

KALYPSO: Software for Simulation of Atomic Collisions at Surfaces

Version 2.1 User Guide

Marcus Karolewski

Department of Chemistry, University of Brunei Darussalam, Brunei, Borneo

Revision: 3 December, 2005

CONTENTS

1. INTRODUCTION.....	5
1.1. PREFACE.....	5
1.1.1 (a) Version 2.1	5
1.1.1 (b) Version 2.00	5
1.2. KALYPSO FUNCTIONALITY	6
1.3. PACKAGE COMPONENTS	9
1.4. COMPUTATION TIME	9
1.5. KALYPSO OUTPUT	10
REFERENCES FOR CHAPTER 1	11
2. INTERATOMIC POTENTIALS.....	13
2.1. THE COMPOSITE POTENTIAL	13
2.2. CUT-OFF DISTANCE.....	13
2.3. SCREENED COULOMBIC POTENTIAL.....	14
2.4. TIGHT-BINDING (TB) POTENTIALS FOR METALS.....	15
2.5. TIGHT BINDING POTENTIALS FOR BIMETALLIC SYSTEMS.....	19
2.6. THE SWITCHING FUNCTIONS	20
2.7. CALCULATIONS WITH TIGHT-BINDING POTENTIALS	26
2.7.1. Cohesive energy.....	26
2.7.2. Other properties.....	28
2.8. SIMULATIONS WITH TIGHT-BINDING POTENTIALS	28
2.9. INTEGRATION METHOD	30
2.10. MORSE POTENTIAL	31
ANNEXE. DATABASE OF SURFACE SEGREGATION ENERGIES [25].	32
REFERENCES FOR CHAPTER 2	33
3. TARGET FILES AND PROJECTILE FILES	35
3.1. FUNCTION OF THE TARGET FILE.....	35
3.2. COORDINATE SYSTEM.....	35
3.3. ANCHOR ATOM.....	36
3.4. TARGET FILE FORMAT.....	36
3.4. OPTIONS FLAGS	38
3.4.1. Flags used by Kalypso	38
3.4.2. Flag: ofUseImagePotential.....	39
3.4.3. Flag: ofNoCool.....	39
3.4.4. Flag: ofSprungAtom.....	39
3.4.5. Flag: ofFixedAtom.....	39
3.4.6. Flag: ofEdgeAtom.....	39
3.4.7. Flag: ofRecorded.....	39
3.4.8. Flag: ofPreImplant	40
3.4.9. Other flags	40
3.5. EXERCISE: GENERATING A Ni(100) LATTICE.....	40
3.6. EXERCISE: TARGET FILE FOR (1×1) METAL MONOLAYER SYSTEM	41
3.7. ORIENTING TARGETS	42
3.8. CHOOSING THE TARGET SIZE	43
3.9 PROJECTILE FILES	43
4. THE RUN FILE.....	45
4.1. FUNCTION OF THE RUN FILE	45
4.2. RUN FILE OPTIONS	45
4.2.1. General specifications.....	45
4.2.2. Periodic boundaries.....	47
4.2.3. Projectile initialisation	47
4.2.4. Termination criteria.....	48
4.2.5. Output	49
4.2.6. Thermal vibrations.....	51
4.2.7. Neighbour lists and timestep.....	53
4.2.7. Multiple impact (MI) simulations (notes).....	56

4.2.8. Periodic boundary conditions (notes)	57
REFERENCES FOR CHAPTER 4	58
5. THE MODEL FILE	59
5.1. FUNCTION OF THE MODEL FILE	59
5.2. MODEL FILE OPTIONS	60
5.2.1. Screened Coulombic potentials	60
5.2.2. Tight-binding potentials	61
5.2.3. Switching functions	62
5.3. MODEL FILES FOR ION-SCATTERING SPECTROSCOPY (ISS) SIMULATIONS	63
6. THE IMPACT FILE	64
6.1. FUNCTION OF THE IMPACT FILE	64
6.2. IMPINGING MODE	65
6.3. RECOILING MODE	65
6.4. MIXED MODE	66
6.5. IMPACT FILE FORMAT	66
6.6. CREATING IMPACT FILES	66
6.7. IMPACT FILE EXAMPLES	70
6.7.1. Normal projectile incidence on Cu(110) surface	70
6.7.2. Non-normal projectile incidence on Cu(100) surface: $\phi = \langle 001 \rangle$	72
6.7.3. Impact file for fcc (111) surface, normal incidence	73
6.7.4. Impact files for multiple impact simulations	75
REFERENCES AND NOTES FOR CHAPTER 6	75
7. THE INELASTIC FILE	76
7.1. INTRODUCTION	76
7.2. LINDHARD-SCHIOTT-SCHARFF (LSS) MODEL	77
7.2.1. THEORY	77
7.2.2. LSS PARAMETERS	78
7.3. OEN-ROBINSON (OR) MODEL	78
7.3.1. THEORY	78
7.3.2. OR PARAMETERS	79
7.4. SHAPIRO-TOMBRELLO (ST) MODEL	80
7.4.1. THEORY	80
7.4.2. ST PARAMETERS	82
7.5. TEMPERATURE CONTROL	82
7.6. IMAGE POTENTIAL EFFECTS	83
7.7. LATTICE SITE SPRINGS	83
7.8. COMPUTE K(LSS), K(OR) TOOL	83
7.9. INELASTIC DATA FILE	84
REFERENCES FOR CHAPTER 7	85
8. KALYPSO	86
8.1. INTRODUCTION	86
8.2. SIMULATION OPTIONS	86
8.3. RUNNING A SIMULATION PROJECT	88
8.3.1. Single simulation project	88
8.3.2. Batch simulation project	88
8.3.3. Kalypso screen output	89
8.3.4. Kalypso user interface	90
8.3.5. Implementation notes	91
9. WINNOW	92
9.1. INTRODUCTION	92
9.2. QUERY LANGUAGE	93
9.2.1. Introduction	93
9.2.2. Syntax	93
9.2.3. Predefined identifiers (floating point variables)	94
9.2.4. Integer variables (<i>rw</i> , <i>rn</i> , <i>ui</i> , <i>fl</i>)	95

9.2.5. Constants	95
9.2.6. Arithmetic Operators and Arithmetic Expressions	95
9.2.7. Functions	96
9.2.8. Conditional expressions and filtering	96
9.2.9. Logical and relational operators	97
9.2.10. Numeric types.....	98
9.2.11. Parser errors.....	99
9.3. FILTERING DATA.....	99
9.4. AVERAGING DATA	100
9.5. COLLATING DATA	101
9.6. FORMAT COLUMNS OPERATION	102
9.6.1. Summary	102
9.6.2. Example: Create a Target file based on a dynamics (*.snk) file.....	102
9.7. CONVERTING DATA	103
9.8. CONSTRUCTING DATA SPECTRA (HISTOGRAMS).....	104
9.9 MERGING AND RE-MERGING DYNAMICS FILES	105
9.10. FIND SPUTTERED CLUSTERS	105
9.11. SPUTTERING STATISTICS	106
9.12. DISPLACEMENTS OPERATION	107
9.13. CROSS-REFERENCE OPERATION	108
9.14 SCATTERING RELATIONS.....	109
9.15. CONVERT SNK TO POV	109
9.16. CONVERT TRG TO SNK	111
9.17. REFORMAT MI DATA	111
9.20. NEIGHBOUR COUNT	112
ANNEXE: QUERY EXPRESSION EXAMPLES	112
FUNCTION (EXPRESSION) SPECIFICATIONS.....	112
CONDITIONAL EXPRESSIONS	113
REFERENCES FOR CHAPTER 9	114
APPENDIX 1: KALYPSO ERRORS AND WARNINGS	115
ERROR MESSAGES.....	115
WARNING MESSAGES.....	117
APPENDIX 2: FUNDAMENTAL PHYSICAL CONSTANTS.....	117

1. INTRODUCTION

1.1. Preface

1.1.1 (a) Version 2.1

This release of the Kalypso package fixes a few problems that were found in the version 2.0 (programs, documentation, example projects), but contains few new simulation capabilities. The root directory of the installation package contains a file (`revisions.txt`) that summarises the changes made to the programs. As always, I am grateful to users who took the trouble to write to me.

1.1.1 (b) Version 2.00

Several years have passed since I released version 1 of *Kalypso*, and only now do I find myself again in surroundings (the island of Borneo) that are conducive to the further development of the program. The *Kalypso* package has been largely rewritten for the version 2 release, a task that has consumed the entire autumn and winter of 2003-2004. The goals of this effort have been to extend the utilitarian value of the package, and to make it easier to use. Another design objective was to make the program easier to maintain and develop as time goes on. *Kalypso* remains free because it is a by-product of my own research interests. However, for the same reason, its capabilities will only evolve slowly as I find time to work on it.

In general, I welcome feedback from users about the package (contact information is given on the front page), and I am grateful to the many people who wrote to me about version 1 of the package.

This version (version 2) of the package has been tested over several months by the author and by others prior to its formal release. All bugs known to me have been fixed, but others will surely come to light eventually. The program appears to be stable over time in the sense that it does not crash during normal operation. For example, I have carried out very demanding batch projects that involve multiple impact (> 200 impacts) simulations over periods of ~ 3 months using 10^5 atom targets without any hint of a problem.

If you have a bug to report, please attach **a set of input files**, and make sure to state the **program version** (see the Help/About box) and **operating system** that you are using. Otherwise, my reply will most likely start with the phrase: “Can you attach a set of input files?”. If you want to know how to do something, be specific about what it is, and what the bombardment conditions are. Otherwise, my reply will most likely start with the phrase: “Can you be more specific...?”.

The **literature citation** for *Kalypso* 2 is in ref. [1] (a preprint copy can be found in the `/docs/papers` directory of the *Kalypso* distribution).

The **User Guide** is the principal help document for *Kalypso*. A **tutorial** that describes the construction and analysis of a simple simulation project (sputtering of Cu(100)) is also provided. In the `/examples/project` directory of the *Kalypso* distribution, you can find input files for number of complete simulation projects, each of which is accompanied by documentation (**readme files**) that describes the simulation strategy and the data analysis procedure for that project. These projects illustrate how to set up the non-standard features of the package such as multiple impact simulations, periodic boundary conditions and angular scans. Context-sensitive **online Help** files are also provided. Most topics in the latter are abbreviated

versions of this User Guide, but the online Help files also contain information about utility functions that is not found in the User Guide.

1.2. *Kalypso* functionality

Kalypso, which comprises a program of the same name, and various utilities, is a Windows software package for molecular dynamics (MD) (also known as classical dynamics, CD) simulations of atomic collisions in (primarily) metallic and bimetallic crystals. *Kalypso* uses centrosymmetric many-body tight-binding (or Gupta) potentials which can describe the material and cohesive properties of (in order of decreasing accuracy): fcc metals, hcp metals, bcc metals. Please note that these potentials are *not* particularly suitable for modelling the cohesive and material properties of semiconductors like Si or GaAs, or ionic materials such as MgO. However, the short-range repulsive potentials for any binary compound material can be correctly simulated by *Kalypso*, which means that *Kalypso* can also be used for energetic *ion scattering* simulations that are determined by hard collisions of a homonuclear projectile species on any binary compound material (e.g. $\text{Ar} \rightarrow \text{KBr}$, $\text{O}_2 \rightarrow \text{KBr}$). Please note the following limitation: if the projectile is a heteronuclear species (e.g. CuNi), the target must be comprised of one or both of the same types of atoms; thus, $\text{CuNi} \rightarrow \text{Cu}(100)$ or $\text{CuNi} \rightarrow \text{Ni/Cu}(100)$ is acceptable, but $\text{CuNi} \rightarrow \text{Ag}(100)$ is not acceptable. This restriction is enforced by *Kalypso*.

The many-body potentials used by *Kalypso* are described in Chapter 2. The systems that can be simulated consist of metallic atoms (up to two types) plus optionally a noble (inert) gas atom. Some examples of processes that could be studied with *Kalypso* are: relaxation of a Cu(100) surface; sputtering of Ni/Cu(111) by Ar projectiles; sputtering of Cu(100) by O_2^+ projectiles; diffusion of Pb adatoms on Cu(100), γ -ray induced recoils in Ni, ion scattering from Ag(110). At present, *Kalypso* does not include any algorithms that can be used to simulate ion neutralisation processes as the program runs. The main reason for this omission is disagreement among researchers in the field about the applicable theory: I welcome correspondence about suitable algorithms.

Table 1 (below) summarises the main features of the program. Typically, energy is imparted to the system by one or more primary projectiles such as an inert gas atom or a metal atom or cluster, or by temperature ramping.

The range of particle energies that can be treated by *Kalypso* is roughly 0.1 eV to 10 keV. The lower energy limit is imposed by quantum effects, while the upper limit is determined by the treatment used for modelling inelastic effects, and by the practical difficulty of containing fast projectiles in small targets. This energy range covers, for example, deposition of metals by evaporation, sputtering and ion scattering phenomena, and numerous less familiar phenomena such as gamma-ray induced Doppler broadening.

Most *Kalypso* simulations will involve the calculation of the average effects of N projectile impacts at a statistical sample of different surface impact points. *Kalypso* can simulate these projectile impacts on a virgin surface ('zero-fluence' simulation) or on a surface which accumulates the damage from prior projectile impacts ('multiple impact' simulation).

Table 1. System requirements and simulation capabilities of *Kalypso*, version 2.

Property	Capability
Platform	32-bit Windows environments (95/98/Me/2000/NT/XP etc.)
Hardware (preferred)	IBM-type PC: 3GHz CPU, 1 GB RAM
Hardware (minimum)	IBM-type PC: 500 MHz CPU, 128 MB RAM
Projectile (optional)	Inert gas atom, metal atom or metal cluster e.g. Ar, Cu, CuNi
Target	Elemental metal or bimetallic crystal, with or without periodic (x, y) boundary conditions e.g. Cu(100), Ni/Cu(100), Ni _x Cu _{1-x} (100). Also, for ion scattering > 50 eV: any binary compound material.
Maximum practical system size	10 ⁴ -10 ⁵ atoms (depends on RAM, CPU speed and simulation goal)
Maximum allowed size	10 ⁶ atoms
Energy range	10 ⁻¹ to 10 ⁴ eV
Interatomic potentials	Composite screened Coulombic potentials + Tight-binding potentials (with switching functions in the cut-off region)
Inelastic effects etc.	Local and non-local electronic stopping (LSS, Oen-Robinson, Shapiro-Tombrello models); thermostat; image potential; lattice site 'springs'
Typical time frame	10 ¹ -10 ⁵ fs

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.

Molecular dynamics is a theory-neutral simulation technique. By this I mean that you can use molecular dynamics to implement the latest theory, or an incorrect theory. The essential goal of molecular dynamics is to provide particle trajectories. Data processing is a necessary operation that follows any atomistic simulation, but the data processing stage is not unique to molecular dynamics simulation.

If you need to model absolute quantities, such as sputter yields, you may be disappointed by the performance of molecular dynamics simulation programs. This is because absolute quantities are quite sensitive to the input parameters (e.g. potential screening length), which may be difficult to choose optimally. 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.

I envisage that *Kalypso* will be mainly of interest for people who are in the business of doing experiments with ions (e.g. ion bombardment, sputtering, ion scattering) that require interpretation in terms of an atomistic model. *Kalypso* can also be used to carry out theoretical enquiries of a more general nature that have no connection to particular experiments, e.g. how do sputtering at glancing and normal projectile incidence differ?

Kalypso permits the evolution of a system (defined by certain initial conditions and an interaction model) to be monitored over a period of time. The simulation of events following each incident projectile trajectory is described as a 'run'. For example, a sputtering simulation might consist of 1000 runs, each lasting for 2000 fs. The initial

conditions at the start of each run will be quite similar, except for the starting position (impact parameter) of the projectile, which will sample different impact points within the surface unit cell. The program uses the Verlet integration algorithm with an adaptive timestep to integrate the classical equations of motion for a system of interacting particles. The output from the simulation is set of particle coordinates and velocities at one or more specified sampling intervals which are stored in the ‘dynamics’ (or ‘trajectory’) file.

Kalypso is not designed to simulate specific experiments, so it is up to the user to manipulate the output trajectory data in a way that can be compared with experimental data (if that is the purpose of the simulation). It is quite easy to generate simulation data, but it is usually more difficult to extract meaningful output in the form of scattering profiles, spectra and so forth. Winnow can be used for many common tasks, but on occasions you might have to write your own programs to process the output *.snk files.

The best way to start learning about the package is to run the program *Kalypso* using the tutorial and example projects provided. The projects can be run immediately by loading the input files into *Kalypso*. A number of references to the primary and review literature are supplied in context and may be found at the end of each chapter of this guide. For absolute beginners, I suggest the online MD primer by Ercolessi [2] and an online course on computational materials science by Zhigilei [3]. There are several texts and review papers that should be mentioned as general references. These are by: Smith *et al.* [4], Eckstein [5], Mashkova and Molchanov [6], Harrison [7], Smith and Harrison [8], Robinson [9], Niehus *et al.* [10], Rabalais *et al.* [11] and Nastasi *et al.* [12]. 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 that are neglected in this user guide, and give leads to the early literature of this field, which remains remarkably relevant and valuable today.

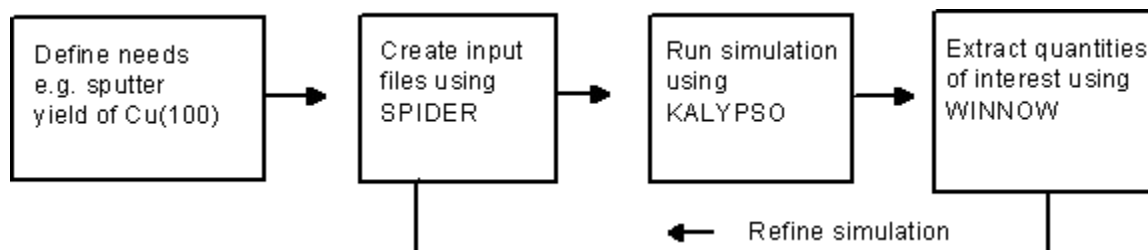
Simulation programs should not be treated as black boxes. Background reading is especially important for those who are new to simulation, since this will give an appreciation of not only *what Kalypso* is doing, but *why* it is done in a certain way and what is being neglected in the model. My Webb pages (see cover page for root URL) provide hyperlinks to literally hundreds of articles by researchers in the field of projectile-surface interactions and their simulation. If you plan to publish your results, I suggest you use papers by simulation researchers as a guide to what information should be provided in your description of the simulation. From correspondence arising from the previous version of the program, I anticipate that *Kalypso* will be more useful for experienced researchers than for students who are still finding their way around the literature.

