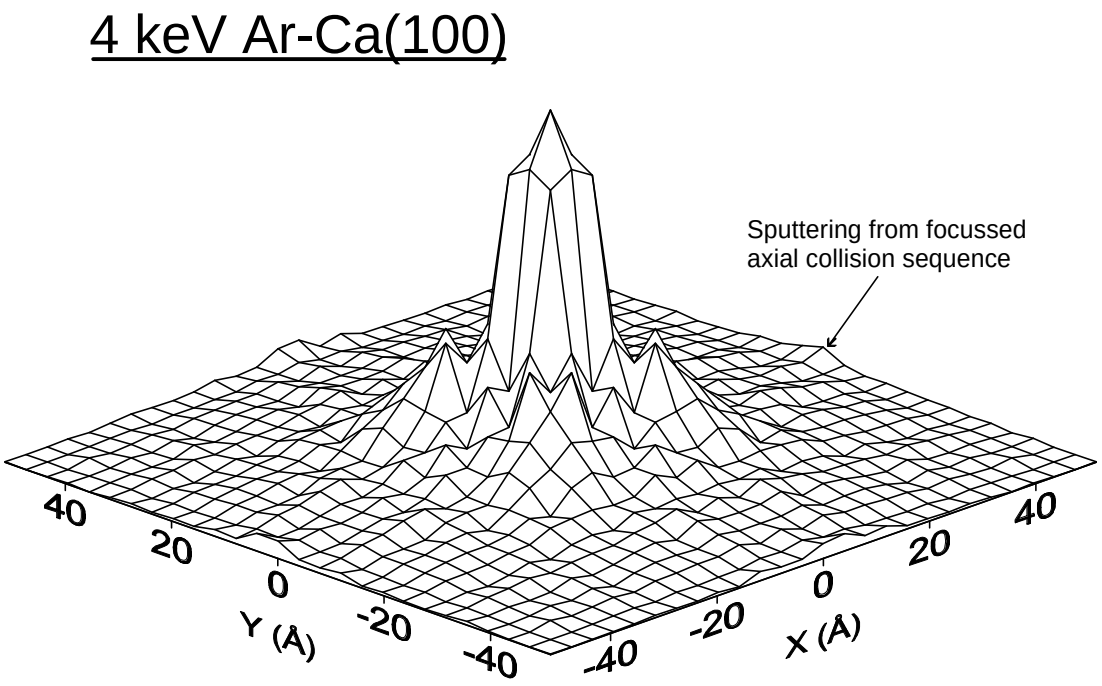


Fig. 5.1. Origin of sputtered atoms during 4 keV Ar bombardment of Ca(100) at normal incidence. The Ca(100) target has 625 atoms/layer, and 9 layers. The original (*x*, *y*) atomic positions of the surface Ca lattice sites lie at the grid intersections. The projectile makes impact in a zone centred at $x = 0$, $y = 0$. The z-axis shows the relative emission intensity for sputtered atoms from lateral (x,y) positions. The data are averaged over 2025 projectile trajectories.[40]

5.2. Sputter coefficients

5.2.1. Method

Sputter coefficients (γ = number of atoms sputtered per incident projectile) for metals are typically of magnitude 0.1-10 for projectiles in the keV range. Sputter coefficients are predicted by measuring the number of atoms which leave the surface during the course of a series of sputtering simulation runs. The method used by the author is essentially as follows.

1. The Run file is set up to record the coordinates of all emitted atoms at the termination of the simulation (a few hundred fs or more). This means any atom for which z > 0.5 Å, and includes the projectile. A sputtering simulation is then executed with (say) N_r runs.
2. The output file, `dynvars.snk`, is then filtered using *Winnow*'s Process|Filter command. A filter condition like the following is applied: `[rw > 0] & [vz > 0.0] & [rz > 6.0E-10]`.* The first term `[rw > 0]`filters out records that refer to the projectile. The second term

* Different systems might require different definitions of what constitutes a sputtered atom.

`[vz > 0.0]` filters out particles that may be drifting back to the surface. The third term `& [rz > 6.0E-10]` filters out particles which are within interaction range of the surface, assumed in this case to be 6.0 Å (see 5.2.2).

3. *Winnow* reports the number of records that satisfied the filter query (N_f). the sputtering coefficient is then obtained as $\gamma = N_f/N_r$.

Table 5.1 summarises predictions for the sputter yield of Ca(100) bombarded by 4 keV Ar projectiles using various Ca-Ca potentials. (The differences between these predictions can be completely explained by the different surface binding energies predicted by the potentials.) The predicted yields are higher than expected, based on the estimated value for polyx Ca. The energy cost to sputter an atom from a surface is reviewed by Garrison *et al*.[41]

Table 5.1. Predicted and experimental total sputter yields for 4 keV Ar projectiles normally incident on Ca(100) surfaces.[40] The predictions used Sutton-Chen (SC) and Morse potentials which were fitted using the properties of either the fcc (A) or bcc (B) phases of Ca.

	SC-A	SC-B	MORSE-A	MORSE-B	EXPTL.
Ca(100)	1.10	1.02	1.28	1.10	
polyx Ca					0.7-1.2

By applying additional filter conditions, partial sputter fractions can be obtained. For example, suppose there are 1000 atoms in the surface layer. Then the addition of another filter condition `[rw < 1001]` will allow you to count only those sputtered atoms which originated from the surface layer. If you want to know how many atoms were sputtered with an altitudinal angle of 70±10° you can add the condition `[altd < 80.0] & [altd > 60.0]`. If you want to count how many sputtered atoms have energies above 10 eV you can add the condition `[ke/ep > 10.0]`. These examples illustrate the usefulness of filtering operations in *Winnow*, but you can also carry out similar operations using database and spreadsheet programs if you prefer.

5.2.2. What can go wrong

It is not always easy to locate the boundary between the flux of sputtered atoms and any bound adatoms (readsorbed layer) that are produced by the projectile impact (i.e. the value to use in the `rz` filter discussed in the preceding section). Fig. 5.2 illustrates the typical situation for an Ar-Ni(100) sputtering simulation which was terminated after 1000 fs.

Near the surface, one or two layers of (re-) adsorbed target atoms (known as 'adatoms') tends to form. The target surface near the projectile impact region also tends to bubble outwards. It is important not to include these atoms in the sputtering estimate. At a greater distance from the surface (about 25 Å in this example), there is a 'pulse' of sputtered atoms. Somewhere between these extremes lies the boundary that optimally defines the sputtered atom region. In this example, it has been chosen to be 6 Å. This is about 4 Å above the 2nd adatom layer (at 3.8 Å). Undoubtedly some low-energy sputtered atoms will be missed by this technique.

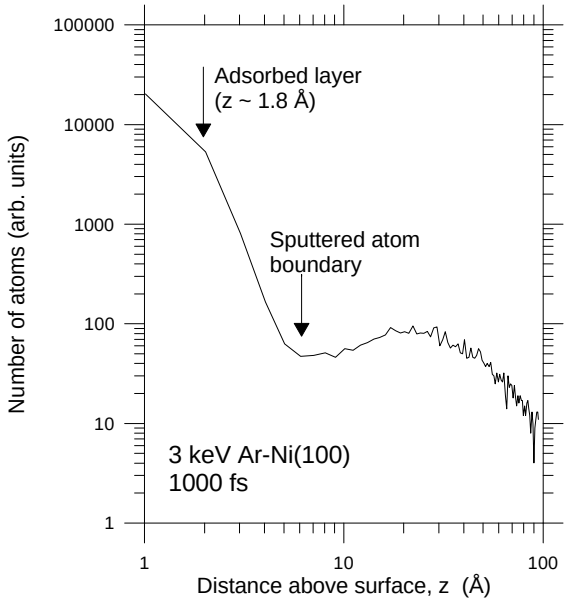
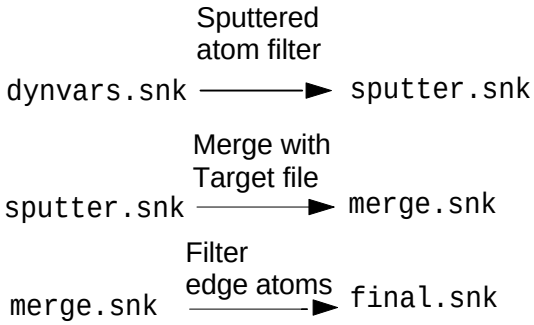


Fig. 5.2. Distribution of Ni atomic positions above the surface for a Ni(100) target 1000 fs after bombardment by 3 keV Ar projectiles (based on ~1000 runs). The potential cut-off used in the simulations was 4 Å. The maximum density of sputtered atoms at termination is at ~25 Å above the surface.

Lattice containment effects are an obvious source of error in sputter coefficient predictions. Edge atoms can be sputtered from sub-surface layers by a focussed knock-on sequence (see Fig. 5.1). Although this has a minimal effect on the total yield, it may cause significant error in the layer-specific yield. The only way to deal with this is to ignore sputtered edge atoms in the yield count. There is no elegant way to do this. The Merge option in *Winnow* allows you to overwrite (replace) coordinates from a simulation output file with those from a Target file, which suggests the following sequence:



In `merge.snk` the (*x*, *y*, *z*) coordinates of the sputtered atoms are replaced by the (*x*, *y*, *z*) coordinates from the Target file. The edge atoms can then be filtered based on the known dimensions of the target. The edge atom filter takes the following form (assuming that atoms from lattice rows located within 30 Å of the origin are to be counted): `[abs(rx) < 30E-10] & [abs(ry) < 30E-10]`*. Optionally, you can also replace the (*x*, *y*, *z*) coordinates in `final.snk` with those appropriate for the sputtered atoms using the Remerge command in *Winnow*.

* The function `abs()` returns the absolute value of its argument.

5.2.3. Prediction accuracy

The absolute values of simulated sputter yields often turn out to be higher than experimental yields, but are usually within 50% or so of the experimental yield. There is no obvious explanation of this, but it should be pointed out that most experimental sputter yields have been determined under highly dynamic bombardment conditions, which implies considerable surface damage. A possible cause of error in the predicted yields is that the projectile-target potential is too hard, thereby overestimating the energy transfer at the surface.

Another possibility is improper handling of inelastic energy losses and cooling effects. The termination time has a role to play here. Longer simulation times do not necessarily mean more accurate predictions of bulk properties, because of evaporation effects. The attractive potential, especially in a pairwise model, may underestimate the strength of binding at the surface.[*] Lastly, simulation models typically ignore cooling due to electron-phonon-coupling,[†] which may overestimate the yields. Relative yields, for different projectile-surface configurations, different interaction models, different termination times etc., do tend to be quite consistent, however. This is especially satisfactory if your interest is mainly in relative sputter yield variations.

5.3. Projectile angular effects in sputtering

There are several distinct kinds of experiments which explore angular effects in sputtering processes. The corresponding simulation procedures are generally similar for each kind of experiment, although those simulations where the projectile incident geometry varies are best handled as batch jobs (see the discussion about ICISS in the following chapter). A simulation is linked to a particular experimental measurement only by the criteria used at the output-processing stage to categorise sputtered particles. For example, the output data from a given simulation could equally well be processed by *Winnow* to give a plot of either the azimuthal or the altitudinal distribution of ejected particles (or of many other kinds of distribution, e.g. the energy distribution): the ultimate use of the output data need not be known or specified at the time that the simulation is run. This flexibility is an important advantage of the *Simulation Kit*, although it is achieved at the expense of increased complexity for the user.

Winnow's predefined angular variables (`phid`, `altd`,[‡] `phi2`, `phi4`, `phi8`[§]) ease the labour of extracting angular information. Suppose you want to make a plot of the azimuthal distribution of ejected particles (excluding the projectile) at an altitudinal angle of 45±3 °. Then you would follow this type of procedure:

---

[*] This can be corrected by application of an additional (small) correction to the surface binding energy (in the MDL file).

[†] Cooling due to electron-phonon coupling can be included in the simulation via the cooling option, if required (and if the physics justifies it).

[‡] `phid` and `altd` if expressed in degrees, or `phi`, `alt`, if expressed in radians.

[§] `phi4`, `phi8` are used to map particle trajectories into the first azimuthal quadrant (ϕ = 0-90°) and octant (ϕ = 0-45°) respectively. They are only useful for (110) and (100) cubic surfaces at normal projectile incidence, where these symmetries can be assumed. `phi2` maps particle trajectories into the first two azimuthal quadrants (ϕ = 0-180°) and can be useful for simulations involving off-normal projectile incidence.

1. Filter the output file to obtain records for sputtered atoms moving at the desired altitude: `[rw > 0] & [rz > 6.0E-10] & [vz > 0.0] & [altd > 42.0] & [altd < 48.0]`.
2. Use *Winnow*'s Process|Spectrum option to generate the distribution of the variable `phid` over the angular range 0.0 to 360.0.

The possible range of the angular variable `altd` is -90.0 to +90.0 (°), but the negative values (implying motion into the bulk) are rarely needed. Positive values of `altd` imply positive values of `vz` (the perpendicular velocity), which means that the term `[vz > 0.0]` in the above filter can be omitted, since it is redundant.

The major difficulty encountered in predictions of the angular distributions of sputtered atoms is the small number of counts in any specified angular direction. Fig. 5.3 shows the altitudinal distribution of sputtered Ca atoms from a Ca(100) surface. The plot is an average over sputtered trajectories (~1000 runs) into all azimuthal directions. The difficulty of obtaining a noise-free plot for a restricted azimuthal range is evident.

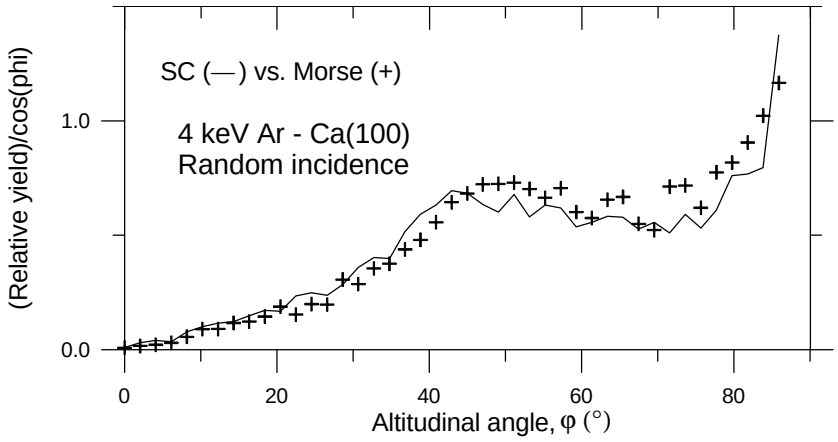


Fig. 5.3. Relative altitudinal angular emission cross-section (averaged over all azimuthal directions) for Ca atoms sputtered from Ca(100) using a random projectile (3 keV Ar) direction of incidence. The simulations used either the Sutton-Chen (SC) or Morse potentials.

It has long been known that sputtered particles are preferentially ejected along directions parallel to the more closely-spaced atomic rows.[42] This can be observed experimentally in the phenomenon of Wehner spots. Wehner spots are prominent features in the images formed by sputtered particles incident upon photographic plates (or other imaging devices) facing the plane of the surface under bombardment (a set-up similar to that used for Laue or electron diffraction). Wehner spots can be investigated through simulations by making plots of the asymptotic sputtered particle momenta in the (x, y) plane. As an example, Fig. 5.4 shows a plot of the quantities (as defined by *Winnow*) `atan(px/pz)` against `atan(py/pz)` for the Ar-Ni(100) system.* The data are produced by *Winnow*'s Format Columns command.

Sputter yields also depend strongly on the projectile angle of incidence. This dependence can be exploited, for example, for structural determinations using angle-variation SIMS. However, the simulation of this type of effect is quite time-consuming, since many projectile directions of incidence (30 or more) need to be sampled. Once the simulations have been run, the sputter

* This projects the spots onto a flat plane, which introduces distortion. For a polar plot, one should use the expression (x-axis): `px*atan(sqrt(px*px+py*py)/pz)/sqrt(px*px+py*py)` (and analogously for y-axis) which will project the spots onto the surface of a sphere (where $r = x^2 + y^2$ represents the angular variable in radians).

yields or other information has to be extracted from the output files. In the author's experience, it is very tedious to do this using *Winnow*: it is better to write a small computer program that will search for output files in the project directory, and process them according to need.