For simulation projects that do not involve inelastic scattering, the key test of program correctness is the conservation of energy. In a simulation, the system moves from a point *A* to a point *B* in phase space over a time interval Δt . The system evolution is effected by using an integration algorithm that depends on forces computed at every timestep. We know from Newtonian mechanics that if this integration is carried out correctly, the energy of the system (which can be calculated from analytic formulae) should be conserved in a purely elastic model. Users of *Kalypso* should therefore always pay attention to energy conservation since this is the best guarantee that the program is working correctly. Energy leakages of >0.5% observed for purely elastic scattering phenomena are indications that something is wrong in the simulation. If you cannot resolve energy discrepancies by modifying

your simulation parameters (normally the timestep) you should contact the author for advice.

1.3. Package Components

The *Kalypso* package consists of several integrated programs, whose relationships are indicated in the flow chart below.



The main programs in the *Kalypso* package are: ***Spider*** (Simulation Program Input Data Editor), ***Kalypso*** (the simulation engine), and ***Winnnow***. *Spider* is a utility program whose function is to design and generate the input data files (*.trg, *.prj, *.run, *.mdl, *.imp, *.inl) used by *Kalypso*. These files are discussed in various chapters of this User Guide. All simulation projects begin with *Spider*, and this is the most important program to master. *Winnnow* is a program that processes the output data created by *Kalypso*, and transforms this data into scientifically useful information (sputter yields, energy spectra etc.). It may be helpful to new users of *Kalypso* to regard the output files produced by *Kalypso* as database files: the purpose of *Winnnow* is to process user-defined queries to this database, and to present the information in a modified form. Users of the *Kalypso* package will undoubtedly have to learn how to use both *Spider* and *Kalypso*. *Winnnow* 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) for analysis of the dynamics file data. The programs should first be studied by running them on the tutorial and demonstration projects.

Another program supplied with the package is *Cone*. This program, which was written by Mr Tan Hean Seng (formerly of the National University of Singapore), is extremely useful for quick calculations of shadow-cone radii in ion scattering experiments [13]. For this release, I have added a graphical user interface to the program.

1.4. Computation Time

With a 2.4 GHz CPU, a single sputtering run can be carried out in about 1 min for an 8000 atom target, while 1.5×10^4 ion scattering runs (two dimensional, periodic target based on the binary collision approximation) can be carried out in the same time. Although individual simulations run quite quickly, the time problem arises because of the need to gather adequate statistics. Sputtering simulations require several hundreds of runs, while ion scattering (ISS) simulations need hundreds of thousands or millions of runs. Serious simulations of atomic collision phenomena therefore make heavy demands on current-day personal computers, and running simulations on fast computers can have a big impact on productivity. Setting up the parameters of a simulation is an iterative process. Much time can be saved if you

avoid doing computations that will subsequently be discarded because an error came to light.

If you plan to buy a computer for running simulations, try to test its performance beforehand. In the author's experience, the CPU speed alone is not a good guide to performance, and you may lose up to ~30% of the processing potential due to a bad memory or chipset configuration. Make sure you have at least 1 GB of memory and 100 GB of disk space to satisfy invariably increasing ambitions.

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 simulation 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 engine, *Kalypso*, can coexist happily with other (well-behaved) Windows programs.¹ The best time-management strategy is to run long simulation projects as background batch jobs. You can even pause or stop 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. Except for multiple-impact simulations, *Kalypso* jobs can also be stopped completely then restarted later if necessary.²

Long calculations can be 'parallelised' by running parts of them on different machines. For example, in a 1000-run simulation, runs 1-500 can be executed on machine #1, and runs 501-1000 can be executed on machine #2. The resulting output data (1.snk, 2.snk dynamics files) can be combined using the Windows command line **binary file** copy command:

```
copy /b 1.snk + 2.snk combined.snk.
```

Kalypso includes an option to start the calculations from the middle of the Impact file, so you should not edit the Impact file by hand when restarting a simulation.

1.5. *Kalypso* output

The output from a successful simulation is the **dynamics file** (*.SNK), which consists of a sequence of binary records. Each record is associated with a specific particle at a certain time during the simulation (often, but not necessarily, at the end of the simulation). The user can specify the conditions under which output data will be produced.

Each record stored in the dynamics file includes the following variables (see section 9.2 for detailed explanations of the fields):

- ti**, time elapsed since the start of the simulation run (units: s);
- rw** (row number), a particle index that is based upon the position of the particle in the projectile or Target files;
- rn** (run number), a run index indicating to which run the data refer;
- ui** (unique identifier), another index that is incremented whenever the output routine of the program is called; if you are combining data from different simulations, you should initialise it manually so that values don't overlap (*Kalypso* options);

¹ Certain Webb browsers and firewalls are not well-behaved, in the sense that they can deplete system resources if used continuously, leading eventually to a 'hung' system.

² See section 9.6 for a description of the procedure involved in restarting a multiple-impact simulation.

rx, ry, rz, vx, vy, vz: particle position (units: m) and velocity (units: m s⁻¹) components;
ms: particle mass (units: kg);
fl: particle options variable (flags) at time of output;
bx, by: these ‘variant’ fields store either the projectile impact parameter, or the projectile incident angles, or other variables, depending on options selected by the user (in the Run file: see Chapter 4 of the User Guide). Units of these fields depend on the type of output selected. New options will be added over time.

The program *Winnow* can manipulate these variables directly in order to produce meaningful information (e.g. energy spectra) that is customised to the user’s requirements. Alternatively, for users who have specialised needs, the output file data can be dumped to a text file for reading and processing with other software.

The dynamics file consists of ‘TSnkRec’ binary records (56 bytes = 10×4 + 4×4 per record) whose Pascal/Delphi definitions are:

```
TSnkRec = record
  rx,ry,rz,vx,vy,vz,ms,ti,bx,by: single; // 4 byte
  rw,rn,fl,ui: integer;                // 4 byte
end;
```

An equivalent structure declaration in C would be:

```
typedef struct
{
  float rx,ry,rz,vx,vy,vz,ms,ti,bx,by; // 4 byte
  long  rw,rn,fl,ui;                  // 4 byte
} TSnkRec;
```

You can quickly count the number of records in a dynamics file by dividing the file size in bytes by 56.¹ Dynamics files originating from different simulations can be combined using DOS binary file copy commands such as:

```
copy/b dv1.snk+dv2.snk+dv3.snk combined.snk
```

This appends *dv3.snk* and *dv2.snk* to *dv1.snk* in a new file called *combined.snk*. This operation is useful if, for some reason, you had to run the simulation in more than one piece.

References for Chapter 1

-
- [1] M.A. Karolewski, *Kalypso: A Software Package for Molecular Dynamics Simulation of Atomic Collisions at Surfaces*, Nucl. Instr. Meth. B 230 (2005) 402.
 - [2] F. Ercolessi, *A Molecular Dynamics Primer*, URL: <http://www.fisica.uniud.it/~ercolessi/md/>

¹ I have recently discovered by chance that the maximum size allowed in WinXP for a SNK file is about 6 GB (gigabytes). After this, no more output is stored, although the simulation continues to run. This seems to be due to an integer overflow error in the compiler or (Windows) operating system which cannot be fixed at this time.

-
- [3] Modelling in Materials Science, L.V. Zhigilei, <http://www.people.virginia.edu/~lz2n/mse524>.
- [4] R. Smith, M. Jakas, D. Ashworth, B. Owen, M. Bowyer, I. Chakarov and R. Webb, Atomic and Ion Collisions in Solids and at Surfaces, Cambridge University Press, 1997.
- [5] Wolfgang Eckstein, Computer Simulation of Ion-Solid Interactions, Springer-Verlag, Berlin, 1991.
- [6] E.S. Mashkova and V.A. Molchanov, Medium-Energy Ion Reflection from Solids, North-Holland, Amsterdam, 1985.
- [7] D.E. Harrison Jr., Sputtering Models - A Synoptic Review, Radiation Effects, 70 (1983) 1-64.
- [8] R. Smith and D.E. Harrison Jr., Algorithms for Molecular Dynamics Simulations of keV Particle Bombardment, Computers in Physics, 3 (1989) 68-73.
- [9] M.T. Robinson, Theoretical Aspects of Monocrystal Sputtering, in Sputtering by Particle Bombardment I, (pp. 73-144) , R. Behrisch ed. (Springer-Verlag, Berlin, 1981).
- [10] H. Niehus, W. Heiland and E. Taglauer, Low-energy Ion Scattering at Surfaces, Surface Science Reports 17 (1993) 213-303.
- [11] J.W. Rabalais (ed.), Low-Energy Ion-Surface Interactions, (Wiley, Chichester, 1994).
- [12] M. Nastasi, J.W. Mayer, J.K. Hirvonen, Ion-Solid Interactions: Fundamentals and Applications, (Cambridge Univ. Press, Cambridge 1996).
- [13] H.S. Tan, M.A. Karolewski, Determination of Shadow Cone Dimensions for 0.5-5 keV Ar, Kr and Xe Projectiles, Nucl. Instr.Meth. B 73 (1993) 163.

2. INTERATOMIC POTENTIALS

2.1. The composite potential

The atomic interactions modelled by *Kalypso* are described by means of **composite potentials**. These potentials consist of a repulsive screened Coulombic potential (V_c) at short internuclear distances, which is joined to an attractive many-body tight-binding (TB) potential (V_a) at internuclear distances that are comparable to chemical bond lengths.

The scheme that is used for constructing composite potentials in *Kalypso* 2 is different from that used in *Kalypso* 1. In *Kalypso* 1, an interpolation function was fitted between the ZBL potential and a pair potential that represented the *effective* two-body potential in the bulk environment. One difficulty with this approach is that it produces a small energy leakage that conflicts with the energy book-keeping scheme. In order to track energy accurately, a different method of joining the core and attractive potentials has been implemented for *Kalypso* 2.

In *Kalypso* 2, the repulsive and attractive potentials are joined smoothly at short distances by means of a switching (i.e. interpolation) function called here the **core switching function**. The attractive TB potential is also terminated smoothly at the cut-off distance by means of another function, the **cut-off switching function**. The potentials and the switching functions are described in detail in later sections of this chapter.

2.2. Cut-off distance

For reasons of computational economy, the terms in the potentials are evaluated only up to the cut-off distance (r_c). However, it is not correct to say that the range of influence of the potential is r_c . For a many-body potential, the force acting on an atom i depends on the number of atoms that coordinate both i and its neighbours. Any atom that interacts with a neighbour of i can thus affect the force acting upon i . Thus the range of *influence* of the potential is $2r_c$.

For example, consider an atom i that is located in the 3rd layer of a fcc (100) surface. If $r_c = a$ (the fcc lattice constant), the coordination shell of atom i is bulk-like (i.e. 12 nearest neighbours and 6 next-nearest neighbours). However, the coordination shells of the neighbours of i that lie between i and the surface are not bulk-like. Therefore, a net force is exerted on i if the atoms are arranged in the ideal fcc lattice configuration. For Cu, the magnitude of this force is about 0.02 eV Å⁻¹.

Baskes et al. have determined the range of forces in fcc and bcc Al using *ab initio* calculations [1]. Most of the force (~80%) on atoms adjacent to monovacancy defect sites comes from nearest neighbours.

It is important to appreciate that the parameters of an attractive interatomic potential are fitted on the assumption of a specific cut-off distance. For example, a potential might be fitted using a cut-off distance that excludes the third and higher coordination shells. In that case, any simulation that uses the potential should also have a cut-off distance that excludes the third and higher coordination shells. In other words, r_c should lie **somewhere between the second and third coordination shells**. You have some freedom to choose exactly where, since this choice will not affect the material parameters of the ideal crystal. Some authors will place r_c in the middle of the shells, but for reasons discussed in Sect. 2.6, a better choice for **fcc metals** is to place r_c

about 0.1 Å below the third coordination shell. If r_c exactly coincides with the third coordination shell, some atoms from that shell will lie within the range of influence of the potential at finite temperature due to thermal displacements (~ 0.1 Å), and surface relaxation parameters might also be affected. For **bcc metals**, placing r_c in the middle of the shells may improve target stability (e.g. to avoid bcc→fcc recrystallisation in the target during the course of the simulation). The optimum choice of r_c may require careful thought for a bimetallic system involving metals that have different lattice constants or different lattice structures in their elemental states (no general solution to this dilemma can be offered - let trial and error be your guide).

2.3. Screened Coulombic potential

The short range potential used by *Kalypso* is classified as a screened Coulombic potential. There are several variants in common use, which are named after their developers: the (Ziegler-Biersack-Littmark) ZBL or ‘Universal’ potential [2], the Molière potential [3] and the Bohr potential [4]. The analytic forms of these screened Coulombic potentials for interacting atoms of atomic number Z_1 , Z_2 , respectively, can be expressed as:

$$V(r_{ij}) = \frac{Z_1 Z_2 e^2}{4\pi\epsilon_0 r_{ij}} \sum_{k=1}^N c_k \exp(-b_k r_{ij} / a), \quad (2.1)$$

where the number of terms, N , is 3 for Molière potential, 4 for the ZBL potential, and 1 for the Bohr potential. The parameters c_k , b_k and a (the screening length) are defined differently for the various potentials, while r_{ij} is the internuclear separation of atoms i and j .

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

$$a = 0.4685 / (Z_1^{0.5} + Z_2^{0.5})^{2/3} \text{ (Firsov form)} \quad (2.2a)$$

$$a = 0.4685 / (Z_1^{2/3} + Z_2^{2/3})^{0.5} \text{ (Lindhard form)} \quad (2.2b)$$

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 used to scale the screening length parameter in screened Coulombic potentials (normally by a factor < 1.0), particularly the Molière potential. A typical value would be 0.80 ± 0.05 (usually chosen by fits to experimental data, e.g. impact collision ion scattering spectroscopy).

The screening length (a , in Å) for the ZBL potential is defined as:

$$a = 0.4685 / (Z_1^{0.23} + Z_2^{0.23}) \quad (2.2c)$$

For the ZBL potential, a value of 1.0 is normally used for the screening length correction, unless there is reason to do otherwise (see below). The standard Molière and ZBL screening lengths can be calculated automatically using *Spider*.

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 [5]. Critical evaluation of the ZBL potential (see below) suggests that it is too hard (too repulsive). You should be wary of changing screening lengths arbitrarily, because unrealistic values could conceivably undermine the credibility of the simulations. 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 fixed form of the ZBL screening length may be regarded as an advantage or a disadvantage, depending on how highly you rate the potential.

For the Ar-Cu system the ZBL potential fits closely the *ab initio* ArCu⁺ potential calculated by Broomfield et al. [6] (the optimum screening length correction over the range $r = 0.44$ - 1.40 Å is 0.999, and over the range $r = 0.6$ - 1.44 Å is 0.959). Kawata et al. [7] compared the ZBL potential to density functional theory (DFT) potentials and also found good agreement for Ar-Cu but quite poor agreement (~50% error) for Ar-Al. Nordlund et al. [8] 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). The author has recently used density functional theory to calculate repulsive potentials for NeCu, ArCu, KrCu, Cu₂, Ni₂, CuNi, ArNi [9]. For energies below 5-10 keV, the ZBL potential is invariably found to be too repulsive. This impacts sputter yield predictions, for example, at the 10% level.

Gamma-ray induced Doppler broadening (GRID) data obtained for Ni-Ni, Fe-Fe and Cr-Cr collisions have been used to fit the exponent in the screening length definition for the ZBL potential (keeping other parameters fixed at the standard values) [10]. 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 significant corrections to the ZBL potential. The Ni-Ni data agree quite well with the *ab initio* predictions in ref. [9].

The conclusion must be that in any serious study of ion-surface collisions one should consider experimenting a little with the standard ZBL potential, to see what effect a softer potential has on the simulation predictions. However, due to time constraints this is rarely done.

The screened Coulombic potential is used by *Kalypso* for all projectile-target atom separations when the projectile is an 'inert' species (such as Ar). In this case, no switching function is used to truncate the potential at the cut-off distance (r_c). Instead, the entire potential is shifted downwards by an amount $V(r_c)$, so that the discontinuity in the potential function at r_c is removed; that is, the screened Coulombic potential function $V(r)$ is replaced by $V(r) - V(r_c)$. The primary purpose of the adjustment is to facilitate energy book-keeping. No adjustment to the discontinuity in the derivative of the potential at r_c is made. The adjustments involved are typically small. For the Ar-Cu system, with $r_c = 4$ Å, $V(r_c) < 0.01$ eV, with $r_c = 2.56$ Å, $V(r_c) < 0.3$ eV. Small values of the cut-off distance might be used for an ISS simulation. With $r_c = 1.8$ Å, $V(r_c) = 6.9$ eV for Ar-Cu and $V(r_c) = 1.7$ eV for He-Cu.

2.4. Tight-Binding (TB) potentials for metals

The attractive potentials used by *Kalypso* are **tight-binding** (TB) potentials based on exponential functions [11] (also known as Gupta potentials [12]). These potentials can be regarded as particular examples of Finnis-Sinclair (FS) potentials [13], and are closely related to the embedded atom method (EAM) potentials [14] which have their

conceptual roots in effective medium theory [15]. The potentials are many-body potentials in the sense that the energy of a system cannot be decomposed into pairwise contributions. The potentials are used with a cut-off distance (typically above the second coordination shell for fcc metals). Functionally, the TB potentials are closely related to the Morse pair potential [16]. The attractive part of the TB potential is a non-linear (square root) function of a sum of pairwise Morse-like terms. What this means is that cohesive energy is a non-linear function of the coordination number for a TB potential.

The TB potentials are expected to work best for fcc transition metals with filled or nearly filled *d*-bands. 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 necessarily a critical problem for sputtering simulations, 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 of ~ 1 ps. However, you may need to increase the size of your target, since it will invariably 'reconstruct' from its edges inwards as the simulation progresses. 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 author has fitted TB potentials for 26 elemental fcc and bcc metals in ref. [17]. The potential parameters based on that paper are collated in Tables 2.1 and 2.2 of this document respectively. In the same paper (paper 1 in the `/docs/papers` directory of the *Kalypso* distribution) you can find a discussion of the properties of these 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. [17].

The assumptions underlying the tight-binding model of metallic cohesion in the second-moment approximation are reviewed by Clari and Rosato [18]. Within this approximation, the band energy of the system is proportional to the square root of the second moment of the density of states. In *Kalypso*, the system potential energy for an element, U_s , is expressed 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:

$$U_s = \sum_i (E_i^R + E_i^B), \quad (2.3)$$

where E_i^R is a repulsive pair potential:

$$E_i^R = \frac{1}{2} \sum_{j \neq i} U_{ij}(r_{ij}), \quad (2.4)$$

$$U_{ij}(r_{ij}) = A' \exp(-p(r_{ij}/r_0 - 1)),^1 \quad (2.5)$$

¹ *Kalypso* actually permits a more general form for the repulsive potential:

$$U_{ij}(r_{ij}) = A \left\{ \exp(-p(r_{ij}/r_0 - 1)) - b \exp(-2q(r_{ij}/r_0 - 1)) \right\}.$$

If $b < 0$ and $\xi = 0$, a Morse-like potential results (see Section 2.10). However, most users will set $b = 0.0$, leading to Eq. 2.5. If you have reason to do otherwise, note the following. If $b \neq 0$ for the

and E_i^B represents the cohesive band energy term:

$$E_i^B = - \left(\sum_{j \neq i} \phi(r_{ij}) \right)^{1/2}, \quad (2.6)$$

$$\phi(r_{ij}) = \xi^2 \exp(-2q(r_{ij}/r_0 - 1)) \quad (2.7)$$

In Eqs. 2.3-2.7, r_{ij} is the separation between atoms i and j , and A' , ξ , p , q , r_0 are adjustable parameters governing the interaction between those atoms. Instead of Eq. 2.4, most TB potentials reported in the literature use a non-intuitive double summation convention for the repulsive functions, which differs from that used for other potentials:¹

$$E_i^R = \sum_{j \neq i} U_{ij}(r_{ij}); \quad U_{ij}(r_{ij}) = A \exp(-p(r_{ij}/r_0 - 1)) \quad (2.4b)$$

In order to reduce (increase?) confusion between these two conventions, *Kalypso* labels the parameters using the literature TB convention (Eq. 2.4b) but requires as inputs not the value A given in Eq. 2.4b but $2A$ ($= A'$ in Eq. 2.5). For example, Cleri and Rosato [18] find $A = 0.0855$ eV for Cu (5th neighbour cut-off). To use their potential in *Kalypso*, enter the parameter $2A = 0.17100$ eV. The parameter q is also input as $2q$.

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 , ξ , 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 (in practice, to about 0.02%, once the coefficients have been rounded off), 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 $E_c/3$.

heteronuclear (i, j) interaction, *Kalypso* **assumes** that the value of q for the same interaction is symmetric with respect to both species, i.e.: $q_{ij} = q_{ji}$.

¹ *Kalypso* conforms with the summation convention used by other potential types, including EAM potentials. Unfortunately, TB potentials (a special class of EAM potential) do not follow the same convention.

TB potentials for a number of fcc (Ni, Cu, Rh, Pd, Ag, Ir, Pt, Au, Al, Pb), bcc (V) and hcp metals (Ti, Zr, Co, Cd, Zn, Mg) and fcc alloys (Cu₃Au, Ni₃Al) have also been fitted by Cleri and Rosato (CR) [18]. These potentials are fitted using a cut-off above the fifth coordination shell, which will cause simulations to run much more slowly than those based on second-neighbour cut-offs. The CR paper (recommended reading) also has references to earlier parameterisations with a cut-off above the first shell. A set of TB potentials was fitted up to the fifth coordination shell for Cu, Ag and Au by Kallinteris et al. [19]. 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 [20]. López and Jellinek give a thorough discussion of the issues involved in fitting TB (Gupta-type) potentials [21]. Apart from the above compilations, there are several other TB potentials in the solid-state and surface literature (do a search through Phys. Rev. B, for example at URL: <http://prola.aps.org>). However, most of these potentials have ranges that are too long to be ideal for sputtering simulations (in the sense that they require longer simulation times).

Table 2.1. Parameters of tight-binding potentials for fcc metals [17]. The potentials should be cut off somewhere between the second and third neighbour distances ($\sqrt{2} r_0 < R_c < \sqrt{3} r_0$). The parameter $b = 0.0$ for all of these potentials.

Z	Element	$2A$ (eV)	r_0 (Å)	p	$2q$	ξ (eV)
13	Al	0.320456	2.8634	7.568129	5.491184	1.507384
20	Ca	0.098483	3.9471	11.21150	5.368150	0.684202
30	Sr	0.051467	4.3027	12.34058	3.620951	0.555708
28	Ni	0.112995	2.4918	14.08666	3.587386	1.400543
29	Cu	0.156525	2.5560	11.18320	4.639412	1.235524
45	Rh	0.217164	2.6901	14.13154	5.110908	1.977562
46	Ag	0.162518	2.889	11.55970	5.663298	1.108113
79	Au	0.386964	2.8838	10.43418	7.894370	1.758066
77	Ir	0.428119	2.7145	12.89860	6.908174	2.708201
82	Pb	0.170293	3.5003	10.06662	6.712517	0.869929
78	Pt	0.581228	2.7746	10.14231	7.575668	2.671486
46	Pd	0.244689	2.7511	11.32250	6.139402	1.519346
90	Th	0.239961	3.5951	9.834413	3.594461	2.093744

Table 2.2. Parameters of tight-binding potentials for bcc metals [17]. The potentials should be cut off somewhere between the third and fourth neighbour distances ($\sqrt{8/3} r_0 < R_c < \sqrt{11/3} r_0$). The parameter $b = 0.0$ for all of these potentials.

Z	Element	$2A$ (eV)	r_0 (Å)	p	$2q$	ξ (eV)
3	Li	0.097512	3.0391	6.367465	2.793757	0.572925
11	Na	0.070641	3.7158	7.853638	3.495437	0.408306
19	K	0.046017	4.6073	9.309300	3.228612	0.316985
37	Rb	0.058311	4.9363	8.153151	3.846991	0.323340
55	Cs	0.053956	5.3174	8.411957	3.886624	0.303616
56	Ba	0.079934	4.3466	10.18347	3.013993	0.616739
23	V	0.514348	2.6223	6.854340	4.377272	2.312561
41	Nb	0.909163	2.6033	5.270157	4.110446	3.630224
73	Ta	0.656266	2.8601	8.276389	4.474236	3.300764
24	Cr	0.081360	2.4981	13.18516	1.798556	1.101227
42	Mo	0.408696	2.7253	10.01545	4.102271	2.509715
74	W	0.498072	2.741	10.37148	3.983120	3.205477
26	Fe	0.236898	2.4824	10.76133	4.075708	1.541808

2.5. Tight binding potentials for bimetallic systems

The TB potential formalism can be extended to describe bimetallic systems [18]. Strictly, the fitting parameters for a bimetallic system cannot be deduced from those of the pure elements alone. However, an approximate combination rule [22] that has been used for 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):

$$\begin{aligned}\phi^{AB}(r_{ij}) &= [\phi^{AA}(r_{ij})\phi^{BB}(r_{ij})]^{1/2}, \\ U^{AB} &= [U^{AA}(r_{ij})U^{BB}(r_{ij})]^{1/2}\end{aligned}\quad (2.8)$$

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:

$$\begin{aligned}A_{AB} &= A_{BA} = \sqrt{A_{AA}A_{BB}}; \quad \xi_{AB} = \xi_{BA} = \sqrt{\xi_{AA}\xi_{BB}}; \\ p_{AB} &= p_{BA} = (p_{AA} + p_{BB})/2; \quad q_{AB} = q_{BA} = (q_{AA} + q_{BB})/2 \\ 2r_{0AB} &= (2p_{AB})/(p_{AA}/r_{0AA} + p_{BB}/r_{0BB}) + (2q_{AB})/(q_{AA}/r_{0AA} + q_{BB}/r_{0BB})\end{aligned}\quad (2.9)$$

The expression for r_{0AB} is complex (probably unnecessarily complex in view of the approximations involved) because there is no unique way to assign a value for this parameter. An average of two methods has been used in Eqs. 2.9, but other methods could be devised. For an ordered or substitutional alloy, the lattice constant may be known (or Vegard's rule, which suggests that alloy properties such as lattice constant can be estimated by linear interpolation of composition, can be assumed). Then it becomes possible to choose r_{0AB} to reproduce the desired lattice constant.¹

TB potentials have been fitted for a number of bimetallic systems in the literature, but these often have a fairly long range. A literature search (especially in Phys. Rev. B, NIM B, and Surf. Sci.) on *Gupta potential(s)*, or *tight-binding potential(s)* is the best way to locate previously fitted potentials for bimetallic systems. For example, Rohart et al. give parameters for CoAu [23], while Mottet et al. give parameters for PdCu [24].

The scheme shown in Eqs. 2.9 is based on the assumption that A - B interactions are energetically intermediate between A - A and B - B interactions. There are 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 are, however, cases where it breaks down completely. In such cases, a potential has to be derived by fitting to a representative database of structural and energetic data (typically obtained from *ab initio* calculations). A useful database of surface segregation energies for transition metal impurities in transition metal hosts is

¹ This entails calculating system energy as a function of lattice parameter, and locating the minimum.

found in ref. [25] and Annexe of this chapter. These data can be used to test the performance of potentials in bimetallic systems. Alternatively they can be tested using thermodynamic enthalpies of solution data [26]. The same data can also be used to fit potentials. Potential fitting is a specialised and tedious research activity which the casual user of *Kalypso* will want to avoid.¹

The TB potentials used in *Kalypso* are more flexible than those used in the literature because the conditions $q_{AB} = q_{BA}$ and $\xi_{AB} = \xi_{BA}$ are not enforced (that is, the parameters that define the attractive part of the potential do not have to be symmetric with respect to the interchange of the atoms).² Due to this flexibility (which can improve the accuracy and/or transferability of potentials in bimetallic systems), *Kalypso*'s TB potentials can also be described as EAM potentials [14] that have a square root embedding function. For reasons of computational efficiency, the pairwise potential parameters ($2A, p$) are required (by *Kalypso*) to be symmetric. It can be shown that this does not lead to any loss in the generality of the potentials (in the language of potential theory, this condition is equivalent to imposing a gauge transformation).

2.6. The switching functions

The switching functions, $S(r)$, used by *Kalypso* have the following properties: $S(r_1) = 1$, $S'(r_1) = 0$ and $S(r_2) = 0$, $S'(r_2) = 0$. Two parameters r_1 and r_2 (where $r_2 > r_1$) define the region of application of the switching functions. The values of r_1 and r_2 are chosen in a way that will not affect the material properties (e.g. cohesive energy) of the equilibrium solid. This requires that no atoms in the equilibrium target lattice should be located in a region of the potential that is modified by the switching function.

The core and cut-off switching functions have similar functional forms, with one adjustable scale parameter (a) that can be modified by the user (in order to improve the smoothness of the composite potential. In the core switching function region the total potential (V) is expressed as:

$$V(r) = V_c(r)S(r) + V_a(r)(1 - S(r)) \quad (2.9)$$

The corresponding force function is:

$$-F(r) = V'_c(r)S(r) + V_c(r)S'(r) + V'_a(r)(1 - S(r)) - V_a(r)S'(r) \quad (2.10)$$

In the cut-off switching function region:

$$V(r) = V_a(r)S(r) \quad (2.11)$$

$$-F(r) = V'_a(r)S(r) + V_a(r)S'(r) \quad (2.12)$$

¹ A Webb search for the words or phrase *fitting interatomic potentials* will provide many links.

² Exception: if $b \neq 0.0$ (rare), the q cross-terms must be symmetric, i.e. $q_{AB} = q_{BA}$. This restriction arises because of the way the potential calculations are implemented in *Kalypso*.

For the core switching function, r_2 *must* have a value smaller than the lattice nearest neighbour distance, while r_1 is chosen to optimise the fit with the screened Coulombic potential. For the cut-off switching function, r_2 *must* be equal to the cut-off distance, while r_1 *must* have a value greater than the outer coordination shell radius. The core switching function region and the cut-off switching function region must not overlap (*Kalypso* will flag error #E026 if this condition is violated). Apart from these restrictions, the switching function regions are chosen heuristically by the user.

For example, consider a Cu lattice with lattice parameter $a = 3.615 \text{ \AA}$ and a TB potential with cut-off between the second and third coordination shells (which requires a choice for r_c that satisfies: $3.615 \text{ \AA} < r_c < 4.427 \text{ \AA}$). In the Cu lattice, the nearest neighbour distance is 2.556 \AA . The core switching function will have $r_2 < 2.556 \text{ \AA}$, while the cut-off switching function will have $r_1 > 3.615 \text{ \AA}$ and $r_2 = r_c$.

Ideally, the lengths of the regions over which the switching functions are applied, $r_2 - r_1$, should be made as small as possible. However, if the length of the cut-off switching function region is too short, this will cause the potential to change rapidly, and may give rise to the appearance of an unphysical minimum in the force function, which is to be avoided wherever possible.

The force function should preferably not have a minimum in the switching function region (it should be smoothly inflected) but for third neighbour cut-off distances this can rarely be achieved using the TB potential functional form. Note that the force function should not oscillate to the point of changing sign, since this indicates the presence of a spurious potential well that can trap a moving particle permanently.

The switching function used by *Kalypso* is similar to one that was proposed by Bazant [27], apart from an adjustable scaling parameter, a :

$$S(r) = \exp\left(a(1-x^{-3})^{-1}\right) \text{ with } x = (r-r_1)/(r_2-r_1); \quad r_2 > r_1. \quad (2.13)$$

Unlike polynomial switching functions, Bazant's switching function does not produce spurious maxima and minima. It has two continuous derivatives at the inner cut-off (r_1), and is smooth at the outer cut-off:

$$S'(r) = dS/dr = -3ax^{-4}(1-x^{-3})^{-2}(r_2-r_1)\exp\left(a(1-x^{-3})^{-1}\right). \quad (2.14)$$

Fig. 2.1 plots the switching function for various values of a . Fig. 2.2 shows the composite potential and force functions for the Cu-Cu interaction in the *attractive* region of the potential. The potential was fitted to the first two coordination shells of the Cu lattice, which means that the cut-off switching function *must* be applied after the second shell, while the cut-off distance *must* be placed before the 3rd shell (about 0.1 \AA before the shell would be optimal, in order to eliminate interactions with the 3rd shell that could arise from vibrational displacements and surface relaxation effects).

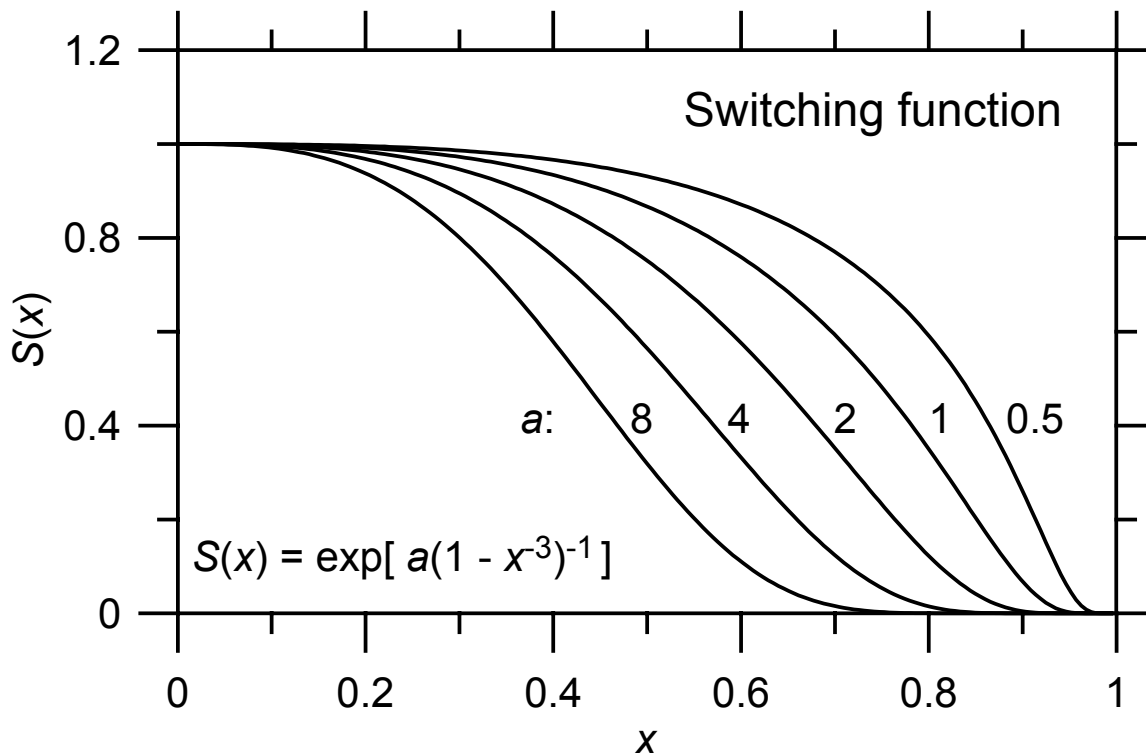


Fig. 2.1. Plots of the switching function, using different values of the scaling parameter a .

The imposition of a cut-off just before the 3rd coordination shell produces a large ‘bulge’ in the force function in the switching potential region. The cause of this ‘bulge’ can be understood when it is noted that any cut-off procedure must conserve the area under the force curve (this area represents the work involved in moving from the zero potential configuration to the configuration at r_1). Therefore, the shorter the switching function region, the more distinct will be the ‘bulge’ in the curve. Note the smooth, symmetric shape of the ‘bulge’ region in Fig. 2.2. A poorly shaped switching function (e.g. a polynomial) will produce much sharper changes of gradient (and possibly oscillations). Such behaviour is to be avoided.

The amplitude of the ‘bulge’ represents the accelerating force.¹ As Fig. 2.2 shows, the use of a switching function produces a large force just inside the cut-off boundary, which in principle is an undesirable artefact. The ‘bulge’ cannot be eliminated by a change of switching function, although its shape can be modified (the ‘bulge’ shown in the figure represents a nearly optimal adjustment of the switching function parameter, a). The use of a potential with a larger cut-off distance will reduce the ‘bulge’, but this is normally not convenient for reasons of computational efficiency.

Fig. 2.3 shows the force function in the *core* switching function region of the Cu-Cu potential. In this example the force curve exhibits a ‘bulge’ that produces an inflection rather than an extremum in the curve. There is more freedom in fitting the switching function in this region than in the cut-off region (the constraints were discussed above). By increasing the width of the region over which the switching function is applied, the ‘bulge’ could be reduced to any desired level. It is difficult to give guidelines as to the optimum procedure for setting up the switching function in this

¹ If this puzzles you, consider a gravitational analogy: one approaching particle drops from a cliff, while another rolls the same vertical distance down a slope.

region. It is probably a good idea to avoid applying the switching function in the attractive part of the TB potential ($V < 0$), where the screened Coulombic potential is known to be incorrect. Typically, one would aim to eliminate the maximum or minimum in the ‘bulge’ in favour of an inflection or smooth change of gradient, while keeping the switching function region width at a reasonable size. (In other words, merge the screened Coulombic and many-body potentials only to the extent that is necessary to achieve a chemically and aesthetically agreeable result.)