Fig. 5.4. Wehner spots predicted for 3 keV Ar bombardment of Ni(100). The limits of the plot correspond to $p_z = 0$, or $\varphi = 0°$, while the origin represents $\varphi = 90°$ (normal emission). The directions of preferential emission (diagonals of plot) are parallel to <001> surface rows.

Small C and Pascal programs can be found under the `examples\Winnow` directory of the *Simulation Kit* or *Kalypso* installation that demonstrate how to read output (SNK) files. These can be modified for custom data processing tasks.

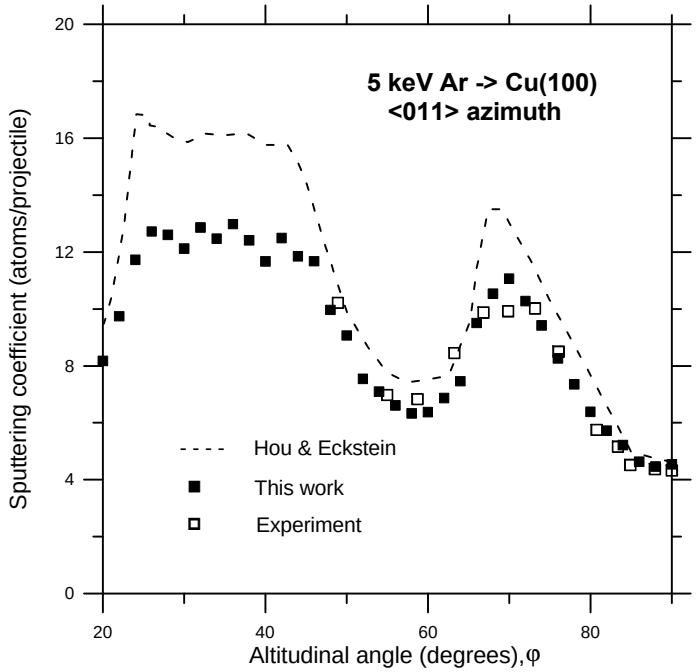


Fig. 5.5. The sputtering coefficient, shown as a function of the altitudinal angle of incidence, for bombardment of a Cu(100) surface by 5 keV Ar projectiles. The target consisted of 9 planes of 225 atoms each (2025 total). The projectiles are incident in the <011> plane. Black squares: sputtering coefficients calculated with Snook. White squares: experimental sputtering coefficients.[43] The dashed line shows the sputtering coefficients calculated by Hou and Eckstein[44] using the MARLOWE BCA program.

Fig. 5.5 compares the dependence of the sputter yield on the projectile altitudinal angle predicted by the *Simulation Kit* (SK) and by the BCA program MARLOWE, with that measured experimentally. The good agreement between the predicted and measured sputter yields is probably fortuitous. The SK simulations were terminated after 200 fs, and would tend to underestimate the yield of some slow sputtered particles compared to a longer simulation time. This is of minor importance if, as is normally the case, the experimental parameter of interest is the relative angular yield.

5.4. Example project: 0.4 keV Ne-Ag(111)

The input files required for the sputtering example project can be found in the `examples\ne-ag111` directory of the SK or *Kalypso* installations. The simulations involve bombardment of a 726 atom Ag(111) surface by 0.4 keV Ne projectiles, which are incident along the normal direction. Fig. 4.2 of the previous chapter shows the reduced impact zone into which the 376 projectile trajectories are directed. You can run this project by loading the input files into either *Snook* or *Kalypso*. You can examine the input files (except the Target and Impact files) using *Spider* (take care not to modify them by accident).

For a research simulation one would use a larger target (1500-2000 atoms) to improve containment. A small target will show a greater tendency to evaporate over long simulation times, and will lose its edge atoms. Both effects artificially raise the sputter yield. The purpose of this example, however, is to illustrate the simulation methodology. The simulations are terminated after 1000 fs. The Ag-Ag potential cuts off at 4.5 Å (between the 2nd and 3rd neighbours).

In the Run file of this project, the output information requirement is specified by the 'user-defined' expression: `[rz > 1E-10]`. This means that a record will be written to disk for any atom (including the projectile) which is found to be more than 1 Å above the surface at termination (t = 1000 fs). Before calculating the sputter yield, it is helpful to have an idea of how the Ag atoms are distributed at the end of the simulation. This is achieved by applying the Spectrum option of *Winnow*[*] to the output file produced by filtering `dynvars.snk` with the condition: `[rw > 0]` (a filter that removes projectile atom records from `dynvars.snk`). The resulting spatial distribution is shown in Fig. 5.6.

The results obtained by *Snook* and *Kalypso* are similar in the region > 6 Å above the surface. The cut-off point for sputtered atoms appears to lie at ≥ 4 Å for the *Snook* results, and at ≥ 6 Å for the *Kalypso* results. There is no agreed way to set this boundary. The potential cut-off defines a possible lower bound (4.5 Å). In this case, a value near 6 Å seems reasonable. This avoids counting the first two adatom layers formed by atomic relocation of Ag atoms during sputtering (see next section). A plot of log(N) vs. z (as in Fig. 5.6) may be helpful in distinguishing between adatoms and sputtered atoms.

The sputter yield is obtained by filtering the output data (in `dynvars.snk`) with the filter expression: `[rw > 0] & [rz > 6E-10] & [vz > 0.0]`. This removes records which refer to (a) projectile atoms (the condition `[rw > 0]`); (b) atoms near the surface (the condition `[rz > 6E-10]` for a 6 Å boundary position); and (c) atoms travelling back to the surface (`[vz >`

[*] Spectrum independent variable: `rz*1.0E10`, with 40 bins in the range 0.0 to 20.0.

`0.0]`). The number of records which satisfy the condition is then divided by the number of incident projectile trajectories (376) to give the sputter yield.

Table 5.2 shows the sputter yields calculated on the assumption of different positions of the boundary between the true sputtered atom flux, and the target surface (including adsorbed matter). The sputter yield falls in the range 2.2-2.4 for various choices of the sputtered atom boundary. The experimental value for polycrystalline Ag is about 1.8;[38] the value for Ag(111) should be similar at this energy. The simulation thus gives fair agreement with experiment, but this cannot be expected for the general case. The sputter yields predicted by *Snook* and *Kalypso* are quite similar.

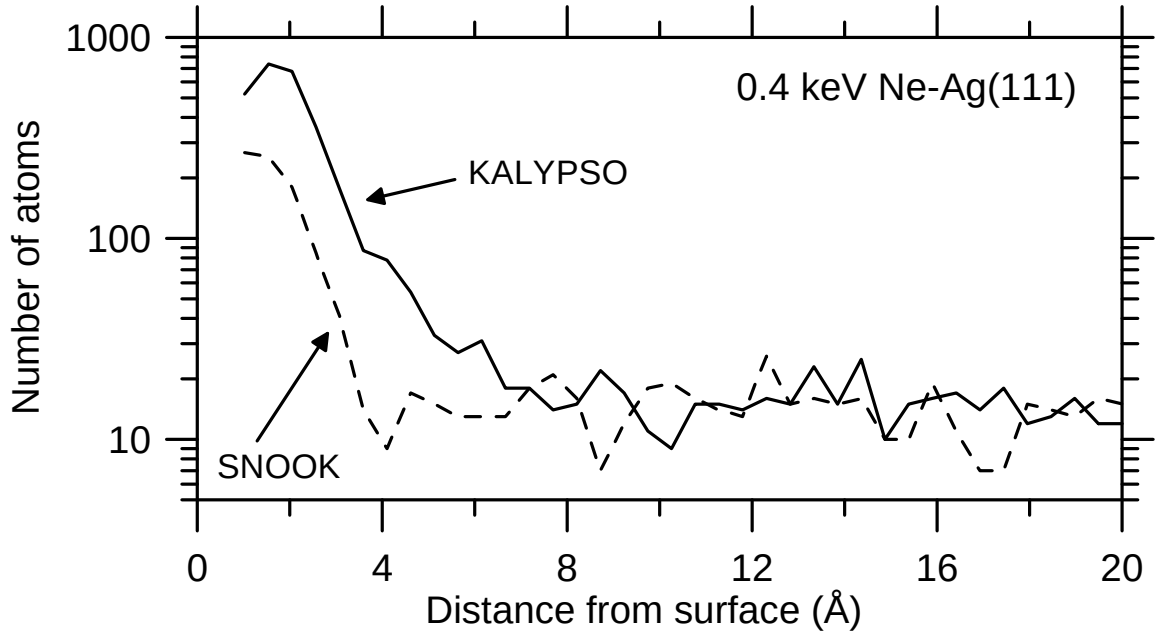


Fig. 5.6. Distribution of Ag atomic positions above the surface for a Ag(111) target 1000 fs after bombardment by 0.4 keV Ne projectiles (based on 376 runs). The potential cut-off used in the simulations was 4.5 Å.

Table 5.2. Number of sputtered atoms (and sputter yields in parentheses) calculated for 0.4 keV Ne-Ag(111), assuming different boundary points in the *z*-direction for the sputtered atom flux.

	Snook	*Kalypso*
$r_z > 6$ Å	880 (2.34)	906 (2.41)
$r_z > 8$ Å	821 (2.18)	845 (2.25)
$r_z > 20$ Å	489 (1.30)	494 (1.31)

Fig. 5.7 shows the energy distribution of atoms sputtered from Ag(111) by 0.4 keV Ne projectiles. The results for the *Kalypso* simulation peak at a higher energy than those for *Snook*. This is most likely due to the different surface binding energies associated with the Morse and TB potentials.[35] The distribution is obtained with the Spectrum command of Winnow (using the expression `ke/ep` over the 0-20 eV energy range.

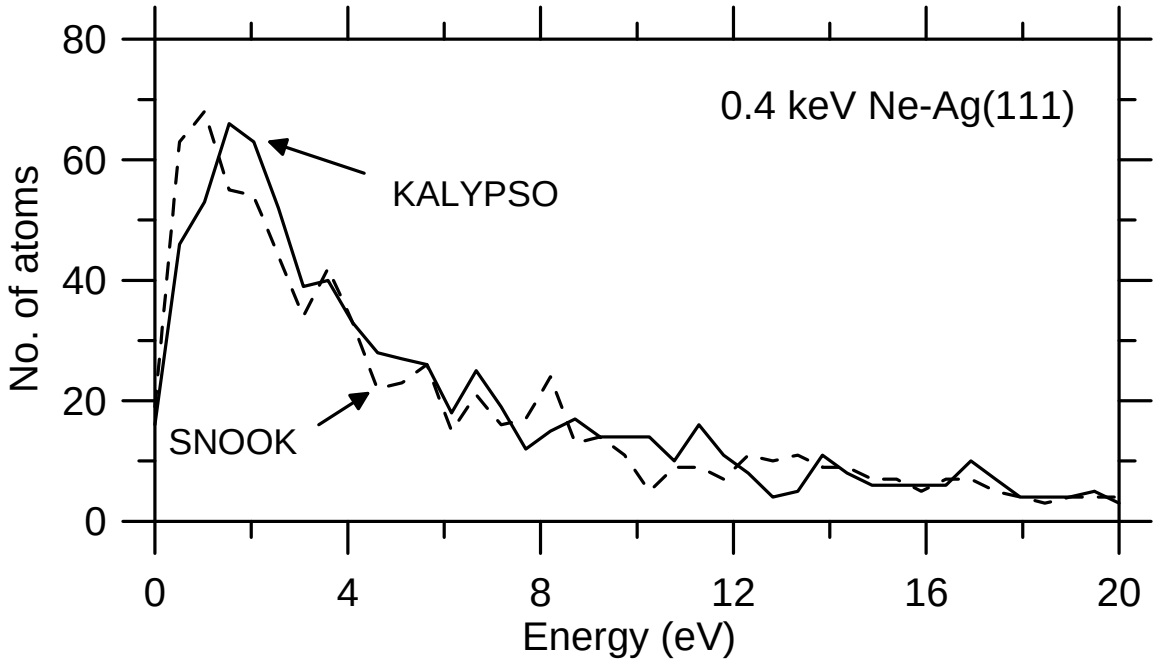


Fig. 5.7. Energy distributions of atoms sputtered from Ag(111) by 0.4 keV Ne projectiles.

5.5. Processes in the target

Atomic relocation in the target is an inevitable consequence of ion bombardment. Fig. 5.8 shows the predicted distribution of Cu atoms after bombardment of a Cu/Ni(100) target. To produce Fig. 5.8, it was necessary to store records for all Cu atoms located in the region of interest at the end of the sputtering simulation. This region was essentially a cylinder centred on the origin with a radius (in the *xy* plane) of 25 Å, and a length (*z* axis) of ±6 atomic layers from the surface. The easiest way to achieve this is to store records for all (or most) of the Cu atoms, then to filter them later as needed using *Winnow*. For simulations of this type, considerable disk space (> 100 MB) may be required.

The *z*-distribution shoen n Fig. 5.8 can be readily generated with Winnow's Spectrum option using the expression `rz*1E10` and a suitable range of depths (expressed in Å).

The simulations on which Fig. 5.8 is based included cooling effects which were applied with a 250 fs cooling period (equivalent to a time constant). Without this procedure, the surface layers will normally retain a molten character, which will obscure the distinct adatom layer structures visible in Fig. 5.8. The choice of cooling period was dictated by a theoretical model of heat transfer via electron-phonon coupling. For more details, the reader is referred to the literature.[45]

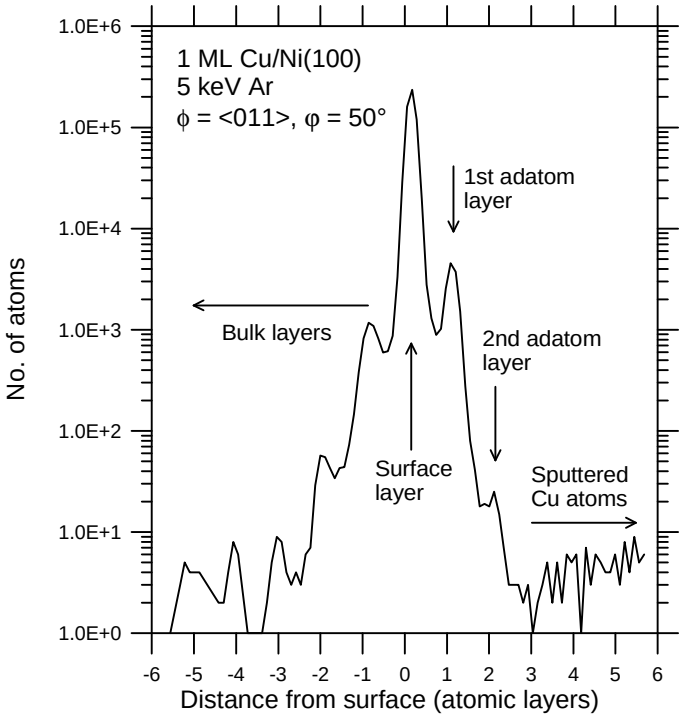


Fig. 5.8. Depth distribution of Cu atoms following 5 keV bombardment of 1 ML Cu/Ni(100) target. The Cu monolayer in the undisturbed surface is located at layer #0.

6. IMPACT COLLISION ION SCATTERING

6.1. Introduction

The repulsive interaction between an incident atomic projectile and a target atom at the surface of a lattice creates a region (known as the shadow cone) into which the primary projectiles cannot penetrate. Impact Collision Ion Scattering Spectrometry (ICISS) is an ion scattering technique which uses the shadow cone effect to derive structural information.[7,46] The shadow cone formalism has also been applied to explain projectile-incidence anisotropies in other techniques/processes such as SIMS,[47] ion-induced secondary electron emission,[48] and ion-induced Auger electron emission.[49]

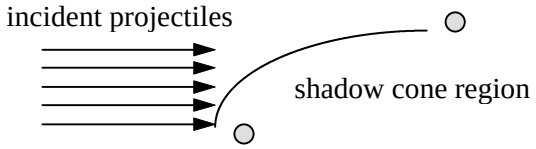


Figure 6.1. Shadow cone region formed by trajectories of projectiles scattered by repulsive potential.

The simulation of measurements made in ICISS experiments is one of the most demanding applications of the *Simulation Kit*.* This is chiefly because ICISS measurements only detect backscattering of projectiles into a narrow angular range (~2°). For this reason, practical ICISS simulations require the use of simplified scattering models and a modest number of target atoms. Using these simplified models, a wide-range ICISS spectrum can be simulated in approximately one working day on a 200 Mhz Pentium PC. It should be mentioned that classical dynamics simulations are not the normal choice for fitting initial trial structures to ICISS data, because of their high computational cost compared with BCA or other types of model calculations. Simulations of ICISS data and other kinds of angular scans are good candidates for running as batch processes (using *Snook*'s batch run option), as described in section 6.2.

The output from an ICISS simulation contains information about projectile scattering at all angles of emission (not just backscattering). Data for the scattering angles of interest is selected later using *Winnow*. This means that a single simulation can generate data for a variety of experimental measurements. Most of the ideas discussed in this chapter are also applicable to ion scattering spectroscopy (ISS) measurements as used in general surface analysis.[50]

In ICISS experiments, a projectile is incident parallel to a row of the target lattice. Scattered projectiles are detected in the same plane, at a scattering angle approaching 180°. Scattering in the plane requires that the projectile's collision partners be coplanar, so typically we consider only those collisions involving projectiles incident in a plane occupied by an atomic row.[51] (see fig. 6.2). If the model includes vibrational effects (recommended) these should be disabled for the *y*-direction (i.e. perpendicular to the scattering plane) using the appropriate simulation option

* For reasons of speed, Kalypso should not be used for ion scattering simulations unless its features are specifically needed, e.g. low-energy self-bombardment processes.

provided by *Snook* or *Kalypso* (see section 6.4). Since ICISS is quite surface sensitive, a target consisting of a small number of atomic layers (3 or 4) is sufficient.*

Large-angle scattering in ISS and ICISS is caused by the repulsion between the core regions of projectile and target atoms. This scattering can be described by a repulsive screened Coulomb potential, such as the ZBL potential. The following discussion assumes that the parameters of the projectile-target interaction potential (notably the screening length correction) are known. In practice, these may have to be determined heuristically.[46]

One method for selecting the screening length correction for an ICISS simulation is to identify a scattering process that involves 2 atoms (shadower and scatterer) which are both located in the surface plane, then use a binary collision program (such as `Cone`) to fit the experimental shadow cone data to a known structure (e.g. the unreconstructed clean surface). This binary collision method is based on the assumption that the projectile does not interact appreciably with other atoms during its approach (this assumption is valid if the critical angle is not too shallow).

When comparing the results of simulations with experimental data, it should be remembered that the latter involve measurements of ion yields incident in a fixed solid angle. The simulated data, however, involve scattering into a fixed angular range. The solid angle subtended by angular ranges $\Delta\varphi$ and $\Delta\phi$ is $\Delta\Omega = \Delta\varphi.\Delta\phi.\cos\varphi$. If the simulation is treated as a 2-dimensional scattering problem (as in this chapter), no correction to the scattered intensities is required. If a full 3-dimensional calculation is undertaken, scattered intensities need to be corrected (for comparison with experimental measurements at fixed solid angle of acceptance) by dividing them by $\cos\varphi$, where φ is the altitudinal emission angle (measured relative to the surface).

6.2. ICISS example project: 1.5 keV-Cu(110)

The input files you will need for the example ICISS project can be found in the `\examples\iciss` directory of your SK installation. The simulation runs in about 6 hours on a 600 MHz Pentium III machine. [This project is not included in the *Kalypso* package because *Kalypso* (for reasons of speed) is not recommended for this type of simulation.] Be sure to read the `readme.txt` file in that directory before you attempt to run the simulation (which uses a batch file, `cu110.bdf`). That file explains how to generate the Impact file, and how to set *Snook*'s simulation options correctly. This project simulates the scattering of 1.5 keV He from a Cu(110) surface at 37 different altitudinal angles of incidence (9°-76°). The geometry of the system is shown in Fig. 6.2.

The input files for each projectile incidence angle are identical, with the exception of the Run files. The various Run files differ only in the specification of the projectile altitudinal angle of incidence, so it is easy to generate Run files for each geometry from a common 'template' Run file simply by editing the altitudinal angle field (using *Spider*'s Run|Open command).

Fig. 6.3 shows the processed results of the ICISS simulation, namely a plot of the 180° backscattering yield as a function of the projectile incident altitudinal angle (see below for data processing details). The solid line in the figure represents the simulated data, while the plotted symbols represent the experimental data (arbitrarily scaled on the intensity axis) as measured by

* The target should only include the layers that participate in backscattering mechanisms. Contrary to what is often claimed, ISS is not a surface-specific technique.

Spitzl *et al*.[52] The backscattered projectile yield is below 200 counts (for a ±1° counting interval and 15,000 trajectories). A real project would probably include more trajectories to improve statistics.

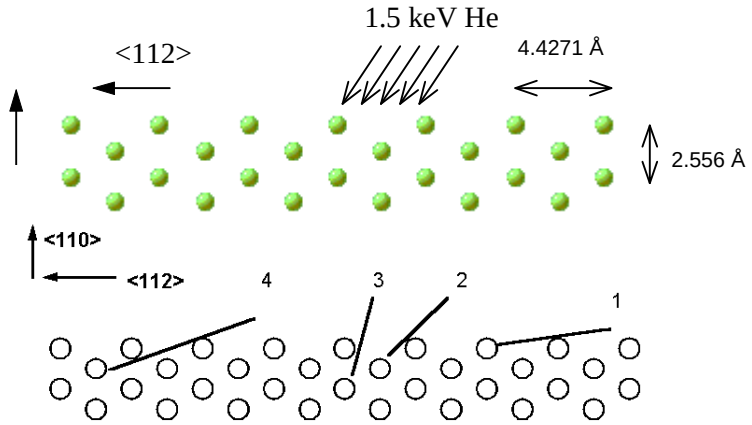


Figure 6.2. 26-atom Cu(110) lattice used for example project (projection in xz plane). Top: The arrows depict the range of incident projectile trajectories in the simulation. Bottom: possible backscattering mechanisms.

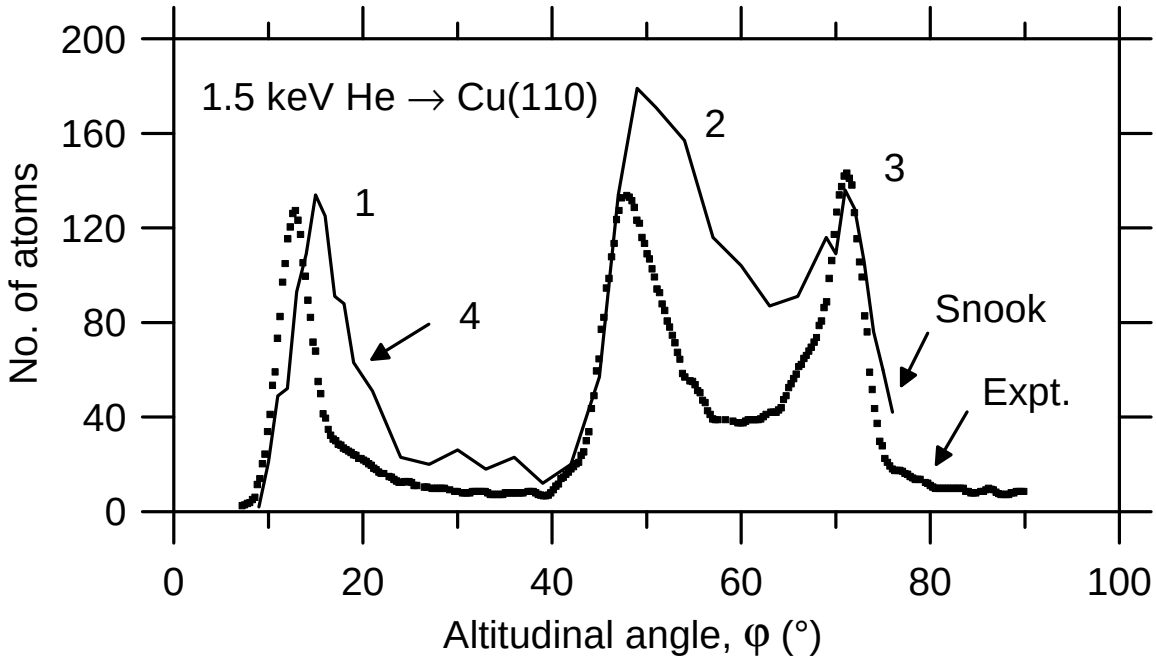


Figure 6.3. Calculated (solid line) and experimental [52] (symbols) direct impact backscattering intensity for 1.5 keV He projectiles incident on Cu(110) along a <112> azimuthal direction (see fig. 6.2). Yields are calculated using a ±1° acceptance range for the altitudinal angle (e.g., a point at φ = 60° refers to the fraction of particles backscattering between 59-60°. Each point is based on 15,000 runs (trajectories). Numbers on the peaks indicate the corresponding scattering mechanism in Fig. 6.2. An uncorrected ZBL screening length was used.

Fig. 6.3 illustrates the influence of 'critical angles' φ_c for 180° ('direct impact' or centre-to-centre) collisions, which are indicated by sharp onsets in scattering intensities. The widths of the peaks are determined by atomic vibrational amplitudes in the target. There are some differences between the simulated and the experimental data (about 2° error for peaks 1 and 2). However, Fig. 6.3 was produced without any adjustment in parameters, and as such, gives quite a good 'first-principles' account of the ICISS spectrum. The simulation neglected surface relaxation effects (about -0.10 Å for the surface layer of Cu(110)), but this would not affect the position of peak 1, since the associated scattering mechanism involves coplanar surface atoms.

The simulation used an uncorrected ZBL screening length, so a small reduction in this parameter might improve the fit. The optimum screening length correction for the He-Cu ZBL potential can be estimated to be about 0.9 using the `Cone` program which ships with the *Simulation Kit*. The critical angle computed by `Cone` for various correction factors is compared to the measured critical angle for coplanar scattering (12.5°), which is conventionally located at ~85% of the experimental peak height. Once a screening length correction has been established in this manner, a full MD simulation can be performed using the corrected screening length.[*]

The simulation model shown in Fig. 6.3 fails to reproduce the relative peak heights accurately. Better agreement can be obtained if a small relaxation is applied to the Cu(110) surface layer. But perfect agreement is not to be expected, since the simulation model takes no account of angular variations in incident ion neutralisation efficiency. The simulated width of the leftmost peak does not agree with experiment: this may be because (a) the vibrational displacement of surface Cu atoms is wrongly estimated, or (b) too much weight is given to the contribution of backscattering mechanism 4 (neutralisation effects should more effectively attenuate this peak, because it originates from sub-surface scattering). Looking at Fig. 6.3, one can suggest that it is probably more useful to compare simulated and experimental data on a peak by peak basis, rather than as full angular plots.

The data shown in Fig. 6.3 were produced by the following method:

1. In the Run file of the ICISS project, it was specified that only emitted projectile data should be written to the output file. (This restriction is not mandatory, as filtering can separate the projectile data later, but it saves unnecessary disk usage.)
2. After the simulation was completed, the resulting output file for each incident angle was filtered to select out the backscattered angular fraction of interest.[†] For example, at 15° incidence: `[altd > 14.0] & [altd < 16.0] & [vx > 0.0]`. The above filter selects particles scattered in the $+x$ direction at altitudinal angles (expressed in degrees) between 14.0-16.0° i.e. at an angle of 15±1°. (The symbol `altd` refers to the altitudinal angle expressed in degrees.) Since the output routine only stored projectile data in this case, there is no need to select projectile particles explicitly (i.e. by using the additional filter condition: `[rw = 0]`). This filter is appropriate for 180° backscattering events.

[*] To correct the screening length for the example project, just open up `\iciss\cu110.mdl` using Spider's Model|Open command. Then edit the screning length correction from 1.0 to 0.9 for the projectile-target screened Coulombic potential.

[†] A quicker, and recommended, approach is to use the `scat-cnt` utility program which can be found in the `examples\winnow` directory.

If the experimental data refer to scattering at some angle less than 180°, then the filter condition should be modified accordingly. For example, for φ =15°, and 140° scattering, it must be modified to: `[altd > 54.0] & [altd < 56.0] & [vx > 0.0]` (Make sure you use a filter `[vx < 0.0]` if the scattered projectiles move in the negative *x* direction, as they would in this example for φ > 50°).

Fig. 6.4 shows the result of repeating ICISS simulation with a ZBL screening length correction of 0.9. This gives a better prediction of the first critical angle, but this is not surprising since the correction factor was fitted (see above) in order to achieve this. The locations of peaks 2 and 3 do not change greatly after this correction,but the intensity of peak 2 (relative to peaks 1 and 3) is seen to increase.

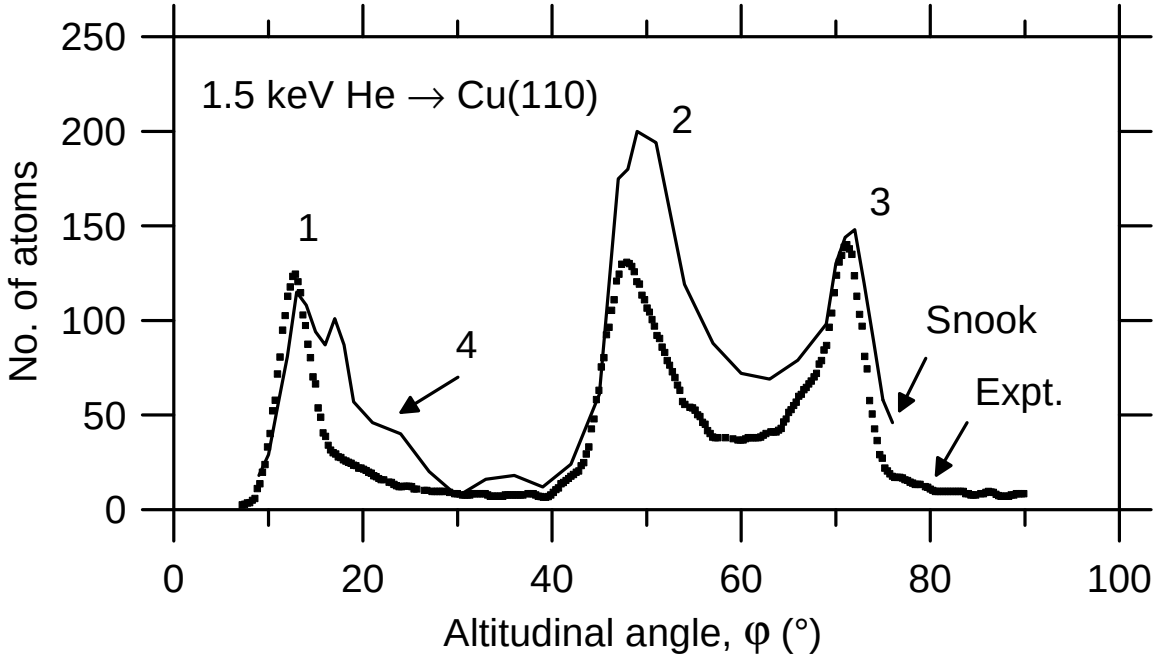


Figure 6.4. Calculated (solid line) and experimental (symbols) direct impact backscattering intensity for 1.5 keV He projectiles incident on Cu(110) along a <112> azimuthal direction (see fig. 6.2). The calculations are identical to those in Fig. 6.3 except that the ZBL screening length has been corrected by a factor 0.9.