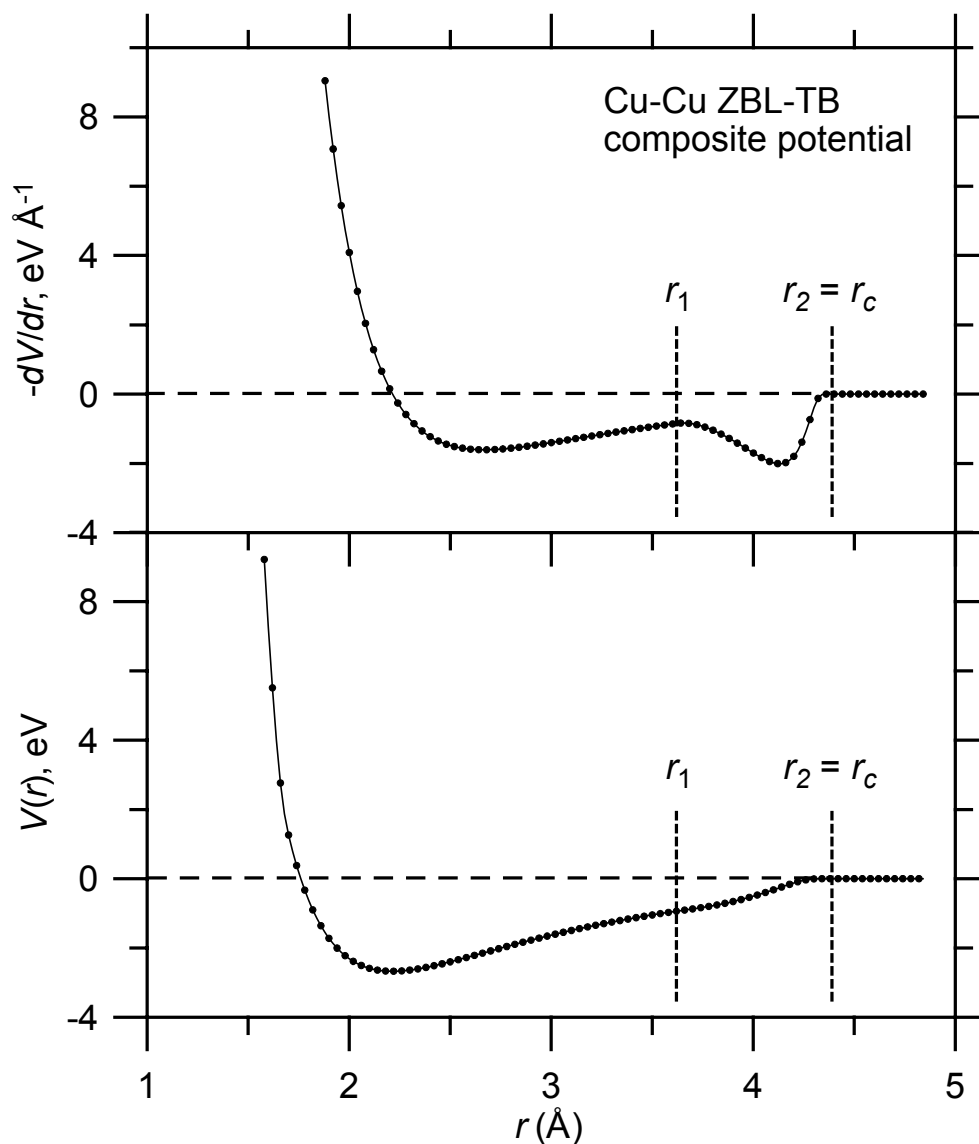


Fig. 2.2. Composite ZBL-TB potential that represents the interaction between two Cu atoms in the attractive region. The cut-off switching function region (between r_1 and r_2 , or 3.62 - 4.39 Å) is indicated ($a = 4$ in this example).

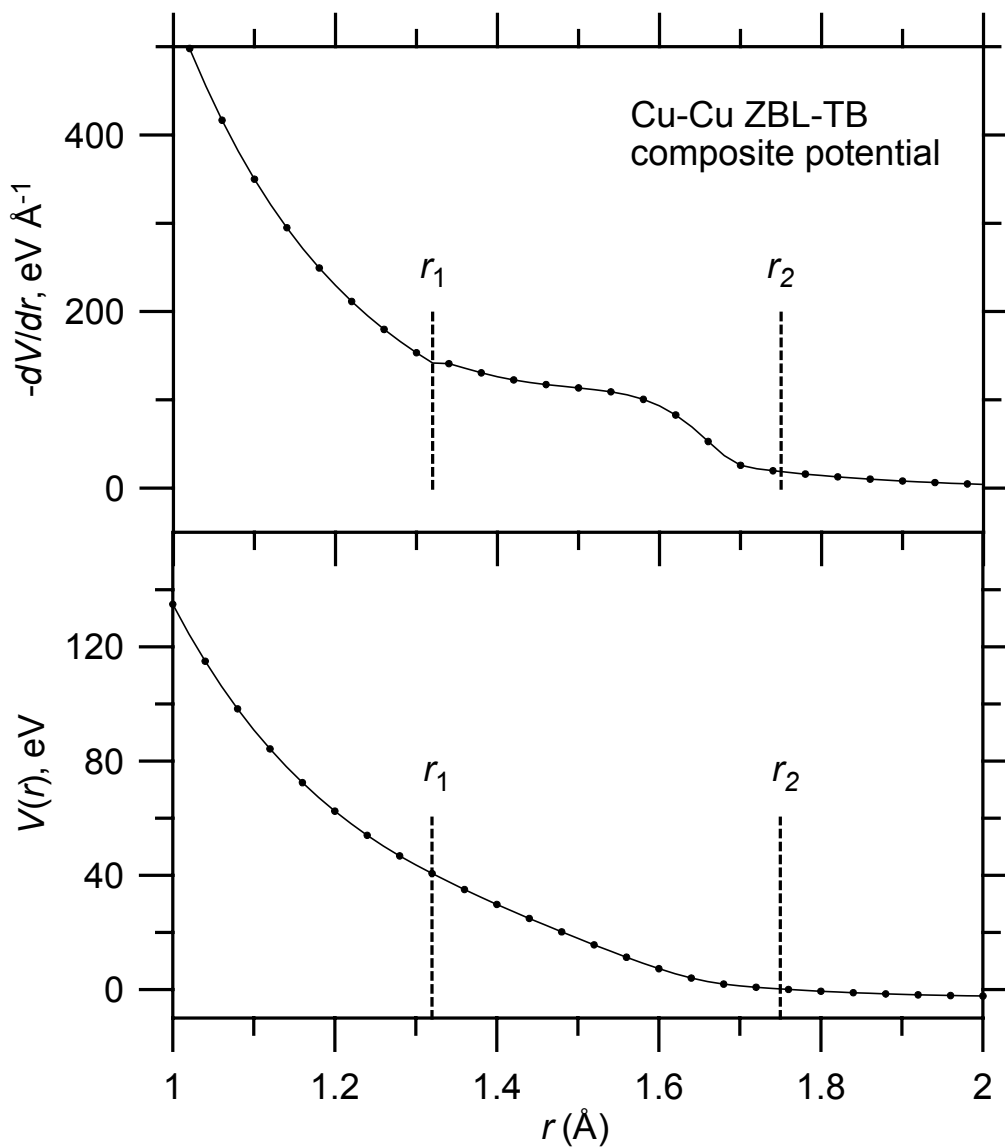


Fig. 2.3. Composite ZBL-TB potential that represents the interaction between two Cu atoms in the repulsive region. The core switching function region (between r_1 and r_2 , or 1.32 – 1.75 Å) is indicated ($a = 3$ in this example).

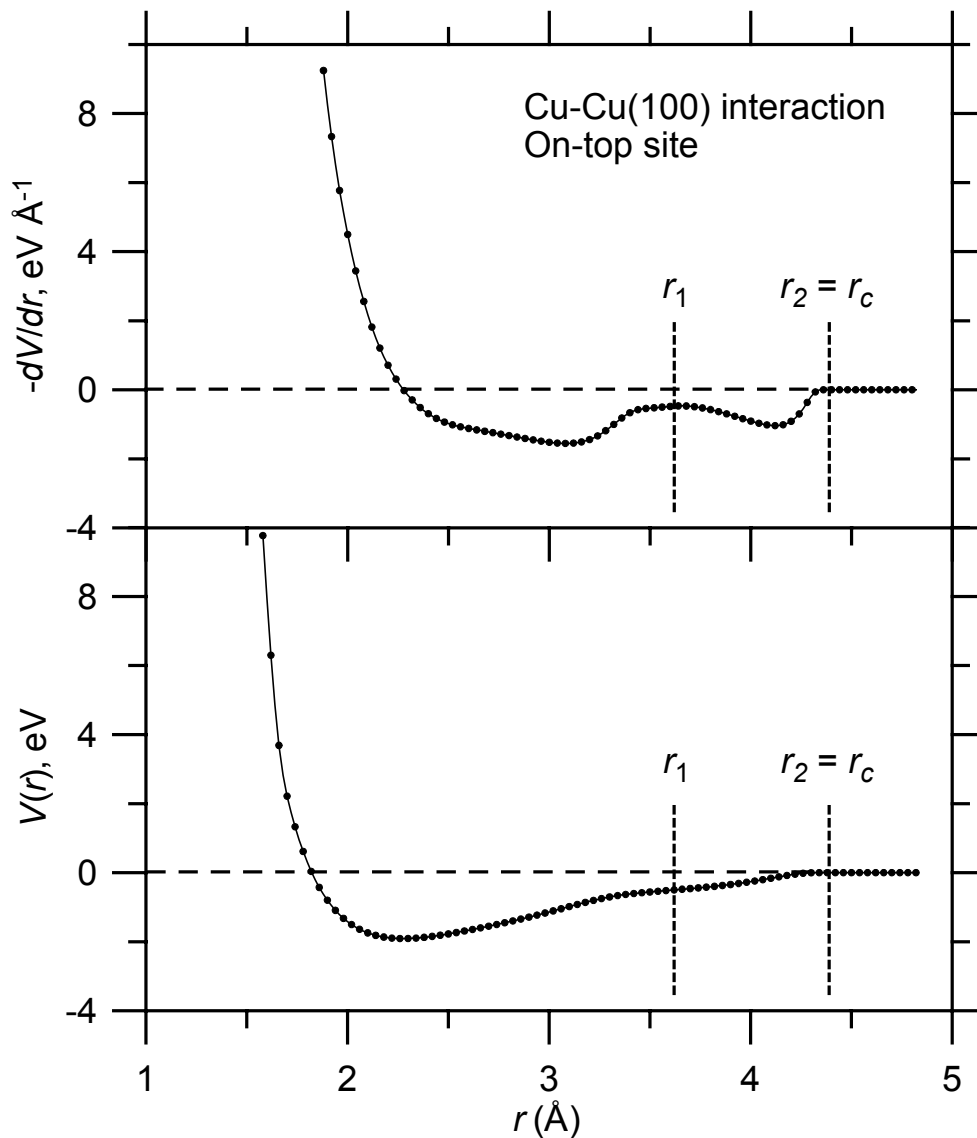


Fig. 2.4. Attractive region of composite ZBL-TB potential for the interaction of a Cu atom with a Cu(100) surface as it approaches an on-top site. Note: r represents the vertical distance of the Cu atom from the Cu(100) surface layer.

Figs. 2.4 and 2.5 show the potential and force functions for a Cu atom approaching a Cu(100) surface. In these figures the potential wells are shallower than for the diatomic case because the surface atom bonds are already highly coordinated. The force curves have a 'lumpy' appearance that is due to simultaneous interactions with two coordination shells (each contributing one of the 'bulges').

The current practice throughout the literature is to set up the interpolating functions using potential curves for diatomic molecules. This has the virtue of simplicity, but it is clear from Figs. 2.4. and 2.5 that the shapes of the resulting force functions may look quite different when the potential is applied to a solid state environment: (a) because of cut-off effects; (b) because of the inherent environmental dependence of many body potentials. As a rule, the greater the bond order (N) of any atom, the weaker its interaction with any specific atom. This is because the repulsive pairwise

part of the potential increases in proportion to N , while the attractive many-body part increases (more slowly) in proportion to $N^{1/2}$.

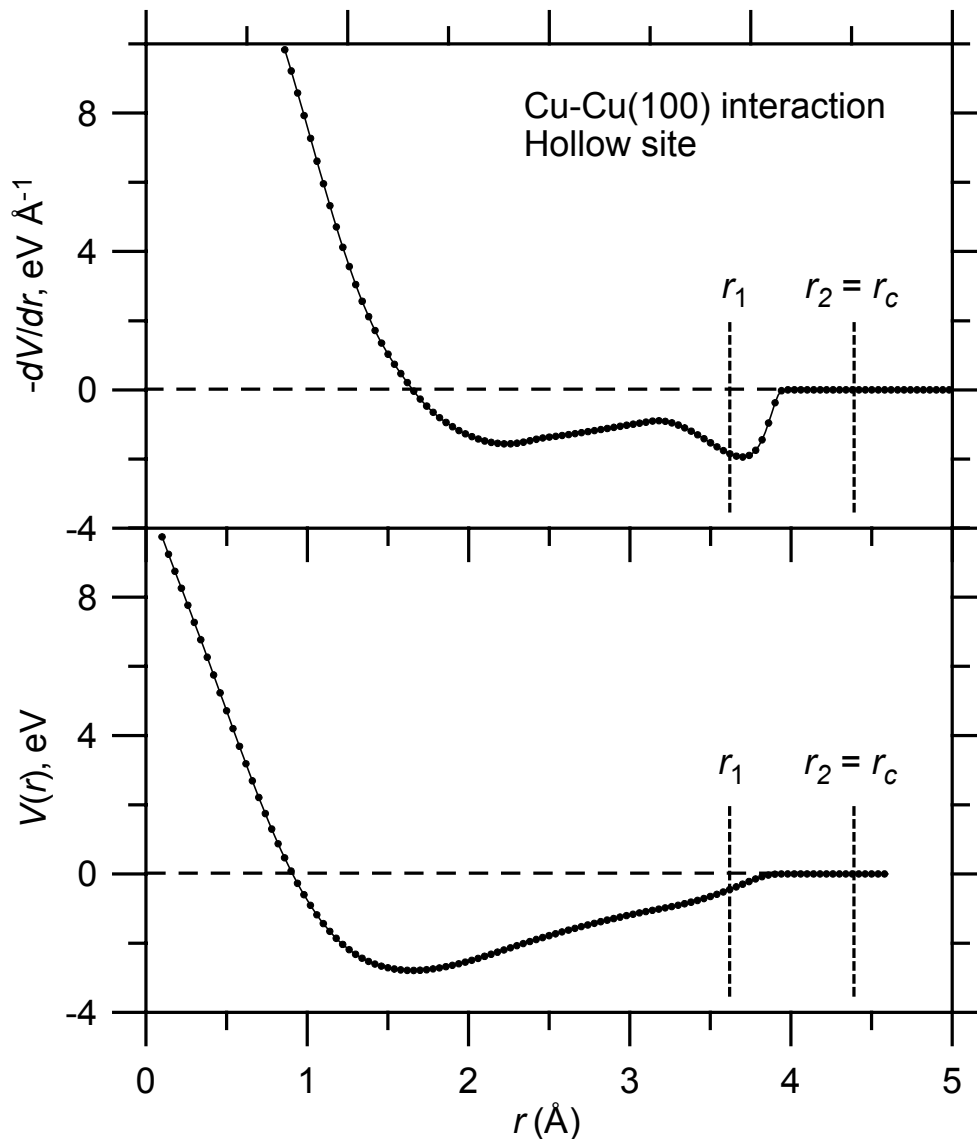


Fig. 2.5. Attractive region of composite ZBL-TB potential for the interaction of a Cu atom with a Cu(100) surface as it approaches a hollow site. Note: r represents the vertical distance of the Cu atom from the Cu(100) surface layer (the lattice atom is located ~ 1.81 Å below the surface)..

2.7. Calculations with tight-binding potentials

2.7.1. Cohesive energy

A calculation of the cohesive energy is a useful check of the parameters of a TB potential. From the parameters given in Table 2.1, we will calculate the predicted cohesive energy (E_c) of Ag.

Note that a fcc metal has 12 neighbours at a distance r_0 , and 6 neighbours at a distance $\sqrt{2}r_0$ (see Table 2.3). 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.

From Eqs. 2.4, 2.5, the repulsive part of the potential energy of the i th bulk atom is:

$$E_i^R = \frac{1}{2} \sum_{j \neq i} U_{ij}(r_{ij}) = A \sum_{j \neq i} \exp(-p(r_{ij}/r_0 - 1)) = A \left[12 + 6 \exp(-p(\sqrt{2} - 1)) \right] \quad (2.15)$$

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

$$\begin{aligned} E_i^B &= - \left(\sum_{j \neq i} \phi(r_{ij}) \right)^{1/2} = -\xi \left(\sum_{j \neq i} \exp(-2q[r_{ij}/r_0 - 1]) \right)^{1/2} \\ &= -\xi \left(12 + \exp(-2q[\sqrt{2} - 1]) \right)^{1/2} \end{aligned} \quad (2.16)$$

You should obtain: $E_i^R = 0.9785$, $E_i^B = -3.9294$, $E_c = -2.9509$ (remember that Table 2.1 tabulates $2A$ and $2q$, not A and q). The E_c value used for the fit was 2.95 eV; the difference, 0.0009 eV, represents the fitting and round-off error.

Whenever you use a new potential, you should check its properties using this type of calculation. Bear in mind that published potential parameters (e.g. Tables 2.1 and 2.2) are normally only quoted to 4 or 5 significant figures in the literature, and that the fitting data are subject to experimental uncertainty (elastic constants, vacancy formation energies: 10%; cohesive energies: 0.5%).

Table 2.4 compares calculated values of E_c for the fcc metals with the experimental values. The two sets of values agree to within 1 meV or better. The calculated cohesive energy of -3.48956 eV for Cu in Table 2.4 can be compared to the value of 3.49009 eV that is obtained when the potential parameters are not rounded off. The equilibrium lattice parameter for Cu is changed by less than 10^{-5} Å after rounding off the parameters.

Table 2.3. List of coordination shell radii for a fcc crystal (expressed in terms of the lattice parameter, a).

Shell (s)	Shell radius (R_s)	No. of atoms
1	$a/\sqrt{2}$	12
2	a	6
3	$a\sqrt{1.5}$	24
4	$a\sqrt{2}$	12
5	$a\sqrt{2.5}$	24

Table 2.4. Comparison of experimental (expt.) cohesive energies for fcc metals with those calculated (calc.) using the (rounded-off) TB potential parameters listed in Table 2.1.

	E_c (expt.)	E_c (calc.)		E_c (expt.)	E_c (calc.)		E_c (expt.)	E_c (calc.)
Al	-3.39	-3.39014	Rh	-5.75	-5.74871	Au	-3.81	-3.81042
Ca	-1.84	-1.84018	Pd	-3.89	-3.89113	Pb	-2.03	-2.03069
Ni	-4.44	-4.43959	Ag	-2.95	-2.95095	Th	-6.20	-6.19872
Cu	-3.49	-3.48956	Ir	-6.94	-6.93931			
Sr	-1.72	-1.72023	Pt	-5.84	-5.84084			

2.7.2. Other properties

Simple formulae for the elastic constants and vacancy formation energy (for fcc metals only) have been catalogued by Paidar et al. [20] for TB potentials that are cut off above the second neighbour distance. If you want to use these formulae with *Kalypso*'s input parameters, you must replace the parameter A in the Paidar et al. formulae by $A/2$, as discussed in section 2.6.

The condition for lattice stability is that the derivative of the lattice energy must be zero for the equilibrium value of the lattice parameter. This produces the following formula [20], which applies to fcc TB potentials fitted to the first *two* coordination shells:

$$\xi = (ARp/2q)(12 + 6Q)^{1/2} \quad (2.17)$$

where:

$$P = \exp\left(-p(\sqrt{2} - 1)\right), \quad Q = \exp\left(-2q(\sqrt{2} - 1)\right), \quad \text{and} \quad R = (\sqrt{2} + P)/(\sqrt{2} + Q) \quad (2.18)$$

Eq. 2.17 provides a simple check that the potential parameters are consistent with the lattice parameter. Table 2.5 applies it to the TB potentials listed in Table 2.1. The values agree to the round-off precision of the potential parameters, i.e. to 4 significant figures.

Table 2.5. Comparison of fitted (fit) values of the TB potential parameter ξ for fcc metals with the ξ values calculated using Eq. 2.17 (test).

	ξ (eV) (fit)	ξ (eV) (test)		ξ (eV) (fit)	ξ (eV) (test)
Al	1.5074	1.507175	Ag	1.1081	1.107414
Ca	0.6842	0.683572	Ir	2.7082	2.708866
Ni	1.4005	1.400908	Pt	2.6715	2.671474
Cu	1.2355	1.236041	Au	1.7581	1.758203
Sr	0.5557	0.55494	Pb	0.8699	0.869481
Rh	1.9776	1.977997	Th	2.0937	2.094313
Pd	1.5193	1.518974			

2.8. Simulations with tight-binding potentials

The centrosymmetric functional form of TB potentials (i.e. absence of explicit angular dependent terms) favours the formation of close-packed structures (fcc, hcp).

This is not absolutely guaranteed, because it is possible to choose the cut-off distance in a manner that favours other structures (e.g. bcc). The switching function complicates analysis of the general case.

Practical experience suggests that crystallites of all structural types have a tendency to reconstruct at the edges on a timescale of about 1 ps, especially when disturbed. An extreme case would be the bcc to fcc transition of a Mo crystallite, which is clearly unrealistic. Close-packed crystals may reconstruct at edge faces or crystallite corners, but the bulk crystal should remain stable almost indefinitely.

Bulk recrystallisation can be identified by visual inspection of the simulation as it runs. Recrystallisation commences at the target edges and moves towards the centre of the target. Various strategies are possible for dealing with recrystallisation: increasing the size of the target; reducing the drifting tendencies of edge atoms (by holding them to their lattice sites with harmonic forces, or by increasing their masses). The key point is that the timescale of the process that you are studying (e.g. sputtering, ~ 1 ps or ion scattering, ~ 20 fs) should be less than the time required for significant structural transformation.

Surface relaxation is invariably observed if simulation targets are based upon ideal (bulk) lattice structures. For many problems this small relaxation, Δd , is inconsequential (the 1st-2nd layer distance relaxation $\Delta d_{12} < 0.05$ Å), but you may want to characterise it (by plotting potential energy versus surface plane position) and make adjustments to your target accordingly. Normally only the first one or two layer positions on the active (bombarded) crystallite face need adjustment (the other exposed faces will adjust quickly as the simulation proceeds). A compilation of experimental and predicted surface relaxations (for Cu, Ag, Au, Ni, Pd, Pt, Al and Pb) may be found in ref. [28].

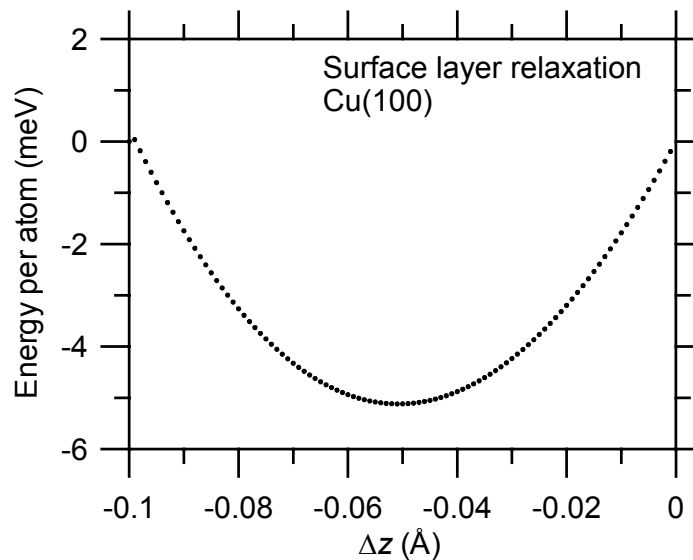


Fig. 2.6. Energetics of surface relaxation for Cu(100).

In Fig. 2.6, which refers to Cu(100), the potential energy per surface atom (relative to the unrelaxed system) is plotted as a function of the displacement (Δz) of the surface layer from its ideal location ($z = 0.0000$ Å). For the calculation shown in Fig. 2.6, the second layer of the crystal has already been placed at its relaxed position (for which $\Delta d_{23} = -0.013$ Å). The location of the energy minimum at a first layer displacement $\Delta z = -0.051$ Å thus implies $\Delta d_{12} = -0.038 \pm 0.001$ Å (referenced to the

ideal lattice). In the ideal lattice, the first and second Cu layers are located at 0.0000 and -1.8074 Å. After relaxation, these values are changed to -0.038 and -1.8204 Å respectively. Experimental estimates place Δd_{12} in the range -(0.02-0.04) Å [28]. The relaxations predicted by TB (and EAM potentials) potentials for Δd_{12} are typically quite accurate, but predictions for Δd_{23} and deeper layers normally have the wrong sign. This reflects a fundamental limitation of the TB functional form. Calculations of equilibrium interlayer spacings using large slabs can be influenced by edge effects, which tend to draw the slabs closer together. Unless this artefact is taken into account, the predicted equilibrium interlayer spacings are probably not accurate to more than 0.01 Å. For examples, the figures given above for Cu(100) correspond to calculations on a large, non-periodic slab. For a periodic (i.e. infinite) slab the predicted distances are -0.029 and -1.8034 Å respectively. The change in Δd_{23} is quite significant

2.9. Integration method

The classical equations of motion are integrated by *Kalypso* using the finite difference ‘velocity Verlet’ integration algorithm [29]:

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{v}_n \Delta t + \frac{1}{2} \mathbf{F}_n \Delta t^2 / m \quad (2.19)$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \frac{1}{2} [\mathbf{F}_{n+1} + \mathbf{F}_n] \Delta t / m \quad (2.20)$$

The meanings of the symbols are: \mathbf{r}_n , \mathbf{v}_n , \mathbf{F}_n : position, velocity, total force vectors at n th timestep; Δt : size of the current timestep. The best guarantee of the accuracy of the integration algorithm 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 simulation project (Chapter 4).

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 . *Kalypso* uses the formula $\Delta E / (KE + |PE|)$ (as a percentage, %) to report energy conservation. Apart from integration errors, small energy discrepancies can arise because *Kalypso* may fail to track inelastic energy losses with sufficient precision. (To isolate these, run the simulation with all inelastic options switched off.)

The forces appearing in Eqs. 2.19 and 2.20 are not necessarily calculated directly from the analytic potential functions. In *Kalypso*, forces that involve the attractive potential 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). 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. However, the system energy is calculated analytically.

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 (this is reported as the `range` parameter in the Log file produced by *Kalypso*).

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 conservatively at ~ 100 , although no more than 60-80 partners are needed in most simulations. Depending on selections made by the user, the neighbour lists for targets with free boundaries are constructed either by a brute force (naive search) method or by the linked-cell method [30]. For periodic boundaries, only the brute force method has been implemented at the time of writing.

2.10. Morse potential

When $\xi = 0.0$, the potential used by Kalypso reduces to a pairwise potential having the form:

$$U_{ij}(r_{ij}) = A \left\{ \exp(-p(r_{ij}/r_0 - 1)) - b \exp(-2q(r_{ij}/r_0 - 1)) \right\} \quad (2.21)$$

This class of potentials includes the Morse potential [16]:

$$U_{ij}(r_{ij}) = D \left\{ \exp(-2\alpha[r_{ij} - r_0]) - 2 \exp(-\alpha[r_{ij} - r_0]) \right\} \quad (2.22)$$

where D is the well-depth, and r_0 is the interatomic distance. The range of the potential is determined by the value of α . Eqs. 2.21 and 2.22 are functionally equivalent.

To set up a Morse potential as written in Eq. 2.22 for use in Kalypso, the following data should be entered into the Model file dialog box: $2A = D$, $b = 2$, $p = 2q = 2\alpha/r_0$, $\xi = 0$. The Morse potential is not suitable for describing the material properties of metals, but it may find a use in certain kinds of simulation.

Annexe. Database of surface segregation energies [25].

TABLE II. Calculated surface segregation energies (in eV) for impurity atoms (columns) at the close-packed surfaces of transition-metal hosts (rows). The second row for (host) Fe (Fe*) corresponds to the segregation energies on the bcc(100) surface. The numbers in this diagonal are the surface energies (in eV/atom).

	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zr	Nb	Mo	Tc	Ru	Rh	Pd	Ag	Hf	Ta	W	Re	Os	Ir	Pt	Au
Ti	1.20	0.10	-0.24	-0.34	-0.41	-0.56	-0.75	-0.94	-0.38	0.03	0.09	-0.06	-0.31	-0.62	-0.93	-1.18	-0.14	0.25	0.35	0.20	-0.04	-0.37	-0.72	-1.05
V	-0.49	1.16	0.30	0.41	0.36	0.15	-0.12	-0.54	-1.08	-0.41	0.10	0.36	0.39	0.13	-0.28	-0.75	-1.00	-0.23	0.31	0.62	0.68	0.51	0.09	-0.39
Cr	-0.72	-0.15	1.46	-0.14	-0.44	-0.67	-0.80	-1.02	-2.05	-1.15	-0.62	-0.45	-0.68	-1.25	-1.70	-1.90	-1.55	-0.98	-0.40	-0.17	-0.29	-0.81	-1.58	-1.98
Mn	-0.83	-0.32	-0.10	1.24	-0.12	-0.26	-0.47	-0.77	-1.28	-0.73	-0.73	-0.48	-0.52	-0.69	-0.93	-1.31	-1.83	-1.03	-0.56	-0.31	-0.32	-0.53	-0.83	-1.23
Fe	-0.39	0.06	0.10	-0.16	1.20	-0.14	-0.65	-0.83	-1.60	-0.65	-0.06	0.10	-0.20	-0.52	-1.05	-1.55	-1.80	-0.35	0.20	0.45	0.25	-0.15	-0.66	-1.36
Fe*	-0.69	0.19	0.16	-0.38	1.73	-0.03	-0.77	-1.37	-2.22	-0.83	0.03	0.24	0.00	-0.53	-1.43	-2.37	-2.15	-0.80	0.42	0.78	0.60	0.08	-0.78	-1.93
Co	-0.33	0.13	0.19	0.10	-0.01	1.07	-0.13	-0.48	-1.40	-0.45	0.00	0.49	0.12	-0.40	-0.60	-0.93	-0.56	-0.24	0.34	0.72	0.56	-0.10	-0.38	-0.76
Ni	-0.12	0.20	0.25	0.00	0.13	0.13	0.95	-0.25	-1.16	-0.26	0.18	0.31	0.10	-0.10	-0.40	-0.80	-0.74	-0.01	0.45	0.53	0.37	0.16	-0.17	-0.69
Cu	0.01	0.25	0.10	0.07	0.28	0.33	0.17	0.77	-0.64	0.00	0.28	0.30	0.20	0.05	-0.20	-0.42	-0.35	0.22	0.57	0.62	0.48	0.23	-0.04	-0.29
Zr	0.06	0.01	-0.47	-0.36	-0.40	-0.45	-0.55	-0.72	1.22	0.19	0.13	-0.01	-0.19	-0.41	-0.68	-0.88	0.15	0.33	0.30	0.15	-0.02	-0.25	-0.50	-0.80
Nb	-0.24	0.12	0.32	0.23	0.29	0.31	0.08	-0.20	-0.65	1.21	0.48	0.70	0.65	0.42	0.05	-0.32	-0.47	0.17	0.70	0.98	1.00	0.77	0.40	-0.03
Mo	-0.14	0.08	-0.01	-0.50	-0.52	-0.72	-0.82	-1.16	-0.90	-0.22	1.60	-0.21	-0.81	-1.28	-1.47	-1.75	-0.98	-0.02	0.22	0.10	-0.45	-1.15	-1.60	-1.94
Tc	-0.82	-0.40	-0.11	-0.20	-0.11	-0.01	-0.24	-0.70	-1.57	-0.77	-0.27	1.47	0.02	-0.11	-0.46	-0.97	-1.26	-0.65	-0.06	0.26	0.37	0.21	-0.16	-0.70
Ru	-0.30	0.15	0.24	-0.40	-0.39	-0.37	-0.71	-1.21	-1.12	-0.31	0.10	0.17	1.48	-0.43	-1.03	-1.72	-0.83	-0.17	0.24	0.37	0.23	-0.20	-0.82	-1.62
Rh	0.12	0.35	0.31	-0.08	-0.01	0.02	-0.08	-0.38	-0.46	0.09	0.44	0.46	0.31	1.15	-0.45	-0.92	-0.15	0.36	0.66	0.71	0.56	0.23	-0.27	-0.87
Pd	0.58	0.78	0.30	0.30	0.35	0.29	0.21	0.04	0.32	0.87	1.08	1.02	0.74	0.36	0.84	-0.26	0.44	1.04	1.37	1.34	1.11	0.70	0.19	-0.22
Ag	0.45	0.63	0.29	0.23	0.41	0.48	0.49	0.22	0.33	0.67	0.74	0.69	0.60	0.42	0.28	0.58	0.40	0.83	0.93	0.88	0.72	0.55	0.34	0.03
Hf	-0.03	-0.04	-0.52	-0.49	-0.51	-0.62	-0.75	-0.93	-0.14	0.68	-0.12	-0.25	-0.44	-0.68	-0.92	-1.16	1.35	0.25	0.17	-0.01	-0.26	-0.53	-0.80	-1.11
Ta	-0.45	-0.03	0.16	0.10	0.13	0.06	-0.18	-0.52	-0.85	-0.21	0.25	0.44	0.40	0.11	-0.26	-0.67	-0.60	1.37	0.47	0.75	0.73	0.49	0.11	-0.37
W	0.02	0.04	-0.13	-0.25	-0.35	-0.45	-0.42	-0.75	-0.81	-0.31	-0.34	-0.55	-1.07	-1.22	-1.27	-1.56	-0.78	-0.13	1.87	-0.27	-0.85	-1.34	-1.66	-1.85
Re	-0.89	-0.42	-0.13	-0.28	-0.18	-0.14	-0.36	-0.83	-1.75	-0.94	-0.43	-0.19	-0.17	-0.32	-0.68	-1.24	-1.51	-0.77	-0.27	1.49	0.04	-0.11	-0.46	-1.05
Os	-0.22	0.18	0.36	-0.21	-0.31	-0.30	-0.62	-1.21	-1.07	-0.27	0.07	0.09	-0.20	-0.70	-1.31	-2.00	-1.04	-0.17	0.13	0.23	1.81	-0.48	-1.25	-2.14
Ir	0.29	0.51	0.35	0.09	0.11	0.16	0.12	-0.12	-0.43	0.10	0.35	0.35	0.23	-0.08	-0.55	-1.00	-0.17	0.26	0.47	0.48	0.32	1.44	-0.58	-1.20
Pt	0.66	0.98	0.60	0.38	0.37	0.46	0.43	0.32	0.30	0.76	0.93	0.85	0.60	0.26	0.00	-0.27	0.47	0.95	1.16	1.11	0.86	0.44	1.03	-0.36
Au	0.46	0.59	0.33	0.30	0.45	0.54	0.56	0.34	0.30	0.61	0.67	0.59	0.52	0.44	0.28	0.00	0.47	0.79	0.92	0.81	0.65	0.50	0.34	0.72

References for Chapter 2

-
- [1] M.I. Baskes, M. Asta, S.G. Srinivasan, *Phil. Mag. A* 81 (2001) 991.
 - [2] J.F. Ziegler, J.P. Biersack, U. Littmark, J.F. Ziegler (Ed.), *The Stopping and Range of Ions in Solids, Vol 1: The Stopping and Range of Ions in Matter* Pergamon, New York, 1985.
 - [3] G. Molière, *Z. Naturforsch. 2a* (1947) 133.
 - [4] M. Born, J.E. Mayer, *Z. Phys.* 75 (1932) 1.
 - [5] Th. Fauster, D. Hartwig, H. Dürr, *Appl. Phys.* A45 (1988) 63.
 - [6] K. Broomfield, R.A. Stansfield and D.C. Clary, *Surface Sci.* 202 (1988) 320.
 - [7] K. Kawata, R.I. Erickson, J.R. Doyle, *Nucl. Meth. B* 201 (2003) 566.
 - [8] K. Nordlund, N. Runeberg and D. Sundholm, *Nuclear Instr. Meth.* 132 (1997) 45.
 - [9] M.A. Karolewski, *Nuclear Instr. Meth. B* (2006) (in press: a preprint can be found in the /docs/papers directory of the *Kalypso* distribution).
 - [10] 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, URL: <http://www.nist.gov>).
 - [11] D. Tomanek, A.A. Aligia and C.A. Balseiro, *Phys. Rev. B* 32 (1985) 5051.
 - [12] R.P. Gupta, *Phys. Rev. B* 23 (1981) 6265.
 - [13] M.W. Finnis and J.F. Sinclair, *Phil. Mag. A* 50 (1984) 45.
 - [14] M.S. Daw, S.M. Foiles, M.I. Baskes, *Mater. Sci. Rep.* 9 (1993) 251; S.M. Foiles, M.I. Baskes, M.S. Daw, *Phys. Rev. B* 33 (1986) 7983; M.S. Daw, M.I. Baskes, *Phys. Rev. B* 29 (1984) 6443.
 - [15] H. Häkkinen, M. Manninen, *Physica Scripta T33* (1990) 210. Online at URL: <http://www.physica.org/secure/archive/T33a00210.pdf>.
 - [16] P.M. Morse, *Phys. Rev.* 34 (1929) 57.
 - [17] M.A. Karolewski, *Radiation Effects and Defects in Solids*, 153 (2001) 239-255. [Note that the parameters given in the paper for Th (thorium) are incorrect, due to fitting with incorrect elastic constants. The parameters given in this document have been corrected.]
 - [18] F. Cleri and V. Rosato, *Phys. Rev. B* 48 (1993) 22.
 - [19] G.C. Kallinteris, N.I. Papanicolaou, G.A. Evangelakis and D.A. Papaconstantopoulos, *Phys. Rev. B* 55 (1997) 2150-2156.
 - [20] V. Paidar, A. Larere and L. Priester, *Modelling. Simul. Mater. Sci. Eng.* 5 (381) 1997.
 - [21] M.J. López, J. Jellinek, *J. Chem. Phys.* 110 (1999) 8899.
 - [22] H. Rafii-Tabar, A.P. Sutton, *Phil. Mag. Lett.* 63 (1991) 217.
 - [23] S. Rohart *et al.*, *Surf. Sci.* 559 (2004) 47.
 - [24] C. Mottet, G. Tréglia, B. Legrand, *Phys. Rev. B* 66 (2002) 045413.
 - [25] A.V. Ruban, H.L. Skriver, K. Nørskov, *Phys. Rev. B* 59 (1999) 15990.
 - [26] F.R. de Boer, R. Boom, W.C.M. Mattens, A.R. Miedema, A.K. Niessen, *Cohesion in Metals*, North-Holland, Amsterdam, 1988.
 - [27] M.Z. Bazant, *Interatomic Forces in Covalent Solids*, PhD Thesis, Harvard University (1997), Chapt. 5, p.117. Available online at URL: <http://www.math.mit.edu/~bazant/thesis/index.html>.
 - [28] J. Wan, Y.L. Fan, D.W. Gong, S.G. Shen, X.Q. Fan, *Modelling Simul. Mater. Sci. Eng.* 7 (1999) 189.
 - [29] R. Smith and D.E. Harrison Jr., *Comp. Phys.* 3 (1989) 68.