Finally, Fig. 6.5 shows the results of another simulation like that of Fig. 6.4, except that in this case the target consisted of only 2 atomic layers (the first two layers of Fig. 6.2). This affects the intensity of peak 2, while peak 3 vanishes completely. The removal of peak 3 is consistent with the shadowing mechanism shown in Fig. 6.2. This example also shows that there are contributions to peak 2 which involve scattering from layers below the second. Fig. 6.2 is thus an oversimplification of the true scattering m

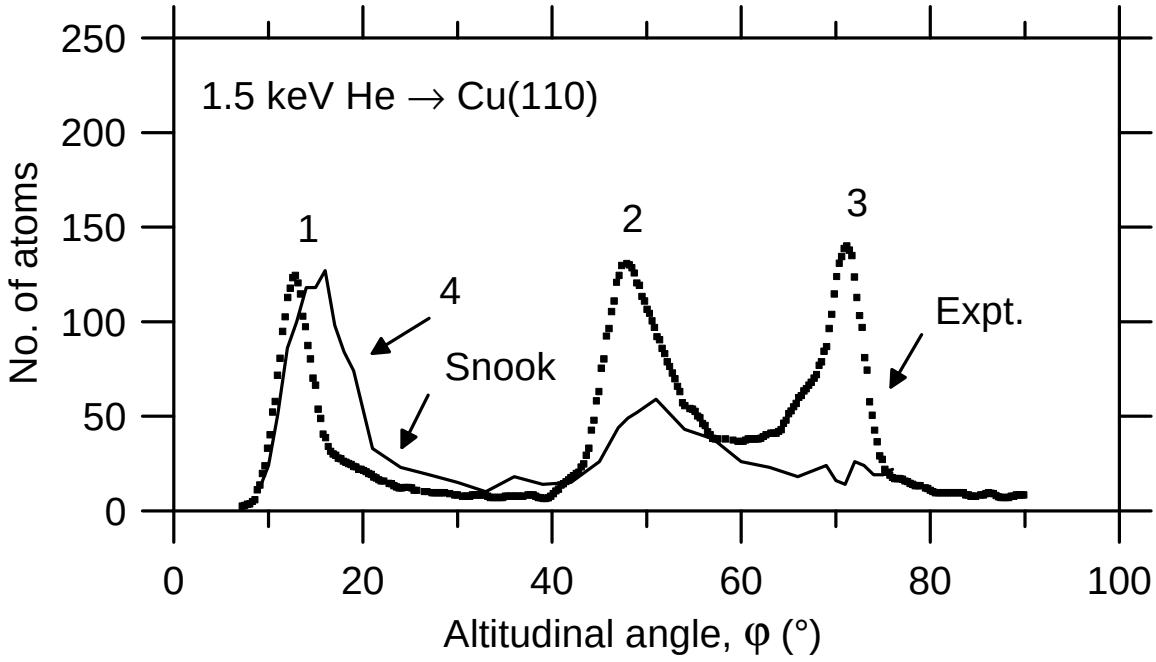


Figure 6.5. Calculated (solid line) and experimental (dashed line) direct impact backscattering intensity for 1.5 keV He projectiles incident on Cu(110) along a <112> azimuthal direction (see fig. 6.2). The calculations are identical to those in Fig. 6.4 except that the target consisted of only two atomic layers.

Fig. 6.6 shows an example of an energy spectrum generated from one of the output files (`51.snk`) from the example project (uncorrected ZBL screening length). The spectrum was obtained by the following steps:

1. Filter the output file using an expression: `[altd > 50.0] & [altd < 52.0] & [vx > 0.0]`. This isolates output records for projectiles which were backscattered within ±1° of the projectile incident trajectory at φ = 51°.
2. Generate a 100-channel spectrum for the expression `ke/ep` in the region 0-1500 [eV] using *Winnow*'s Spectrum command.

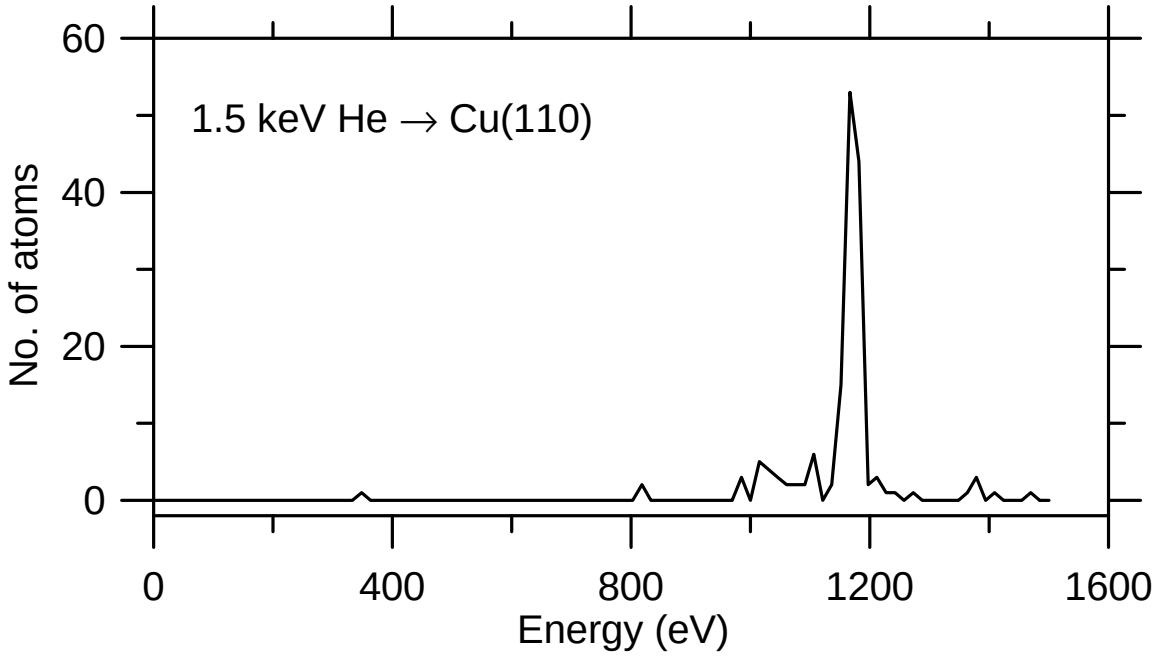


Fig. 6.6. 180° ISS energy spectrum simulated by *Snook* for 1.5 keV He projectiles incident on Cu(110) along a <112> azimuthal direction for φ = 51° (using uncorrected ZBL screening length).

The resulting spectrum has a sharp single-scattering peak situated at 1167±8 eV, as well as some low-intensity features due to multiple scattering which extend down to energies below 1000 eV. Theoretically, the 180° single-scattering peak is expected at 1165.5 eV (computed using *Winnow*'s Scattering Relations gadget). Plots such as Fig. 6.6 become more interesting if the target surface consists of more than one component, because of potential applications in quantitative analysis.

6.3. Angular conventions

There is some possibility for confusion in the angular conventions used by the *Simulation Kit*. The angular variables used by *Winnow* for filtering and other options are based on a self-consistent system (see *Winnow* on-line Help for details). The user-programmed option in the Run file dialog of *Spider* also follows the same system, because it refers to output data. However, the projectile incident altitudinal angle is defined in the Run file in a manner similar to what is used by experimentalists. (It defines the orientation of the position vectors that join the surface 'impact point' to the starting projectile position.) For this reason, the incident altitudinal angle is always entered in the Run file dialog as a *positive* number, although strict geometry (i.e. the convention used by *Winnow*) would indicate a negative value (since $v_z/\sqrt{(v_x^2 + v_y^2)} < 0$). Fig. 6.5 summarises the geometrical conventions.

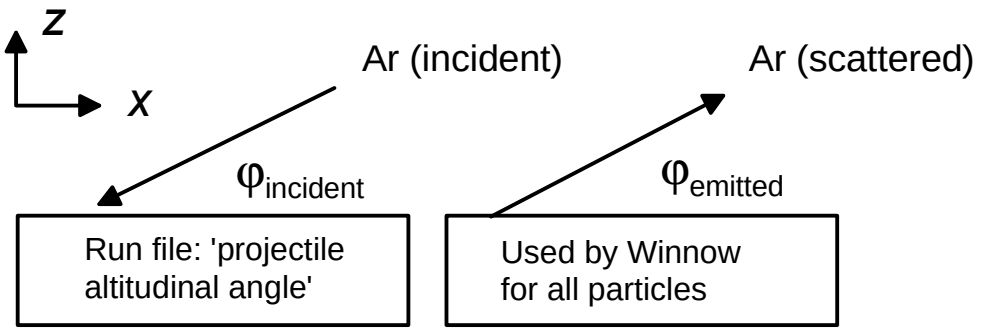


Fig. 6.5. Definitions of the altitudinal angle used for incident projectiles (left) and for emitted particles (right). For incident projectiles, φ is always positive. For emitted particles, φ is positive when the particle has a positive z-velocity (when it travels out of the target), and is negative when it has a negative z-velocity (when it travels into the target - a situation which is normally of little experimental interest).

6.4. Vibrational effects in ICISS

The ICISS example just described incorporates the effects of thermal vibrational displacements in the target lattice. Three-dimensional displacements increase the computational problem immensely, because of the now small probability of occurrence of a direct-impact event (which requires a scattering configuration in which both scattering atoms lie coaxial with the projectile path).

In order to work around this problem, *Snook* offers the option of suppressing thermal vibrational displacements in the *y*-direction, while applying them in the normal way along the *x*- and *z*-directions. This option was used by the example project in the production of Fig. 6.3.

This workaround (the suppression of *y*-vibrations) relies on the assumption that the projectile's azimuthal direction of incidence lies parallel to the *x*-axis. This is achieved by setting the projectile azimuthal angle parameter to 0.0 (its default value) in the Run file. Thermal displacements applied in this preferential manner maintain the coplanarity of lattice atoms in the projectile plane of incidence. To activate this option in *Snook*, it is only necessary to select the "No *y*-vibration" item which appears in the Simulation Options dialog box (on the Options menu).[*]

If an ICISS simulation is run without applying any vibrational displacements, the critical edges will appear as sharp spikes. Some users may prefer this approach for locating the critical angles, but it makes comparison with experimental peak intensities more difficult.

6.5. Concluding remarks

ICISS simulations are quite time-consuming, because it is necessary to repeat the simulation many times for different angles of incidence. This is feasible if the surface structure is known, but if the purpose of the simulations is to determine the structure then a BCA model is recommended. *Snook* is more useful for ISS simulations which involve a fixed angle of incidence (i.e. surface analysis using ISS).

[*] The "thermal vibrations" option must, of course, be enabled in the Model file used by the simulation. Otherwise the option discussed here has no effect.

ICISS simulations (and other simulations which involve many different projectile angles of incidence) also require a significant post-simulation processing effort. In the case of ICISS simulations, this processing may best be achieved by automating it with a custom-written computer program, rather than relying on the general (but slow) capabilities of *Winnow*. The `examples\Winnow` directory includes a simple program (`scat-cnt`) which can be used for batch extraction of the kind of data plotted in Fig. 6.3, or can be adapted for other purposes.

7. THERMAL VIBRATIONS

7.1. Introduction

The choice of atomic thermal displacements for atomic collision simulations is a non-trivial problem.[53] Accordingly, the basis of the computations made by *Snook* and *Spider* is outlined in some detail here. The discussion in this chapter, which is certainly not the only way to approach the problem, should allow users to decide for themselves on the value of the treatments used. Before discussing the equations, it is worth drawing attention to the following general points. The treatment of thermal vibrations used by *Kalypso* is identical to that used by *Snook*.

- Users can always override the mean square thermal vibrations calculated automatically by *Spider*, by simply editing the values in the resulting MDL file. (Refer to the File Formats topic in *Spider*'s online Help).
- Most experiments based on keV ion bombardment are not particularly sensitive to thermal vibration effects. Errors in the treatments of vibrational effects will only significantly influence simulations of physical processes if the root mean square (rms) vibrational amplitude error is significant in comparison with the scattering cross section (collision radius).
- Most published sputtering simulations have not included thermal vibration effects, but you will get unrealistic results if you do not include them in ICISS simulations.
- Take care not to confuse the rms vibrational amplitude $\sqrt{<r^2>}$, with the mean square thermal vibrational amplitude, $<r^2>$. Likewise, do not confuse the isotropic mean square amplitude $<r^2>$ with the unidirectional mean square amplitudes $<x^2>$, $<y^2>$, $<z^2>$.
- Lastly, note that experimental Debeye temperatures are uncertain to at least ±10%, and often more.

7.2. Theory

According to the Debeye theory, the isotropic mean square thermal vibration amplitude, $<r^2>$, of atoms in a monatomic solid is given by the following expression:

$$< r^2 > = 9(h/2p)^2 T/(Mk\Theta_D^2).[f(\Theta_D/T) + \Theta_D/4T] \quad (7.1)$$

where Θ_D is the Debeye temperature, k is Boltzmann's constant, T is the absolute temperature and M is the mass of atoms in the solid. $\phi(x)$ is the following function, where $x = \Theta_D/T$:

$$\phi(x) = (1/x) \cdot \int_0^x \frac{y dy}{(e^y - 1)} \quad (7.2)$$

Eq. (7.1) refers to the mean square amplitude of an isotropic oscillator. This is the quantity which *Snook* reads from the MDL file created by *Spider*. For vibrations in a specific direction (x, y or z), the right hand side of Eq. (7.1) must be divided by 3:

$$<x^2> = <y^2> = <z^2> = <r^2>/3 = 3\hbar^2 T/(Mk\,\Theta_D^2) \,.\, [\phi(\Theta_D/T) + \Theta_D/4T] \quad (7.3)$$

These unidirectional amplitudes are the quantities which are calculated and used by *Snook*.

If we substitute for the physical constants, and interpret M as the mass in atomic mass units (amu), then with the substitution ($x = \Theta_D/T$) the expression (7.3) simplifies to:

$$\langle x^2 \rangle = 145.5/M\Theta_D \,.\, [\phi(x)/x + 1/4]\ \text{Å}^2 \qquad (7.4)$$

using $\hbar = 1.05459\times10^{-34}$ J s; $k = 1.38066\times10^{-23}$ J K^{-1} ; 1 amu = $(1000N_A)^{-1} = 1.6606\times10^{-27}$ kg.

7.3. Comments on the Literature

There is some confusion in the equations published in the literature, in that Eq. 7.3 is sometimes associated with the isotropic vibrations (more properly described by Eq. 7.1). This presumably occurs because of typographical errors. Despite this, and other discrepancies, most authors seem to broadly agree on the constant pre-factor 145.5 in Eq. 7.4 (although physicists tend to calculate the pre-factor using nuclear rather than atomic masses, giving a value of 146.0). The reader is urged to be cautious when reading the literature on this subject (including this article!).

7.4. Implementation in *Snook* and *Kalypso*

Spider uses equation 7.4 to calculate $\langle x^2 \rangle$ for bulk and surface atoms, based on the values for T and Θ_D specified by the user (Model file), and the approximations to $\phi(x)$ discussed below. These unidirectional $\langle x^2 \rangle$ values are read by *Snook* and *Kalypso* from the Model file. Two sets of values are read by *Kalypso* (not *Snook*) if the target is a binary compound, corresponding to different atomic types.[*]

The displacements which are added to each lattice atom x, y and z coordinate are drawn (using the Box-Müller method) from a random distribution having Gaussian deviates characterised by a variance $\langle x^2 \rangle$. Bulk, surface parallel and surface perpendicular displacements respectively can be drawn from distributions with different standard deviations. However, a requirement imposed on the displacements calculated by *Snook*/*Kalypso* is that they shall not exceed 2.5 standard deviations (to prevent highly improbable events). For purposes of the vibrational correction, a lattice atom is classified as a 'surface atom' if its Target file z-coordinate, z[n], places it above the anchor atom (i.e. z[1]) or no more than 1 Å below it.

The method for calculation of the Debeye function $\phi(x)$ in equation 7.4 is now explained.

$\phi(x)$ can be calculated by numeric integration of Eq. 7.2. Eckstein[2] provides the following approximation:

$$\phi(x) = (1/x)\cdot\int_0^x \frac{y\,dy}{(e^y - 1)} = 1 - x/4 + \frac{x^2}{36} - \frac{x^4}{3600} + \ldots \qquad (7.5)$$

The same approximation is used by *Snook* for x in the range 0 to 3.0, i.e. in the range $T = 0.33\Theta_D$ to "infinity". Recall that $x = \Theta_D/T$. For $T = 0.33\Theta$D, $\phi(x) = 0.48$.For "infinite" T, $\phi(x) = 1.0$. As $T \rightarrow 0$, so too $\phi(x) \rightarrow 0$.

[*] The parameters of the second type, if different from the first, have to be entered by hand into the Model file, using a text editor. You should make the file read-only to avoid accidental modification later.

For values of $x > 3.0$ the following (low-temperature) approximation is used:

$$\phi(x) = [\pi^2/6 - (x + 1).\exp(-x) - (x/2 + 1/4).\exp(-2x) - ...]/x \quad (7.6)$$

For $x >> 1$, $\phi(x)/x$ becomes small in comparison to the ¼ term in Eq.7.4, and the lattice vibrations approach their zero-point levels.