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

3. TARGET FILES AND PROJECTILE FILES

3.1. Function of the Target file

Target files, which have the extension .TRG, store information about the target lattice. To generate a target lattice within *Spider*, select Target|New¹ then (for example) Face-centred cubic|(100) surface. This selection will allow you to generate a fcc target whose *xy* surfaces have a (100) orientation. Once you understand the file format (section 3.4), you can also generate your own TRG files (via a spreadsheet, computer program etc.). Other programs that can be used to generate lattice atom coordinates for inclusion in Target files include the Makextal utility in the Camelion package (<http://www.tm.tudelft.nl/secties/fcm/matphy/software/software.htm>) and the Atoms utility (<http://feff.phys.washington.edu/~ravel/software/exafs/>).

3.2. Coordinate system

Coordinates in a TRG file are expressed in Ångströms. The *z*-direction corresponds to the surface normal. By default, the surface plane coincides with *z* = 0.0, although this is not mandatory. The positive *z*-direction is directed away from the target surface, while the negative *z*-direction is directed into the surface (Fig. 3.1).

The coordinates in a TRG file refer to the atomic positions of target atoms in the laboratory system of coordinates. Apart from thermal vibrational effects, target atoms are stationary at the start of a simulation.

Most simulations include one or more moving atoms that are designated as *projectiles*. In many cases, the projectile is a single atom, but *Kalypso* also allows the use of clusters of atoms.

The projectile initial position and velocity depend on the Projectile file (*.PRJ), on the projectile incident angles (φ , ϕ) specified in the Run file, on the 'projectile mode' parameter, which is also set in the Run file (see Chapter 4), and finally on the impact parameter (specified in the Impact file).

In most cases, the projectile mode will be that of an *impinging atom* (or cluster). This mode applies to sputtering, ion scattering and any other type of projectile bombardment simulation. An *impinging* projectile approaches the target surface from above, travelling in the negative *z* direction (and if the angle of incidence is non-normal, in the negative *x* or *y* directions), as shown in Fig. 3.1.

For an impinging projectile mode, the velocity initialisations are carried out as follows:

$$\begin{aligned}v_{0x} &= -\sqrt{E/2m} \cos \varphi \cos \phi \\v_{0y} &= -\sqrt{E/2m} \cos \varphi \sin \phi \\v_{0z} &= -\sqrt{E/2m} \sin \varphi\end{aligned}\tag{3.1}$$

The altitudinal angle of approach ($\varphi = 0-90^\circ$) is defined in the RUN file. The angle φ is a polar-type angle defined relative to the surface. Normal incidence corresponds to $\varphi = 90^\circ$. The direction of approach in the *xy* plane can be specified by choosing the

¹ The expression 'select Target|New' means 'select menu command Target, then sub-menu command New'.

azimuthal (ϕ) angle in the range 0-360° (also specified in the RUN file), although it is generally much easier to choose $\phi = 0^\circ$ (which aligns the projectile with the x -axis) and to rotate the target as needed. Further discussion on projectile velocity initialisation is given in Chapter 4.

3.3. Anchor atom

The first atom specified in the TRG file is known as the **anchor atom**. *Spider's* lattice generation options will assign this target atom to the position (0,0,0) by default, but any other location is allowed. When you later define the starting coordinates of the impinging projectile (in the Impact file), these will be expressed *relative to the position of the anchor atom*, for example:

Anchor atom: (0.0, 0.0, 0.0)
Impact file: (0.0, 0.0, 3.0)
Projectile initial position (normal incidence): (0.0, 0.0, 3.0)

Anchor atom: (0.0, 2.5, 3.2)
Impact file: (0.0, 0.0, 3.0)
Projectile initial position (normal incidence): (0.0, 2.5, 6.2)

3.4. Target file format

Target files can contain references to **one** or **two** distinct types of atoms, but not more. Atomic type is defined by the atomic number, Z (different isotopes of the same atomic type may be included). The following excerpt from a Target file illustrates the format:

0.00000	0.00000	0.0000	19	39.10000	0	K
7.38238	-0.00000	0.5000	35	79.91000	0	Br
7.38238	-0.00000	-3.30150	19	39.10000	0	Br

Columns 1-3: (x, y, z) coordinates in Å (real numbers)

Column 4: Atomic number, Z (integer)

Column 5: Atomic mass in amu (real number)

Column 6: Options flags variable (integer)

Column 7: Label (e.g. chemical symbol); this is ignored by *Kalypso*

All numeric data are space-delimited (i.e. the columns must be separated by one or more blank/tab spaces, but they do not have to be aligned - this is known as 'free-formatting'). Real numbers can be entered in a variety of formats (32.0, 32, 3.2E1, 3.2e1 etc.).

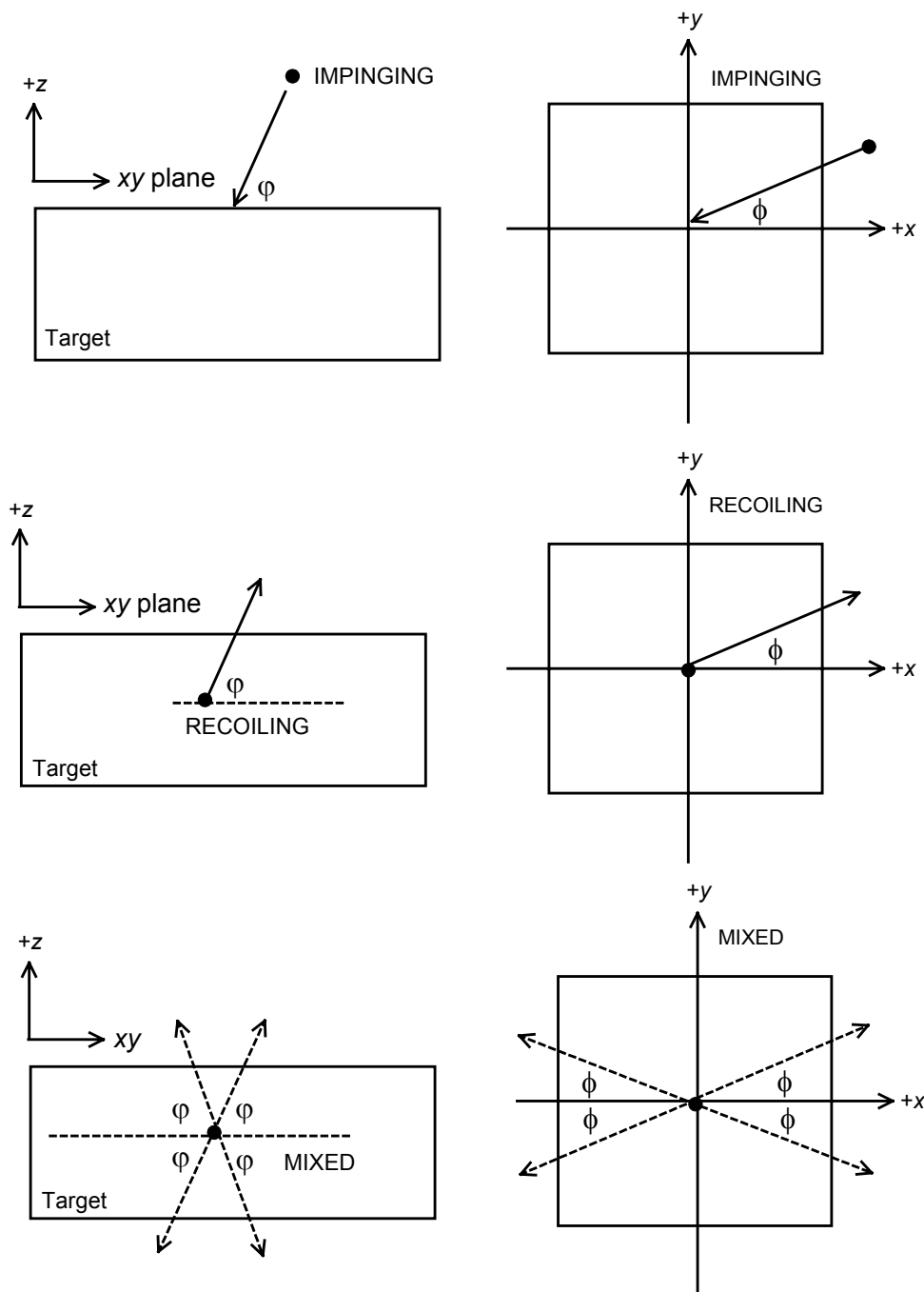


Fig. 3.1. Configurations of projectile and target for 'impinging' (top), 'recoiling' (centre) and 'mixed' (bottom) projectile modes respectively. In the 'mixed' mode, the projectile direction of motion is randomly chosen from several possibilities as shown (for explanation, refer to section 4.2.3).

Target files can be modified in any text editor. The only thing to be careful about is the end-of-file position: if you create Target files yourself using a text editor make sure that the cursor is flush with the left-most margin of the last (empty) row when you save the file (i.e. no blank spaces after the last linefeed character). If not, you will probably get an error message E023 (see Appendix for explanation). If in doubt, check that *Kalypso* reports the expected number of target atoms. **Except for the anchor atom, the order of the atoms in the Target file does not matter.** Labels in Column 7 are included only for the user's information - they are ignored by *Kalypso*.

3.4. Options flags

3.4.1. Flags used by *Kalypso*

Kalypso uses an array of 32-bit integers to store information relating to individual particles (up to 32 properties per particle can be stored). This type of variable is known as a bitmapped field, and the technique of mapping bits will be familiar to anyone with programming experience. The specific bits of each *options flags variable* (array element) contain information about different properties (or options) for the corresponding particle.

The individual bits in the options variable are known as *flags*. In *Kalypso*, these options flags are designated by names of the form ofXXX, and they can either be set (flag = 1) or cleared (flag = 0). The flags defined in version 2 of *Kalypso* and their values are listed in Table 3.1.

Table 3.1. Options flags (ofXXXX) used by *Kalypso* and their values. Flags indicated in **bold face** can be modified by the user.

Flag name	Decimal value	Binary value	Remarks
ofUseImagePotential	1	00000001	All particles
ofNoCool	2	00000010	All particles
ofSprungAtom	4	00000010	Target particles
ofFixedAtom	8	00000100	All particles
ofEdgeAtom	16	00001000	All particles
ofRecorded	32	00010000	All particles
ofType1Atom	64	00100000	[Do not modify]
ofProjectile	128	01000000	[Do not modify]
ofPreImplant	256	10000000	Projectiles
ofReserved	512-2048	-	May be used in future

The options flags variables are read (as integers) from the Target and Projectile files of a simulation project. The default value for all flags is zero. **Many users of Kalypso will have no reason to change this default behaviour.** The functions of the various flags are explained in the following sections.

If you wish to incorporate one or more of the flags-related features into your simulation, you 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 Decimal Value column of Table 3.1. Thus an options flags variable of 3 indicates that the ofImagePotential and ofNoCool bits have both been set ($1+2 = 3$). If you are in doubt about how to add bits, ask a computer programmer for advice.

Spider provides two methods for modifying the flags settings in **Target files**: (a) the Target file Visualiser, which allows flags to be updated for spatially localised sets of

atoms; (b) the Edit Flags dialog box on the Target menu, which permits flag updates for specified atoms. Flags in both **Target** and **Projectile files** can be modified by hand using a text editor.

There may be some situations in which you choose to set new flags for your own purposes (see section 3.4.5). In this case, you are advised not to use the (currently unused) flags in the numeric range 256-2048, because they may acquire a specific meaning in future releases of *Kalypso*. Suitable values are 4096, 8192, ... etc.

3.4.2. Flag: ofUselImagePotential

If you plan to include image potential effects in your simulation, you must set this flag for each (projectile or target) atom to which the image potential applies. Image potential effects will be ignored for any atoms which do not have this flag set. Example: if you only want to apply the image potential to the projectile species, just set the projectile flags parameter to 1.

3.4.3. Flag: ofNoCool

Atoms which have this flag set are ignored for purposes of calculating and correcting the target temperature. The flag has no effect unless the temperature control feature (Inelastic file) is included in the simulation. Typically the flag is set for atoms in the upper layers of a target (no cooling applied), while cooling effects are applied to the lower layers of the target.

3.4.4. Flag: ofSprungAtom

This flag has no effect unless the lattice site springs feature is used by the simulation (Inelastic file). Lattice site springs are then applied to any atoms that have this flag set but not to other atoms. Typically, lattice site springs are only ever used for atoms at the edges of a target with free boundaries (in order to stabilise the structure).

3.4.5. Flag: ofFixedAtom

When set, this (rarely useful) flag prevents updates of the velocity and position of the associated particle (similar to the behaviour of a particle of infinite mass). The particle continues to interact with nearby particles as usual.

3.4.6. Flag: ofEdgeAtom

This flag has no effect on the simulation. It is only used for marking atoms at the edge the Target file, so that later they can easily be filtered out using *Winnnow*. The significance of this flag (and any flag whose value is undefined in Table 3.1) is only determined by the interpretation that the user puts upon it. For example, it is possible to set the ofEdgeAtom flag for atoms in the second atomic layer of the target, instead of for those at the edge of the target.

3.4.7. Flag: ofRecorded

This flag 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: 'Do not write more than one record per run' has been selected. Under these circumstances, a set flag will suppress output for the atom in question. If you set the flag manually for a particle in the Target or Projectile file, it will prevent any output being written to disk for that particle during the simulation.

3.4.8. Flag: ofPreImplant

This flag is used only if you want to run a simulation of a target that includes implanted inert gas atoms. It identifies the (implanted) projectile atoms that are located in the target at the start of the simulation. The flag also affects how the positions and velocities of the atoms are initialised.

For any projectile atoms that have the flag set, the (x, y, z) coordinates will be initialised to the values read from the projectile file (in Å), and the velocities will be initialised to zero.

For example, the following Projectile file input is appropriate for a target which has a stationary Ar atom implanted at $(0, 0, -2.2556)$ [Å] (line 2) and is bombarded by a single 2 keV Ar (98290 m s^{-1}) projectile (line 1) :

0.0	0.0	0.0	18	39.948	0	98290.86202	Ar
0.0	0.0	-2.556	18	39.948	256	0.0	Ar

Note in particular that the coordinates of the atom referenced in line 2 are completely unaffected by the data in the Impact file, unlike the 'normal' projectile atom referenced in line 1.

The `ofPreImplant` flag cannot be used in a multiple impact simulation (an error condition will be raised).

3.4.9. Other flags

The `ofType1Atom` and `ofProjectile` flags are used internally by *Kalypso*. The user should take care not to modify them.

3.5. Exercise: Generating a Ni(100) lattice

Select from the menu Target|New|Face-centred cubic|(100) surface. From the drop-down symbol box select 'Ni'. The data shown in the table below come up in the dialog box. All data input fields (including the symbol) can also be edited manually.

Fig. 3.2. Target file dialog box, showing settings that are appropriate for a small Ni(100) target.

Click SaveAs to generate the lattice, and save it in a file which you specify (e.g. 'Ni100.trg'). Examine the file in a text editor (e.g. the Editor|Open option in *Spider*). There are 121 atoms per layer in this Target file. The first line is the anchor atom, followed by other atoms in the outermost layer of the lattice ($z = 0$):

```
0.00000 0.00000 0.00000 28 58.71000 0 Ni fcc
12.45922 12.45922 0.00000 28 58.71000 0 Ni fcc
12.45922 9.96738 0.00000 28 58.71000 0 Ni fcc
```

The second layer, identified by its z value (-1.762 \AA), begins at line 122:

```
13.70514 13.70514 -1.76200 28 58.71000 0 Ni fcc
```

If you examine the Target file further, you will notice that the atomic positions in each layer (except for the anchor atom) are sorted systematically according to their x values.

Suppose you wanted to apply an *inward* relaxation of 0.1 \AA on the outermost layer of the Ni target (relative to the ideal positions). One way to achieve this is to enter the value -0.1 for the layer relaxation parameter edit box (in Fig. 3.2). Alternatively, you can modify a previously generated Target file using the search-and-replace capabilities of a text editor:

REPLACE: '0.00000 28 58.71000' WITH: '-0.10000 28 58.71000'

This operation will only modify the z values of the first layer. If you did a search-and-replace on the string '0.00000' by itself (wrong!), you would also modify any x or y coordinate with the same value.

3.6. Exercise: Target file for (1×1) metal monolayer system

The simplest type of metal monolayer is the pseudomorphic (1×1) type, e.g. Cu/Ni(100), where to a first approximation the Cu overlayer simply replaces Ni atoms in the surface layer. To create such a target, we begin by creating a Target file 'Cu.trg' that consists of a single (100) layer of Cu atoms based on the Ni lattice constant (Table 3.2).

Table 3.2. Target file input data for a (100) layer of Cu on Ni(100).

Symbol	Cu	Lattice const.	3.524
Atomic No.	29	Atomic mass	63.54
X-width	4	X-origin	0
Y-width	4	Y-origin	0
Z-depth	1	Z-origin	0
Relaxations		0, 0, 0	

We then create a second Target file 'Ni.trg' that consists of a Ni(100) crystallite as described in the previous section. Open this file in a text editor, and cut out all references to atoms in the first layer of Ni atoms. Now load Cu.trg in a text editor, then copy and paste the entire file at the top of the modified file Ni.trg, saving the result. This produces a Target file for Cu/Ni(100) with the Cu atoms in registry with

Ni atoms in the second layer. In most cases, a vertical relaxation would be applied to the Cu layer.

Other types of target can be quite difficult to create using *Spider*. A knowledge of basic surface crystallography and some ingenuity is necessary. The example of a $c(2 \times 2)$ structure on a (100) fcc surface is dealt with in the next section.

3.7. Orienting targets

The Target|New command generates a lattice with a specified surface orientation in the z -direction. Often, however, you may have to modify the orientation of the lattice in the (x, y) plane. Table 3.3 summarises the default orientations generated by *Spider*'s Target|New... option.

Table 3.3. Target lattice orientations in (x, y) plane generated by *Spider*.

Lattice type	Orientation	y	x
fcc	(100)	[011]	[01-1] ^b
fcc	(110)	[-110]	[001]
fcc	(111)	[-1-12] ^a	[-110]
bcc	(100)	[-100]	[010]
bcc	(110)	[001]	[-110]
diamond & ZnS	(100)	[011]	[01-1]
diamond & ZnS	(110)	[-110]	[001]
diamond	(111)	[-1-12] ^a	[-110]
hcp	(0001)	[-1010]	[-12-10]

^a The edges of these lattices lie at 60° to the x -axis, not parallel to the y -axis, and they are also $\langle 110 \rangle$ type edges.

^b The notation [01-1] is used in this document to mean $[0\bar{1}1]$.

The orientation of a Target file can be modified using the Target|Visualiser menu command. This brings up a dialog box with a display of your lattice. The detailed instructions for using this utility are found in the online help, which is context sensitive. Some of the main features are:

1. There are options for displaying Target files, and for re-orienting them.
2. The orientation options can be tested without committing changes to disk.
3. Orientation options include the ability to rotate, clip (cut edges off) and translate lattices.
4. Display options allow you to view the lattice from different directions, zoom in/out, translate and rotate.
5. There is an option to superimpose coordinates from another Target file or an Impact file on the current display. This allows you to check, for example, that the Impact file correctly reflects surface structure and symmetry.

A very frequent operation is to create a fcc(100) target with $\langle 001 \rangle$ edges: this example is discussed in the online Help, and involves applying a rotation of 45° in the (x, y) plane to the $\langle 011 \rangle$ terminated default lattice, followed by a trimming operation that reveals the $\langle 001 \rangle$ edges. You can also write a computer or spreadsheet program that achieves the same goal. More complex operations can be envisaged (e.g. creating a Cu lattice which exposes the (210) surface), but in practice these will rarely be used.

It is a good idea to start off with large Target files, then trim them down to your requirements when all operations have been completed. Otherwise, you may find that the rotated overlayer does not cover the corners of your substrate lattice, and you will have to start all over again.

Section 3.6 described how to create the Target file for a (1×1) metal overlayer system. On occasion you may have to work with a more complex system, for example, a $c(2 \times 2)$ overlayer on a (100) fcc surface, more properly known as a $(\sqrt{2} \times \sqrt{2})R45^\circ$ overlayer. The latter is equivalent to a fcc (100) layer which is rotated by 45° with respect to the substrate, and which has a lattice constant expanded by a factor $\sqrt{2}$ relative to that of the substrate (a). Thus, one way of creating such an overlayer is as follows:

1. Create a Target file with 1 atomic layer, and a lattice constant $\sqrt{2}a$.
2. Use the Visualiser utility to rotate this layer by 45° with respect to the substrate.
3. Translate the x and y coordinates by an amount $a/2\sqrt{2}$, to place the overlayer atoms in registry with (i.e. above) the substrate hollow sites.
4. Combine the rotated Target file with that for the substrate, as in section 3.6.
5. Use the Visualiser utility to trim unwanted edges from the composite target.

This example illustrates the importance of understanding the surface crystallography of your simulation problem.

3.8. Choosing the target size

For a keV sputtering simulation you will need a target with at least 1500 atoms. A more typical number these days would be 5000-10,000 atoms. In small targets, the failure to contain collision cascades laterally may introduce errors in the sputter yield and other predicted properties.

For an ion scattering spectroscopy (ISS) simulation, the 'lattice' will consist of 2-20 atoms typically, or as few as is required to illustrate the physics of the problem. One reason for this is that ISS simulations run for about 10 fs, which means that interactions between target atoms can be neglected. Furthermore, ISS data require only realistic modelling of the hard, short-range interactions experienced by the projectile in the topmost 1-3 target layers.

The CPU time required for simulation is roughly a linear function of target size. For ISS simulations, a careful choice of the termination time can also reduce the CPU time. However, in a sputtering simulation there is little to gain from reducing the termination time from (say) 2000 fs to 1000 fs because the timestep becomes quite large near the end of the simulation.

3.9 Projectile files

The Projectile file (extension PRJ) contains the projectile characteristics: symbol, atomic number and mass, velocity (in m s^{-1}) and the projectile flags (see section 3.4). For most purposes you can ignore the projectile flags (but if you apply a thermostat to the target you will probably want to disable cooling effects for the projectile).

The Projectile file also contains a set of positional coordinates (x_p , y_p , z_p) for each projectile atom (the order in which atoms are specified is unimportant). By default, the coordinates are written as (0, 0, 0) for every projectile atom. These coordinates are used to define the relative locations of atoms in a cluster projectile, and for a cluster projectile they **must** be edited appropriately in a text editor (see also section 6.2). For an atomic projectile, the coordinates should normally be left at the default value of (0, 0, 0).

The Projectile file data is thus very simple, and little further explanation is required. Just click Projectile|New in *Spider* and enter the projectile characteristics (Fig. 3.3).

If your simulation involves a cluster projectile, you should enter the number of atoms in the cluster into the dialog box (e.g. 3 for the Cu₃ cluster projectile). The coordinates in the Projectile file that is generated must be modified by hand to reflect the cluster structure (the default values will cause *Kalypso* to crash). It is a good idea, but not mandatory, to express the cluster coordinates relative to the cluster centre of mass.

Note that several simulation parameters defined in other files depend on the projectile's atomic attributes. These dependencies are listed in Table 3.4 for reference. As a result, changing the projectile atomic species for a new simulation (e.g. from Ar⁺ to He⁺) is not as trivial as it may seem, because parameters must be updated in several files, as Table 3.4 shows.

Projectile file

Specifications

Elemental symbol: Ar

Energy (keV): 3.0

Atomic number: 18

Atomic mass (amu): 39.948

No. of atoms: 1

Edit coordinates by hand if > 1 atom

Save Save As Close Help

Fig. 3.3. Projectile file dialog box.

Table 3.4. Input file dependencies on projectile type.

<i>Simulation parameter(s)</i>	<i>File</i>	<i>Remarks</i>
Projectile mass, atomic number, energy	Projectile	
Projectile-target potential	Model	Depends on projectile Z
Timestep	Run	Depends on projectile velocity
Inelastic energy loss models	Inelastic	Depend on projectile mass, atomic number

4. THE RUN FILE

4.1. Function of the Run file

The Run input file (*.RUN) is the repository for miscellaneous input data that do not involve particle coordinates, potentials, or inelastic energy losses. Some, but not all, of the Run file parameters control the manner in which the simulation runs and terminates. The Run file input dialog box presented by *Spider* consists of two pages, shown in Fig. 4.1. The simulation choices available to the user in this dialog box will now be explained.

4.2. Run file options

4.2.1. General specifications

Random number seed: this (positive integer) value is used to initialise the random number generator at the start of the first run in a simulation. Any value $\neq 0$ will give a reproducible sequence of random numbers for each simulation. A value = 0 will seed the random number generator using the system clock, thereby generating a different, non-reproducible random number sequence for each simulation.

Ignore interactions between target atoms: if this option is checked (☒) , the simulation takes into account projectile-target interactions, but ignores all target-target interactions. Target atoms only accelerate due to interactions with the projectile. This speeds up simulations of repulsive projectile-surface interactions (e.g. for ISS simulations) but should not be used in situations where attractive interactions in the target are important (e.g. sputtering). Typically, this option is used in conjunction with a very short potential cut-off (less than the nearest neighbour distance), in order to reduce calculation time. This option cannot be used in conjunction with item 3 (MI simulation).

Multiple-impact simulation: if this option is checked (☒) , a multiple-impact simulation is executed (see also item 4). This means that the target atom positions are not refreshed at the end of each run. Instead, the new projectile species (atom or cluster) is directed at the target that existed at the end of the previous run. The projectile species in a MI simulation may be an inert atom or cluster, or a metallic atom or cluster, but not a mixture of the two (Ar, Ar₂, Cu₂, CuNi are acceptable, but ArCu is not acceptable). See section 4.2.7 for further information about running multiple impact simulations.

No. of replica atoms: this input item value is only required for a multiple-impact simulation (see item 3). At the end of each run in a multiple impact simulation, the properties of the current projectile atoms (position, velocity, flags etc.) are copied into **replica atoms**, before being reinitialised in preparation for the next projectile impact. As a result, the number of atoms in the system increases after completion by each run by an amount N_p , the number of atoms in the projectile species. The use of replica atoms means that the atomic indices used to label atoms (i.e. the row number or r_w parameters) are manipulated dynamically as the simulation runs. The manner in which this is done depends on whether the projectile consists of inert or metallic atoms.

<p>General specifications</p> <p>Random number seed <input type="text" value="1723983"/></p> <p><input type="checkbox"/> Ignore interactions between target atoms</p> <p><input type="checkbox"/> Multiple-impact simulation</p> <p>No. of replica atoms <input type="text" value="0"/> = N(proj. atoms)xN(impacts)</p> <hr/> <p>Periodic boundaries</p> <p><input type="checkbox"/> Use periodic (x, y) boundaries</p> <p>Period, Lx (Å) <input type="text" value="0.00"/></p> <p>Period, Ly (Å) <input type="text" value="0.00"/></p> <p>Boundaries extend from -Lx/2 to Lx/2, and -Ly/2 to Ly/2</p> <hr/> <p>Projectile initialisation</p> <p>Altitudinal angle (°) (φ) <input type="text" value="90.0"/></p> <p>Azimuthal angle (°) (ϕ) <input type="text" value="0.00"/></p> <p>Values < 0.0 have a special meaning (see Help)</p> <p><input type="checkbox"/> Randomly rotate projectile (for clusters only)</p> <div style="border: 1px solid black; padding: 2px; width: fit-content; margin-top: 5px;"> <p>Projectile mode</p> <p><input checked="" type="radio"/> Impinging atom(s)</p> <p><input type="radio"/> Recoiling atom(s)</p> <p><input type="radio"/> Mixed</p> </div> <hr/> <p>Termination criteria</p> <p>Termination time, minimum (fs) <input type="text" value="0.00"/></p> <p>Termination time, maximum (fs) <input type="text" value="2000.0"/></p> <p>Termination energy (eV) <input type="text" value="0.00"/></p> <p><input type="checkbox"/> Perform energy tests on projectile only</p> <p>Projectile z(min), z(max) (Å) [no effect if = 0] <input type="text" value="0.0"/> <input type="text" value="0.0"/></p>	<p>Output</p> <p>When to write output</p> <p><input type="checkbox"/> At start of each run (t = 0)</p> <p><input checked="" type="checkbox"/> At end of each run</p> <p><input type="checkbox"/> Periodically during each run...</p> <p style="margin-left: 20px;">Period (timesteps): <input type="text" value="1"/></p> <p><input type="checkbox"/> Do not write more than one record per run for any atom</p> <p><input type="checkbox"/> Output log of inelastic events</p> <hr/> <p>Which atoms to write output for</p> <p><input checked="" type="radio"/> All atoms in system</p> <p><input type="radio"/> Projectile atom #1 only</p> <p><input type="radio"/> All particles with KE > 10 eV</p> <p><input type="radio"/> All ejected atoms with rz > 5 Å</p> <p><input type="radio"/> Ejected projectile atom #1 with rz > 5 Å</p> <p><input type="radio"/> Atoms that satisfy the output condition specified below</p> <hr/> <p>Output condition</p> <p>[ke > 20.0] & [vz > 0.0] & [rz > 5.0e-10] outgoing particles with KE > 20 eV</p> <hr/> <p>Meaning of tag fields in output records (advanced option)</p> <p><input checked="" type="radio"/> Normal (proj: alt, phi; target: bx, by)</p> <p><input type="radio"/> Inverted (proj: bx, by; target: alt, phi)</p>
--	--

<p>Thermal vibrations</p> <p><input checked="" type="checkbox"/> Apply vibrational displacements</p> <p><input type="checkbox"/> Apply thermal velocities</p> <p>Lattice vibrational temperature (K) <input type="text" value="300.0"/></p> <p>Location of surface/bulk boundary (on z-axis) (Å) <input type="text" value="0.00"/></p> <p><input type="checkbox"/> Do not apply y-axis vibrational displacements (e.g. for 2D simulations)</p> <div style="border: 1px solid black; padding: 5px; margin-top: 5px;"> <p>Mean square vibrational displacements (type 0 atoms)</p> <p>Bulk (Å²) <input type="text" value="0.0"/></p> <p>Surface perpendicular (Å²) <input type="text" value="0.0"/></p> <p>Surface parallel (Å²) <input type="text" value="0.0"/></p> <p style="text-align: right;">Calculate</p> </div> <div style="border: 1px solid black; padding: 5px; margin-top: 5px;"> <p>Mean square vibrational displacements (type 1 atoms)</p> <p>Bulk (Å²) <input type="text" value="0.0"/></p> <p>Surface perpendicular (Å²) <input type="text" value="0.0"/></p> <p>Surface parallel (Å²) <input type="text" value="0.0"/></p> <p style="text-align: right;">Calculate</p> </div>	<p>Neighbour lists and timestep</p> <div style="border: 1px solid black; padding: 2px; margin-bottom: 5px;"> <p>Range search</p> <p><input type="radio"/> Brute force (faster for very small targets)</p> <p><input checked="" type="radio"/> Cell-index (normal choice)</p> </div> <p>Cell-index search cell size (Å) <input type="text" value="5.0"/></p> <p>Range test constant <input type="text" value="1.0"/></p> <p>Maximum number of partners <input type="text" value="100"/></p> <p>Neighbour update period (timesteps) <input type="text" value="10"/></p> <p>Time to start timestep updates (fs) <input type="text" value="10.0"/></p> <p><input type="checkbox"/> Fixed timestep</p> <p>Initial timestep (fs) <input type="text" value="0.04"/></p> <p>For optimum performance:</p> <p>Cellsize = $1.05 * [Rc + v0 * t0 * Rtest * (N-1)]$</p> <p>where:</p> <p>Rc = potential cut-off (model file)</p> <p>v0 = incident speed of projectile (based on energy specified in projectile file)</p> <p>t0 = initial timestep (this box)</p> <p>N = neighbour update period (this box)</p> <p>Rtest = range test constant (this box)</p>
---	--

Fig. 4.1. Run file input dialog box (showing both pages).

4.2.2. Periodic boundaries

1. **Use periodic (x, y) boundaries:** if this option is checked (☒) , periodic boundary conditions (minimum image convention) will be applied to the simulation system in the x and y dimensions (i.e. parallel to the target surface). Otherwise, the target is assumed to have free boundaries. There is no provision in *Kalypso* for periodic conditions in the z dimension. See also item 2 and section 4.2.8.
2. **Period L_x (L_y):** these values specify the lengths of the periodic cells in the x and y dimensions in Å. The periodic cells run from $-L_x/2$ to $+L_x/2$, and from $-L_y/2$ to $+L_y/2$, respectively. The lengths of the cells should be greater than twice the cut-off distance for the potential, i.e. $L_x > 2R_c$.

4.2.3. Projectile initialisation

3. **Altitudinal angle (φ):** this parameter determines the altitudinal angle of approach of the projectile at the start of the simulation. If the value specified for φ is in the range 0 to 90 ($^\circ$), the value will be applied directly in the simulation. If a negative value is specified for φ in the range -90° to -0.001° , a random value in the range between $|\varphi|$ and 90° will be applied in the simulation. See also item 3 and the discussion below item 4 for further information about the interpretation of this parameter. If $\varphi = 90^\circ$, the projectile approaches the target from the $+z$ (normal) direction, while grazing incidence corresponds to $\varphi \rightarrow 0^\circ$.
4. **Azimuthal angle (ϕ):** this parameter determines the azimuthal angle of approach of the projectile at the start of the simulation. If the value specified for ϕ is in the range 0 to 360 ($^\circ$), the value will be applied directly in the simulation. If a negative value is specified for ϕ in the range -360° to -0.001° , a random value in the range between 0° and $|\phi|$ will be applied in the simulation. If $\phi = 0^\circ$, the projectile approaches the target from the $+x$ direction. If $\phi = 90^\circ$, the projectile approaches the target from the $+y$ direction.
5. **Projectile mode:** this option determines how the projectile position and velocity are initialised at the start of each run. For sputtering, ion scattering and thin film growth simulations select the impinging mode. For simulating particle ejection from a lattice site towards the surface, select the recoiling mode (e.g. FAN-type ion backscattering/shadowing simulations). For simulating particle recoils in random directions from a lattice site (e.g. a GRID simulation) selected the mixed mode. For further discussion, see below item 4.
6. **Randomly rotate projectile:** (applies only to polyatomic or cluster projectiles) if this option is checked (☒) , the projectile cluster is randomly rotated (polar and azimuthal rotations) around its **centre of mass** prior to each run in the simulation.