It is worth pointing out that these thermal displacements are computed without reference to the potential function (Morse) used for the target-target interactions in your lattice. This means that the thermodynamic temperatures associated with the atomic displacements in the target lattice (which depend on the potential parameters) will not correspond to the temperature which you specified in the model file. The true lattice temperature is related to the mean difference in potential energies, $<V_1 - V_0>$, of the lattice with (V_1), and without (V_0), thermal displacement effects applied. For a system of oscillators interacting via quadratic (harmonic) terms, theory suggests a value of $<V_1-V_0> = 3/2$kT (which we shall use as an approximation for other potentials too; this result comes from either the Equipartition principle, or the Virial theorem, depending on your point of view).

The discrepancy between theory and the values returned by *Snook* depends on the potential function used by *Snook* or *Kalypso*. This discrepancy has to be investigated by explicit temperature monitoring (which is available in these programs). Further discussion of these matters would take us too far afield,[54] but the conclusion must be that the choice of an appropriate thermal displacement depends to some extent on the purpose of the simulation. For a study of ion scattering or channelling, the primary focus of interest is the displacement itself. However, for a study of lattice dynamics (melting, film growth, diffusion etc.), the potential energy associated with (or implied by) the displacements should be the primary consideration. Sputtering - one of the main applications of such simulations - falls in the area between these extreme cases.

7.5 Lattice atom velocities

The simulation options dialog box in *Snook* allows the user to initialise the velocities of the target lattice atoms. The purpose of the option is to allow the user to set up the target lattice as a dynamical system with a specific temperature (see, however, the preceding section for qualifications to this remark).

The velocities are randomly applied, according to a Maxwellian distribution. No attempt is made to ensure that $<v_x>$, $<v_y>$ and $<v_z>$ are precisely zero, although in practice this very nearly happens in large lattices, as expected. According to the Equipartition Principle (from statistical mechanics), the mean translational energy, <KE>, of an atom in the classical limit is 3/2 kT (k is the Bolzmann constant). This can be broken down into contributions of ½kT for each direction of motion.

The velocity distribution is deduced from the following assumptions:
(a) atomic energies are distributed according to a Boltzmann distribution: i.e.

$$P(E) = \exp-((\mathrm{KE} + V)/\mathrm{kT}) = \exp(-\mathrm{KE}/\mathrm{kT}).\exp(-V/\mathrm{kT}); \quad (7.7)$$

(b) atomic positions and momenta are distributed with independent probability (P) distributions: i.e.

$$P(\mathrm{E}) = P(\mathrm{V}).P(\mathrm{KE}). \tag{7.8}$$

8. INELASTIC ENERGY LOSSES

8.1. Introduction

Up to this point the discussion has considered only the effects of elastic scattering. Energetic particles can also lose energy through excitation of valence or core electrons, or lattice vibrations. These energy losses are termed inelastic losses.[1,2,55] The term electronic stopping is also used to refer to electronic excitations, which are the predominant loss mechanism at keV particle energies and above. The reader should be aware at the outset that the main pitfall in the literature of electronic stopping is the lack of any standard system of units. The equations in this chapter should be used with SI units.

The *Simulation Kit* and *Kalypso* permit the modelling of inelastic effects via a combination of three distinct electronic stopping models. These models are: (a) the Lindhard-Schiott-Scharff (LSS) model; (b) the Oen-Robinson (OR) model; (c) the Shapiro-Tombrello (ST) model. Temperature clamping (both heating and cooling) can also be applied to the sample, and is treated as an additional inelastic loss mechanism.

These energy transfer models can be included singly, not at all, or in any weighted combination according to the preference of the user. The simulation of inelastic effects at keV energies usually has to be done on an *ad hoc* basis because of the general theoretical uncertainty surrounding the mechanisms of inelastic loss. For example, an equipartition (0.5:0.5 weighting) of LSS and OR losses is often used.

The parameters for each of the inelastic models are specified in the Inelastic (INL) file using *Spider*. The INL file is an optional part of a simulation project, in the sense that you only have to create it if you plan to include inelastic effects in your simulation. The actual inelastic model(s) to be used at run-time must also be selected in *Snook*'s Options box, otherwise the INL file is ignored. Examples of inelastic files (for Ar/Ca and Ar/Ni-Cu targets) can be found in the `\examples\inelast` directory.

The LSS model requires the specification of a set of parameters for every kind of atom (i.e. of distinct atomic number, Z_1) for which inelastic losses are to be tracked. For instance, 2 sets of parameters are required for the Ar ® Cu(100) system (for Ar and Cu particles respectively). The OR and ST models require a set of parameters for each pair of collision partners. For example, 2 sets of parameters are required for the Ar ® Cu(100) system, representing the pairs Ar-Cu and Cu-Cu respectively. For all models, incomplete sets of parameters are allowed, but this will lead to neglect of the corresponding energy loss channel (for example, if you forget to include parameters for the Cu-Cu interaction). The *Spider* online Help topics explain how you should enter the parameters for these various inelastic loss models.

In implementing these models, the author has been forced to make decisions with which the user of a program may not necessarily agree. For example, in *Snook*, any inelastic loss corrections for a given timestep are applied after the update of the elastic corrections (positions and velocities) for that timestep. The distinction may seem trivial, but it does have observable consequences.

8.2. Lindhard-Schiott-Scharff (LSS) model

The LSS model asserts that the electronic energy loss (*dE*) associated with the movement *dx* of a particle (atomic number Z_1) in a medium (atomic number Z_2) is proportional to the particle velocity (*v*): [2,56]

$$-\frac{dE}{dx} = 8p\,N\left(\frac{e^2}{4pe_0}\right).a_B\,\frac{Z_1^{7/6}Z_2}{(Z_1^{2/3}+Z_2^{2/3})^{3/2}}\,v/v_B \tag{8.1}$$

where *N* is the atomic density of the medium (atoms m $^{-3}$), a_B is the Bohr radius, and v_B is the Bohr velocity (= *c*/137). Equation 8.1 simplifies to:

$$-\frac{dE}{dx} = K(LSS).v \tag{8.2}$$

where *K*(LSS) is a constant. The LSS model thus views the target as a viscous medium, and describes a continuous electronic energy loss process that arises from passage through this medium.

Spider allows the definition of *K*(LSS) given in Eqs. 8.1 and 8.2. to be scaled by an arbitrary factor (e.g. a factor of 0.5 is appropriate if you plane to use an equipartition of inelastic losses between the LSS and OR models). *dE* is calculated at each timestep, once an atom's displacement, *dx*, has been computed for that timestep.

For a mixed target material (e.g., a Cu-Ni alloy), the correct choice of the Z_2 parameter in equation 8.1 requires some careful thought, which will be guided by the user's physical intuition about the correct 'effective' atomic number of the target.

It should be noted that the application of inelastic corrections according to the LSS model does not conserve linear momentum. (This remark does not apply to the OR and ST models.)

8.3. Oen-Robinson (OR) Model

The OR model estimates the energy loss (ΔE) arising from a single isolated binary atomic collision in which the distance of closest approach (apsidal distance) is R_0:[2,57]

$$-\Delta E = 8p\left(\frac{e^2}{4pe_0}\right).a_0\,\frac{Z_1^{7/6}Z_2}{(Z_1^{2/3}+Z_2^{2/3})^{3/2}}\,v/v_B\,\frac{(c/a)^2}{2p}\,e^{-(c/a)R_0} \tag{8.3}$$

Here *c* is a constant term (normally *c* = 0.3, but this can be specified arbitrarily in *Spider*), and *a* is the uncorrected Moliere-Lindhard screening length. Z_1 is the atomic number of the 'projectile' species, while Z_2 is the atomic number of the 'target' atom species. (The roles of the two atoms involved in the collision can be symmetrized in *Spider*, if the user so desires.) As implemented by *Snook* and *Kalypso*, the term *v* in Eq. 8.3 represents the relative velocity ($|\mathbf{v}_1 - \mathbf{v}_2|$) of the collision partners at 'infinite separation', which is computed as follows:

$$v = \sqrt{\frac{1}{2}\mu.v(t)^2 + V(r(t))} \tag{8.4}$$

where $v(t)$ and $V(r(t))$ respectively represent the magnitude of the relative velocity at some time t, and the potential energy at the same time (the t used by *Snook*/*Kalypso* corresponds to the apsidal point); μ is the reduced mass of the system.

The OR model postulates the occurence of one (large) energy loss event per close encounter, in contrast to the LSS model which involves (small) energy losses on each timestep for each atom. The connection between Eqs. 8.1 and 8.3 is brought out if the latter is written in the form:

$$-\Delta E_{OR} = 8p\left(\frac{e^2}{4pe_0}\right)a_B \frac{Z_1^{7/6} Z_2}{(Z_1^{2/3} + Z_2^{2/3})^{3/2}}.F_{OR}.v / v_B \tag{8.5}$$

$$-\Delta E_{OR} = K(OR).F_{OR}.v \tag{8.6}$$

where F_{OR} is the 'Oen-Robinson factor' used by *Spider* and K(OR) is a constant for a given collision pair. Note the absence of any atomic density (N) term in equations 8.3 and 8.5. K(OR) has very different physical dimensions from K(LSS) defined in the preceding section. A scale factor may optionally be specified in *Spider* to adjust the magnitude of the energy loss described by Eqs. 8.3, 8.5 and 8.6.

In contrast to the LSS model, the OR model has the character of a discrete energy loss, because it associates a single loss event with each close encounter. One problem with implementing the OR model is in finding a technique to deal with collision events that cannot be approximated as binary encounters. *Snook* circumvents this problem, perhaps inelegantly, by requiring the user to impose (heuristically) an upper limit R_{MAX} on R_0 (the distance of closest approach). Thus, the OR energy loss is not computed unless the colliding atoms approach within a distance R_{MAX}. Typically, R_{MAX} would be on the order of 1-2 Å. (The author normally sets R_{MAX} to about half the lattice nearest-neighbour distance.)

The way in which the computed OR energy loss (ΔE) is applied within the simulation routine is similar to the method used for the ST model described in the next section.

8.4. Shapiro-Tombrello (ST) model

The ST model attempts to incorporate collision-induced core electron promotion effects into a classical dynamical scattering model. The main drawback of the model is the uncertainty surrounding the correct choice of parameters (p, R_C, ΔE: see below). A brief summary of the model as implemented in *Snook* will now be given. For a deeper review of the physics involved, the reader is referred to the original literature.[33]

The idea underlying the ST model is that an inelastic (inner-shell electron promotion) transition can occur if a colliding pair of atoms approaches closer than some critical distance (R_C). This energy loss may involve up to N_{MAX} electrons from the inner shell. For each electron promoted, an amount of inelastic energy ΔE is lost (the maximum loss possible is thus $\Delta E^* N_{MAX}$). The

number of electrons considered for promotion (N) depends on the relative radial kinetic energy ($K_R = \frac{1}{2}\mu v_R^2$) available to the collision pair at the moment when R_C is passed; i.e., N must be consistent with the condition: $K_R \geq N\Delta E$. Finally, for each of the N electrons, a probability factor (p) is compared with a random number to determine whether or not that electron is actually promoted (for instance, if $p = 0.5$, only ~N/2 electrons will be promoted on average).

From the foregoing, we see that the total energy loss (ΔE_T) computed for a particular collision configuration satisfies the conditions:

$$\begin{aligned} &\Delta E_T \leq N\Delta E \\ &\Delta E_T \subset \{\Delta E, 2\Delta E ... N\Delta E\} \end{aligned} \tag{8.7}$$

(the equality applies if p = 1, in which case $\Delta E_T = N\Delta E$), while the average energy loss over many such collision configurations is:

$$\langle \Delta E_T \rangle = Np\Delta E \tag{8.8}.$$

The inelastic energy loss corrections for both the ST and OR models are applied at the apsis of the collision.[*] At the apsis, the radial kinetic energy is zero. The energy loss correction (ΔE_T) is applied by reducing the potential energy of the interacting atoms: this entails translating them instantaneously along the line joining their centres by a distance Δr, which changes the potential energy by an amount ΔV, such that $\Delta V = \Delta E_T$. The translation distance Δr was originally estimated by ST via the relation: $\Delta r = \Delta V/F(r)$, where $F(r)$ is the force at the apsidal point. This equation would be exact if the potential declined linearly with separation (r).

Snook and *Kalypso* use a similar procedure, but apply it twice (both at the apsidal point, and at the first estimated displacement). In addition, the first application of the formula uses a heuristic correction factor of 1.2 to compensate for the rapid decline of the force as r is increased. The procedure used by *Snook* and *Kalypso* typically computes the correct displacement Δr to within ~10^{-4} Å or better, but such is the nature of the potential that errors of this magnitude do lead to errors at the ~1 eV level in the energy book-keeping routines. For this reason, it is important that you initially test the parameters of your simulation (in particular, the timestep) with the ST and OR loss effects disabled. This will give a true estimate of the integration error. Subsequent errors in energy conservation can then be attributed to inelastic loss book-keeping errors. The ST and OR energy loss algorithms conserve linear momentum, but not angular momentum.

8.5. Temperature clamping

Snook and *Kalypso* allow the target to be cooled or heated to a specified temperature (T). This capability may be useful for certain kinds of simulations, especially those involving low-energy bombardment. The user is required to specify the target temperature (T_0), the cooling or heating period (τ), and the time interval (start and end) of the period during which the temperature corrections are to be applied. The equations of motion of each particle affected by the temperature correction is written as:[1]

[*] In practice, at the first timestep after the apsis. It could be argued that the ST correction should be applied at the moment that R_c is crossed, rather than at the apsis. However, in practice this procedure would make very little difference to the collision dynamics (and no difference at all for direct impact collisions).

$$m\frac{d^2\mathbf{r}}{dt^2} = F(\mathbf{r}) - \frac{m}{2\tau}(1 - T_0 / T)\frac{d\mathbf{r}}{dt} \tag{8.9}$$

where m is the mass of the particle, and $\mathbf{F}(r)$ is the usual force due to the potential. For $T > T_0$, kinetic energy is removed from the system, and vice versa. The instantaneous temperature is computed from the mean particle kinetic energies (<KE> = 3/2kT).

If the instantaneous lattice temperature is absolute zero (T = 0), Eq. (8.9) cannot be applied, and the correction (and current temperature report) is skipped for that timestep. (This situation may arise, for example, during the first iteration of a simulation based on a static lattice.) You will therefore never see the current temperature reported as 0 K.

If you want to apply cooling or heating effects only to a subset of the atoms in the target (e.g. a base layer with constant temperature that acts as a thermal sink), you can set the `ofDontCool` option flag of the particles which are to be ignored (a) when calculating the current temperature, and (b) when applying the temperature correction, as in Eq. 8.9. For example, if you want to ignore the energy of the projectile when calculating/correcting temperature, just set the `ofDontCool` flag for the projectile (in the Projectile file). Regardless of their option flag settings, only particles in the lattice region are considered when evaluating the temperature and applying the correction. Other particles (those that have left the target) are ignored.

Temperature clamping processes do not conserve energy or momentum. Also, the temperature clamping algorithm will not perform well if a large amount of energy is leaving the system through sputtering,. Likewise, the algorithm does not perform well if hard energetic collisions take place, since these tend to absorb and release kinetic energy. In cases such as these, you can expect temperature overshooting and oscillations. However, for an isolated system near equilibrium, the temperature fluctuation is on the order of 1 K.

9. ADVANCED TOPIC: FLAGS

9.1. Introduction

Snook and *Kalypso* both use an array of 32-bit integers ('options flags') to store information relating to individual particles (32 bits per particle). This technique will be familiar to anyone with some programming experience. Each array can hold up to 32 bitmapped parameters (but less than 10 are currently used). That is, the specific bits of the array element may each contain information about different options in use for the corresponding particle. The individual bits, more usually known as (options) flags (and designated by names of the form `ofXXX`), can either be set (equal to 1) or cleared (equal to 0). The flags defined in version 2.4 of the *Simulation Kit*, and version 1.0 of *Kalypso* and their values are listed in Tables 9.1 and 9.2. Note that (a) not all flags are used by both programs; (b) flag values are not the same for both programs.

Table 9.1. Options flags (`ofXXXX`) used by ***Simulation Kit* 2.4** and their values.

Flag name	Decimal value	Binary value	Applies to
`ofEmitted`	1	00000001	All particles
`ofNoForce`	2	00000010	Target particles
`ofRepulsive`	4	00000100	Target particles
`ofNotContained`	8	00001000	Target particles
`ofDontCool`	16	00010000	All particles

Table 9.2. Options flags (`ofXXXX`) used by ***Kalypso* 1.0** and their values.

Flag name	Decimal value	Binary value	Applies to
`ofTypeBAtom`	1	00000001	All particles
`ofEmitted`	2	00000010	Target particles
`ofNotContained`	4	00000100	Target particles
`ofDontCool`	8	00001000	All particles
`ofRecorded`	16	00010000	All particles

The options flags are read in as integers from the Target and Projectile files of a simulation project (as the last numeric column). The default value for all flags is zero. Most users of the *Simulation Kit* will have no reason to change this default behaviour, and can ignore this chapter. However, *Kalypso* simulations involving binary compound targets require the setting of the `ofTypeBAtom` flag for some atoms in the target, but this is quite straightforward since it has the value 1. The effects of the various flags are explained in the following sections. Note from the Tables that not all flags affect the projectile.

If you wish to incorporate one or more of the associated options into your simulation, you simply need to set the corresponding bit in the Target file and/or Projectile file of your simulation project. To set individual bits, you must total up the respective numbers in the Value column of

Tables 9.1 or 9.2. Thus (Table 9.1) an options flag of 3 means that the `ofEmitted` and `ofNoForce` bits have both been set (1+2 = 3).

9.2. `ofEmitted` flag

This flag is set by *Snook* or *Kalypso* after (a) a projectile or target particle has been ejected from the surface region of the target, and (b) a surface binding energy correction has been applied. The effect of setting this flag manually is to override (ignore) the surface binding energy correction. However, if the surface binding energy is zero (0.0), the flag has no effect.

9.3. `ofNoForce` flag

This flag is only used by *Snook*. It suppresses the interaction between 2 target atoms in the Morse potential region. That is, if the flag is set for either of the colliding atoms, the interaction between them will be ignored in this region. This flag only has an effect if the target is complex (consists of more than one kind of atom). The flag has no effect in the spline or screened Coulombic region of a composite potential (it would not make sense physically to neglect these stronger interactions). The `ofNoForce` flag may be usefully set for an atom that interacts weakly or repulsively with its neighbours, such that the Morse potential would be a poor representation of the potential (for example, Xe atoms in a system consisting of a Xe monolayer adsorbed on a Cu surface).

9.4. `ofRepulsive` flag

This flag is only used by *Snook*. It affects the interaction between 2 target atoms in the Morse potential region only. It replaces these interactions with a screened Coulombic interaction, but only when *both* interacting atoms have the ofRepulsive flag set. This flag takes precedence over the `ofNoForce` flag. This flag only has an effect if the target is complex (consists of more than one kind of atom).

Consider the following flag settings for the collision system Ar ® Cl/Cu(100):

Atom	ofNoForce	ofRepulsive
Cu	Clear	Clear
Cl	Set	Set

Based on the above flag settings, the atoms in the system will interact at short range (i.e. the Morse region defined in the Model file) according to the following scheme:

Atom pair	Type of interaction
Cu-Cu	Morse
Cl-Cu	None
Cl-Cl	Screened Coulomb

The benefit of this approach is that it results in a stable target configuration (apart from the Cl-Cl interactions, which are presumably weak anyway because of the large Cl-Cl distance). The other approach, of treating the Cu-Cl and Cl-Cl interactions with the same Morse potential as the Cu-Cu interaction has obvious difficulties, although it must be admitted that these are irrelevant for certain kinds of simulations (e.g. projectile ranges, ISS and other fast processes). If, in addition, the `ofEmittedFlag` was set for the Cu atoms but not Cl atoms, then Cl (but not Cu) would effectively be bound to the surface by any attractive surface planar potential chosen by the user. Thus, by clever use of the flags in combination with the surface binding energy option it is possible to bind Cl to the surface without using a Morse potential. (Note, however, that this type of simulation can be more naturally carried out with *Kalypso*.)

9.5. `ofNotContained` flag

This flag is set by *Snook* and *Kalypso* after (a) a projectile or target particle has been ejected from any of the 4 sides of the target lattice, and (b) a bulk binding energy correction has been applied. The effect of setting this flag manually is to override (ignore) the bulk binding energy correction. However, if the latter is zero (0.0), the flag has no effect.

9.6. `ofTypeBAtom` flag

This flag is used only by *Kalypso*. It is used to identify an atom as type A (default, flag not set) or type B (flag set). To carry out a simulation for a binary compound target (AB), you *must* set this flag yourself (in a text editor, or using the Edit Flags option in *Spider*) for type B atoms. Otherwise, they will not be recognised as such, and the simulation will return erroneous results.

9.7. `ofDontCool` flag

This flag is used by both *Snook* and *Kalypso*. Atoms which have this flag set are completely ignored for purposes of calculating and correcting the target temperature. The flag has no effect unless the temperature clamping option is enabled in *Snook* or *Kalypso*.

9.8. `ofRecorded` flag

This flag is used only by *Kalypso*. It is set after the coordinates of a particle have been written to a disk output file. The flag is only used by *Kalypso* if the Run file option 'Limit to 1 record' has been selected in conjunction with the periodic sampling mode. (The flag has no effect on records written to disk after termination.) The option is used when you want to write at most one record for a given particle, e.g. to record the precise moment when it left the surface. If you set the flag manually, it will prevent any output being written to disk for that particle during the periodic sampling process - but it will have no effect on data written after termination (if this option is selected).

10. ATTRACTIVE INTERATOMIC POTENTIALS

10.1. Introduction

Most attractive interatomic potentials used by the molecular dynamics community were fitted on the assumption of a cut-off distance greater than what is desirable for simulations of sputtering and projectile scattering.[*] Typically, one aims for a cut-off beyond the 1st, 2nd or 3rd nearest neighbour ("1NN, 2NN or 3NN"), in order to reduce computation time.[†] If such potentials are not available, you will have to fit the potential yourself. However, this chapter suggests Morse, Sutton-Chen and Tight-Binding potentials for most of the metals that you will encounter. A very extensive bibliography of interatomic potentials can be found on the Web (March 2001) at `http://dfw.jst.go.jp/pdb/pdb3.html`.

There is no definitive reference on fitting interatomic potentials, but many such papers for particular metals have appeared in the past decade in journals such as *Philosophical Magazine (A*, *B* and *Letters)*, *Modelling and Simulation in Materials Science*, and *Physical Review B*. There are nice reviews of various types of many-body potentials for (mostly fcc) metals in Daw *et al*.,[58] Carlsson[59], and Ercolessi *et al*.[60] The issues involved in fitting many-body potentials to hcp metals (difficult for long-range potentials) are reviewed in refs. [24,61,62]. I will fit some TB potentials for hexagonal metals in due course. *Semiconductors require a different many-body formalism from that used for metals, which is not currently supported by Kalypso.*

To fit a many-body potential, you need to input (at a minimum) the lattice constant, cohesive energy and 3 elastic constants of the target lattice. For a Morse potential, one normally uses the lattice constant, cohesive energy and bulk modulus. A good many-body potential should also be fitted using the vacancy formation energy. Surface energies are sometimes used, but the accuracy of the experimental values is questionable. It is not possible to fit any type of potential definitively, since the outcome depends on the number, values and weights of properties used in the fit. The recommended values of physical properties (such as elastic moduli) change over time. Moreover, optimisation techniques are rarely guaranteed to find the absolute (global) minimum in the parameter space used for the fitting procedure.

Potential truncation is not advisable for long-range power-law potentials like the Sutton-Chen potential, but you may be able to get away with it if you are using the TB potential, which falls off exponentially (see 10.4). There are two problems with potential truncation: (a) the cohesive energy of the lattice is underestimated; (b) the lattice becomes unstable, and will tend to relax during the course of the simulation.

It is *always* a good idea to check the elementary properties of potential parameters (e.g. predicted cohesive energy), wherever possible. This is possible for a Morse potential in the *Simulation Kit* via *Spider*'s Target|Bulk Properties command. For the *Kalypso* package, a similar option exists within *Kalypso* itself (see the Help).

---

[*] One reason for this is that melting and phonon behaviour is difficult to model with a short-range potential.

[†] An atom in a fcc lattice has the following neighbours: 12 at 1NN, 6 at 2NN, 24 at 3NN, 12 at 4NN, 24 at 5NN, 8 at 6NN and 48 at 7NN.

10.2. Morse potentials

10.2.1. Sources

Eckstein summarises Morse potentials from various sources for a variety of metals,[2] which are listed below in Table 10.1. The parameters in Table 10.1 (a) should not be used for simulation work, since they require a very large potential cut-off distance. Table 10.2 lists parameters fitted for various cut-off distances by the author for a random selection of materials, with their associated properties. These potentials were fitted to reproduce the lattice constants, cohesive energies and bulk moduli exactly. The small discrepancies between the parameters in Table 10.1 and 10.2 probably arise from the use of different values of the cohesive energy, lattice constant or bulk modulus in the fitting process. Other fitting strategies are possible. In particular, one normally chooses to fit a larger number of properties inexactly, in preference to fitting a small number of properties exactly (as here), because this improves the *transferability* of the potential to coordination environments other than the bulk reference environment. (The potential for Ca in Table 10.2 was fitted in this way.)

The Morse potential provides an approximate description of solid-state properties. its main advantage is speed, not accuracy. Thus, it is suggested that the Morse potential be used with a 1NN or 2NN cut-off. The 1NN cut-off results in fast simulations, and has the added advantage that 1NN Morse potentials can be easily fitted for any metal (this topic is discussed in the following sections). Morse potentials cut off above 1NN also do not produce surface relaxation, which may be considered an advantage for most applications.

The following sections show you how to fit Morse potentials with 1NN and 2NN ranges.

Table 10.1. Morse potential parameters for metals. (a) L.A. Girafalco, V.G. Weizer,Phys.Rev. 114 (1959) 687; 87; (b) D.E. Harrison Jr, CRC Crit. Rev. Solid State Sci. 14 suppl. 1 (1988) 1; (c) A. Anderman of Atomics International (cited in: W. Eckstein, 'Computer Simulation of Ion-Surface Interactions,' Springer-Verlag (1991)).

	D (eV)	α	r_0 (Å^{-1})	
(a) All infinite range				
Na	0.06334	0.58993	5.336	
Al	0.2703	1.1646	3.253	
K	0.05424	0.49767	6.369	
Ca	0.1623	0.80535	4.569	
Cr	0.4414	1.5721	2.754	
Fe	0.4174	1.3885	2.845	
Ni	0.4205	1.4199	2.780	
Cu	0.3429	1.3588	2.866	
Rb	0.04644	0.42981	7.207	
Sr	0.1513	0.73776	4.988	
Mo	0.8032	1.5079	2.976	
Ag	0.3323	1.3690	3.115	
Cs	0.04485	0.41569	7.557	
Ba	0.1416	0.65698	5.373	
W	0.9906	1.4116	3.032	
Pb	0.2348	1.1836	3.733	
(b) All cut off above NN2				
Cu	0.481	1.405	2.628	
Mo	0.997	1.500	2.800	
Rh	0.7595	1.560	2.750	
W	1.335	1.200	2.894	
Au	0.560	1.637	2.922	
(c)				
Al	0.46371	1.17984	2.94950	NN2
Al	0.34951	1.14008	3.14172	NN3
Cu	0.48056	1.40465	2.62768	NN2
Cu	0.37598	1.36752	2.76953	NN3

Table 10.2. Morse potential parameters for various metals and Si. Except for Ca, all potentials are fitted *exactly* to the indicated lattice constants (a), cohesive energies (E_c) and bulk moduli (B) using cut-off distances as specified (for example, 2N means cut-off beyond 2nd neighbour distance). Fitted by the author.

Element	a (Å)	Cut-off	E_c (eV)	B (GPa)	D (eV)	α(Å^{-1})	r_0 (Å)
Al	4.0495	2N	3.39	88	0.4624	1.187	2.963
Al	4.0495	3N	3.39	88	0.3509	1.148	3.150
Cu	3.6147	2N	3.49	142	0.4821	1.406	2.636
Fe	2.8664	2N	4.28	168	0.6497	1.395	2.590
Si	5.4305	3N	4.63	99	1.017	1.319	2.803
Si	5.4305	1N	4.63	99	2.315	1.475	2.351
Ni	3.524	2N	4.44	188	0.6107	1.4145	2.573
Pd	3.8907	2N	3.89	195	0.5652	1.633	2.805
Ca*	5.583	3N	1.84	15	0.2012	0.8974	4.280
Ag	4.0857	2N	2.95	109	0.4208	1.432	2.957
Pb	4.9502	2N	2.04	41	0.2899	1.162	3.586
Rh	3.8044	2N	5.75	269	0.8223	1.556	2.752
W	3.165	2N	8.90	323	1.389	1.413	2.852
Mo	3.1469	2N	6.82	272	1.054	1.478	2.833
Pt	3.9239	2N	5.84	288	0.8495	1.627	2.828

* The potential for Ca was fitted to a wide range of physical properties. The bulk modulus predicted by the potential is 21 GPa.

10.2.2. Analytic formulae for Morse potentials

10.2.2.1. First neighbour cut-off

For a cut-off at the *first* neighbour, the Morse potential can be fitted easily for most lattices. The Morse potential is defined as:

$$V(r) = D\exp[-2\alpha(r - r_0)] - 2D\exp[-\alpha(r - r_0)] \tag{10.1}$$

where r is the separation of two interacting atoms.

To ensure lattice stability, the r_0 parameter must be equated with the lattice nearest neighbour distance, r_1. Consider first the fcc lattice, which has 12 nearest neighbours. This means that the total cohesive energy of the lattice, E_c, can be evaluated as $6D$:[*]

$$\begin{aligned} -E_c &= \frac{1}{2}\sum_N D\exp[-2\alpha\,(r_1 - r_0)] - 2D\exp[-\alpha\,(r_1 - r_0)] \\ &= 6D - 12D = -6D \end{aligned} \tag{10.2}$$

where $N = 12$ for a fcc lattice. Thus, only the parameter α remains to be fitted. To do this, refer to the definition of the bulk modulus (B):

$$B = \frac{1}{18\Omega}\cdot\sum_N r_1^2 V''(r_1) - r_1 V'(r_1) \tag{10.3}$$

where Ω is the atomic volume,[†] and the primes denote differentiation with respect to r:

$$\begin{aligned} \sum_N r_1 V' &= r_1\sum_N -2\alpha D\exp[-2\alpha\,(r_1 - r_0)] + 2\alpha D\exp[-\alpha\,(r_1 - r_0)] \\ &= 24 r_1 \alpha D(-\exp[-2\alpha\,(r_1 - r_0)] + \exp[-\alpha\,(r_1 - r_0)]) \\ &= 0 \end{aligned} \tag{10.4}$$

$$\begin{aligned} \sum_N r_1^2 V'' &= r_1^2\sum_N 4\alpha^2 D\exp[-2\alpha\,(r_1 - r_0)] - 2\alpha^2 D\exp[-\alpha\,(r_1 - r_0)] \\ &= 24\alpha^2 r_1^2 D(2\exp[-2\alpha\,(r_1 - r_0)] - \exp[-\alpha\,(r_1 - r_0)]) \\ &= 24\alpha^2 r_1^2 D \end{aligned} \tag{10.5}$$

Thus:

$$\alpha = \frac{\sqrt{0.75 B\Omega / D}}{r_1} \tag{10.6}$$

Thus, to summarise the Morse potential parameters for the fcc lattice with a first neighbour cut-off:

---

[*] The factor of ½ is required in order to partition the mutual potential energy between the atom of interest and its interacting neighbours.

[†] For a cubic lattice with lattice constant a, $\Omega = a^3/N$, where N (the number of atoms per unit cell) is 4 for a fcc lattice, 2 for a bcc lattice, and 8 for a diamond lattice. For an ideal hexagonal lattice, $\Omega = (\sqrt{2}a_1)^3/4$, where a_1 is the nearest neighbour distance.

$$\begin{aligned} &\textit{Face}-\textit{centred cubic}: \\ &\alpha = \frac{\sqrt{0.75B\Omega / D}}{r_1} \\ &D = E_c / 6, \\ &r_0 = r_1 \end{aligned} \tag{10.7a}$$

Example: Al has a lattice parameter of 4.0495 Å (which gives r_1 = 2.863 Å, $\Omega = 1.660\times10^{-29}$ atoms m^{-3}), E_c = 3.39 eV, B = 81 GPa (8.1×10^{10} N m^{-2}). Thus D = 3.39/6 = 0.565 eV (9.051×10^{-20} J), and $\alpha = (9\times8.1\times10^{10}\times1.660\times10^{-29}/9.051\times10^{-20})^{½} = 1.166$ $Å^{-1}$.

Eqs. 10.6 and 10.7 also hold for an ideal hcp lattice. Similar formula can be obtained for bcc and diamond lattices (with 8 and 4 nearest neighbours respectively), as follows:

$$\begin{aligned} &\textit{Body}-\textit{centred cubic}: \\ &D = E_c / 4, \\ &r_0 = r_1 \\ &\alpha = \frac{\sqrt{1.125B\Omega / D}}{r_1} \end{aligned} \tag{10.7b}$$

$$\begin{aligned} &\textit{Diamond}: \\ &D = E_c / 2, \\ &r_0 = r_1 \\ &\alpha = \frac{\sqrt{2.25B\Omega / D}}{r_1} \end{aligned} \tag{10.7c}$$

The Morse parameters for an elemental lattice of arbitrary type, with a 1st neighbour cut-off, are:

$$\begin{aligned} &\textit{Any lattice}: \\ &D = 2E_c / N, \\ &r_0 = r_1 \\ &\alpha = \frac{\sqrt{9B\Omega / (ND)}}{r_1} \end{aligned} \tag{10.8}$$

where N is the number of nearest-neighbour atoms.

In Eq. 10.10, all neighbouring atoms are assumed to reside in chemically equivalent lattice sites (with respect to the atom with which they interact). Thus, molecular solids like sulphur are excluded.

10.2.2.1. Second neighbour cut-off

The analytical approach to Morse potential parameterisation becomes very clumsy when the second shell of neighbours is included, because r_0 can no longer be equated with r_1, and the fitting process is best achieved using numerical computer methods.

A simple program, *MorseFit* is supplied with the SK for fitting Morse potentials that have second neighbour cut-offs and either FCC, BCC or (ideal) HCP lattices.[*] The user must supply the lattice parameter, cohesive energy and bulk modulus of the lattice.

The fitting procedure used by *MorseFit* makes use of the following relationships, which can be derived using the formulae given in the previous section (by extending the summations to include the influence of the second neighbour shell):

$$\frac{dE_c}{dr_1} = Da\left[\sum_N (x^2 - x) + r\sum_M (y^2 - y)\right] = 0 \tag{10.9a}$$

$$-2E_c = D\left[\sum_N (x^2 - 2x) + \sum_M (y^2 - 2y)\right] \tag{10.9b}$$

$$2r_1^2 Da^2\left[\sum_N (2x^2 - x) + r^2\sum_M (2y^2 - y)\right] = 18\Omega B \tag{10.9c}$$

where:

$$x = \exp(-a(r_1 - r_0))$$
$$y = \exp(-a(r_2 - r_0)) \tag{10.10}$$
$$r = \frac{r_2}{r_1}$$

Here r_1 and r_2 are the first and second neighbour distances, while N and M are the numbers of first and second neighbours respectively. Eq. 10.9a defines the condition for lattice stability, Eq. 10.9b defines the cohesive energy at a lattice site, and Eq. 10.9c defines the bulk modulus (exploiting the fact that the first derivative of the potential energy with respect to r_1 is zero; see Eq. 10.3 and 10.9a).

10.3. Sutton-Chen (SC) potentials

Sutton-Chen (SC) potentials are available with a cut-off of $2a$ for a number of fcc metals.[25,63,64] Parameters for these metals are shown in Table 10.3. The lattice sums (S_n) required by *Spider* are also shown. These are defined as:

$$S_n = \sum_{n=1}^{NMAX}\left(\frac{a}{r_n}\right)^M \quad ,$$

[*] The program also fits potentials for FCC and BCC lattices up to the third neighbour.

where the r_n represent the distances of the *NMAX* neighbouring atoms found within the range of the potential (*NMAX* = 140 for a 2*a* cut-off).*

Ref. [64] has a very good description of the potential fitting procedure, including all necessary equations for a do-it-yourself approach. Because of their long range, the parameterised potentials given in Table 10.3 can only be recommended for computationally non-intensive problems (lattice energy calculations, diffusion etc.).

Table 10.3. Parameters sets for the Sutton-Chen many body potential for fcc transition metals. a) Quantum Sutton-Chen, new parameters with quantum correction; b) SC-new, new parameters developed by classical calculations; c) original parameters by Sutton and Chen as revised by Rafi-Tabar and Sutton.[63] The table is taken from ref. 64. Parameter sets (a) are recommended. Note that α is also the lattice parameter of the fcc targets.

	N	*M*	*D* (eV)	*c*	α	S_n
Ni						
a)	10	5	7.3767E-3	84.745	3.5157	89.94
b)	10	5	7.5144E-3	83.073	3.5157	89.94
c)	9	6	1.5714E-2	39.756	3.5200	113.66
Cu						
a)	10	5	5.7921E-3	84.843	3.6030	89.94
b)	10	5	5.9066E-3	83.073	3.6030	89.94
c)	9	6	1.2351E-2	39.756	3.6100	113.66
Rh						
a)	13	5	2.4612E-3	305.499	3.7984	89.94
b)	13	5	2.5027E-3	299.946	3.7984	89.94
c)	12	6	4.9371E-3	145.658	3.8000	113.66
Pd						
a)	12	6	3.2864E-3	148.205	3.8813	113.66
b)	12	6	3.3401E-3	145.658	3.8813	113.66
c)	12	7	4.1260E-3	108.526	3.8900	150.42
Ag						
a)	11	6	3.9450E-3	96.524	4.0691	113.66
b)	11	6	4.0072E-3	94.948	4.0691	113.66
c)	12	6	2.5330E-3	145.658	4.0900	113.66
Ir						
a)	13	6	3.7674E-3	224.815	3.8344	113.66
b)	13	6	3.8060E-3	222.348	3.8344	113.66
c)	14	6	2.4524E-3	337.831	3.8400	113.66
Pt						
a)	11	7	9.7894E-3	71.336	3.9163	150.42
b)	11	7	9.8721E-3	70.743	3.9163	150.42
c)	10	8	1.9768E-2	34.428	3.9200	204.55
Au						
a)	11	8	7.8052E-3	53.581	4.0651	204.55
b)	11	8	7.8863E-3	53.082	4.0651	204.55

* The number and distances (in units of *a*) of neighbours of an atom in a fcc lattice are 12@1/√2, 6@1, 24@√1.5, 12@√2, 24@√2.5, 8@√3, 48@√3.5, 6@2.

c)	10	8	1.2896E-2	34.428	4.0800	204.55

10.4. Tight-Binding (TB) potentials for metals

Recently the author has fitted TB potentials for 26 fcc and bcc metals.[65] These are the potentials which I recommend for use with *Kalypso*, because they have the advantage of a much shorter range than the SC potentials. The potential parameters based on that paper are collated in Tables 10.4 and 10.5 respectively. The values given in the Tables can be entered directly into the relevant fields of the MDL file dialog box in *Spider*, which is obviously very convenient. (Just be sure, when you select a cut-off distance, that it falls within the range indicated.)

In the same paper you can find a discussion of the properties of the potentials which will not be repeated here. The fcc potentials were fitted using the first and second neighbour interactions, while for the bcc potentials the third neighbour interactions were also included. The following remarks are taken from ref. [65].

The assumptions underlying the tight-binding model of metallic cohesion in the second-moment approximation are reviewed by Clari and Rosato.[67] Within this approximation, the band energy of the system is proportional to the square root of the second moment of the density of states. The system potential energy, U_S, is written in the following form, where E_i^R and E_i^B represent respectively a repulsive core interaction and the band energy associated with the *i*th atom:

$$U_S = \sum_i (E_i^R + E_i^B), \tag{10.11}$$

where E_i^R is a repulsive pair potential:

$$E_i^R = \sum_{j \neq i} U_{ij}(r_{ij}), \tag{10.12}$$

$$U_{ij}(r_{ij}) = A \exp(-p(r_{ij}/r_0 - 1))^*, \tag{10.13}$$

and E_i^B represents the cohesive band energy term:

$$E_i^B = -\left(\sum_{j \neq i} f(r_{ij}) \right)^{1/2}, \tag{10.14}$$

$$f(r_{ij}) = x^2 \exp(-2q(r_{ij}/r_0 - 1)) \quad . \tag{10.15}$$

- In Eqs. 10.12-10.15, r_{ij} is the separation between atoms *i* and *j*, and A, x, p, q, r_0 are adjustable parameters governing the interaction between those atoms.
- The lattice sum, S_n, which is listed in Tables 10.4 and 10.5, and is required as input by *Spider*, is the value computed for the function $(E_i^B/x)^2$ in the undisturbed lattice.

Eqs. 10.11 to 10.15 are similar to the Finnis-Sinclair (FS) scheme, 66 but Eq. 10.12 uses a double summation convention for the repulsive functions, whereas the convention in the FS scheme is to

---

* Kalypso actually permits a more general form for the repulsive potential:
$U_{ij}(r_{ij}) = A \exp(-p(r_{ij}/r_0 - 1)) + b \exp(-q(r_{ij}/r_0 - 1))$.
However, most users will set $b = 0.0$, leading to Eq. 10.13. If you set b < 0 and ξ = 0, a Morse-like potential results.

sum this term for j > i rather than j ≠ i. **Note: *KALYPSO* USES THE FS CONVENTION**. *Kalypso* also drops the factor of 2 appearing in the exponent in Eq. 10.15. This means that the parameters A and q which appear in tables of TB parameters in the literature must be scaled by a factor 2.0 for use with *Kalypso* (this has already been done for Tables 10.4 and 10.5, but you will need to apply the correction yourself if you use TB potentials taken from the literature).

For (s, p)-bonded metals there is no strong theoretical motivation for representing the band energy part of the potential by a square-root term. However, this functional form can be rationalised as an empirical representation of the volume-dependent term required by the electron gas model of simple metals.

The length scale parameter, r_0, in Eq. 1 can be set without loss of generality to the lattice nearest neighbour distance. The remaining parameters (A, x, p, q) of the TB potentials are then fitted for each element using the lattice constant, cohesive energy (E_c), elastic constants (C_{11}, C_{12}, C_{44}) and vacancy formation energy (E_v). The uncertainties in experimental values of elastic constants and vacancy formation energies are typically on the order of 10-20%. The lattice constant and cohesive energy were fitted exactly, while the remaining properties were fitted using equal weights. Fitting was carried out using a combination of genetic algorithm and downhill simplex methods. The cut off distance (r_{cut}) used for the fitting procedure was chosen to lie between the second and third neighbour distances for the fcc elements, and between the third and fourth neighbour distances for the bcc elements. For those metals whose elastic constants approximate the Cauchy relation ($C_{12} = C_{44}$) - for example Rh, Ir, Th, Ca and Sr - the vacancy formation energy is the only property in the fitting set which strongly manifests many-body behaviour. The inclusion of vacancy formation energies in the fitting procedure should thus be particularly important for fixing the parameters of the potential for these metals. Unfortunately, reliable experimental measurements of E_v are not available for Ca, Rh or Th, so E_v was estimated in these cases as $E_c/3$.

TB potentials for a number of fcc (Ni, Cu, Rh, Pd, Ag, Ir, Pt, Au, Al, Pb) and hcp metals (Ti, Zr, Co, Cd, Zn, Mg) have also been fitted by Cleri and Rosato (CR).[67] These potentials are fitted using a cut-off at 5NN. The CR paper also has references to earlier parameterisations with a 1NN cut-off. Because of differences in formalism, you *must* multiply the CR A and q parameters by a factor of two (2.0) in order to use them with *Kalypso* (see remarks following Eq. 10.15 and footnote to section 3.4). The same remark applies to a set of TB potentials fitted with the same cut-off for Cu, Ag and Au by Kallinteris et al.[54]. If you use any of these potentials you will also have to calculate the associated lattice sum S_n required by *Spider*. (See definition following Eq. 10.15. This can be done by hand, or by clicking the S_n button in *Spider*'s Model file dalog box.)

A paper by Paidar et al. gives an excellent analysis of the solid state properties and relationships predicted by fcc TB potentials fitted up to the second neighbour distance.[68] Apart from the above compilations, there are many other TB potentials in the literature (do a search through Phys. Rev. B, for example). However, most of them have too long a range to be useful for sputtering simulations.

Exercise: use the parameters given in Table 10.4 to calculate the predicted cohesive energy (E_c) of Ag. *Method*: Note that a fcc metal has 12 neighbours at a distance r_0, and 6 neighbours at a distance $\sqrt{2}r_0$. Since the potential is cut off before the third neighbour shell, no other interactions need to be considered. The cohesive energy is the same for all bulk atoms in the lattice.

(a) From Eqs. 10.12 and 10.13, the repulsive energy of the *i* th bulk atom is:

$$E_i^R = \sum_{j\neq i} U_{ij}(r_{ij}) = A\sum_{j\neq i} \exp(-p(r_{ij}/r_0 - 1)) = A\left[12 + 6\exp(-p(\sqrt{2}-1))\right] \tag{10.16}$$

You have to remember, however, that the parameter *A* used in *Kalypso*'s convention is 2.0 times greater than that used in Eq. 10.12 (the TB convention). [Seems confusing, but it is really the TB convention which is unorthodox, not the FS convention.] So you should use *A* = 0.1624/2.0 to evaluate Eq. 10.18.

(b) From Eqs. 10.14 and 10.15, the band energy of the *i* th atom is:

$$\begin{aligned} E_i^B &= -\left(\sum_{j\neq i} f(r_{ij})\right)^{1/2} = -x\left(\sum_{j\neq i} \exp(-2q[r_{ij}/r_0 - 1])\right)^{1/2} \\ &= -x\left(12 + \exp(-2q[\sqrt{2}-1])\right)^{1/2} \end{aligned} \tag{10.17}$$

Again, you have to remember to divide the value of *q* used by *Kalypso* (Table 10.4) by 2.0, i.e. *q* = 5.6632/2.0.

(c) You should obtain: $E_i^R = 0.97853; E_i^B = -3.9294; \rightarrow E_c = -2.9509$
[The E_c value used for the fit was 2.95 eV; the difference, 0.0009 eV, represents the fitting error.]

(d) Comment: whenever you use a potential, you should check its properties using this calculation (write a simple spreadsheet for it). Some other properties (for fcc metals) have been catalogued by Paidar et al.[68] .

10.5. Tight binding potentials for bimetallic systems

The TB potential formalism can be extended to describe bimetallic systems. In general, the fitting parameters for a bimetallic system cannot be deduced from those of the pure elements alone. However, an approximate combination rule which is often used for TB and Finnis-Sinclair potentials may be useful in the absence of specific parameterisations.* This entails choosing the potential parameters in such a way that the heteronuclear interaction terms (*AB*, for interactions between elements *A* and *B*) correspond to the geometric means of the respective elemental terms (*AA*, *BB*):

$$\begin{aligned} f^{AB}(r_{ij}) &= [f^{AA}(r_{ij}) f^{BB}(r_{ij})]^{1/2}, \\ U^{AB} &= [U^{AA}(r_{ij}) U^{BB}(r_{ij})]^{1/2} \end{aligned} \quad . \tag{10.18}$$

* Unfortunately, most potentials fitted in the literature assume ranges longer than what is desirable for simulations of ion-surface interactions.

The accuracy of this approximation needs to be evaluated on a case-by-case basis using suitable thermodynamic measures, and corrections may be required. The author has used the following (approximate) geometric mean scheme to obtain parameters for bimetallic systems:

$$\begin{aligned} A_{AB} &= \sqrt{A_{AA}A_{BB}} \\ x_{AB} &= \sqrt{x_{AA}x_{BB}} \\ p_{AB} &= (p_{AA} + p_{BB})/2 \\ q_{AB} &= (q_{AA} + q_{BB})/2 \\ 2r_{0AB} &= (2p_{AB})/(p_{AA}/r_{0AA} + p_{BB}/r_{0BB}) + (2q_{AB})/(q_{AA}/r_{0AA} + q_{BB}/r_{0BB}) \end{aligned} \tag{10.19}$$

The expression for r_{0AB} is complex because there is no unique way to assign a value for this parameter (an average of two methods has been used, but other methods could be devised). Thus it should be recognised that this element of the scheme is essentially arbitrary. Furthermore, the scheme is based on the assumption that *A*-*B* interactions are energetically intermediate between *A*-*A* and *B*-*B* interactions. You can find many examples in the literature of TB potentials which use this assumption for bimetallic systems, as well as examples of more rigorous fitting procedures. The geometric means assumption works best for systems in which the heat of solution of dilute *A* in bulk *B* has a similar magnitude, but opposite sign, to the heat of solution of dilute *B* in bulk *A*. There may be cases where it breaks down completely.

Table 10.4. Parameters of tight-binding potentials for fcc metals.* The potentials should be cut off somewhere between the second and third neighbour distances ($\sqrt{2}\,r_0 < r < \sqrt{3}\,r_0$). The parameter S_n is a lattice sum required for input to *Kalypso*.

	A (eV)	ξ (eV)	p	b	q	r_0 (Å)	S_n
Al	0.3204	1.5074	7.5681	0.0	5.4912	2.8634	12.617
Ca	0.0984	0.6842	11.2115	0.0	5.3682	3.9471	12.652
Ni	0.113	1.4005	14.0867	0.0	3.5874	2.4918	13.359
Cu	0.1566	1.2355	11.1832	0.0	4.6394	2.5560	12.881
Sr	0.0514	0.5557	12.3406	0.0	3.621	4.3027	13.337
Rh	0.2172	1.9776	14.1315	0.0	5.111	2.6901	12.723
Pd	0.2446	1.5193	11.3225	0.0	6.1394	2.7511	12.475
Ag	0.1624	1.1081	11.5597	0.0	5.6632	2.8890	12.574
Ir	0.4282	2.7082	12.8986	0.0	6.9082	2.7145	12.341
Pt	0.5812	2.6715	10.1423	0.0	7.5756	2.7746	12.257
Au	0.387	1.7581	10.4342	0.0	7.8944	2.8838	12.229
Pb	0.1702	0.8699	10.0667	0.0	6.7126	3.5003	12.369
Th	0.24	2.0937	9.8344	0.0	3.5944	3.5951	12.674

Table 10.5. Parameters of tight-binding potentials for bcc metals.* The potentials should be cut off somewhere between the third and fourth neighbour distances ($\sqrt{8/3}\,r_0 < r < \sqrt{11/3}r_0$).The parameter S_n is a lattice sum required for input to *Kalypso*.

	A (eV)	ξ (eV)	p	b	q	r_0 (Å)	S_n
Li	0.0976	0.5729	6.3675	0.0	2.7938	3.0391	13.943
Na	0.0706	0.4083	7.8536	0.0	3.4954	3.7158	12.809
K	0.046	0.3170	9.3093	0.0	3.2286	4.6073	13.199
V	0.5144	2.3126	6.8543	0.0	4.3772	2.6223	11.799
Cr	0.0814	1.1012	13.1852	0.0	1.7986	2.4981	16.386
Fe	0.2368	1.5418	10.7613	0.0	4.0758	2.4824	12.103
Rb	0.0584	0.3233	8.1532	0.0	3.847	4.9363	12.362
Nb	0.9092	3.6302	5.2702	0.0	4.1104	2.6033	12.069
Mo	0.4086	2.5097	10.0154	0.0	4.1022	2.7253	12.076
Cs	0.054	0.3036	8.4120	0.0	3.8866	5.3174	12.313
Ba	0.08	0.6167	10.1835	0.0	3.014	4.3466	13.542
Ta	0.6562	3.3008	8.2764	0.0	4.4742	2.8601	11.710
W	0.498	3.2055	10.3715	0.0	3.9832	2.7410	12.201

* These parameters are expressed in a form suitable for direct entry into Spider.

INDEX

REFERENCES

[1] R. Smith, M. Jakas. D. Ashworth, B. Oven, M. Bowyer, I. Chakarov and R. Webb, Atomic and Ion Collisions in Solids and at Surfaces, Cambridge University Press, 1997.

[2] Wolfgang Eckstein, Computer Simulation of Ion-Solid Interactions, Springer-Verlag, Berlin, 1991.

[3] E.S. Mashkova and V.A. Molchanov, Medium-Energy Ion Reflection from Solids, North-Holland, Amsterdam, 1985.

[4] D.E. Harrison Jr., Sputtering Models - A Synoptic Review, Radiation Effects, vol. 70 (1983) 1-64.

[5] R. Smith and D.E. Harrison Jr., Algorithms for Molecular Dynamics Simulations of keV Particle Bombardment, Computers in Physics, vol. 3 (1989) 68-73.

[6] M.T. Robinson, Theoretical Aspects of Monocrystal Sputtering, in Sputtering by Particle Bombardment I, (pp. 73-144) , R. Behrisch ed. (Springer-Verlag, Berlin, 1981).

[7] H. Niehus, W. Heiland and E. Taglauer, Low-energy Ion Scattering at Surfaces, Surface Science Reports vol. 17 (1993) 213-303.

[8] J.W. Rabalais (ed.), Low-Energy Ion-Surface Interactions, (Wiley, Chichester, 1994).

[9] M. Nastasi, J.W. Mayer, J.K. Hirvonen, Ion-Solid Interactions: Fundamentals and Applications, (Cambridge Univ. Press, Cambridge 1996).

[10] Herbert Goldstein, Classical Mechanics, 2 nd ed. (Addison-Wesley, Reading MA, 1981).

[11] L.D. Landau and E.M. Lifshitz, Course of Theoretical Physics, Volume 1: Mechanics, 3rd ed. (Pergamon, Oxford, 1976).

[12] H.S. Tan and M.A. Karolewski, Nuclear Instr. Meth. B73 (1993) 163.

[13] M. Gryzinski, Phys. Rev. 138, (1965) A305.

[14] C. Lehmann and M.T. Robinson, Phys. Rev. 134 (1964) A37-A44; M.T. Robinson and I.M. Torrens, Phys. Rev. B9 (1974) 5008-5024.

[15] M.H. Mendenhall, R.A. Weller, Nuclear Instr. Meth. B58 (1991) 11-17; O.S. Oen, Surface Sci. 131 (1983) L407-L411; J.P. Blanchard, N.M. Ghoniem and S.P. Chou, J. Appl. Phys. 61 (1987) 3120; J.F. Ziegler, J.P. Biersack and U. Littmark, The Stopping and Range of Ions in Matter, vol. 1 (Pergamon, New York, 1985).

[16] A.M. Hiltebeitel, Amer. J. Math. XXXIII (1911) 337-362; C.G. J. Jacobi, Gesammelte Werke: Vorlesungen über Dynamik, (Chelsea Publishing Co., New York, 1969) pp. 221-231; H.C.

Corben and P. Stehle, Classical Mechanics, 2nd Edition (Wiley, New York,1960) pp. 201-213; V. Arnold, Les Méthodes Mathématiques de la Méchanique Classique, (Pub. Éditions Mir, Moscow, 1976), pp. 255-264; E.T. Whittaker, A Treatise on Analytical Dynamics of Particles and Rigid Bodies, 3rd Edition, (Cambridge Univ. Press, Cambridge, 1927), pp. 97-99; L.D. Landau, E.M. Lifshitz, Mechanics, 3rd Edition, (Pergamon, Oxford, 1969), Chap. 7 (section 48, Problem 2, p154).

[17] R.S. Daley, D. Farelly and R.S. Williams, Surface Sci., 234 (1990) 355; R.S. Williams, M. Kato, R.S. Daley and M. Aono, Surface Sci. 225 (1990) 355-366; S. Chaudhury and R.S. Williams, Surface Sci. 255 (1991) 127-134; S.J. Anz and R.S. Williams, Surface Sci., 372 (1997) L323; R.S. Williams, Quantitative Analysis of Low-Energy Scattering and Recoiling from Crystal Surfaces, in ref. [8], pp. 1-54.

[18] Public domain atomic collision simulation programs of various types can be obtained through the following Internet sites:

(a) MARLOWE: http://www.ssd.ornl.gov/marlowe/marlowe.html;

(b) SRIM (includes TRIM): http://www.research.ibm.com/ionbeams/;

(c) CRYSTAL-TRIM: http://www.fz-rossendorf.de/FWI/FWIT/ctrim.e.html;

(d) MDRANGE: http://beam.helsinki.fi/~knordlun/mdh/mdh-program.html;

(e) KSAN: http://ksan.ms.nwu.edu/ksan.html;

(f) TRIDYN: http://www.cpc.cs.qub.ac.uk/cpc/.

[19] D.C. Rapaport, Comp. Phys. 4 (1997) 337.

[20] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, Numerical Recipes in Pascal (editions also in Fortran or C) (Cambridge University Press, Cambridge, 1989).

[21] Derek Wood, Data Structures, Algorithms and Performance (Addison-Wesley, New York, 1993); Robert Sedgewick, Algorithms (2nd Ed.), (Addison-Wesley, Reading, MA, 1988), chapter 26 (“Range Searching”).

[22] K. Gärtner et al., Nuclear Instr. Meth. B 102 (1995) 183.

[23] D.E. Harrison, C.E. Carlston and G.D. Magnuson, Phys. Rev. 139 (1965) A737-A745.

[24] A.S. Goldstein and H. Jónsson, Phil. Mag. B 71 (1995) 1041-1056.

[25] A.P. Sutton and J. Chen, Phil. Mag. Lett. 61 (1990) 139.

[26] R.P. Gupta, Phys. Rev. B 23 (1985) 6265; D. Tomanek, A.A. Aligia and C.A. Balseiro, Phys. Rev. B 32 (1985) 5051.

27 M.W. Finnis and J.F. Sinclair. Phil. Mag. A 50 (1984) 45.

28 M.H. Shapiro and T.A. Tombrello, Nuclear Instr. Meth. B 84 (1994) 453-464.

[29] K. Broomfield, R.A. Stansfield and D.C. Clary, Surface Sci. 202 (1988) 320.

[30] K. Nordlund, N. Runeberg and D. Sundholm, Nuclear Instr. Meth. 132 (1997) 45.

[31] N. Stritt, J. Jolie, M. Jentschel, H.G. Börner, C. Doll, J. Res. Natl. Inst. Stand. Technol. 105 (2000) 71 (available online at the NIST Web site, http://www.nist.gov).

[32] M.H. Shapiro and T.A. Tombrello, *Nucl. Instr. Meth*. *B* 84 453 (1994) .

[33] M.H. Shapiro and T.A. Tombrello, Nuclear Instr. Meth. B 90 (1990) 473; M.H. Shapiro and T.A. Tombrello, Nuclear Instr. Meth. B 94 (1994) 186; M.H. Shapiro and T.A. Tombrello, Nuclear Instr. Meth. B 102 (1995) 277.

[34] G.J. Ackland and V. Vitek, Phys. Rev. B 41 (1990) 10324.

[35] B.J. Garrison, N. Winograd, D.M. Deaven, C.T. Reimann, D.Y. Lo, T.A. Tombrello, D.E. Harrison Jr. and M.H. Shapiro, Phys. Rev. B 37 (1988) 7197.

[36] M.A. Karolewski, Surface and Interface Analysis, 27 (1999) 114-122.

[37] M.A. Karolewski, R.G. Cavell, Surf. Sci. (to be published in 2001).

[38] H.H. Andersen and H.L. Bay, Sputtering Yield Measurements, in R. Behrisch (Ed.), Sputtering by Particle Bombardment I, (Springer-Verlag, Berlin, 1981), pp. 145-209.

[39] H.E. Roosendaal, Sputtering yields of Single Crystalline Targets, in R. Behrisch (Ed.), Sputtering by Particle Bombardment I, (Springer-Verlag, Berlin, 1981), pp. 219-254.

[40] M.A. Karolewski, Nuclear Instruments and Methods in Research, B 159 (1999) 28.

[41] B.J. Garrison, N. Winograd, D. Lo, T.A. Tombrello, M.H. Shapiro and D.E. Harrison, Surface Sci. 180 (1987) L129-L133.

[42] W.O. Hofer, Angular, Energy and Mass Distribution of Sputtered Particles, in R. Behrisch and K. Wittmaack (Ed.), Sputtering by Particle Bombardment III, (Springer-Verlag, Berlin, 1981), pp. 15-89.

[43] D. Onderlinden, Can. J. Phys. 46, 739 (1968).

[44] M. Hou and W. Eckstein, Nucl. Instr. and Meth. B **13**, 393 (1986).

[45] M.A. Karolewski, R.G. Cavell, Surface Sci. (to be published in 2001).

[46] Th. Fauster, D. Hartwig and H. Dürr, Appl. Phys. A45 (1988) 63-67.

[47] C.C. Chang, Surface Interface Anal. 15 (1990) 79-84; D.W. Moon, R.J. Bieler and N. Winograd, J. Chem. Phys. 85 (1986) 1097-1103; N. Winograd and C.C. Chang, Phys. Rev. Lett. 62 (1989) 2568-2569; C.C. Chang and N. Winograd, Surface Sci. 230 (1990) 27-34; C.C. Chang and N. Winograd, Phys. Rev. B 39 (1989) 3467.

[48].J.W. Rabalais, H. Bu and C.D. Roux, Phys. Rev. Lett. 69 (1992) 1391.

[49] L. Wong, P.F.A. Alkemade, W.N. Lennard and I.V. Mitchell, Nucl. Instr. Meth. B45 (1990) 637.

[50] L.C.A. van den Oetelaar, H.E. van Benthem, J.H.J.M. Helwegen, P.J.A. Stapel and H.H. Brongersma, Surface Interface Abnal. 26 (1998) 537-548.

[51] H. Dürr, R. Schneider and Th. Fauster, Phys. Rev. B 43 (1991) 12187.

[52] R. Spitzl, H. Niehus and G. Comsa, Surf. Sci. Lett. 250 (1991) L355.

[53] R.C. Shukla, Phil. Mag. Lett. 70 (1994) 255-259.

[54] G.C. Kallinteris, N.I. Papanicolaaou, G.A. Evangelakis and D.A. Papaconstantopoulos, Phys. Rev. B 55 (1997) 2150-2156.

[55] M. Aono and R, Souda, Nucl. Instr. Meth. B27 (1987) 55-64.

[56] J. Lindhard, M. Scharff and H.E. Schiøtt, K. Dan. Vidensk. Selsk. Mat. Fys. Medd. 33, 14 (1966); J. Lindhard and M. Scharff, Phys. Rev. 124, 128 (1961).

[57] O.S. Oen and M.T. Robinson, Nucl. Instr. and Meth. 132, 647 (1976).

[58] M.S. Daw, S.M. Foiles and M.I. Baskes, Materials Sci. Reports 9 (1993) 251-310.

[59] A. Carlsson, Solid State Phys. 43 (1990) 1.

[60] F. Ercolessi, M. Parrinello and E. Tosatti, Phil. Mag. A 58 (1988) 213-216.

[61] M. I. Baskes, R.A. Johnson, Modelling. Simul. mater. Sci. Eng. 2 (1994) 147-163.

[62] M. Igarashi, M. Khantha and V.Vitek, Phil. Mag. B 63 (1991) 603-627.

[63] H. Rafi-Tabar and A.P.Sutton, Phil. Mag. Lett. 63 (1991) 217-224.

[64] Y. Kimura, Y. Qi, T. Cagin, and W.A. Goddard, Quantum Sutton-Chen Many-Body Potential for Properties of fcc Metals. [Unpublished: on the Web at: http://www.wag.caltech.edu/home-pages/tahir/psfiles/51.ps]

[65] M.A. Karolewski, Radiation Effects and Defects in Solids, 153 (2001) 239-255. [Note that the parameters given in this paper for Th (thorium) are incorrect, due to fitting with incorrect elastic constants. The parameters are given in Table 10.4 of this document have been corrected.]

[66]. M.W. Finnis and J.E. Sinclair, *Phil. Mag. A* 50, 45 (1984) .

[67] F. Cleri and V. Rosato, Phys. Rev. B 48 (1993) 22-48.

[68] V. Paidar, A. Larere and L. Priester, Modelling. Simul. Mater. Sci. Eng. 5 (381) 1997.