Refer to Fig. 3.1 for a summary of the various incident projectile geometries. The pseudo-code shown in Fig. 4.2 demonstrates how the projectile **velocity** components are initialised for the various projectile modes (**vin** is the projectile speed, **sin_alt** = $\sin \varphi$, **cos_phi** = $\cos \phi$ etc.). The difference in the velocity initialisation between impinging mode and recoiling mode is that the Cartesian velocity components have their signs reversed. The difference in the velocity initialisation between impinging mode and mixed mode is that the Cartesian velocity components randomly switch signs. Fig. 3.1 shows the significance of the projectile altitudinal angle parameter in impinging and recoiling mode respectively. Consider a projectile whose (φ , ϕ) values are (45, 0), such that it travels *from* the north-east direction parallel to the x-axis in impinging mode. In recoiling mode, a projectile with the same parameters would

travel *towards* the north-east (i.e. from the south-west). In mixed mode, a projectile with the same parameters would travel randomly towards either the north-east, north-west, south-east or south-west.

```

if PrjMode = IMPINGING then
begin
  vz[n] := -sin_alt*vin;
  vx[n] := -cos_alt*cos_phi*vin;
  vy[n] := -cos_alt*sin_phi*vin;
end
else if PrjMode = RECOILING then
begin
  vz[n] := sin_alt*vin;
  vx[n] := cos_alt*cos_phi*vin;
  vy[n] := cos_alt*sin_phi*vin;
end
else if PrjMode = MIXED then
begin // sign = ±1.0 (randomly)
  vz[n] := sign*sin_alt*vin;
  vx[n] := sign*cos_alt*cos_phi*vin;
  vy[n] := sign*cos_alt*sin_phi*vin;
end

```

Fig. 4.2. Pseudo-code that illustrates the initialisation of the projectile velocity components for different projectile modes (v_{in} is the incident velocity).

In recoiling and mixed mode, the projectile initial **position** coordinates do not depend on the direction of motion, i.e. on (φ, ϕ) . This is in contrast to impinging mode. The details of position initialisation in impinging mode are discussed in Chapter 6.

4.2.4. Termination criteria

The termination criteria determine how and when each run in the simulation will end.

7. **Termination times:** the values given here determine the minimum and maximum times before termination of a run. The run will not terminate before the minimum time (items 2-4 will be ignored), and it will terminate shortly after the maximum time (on the next neighbour update).
8. **Termination energy:** the simulation will terminate if the **kinetic energy** of all particles in the system falls below this value. The energy test can be restricted to the projectile only if desired (see item 3).
9. **Perform energy tests on projectile only:** if this option is checked (☒) , the termination tests in item 2 will be applied to the projectile only. This parameter has no effect if it is set to zero (0.0).
10. **Projectile $z(\text{min})$, $z(\text{max})$:** the run will terminate if the z -coordinate of the projectile falls below the minimum value specified here, or if it exceeds the maximum value specified. These parameters have no effect if they are set to zero (0.0). This condition is useful for speeding up ISS simulations.

The termination criteria can be applied simultaneously, so that a given run may terminate for a different reason from the previous run. The termination criteria should

be optimised with a view to avoiding wasteful calculation. For example, in ion scattering simulations the z_{\min} and z_{\max} criteria should reflect the dimensions of the ion-surface interaction region. There is no point in tracking the scattered projectile when it is far away from the scattering region.

4.2.5. Output

The main output from *Kalypso* is the dynamics file (*.SNK). There are two aspects to defining the output characteristics: *what* output to write, and *when* to write it.

Typically, for sputtering or ion scattering simulations we only write output data at the termination (**end**) of each run. There may be situations, however, when you need to record output at the **start** of a run ($t = 0$), or periodically during a run (if you want to examine the development of a collision cascade, for example). Any or all of these options may be selected. In the latter case, you must also specify the period (in timesteps) between write operations in the **Period (timesteps)** box of the Run file dialog.

Do not write more than one record...: this (rarely-used) option allows you to restrict the output associated with any one atom to one record per run (e.g. when a specified property of the atom crosses attains some threshold value).

Output log of inelastic events: another rarely-used option: if your simulation project incorporates LSS, ST or OR inelastic effects, you can dump some information about inelastic losses to a log file (the name of the file is specified in *Kalypso*). The file can also be used to generate a list of the binary collisions that occurred during the simulation. See section 7.9 for more information about this file.

Output data written to the dynamics file consists of dynamical variables information for some or all particles in the system. The data are written to disk in the form of a record that contains the following information (one record per particle recorded):

- ti**, time elapsed since the start of the simulation run;
- rw** (row number), a particle index that is based upon the position of the particle in the projectile and Target files;
- rn** (run number), a run index indicating to which run (i.e. impact file line) the data refer;
- ui** (unique identifier), another index that is incremented whenever the output routine of the program is called - **ui** uniquely labels a block of output data that are written at the same time; **ui** is often, but not necessarily, equal to **rn**; **ui** is used internally for various tasks by Winnow, but users of Winnow can generally ignore it.
- rx, ry, rz, vx, vy, vz**: particle position and velocity components;
- ms**: particle mass;
- fl**: particle options variable (flags) at time of output;
- bx, by**: these 'tag fields' contain either the projectile impact parameter, or the projectile incident angles (see discussion below).

These records are stored in a binary file with extension SNK (called by default `dynvars.snk`). **All dynamical variables in the SNK file records are stored as SI units (kg m s).** The `\src` directory of the *Kalypso* package contains small programs (with source code) that show how to read these files.

The user must specify the particles for which data should be captured by the simulation. There are several standard choices, that can be selected by clicking the appropriate radio-button item [☉] in the dialog box:

1. All atoms in the system;
2. Projectile atom #1 only (if there is no projectile in the simulation, this choice writes data for target atom #1);
3. All particles with kinetic energy above 10 eV;
4. All ejected atoms with $r_z > 5 \text{ \AA}$ with $r_z > 5 \text{ \AA}$;
5. Ejected projectile atom #1 with $r_z > 5 \text{ \AA}$.

The difference between these choices is that they record the information for a subset of particles whose momentary dynamical characteristics satisfy certain conditions. The purpose of having these choices is to avoid recording data that are irrelevant to the purpose of the user. For example, if the purpose of the simulation is to calculate sputtering coefficients, you can simply record data for ejected particles. If your simulation is modelling ion scattering processes you can record data for the projectile only. Be careful not to throw away data that you might need.

The final output specification item is:

6. Atoms that satisfy the (user-programmed) output condition.

User-programmed **output conditions** allow the user to specify a condition that must be fulfilled by a particle's dynamical (and other) variables before its data are recorded. The conditional expression specified by the user will be interpreted by *Kalypso* at run-time. The language in which conditional expressions are framed is similar to the query language used by *Winnow*.

As a simple example, the following expression will achieve the recording of data only for particles which leave the surface (arbitrarily bounded at 5 Å):

```
[rz > 5.0e-10] & [vz > 0.0]
```

Here r_z represents the particle's z -coordinate, while v_z represents the z -component of its velocity vector. The '&' symbol is the logical AND operator. In pseudo-code this specification is programmed as follows:

```
for n = 1 to NATOMS do
  begin
    if (rz[n] > 5.0e-11) AND (vz[n] > 0.0) then
      WriteToFile(rx,ry,rz,vx,vy,vz...)
    end;
```

Similarly, the expression: $[ke > 100.0]$ is satisfied if a particle's kinetic energy (ke) is greater than 100 eV. Therefore the expression specifies that data are only to be recorded for those particles with more than 100 eV kinetic energy.

User-programmed expressions may be up to 255 characters in length, but long expressions should be avoided. Where possible, use the standard choices since they are more efficiently coded and do not require parsing. Note that comments bracketed within curly braces {like this} will be ignored by the expression parser, and they can be used as required to improve readability. Please note that the validity of the syntax of any user-programmed option is not checked by *Spider*. Instead, syntax

checking is carried out by *Kalypso* at run-time, and you will be informed if an error is found.

The output records produced by *Kalypso* include two ‘tag fields’ designated as b_x and b_y . Depending on options selected by the user, the tag fields can store a variety of information, which may be useful for users who wish to access this information via *Winnnow*.

1. The information content of these fields by default (**normal** case) is as follows: (a) if the atom is a projectile, $b_x = \phi$ and $b_y = \phi$ (both expressed in $^\circ$); (b) if the particle is not a projectile atom, b_x and b_y represent the (x, y) coordinates (in metres) of the currently specified projectile impact point (as read from columns 1 and 2 of the Impact file).
2. The default option can be **inverted** by selecting the second radio-button item ☐ in the box labelled **Meaning of tag fields...**. In this case, (a) if the atom is a target atom, $b_x = \phi$ and $b_y = \phi$ (both expressed in $^\circ$); (b) if the particle is a projectile atom, b_x and b_y represent the (x, y) coordinates (in metres) of the currently specified projectile impact point.
3. **Caloric data** can be written to the tag fields if the third radio-button item is selected. In this case, b_x contains the current system temperature, while b_y contains the total system energy (i.e., the internal energy, $U(T)$).
4. **Custom data**: (a) if the atom is a target atom, the x and y coordinates of the atom in the Target file are recorded, i.e. $b_x = x_0$ and $b_y = y_0$ (both expressed in metres); (b) if the particle is a projectile atom, b_x and b_y represent the (x, y) coordinates (in metres) of the currently specified projectile impact point.

4.2.6. Thermal vibrations

Atoms in real lattices are displaced from their ideal lattice positions as a result of thermal vibrations. It is often desirable to include atomic **vibrational displacements** of this kind in a simulation. Less commonly, it may be necessary to take into account the kinetic energy associated with thermal vibrations (**thermal velocities**) at some specified temperature.

Vibrational displacements and thermal velocities are handled separately by *Kalypso*. To include these effects in a simulation, select the appropriate check-box ☒ in the Run file dialog box.

The **lattice vibrational temperature** parameter determines the thermal velocities that will be applied (to target atoms only), according to the relationship:

$$\left\langle \frac{1}{2} m v_x^2 \right\rangle = kT/2 \quad (4.1)$$

(with a similar relationship for the y and z dimensions). The thermal velocities are applied in such a way that the net momentum of the target is zero in any direction.

In order to apply lattice vibrational displacements, the user must specify appropriate values for the mean square vibrational displacements, $\langle \sigma^2 \rangle$ for each type of target atom. These can be entered directly (e.g. literature values), or they can be calculated from Debye temperature data by clicking the Calculate button.

According to the Debye theory, the isotropic mean square thermal vibration amplitude, $\langle r^2 \rangle$, of atoms in a monatomic solid is given by the following expression:

$$\langle r^2 \rangle = 9\hbar^2 T / (Mk\Theta_D^2) [\phi(\Theta_D/T) + \Theta_D/4T] \quad (4.2)$$

where Θ_D is the Debye 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 $x = \Theta_D/T$:

$$\phi(x) = (1/x) \cdot \int_0^x \frac{y dy}{(e^y - 1)} \quad (4.3)$$

Eq. 4.3 refers to the mean square amplitude of an **isotropic oscillator**. For vibrations in a specific direction (x , y or z), the right hand side of Eq. 4.2 must be divided by 3:

$$\langle x^2 \rangle = \langle y^2 \rangle = \langle z^2 \rangle = 3\hbar^2 T / (Mk\Theta_D^2) [\phi(\Theta_D/T) + \Theta_D/4T] \quad (4.4)$$

These **Cartesian** amplitudes are the quantities that are calculated by *Spider* and used by *Kalypso*. If we substitute for the physical constants, interpret M as the mass in atomic mass units (amu), and make the substitution ($x = \Theta_D/T$) then Eq. 4.4 simplifies to:

$$\langle x^2 \rangle = 145.53 / (M\Theta_D) [\phi(x)/x + 1/4] \text{ \AA}^2 \quad (4.5)$$

using $\hbar = 1.054572 \times 10^{-34}$ J s; $k = 1.380650 \times 10^{-23}$ J K⁻¹; 1 amu = 1.660539×10^{-27} kg.

There is some confusion in the equations published in the literature, in that Eq. 4.4 is sometimes associated with the isotropic vibrations (more properly described by Eq. 4.2). 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 user guide!).

Spider uses equation 4.5 to calculate $\langle x^2 \rangle$ for bulk and surface atoms, based on the values for T and Θ_D specified by the user, and the approximations to $\phi(x)$ discussed below. These unidirectional $\langle x^2 \rangle$ values are read by *Kalypso* from the Model file.

The vibrational displacements that 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 *Kalypso* is that they shall not exceed 2.5 standard deviations (to prevent highly improbable events).

For purposes of the vibrational correction, a target atom is classified as a 'surface atom' if its Target file z -coordinate, $z[n]$, places it above the **location of the surface/bulk boundary** that the user specifies in the Run file dialog box.

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

The function $\phi(x)$ that appears in Eqs. 4.2-4.5 can be calculated by numerical integration of Eq. 7.2. Eckstein [1] provides the following approximation:

$$\phi(x) = (1/x) \cdot \int_0^x \frac{y dy}{(e^y - 1)} = 1 - x/4 + \frac{x^2}{36} - \frac{x^4}{3600} + \dots \quad (4.6)$$

The same approximation is used by *Kalypso* 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 $\phi(x) \rightarrow 0$. For values of $x > 3.0$ the following (low-temperature) approximation is used:

$$\phi(x) = [\pi^2/6 - (x+1) \cdot \exp(-x) - (x/2 + 1/4) \cdot \exp(-2x) - \dots]/x \quad (4.7)$$

For $x \gg 1$, $\phi(x)/x$ becomes small in comparison to the $1/4$ term in Eq.7.4, and the lattice vibrations approach their zero-point levels.

4.2.7. Neighbour lists and timestep

1. **Range search method:** For targets with free boundaries *Kalypso* offers two range search methods, that differ only in their speed of execution (see Fig. 4.3 for a comparison). The **brute force** method (a naive search of the entire target) may be slightly faster for small simulation systems (< 1000 atoms, e.g. for ion scattering simulations). For targets with more than 1000 atoms the **cell-index** method should normally be used. With either method, the simulation results should be identical. *At the time of writing, only the brute force method had been implemented for periodic systems, and this case is handled automatically by Kalypso.*
2. **Cell-index search cell size:** this is a parameter used by the cell-index search. Its value is not critical, but sub-optimal values may have an impact on the execution speed. The aim is to choose a cell size that contains an atom plus all of the neighbouring atoms that might interact with it before the next update of the neighbour lists. If in doubt, experiment and/or set the value too large rather than too small. An optimum cell size (x) can be computed using the formula: $x = 1.05(R_c + v_0 t_0 R_T [N-1])$, where R_c is the potential cut-off distance, v_0 is the incident projectile speed (written to the Projectile file), t_0 is the initial timestep (see item 8 below), R_T is the range test constant (see item 3 below) and N is the neighbour update period (number of timesteps between updates, see item 5 below).
3. **Range test constant:** this parameter (R_T) should be set somewhere between 0.7-2.0. The quantity $\Delta r = v t R_T [N-1]$ represents an estimate of the distance that an atom with velocity v can travel after $[N-1]$ timesteps (t). This estimate is used by *Kalypso* to determine which neighbouring atoms should be included in the neighbour list. *Kalypso* bases v on the fastest atom in the system. A relatively safe estimate is $R_T = 2$, which assumes that the central atom moves towards its neighbour in a straight line with velocity v , and that the neighbour does likewise. In practice, however, smaller values of R_T can be used without any impact on accuracy (i.e. energy conservation) because (a) most atoms have velocities much smaller than v , and (b) because centre-to-centre collisions are infrequent events.
4. **Maximum number of partners:** this parameter determines how much memory is allocated for the neighbour lists. The number of partners specified has no impact on execution speed. If the value is too small, your simulation will terminate with error #E017, so if in doubt, set a value that is too large. This parameter represents (approximately) the number of atoms that may be compressed into a cubic region

with edge lengths x , as defined in item 1. For most problems, 100 atoms is sufficient (this number will increase rapidly if you use a potential with a large cut-off distance), so use this value if your computer has plenty of memory. For few-atom problems, such as ion scattering simulations, you can reduce the value to match the number of atoms in the system if you wish.

5. **Neighbour update period:** this parameter determines how often the neighbour lists are refreshed. Typical values are 8-12 (timesteps). Sub-optimal settings may impact the speed of the simulation, but they will not affect its accuracy.
6. **Time to start timestep updates:** most *Kalypso* simulations employ an adaptive timestep logic (the timestep increases as atom speeds decrease); the initial timestep is set by trial and error in order to ensure energy conservation. Experience suggests that better results are obtained if the adaptive timestep logic is turned on after the first hard projectile-surface collision has taken place. A value of about 10 fs is suggested.
7. **Fixed Timestep:** if this check-box is checked ☒, the timestep will not be adjusted as the simulation runs. This will slow down the simulation, possibly by 1-2 orders of magnitude in the case of a long sputtering simulation. However, this option is necessary if you want to obtain an ion scattering trajectory that is sampled at constant time intervals.
8. **Initial timestep:** this sets the timestep used initially by the simulation. The timestep adapts automatically as the simulation runs between the value specified here and (if necessary) a maximum value of 10 fs. If the timestep is fixed (see item 7) adaptation does not take place and the maximum limit does not apply (however, you are unlikely to need to exceed 10 fs). The timestep should be chosen by trial and error, in order to produce satisfactory energy conservation (e.g. better than 0.5%). The Velocity Calculator tool provided in *Spider* can help you to set the timestep. As a rule-of-thumb, the timestep is set to be approximately equal to the time required for the projectile to travel 0.05-0.1 Å. If in doubt, set the timestep smaller rather than larger (but not too small, since this will slow the simulation). Energy leakages due to timestep errors can take place both at the **start** of a simulation (when the projectile is moving quickly) and at the **end** of a simulation (when the timestep is large). If you run the simulation using *Kalypso*'s verbose output mode, a plot of total energy against time or timestep will indicate where or why energy leakage is taking place (Fig. 4.4).

Notes on timestep adaption: During the simulation, the timestep (Δt) will adapt gradually on the basis of the speed of the fastest atom in the simulation system volume according to the prescription: $\Delta t(\text{new}) = (1/3)\Delta t(\text{ideal}) + (2/3)\Delta t(\text{old})$, where $\Delta t(\text{ideal})$ represents the goal timestep that will produce the desired displacement. The simulation system volume is a Cartesian box that contains the target and projectile atoms at the start of the simulation run (and includes a boundary region equal to the cut-off distance). Atoms that move outside this box during the course of the simulation will not be used for making decisions about the timestep size. In a multiple impact simulation (section 4.2.7) the box is defined by the positions of the current projectile for that run, plus the lattice coordinates in the first run. These choices ensure that the speed of simulations will not be unduly influenced by atoms that have travelled away from the lattice. If the target has periodic boundaries, the simulation may slow down because atoms fast and slow are constrained to remain in the lattice region in the two periodic dimensions. However, they may escape the box in the z -

dimension, in which case they will no longer influence the development of the timestep.

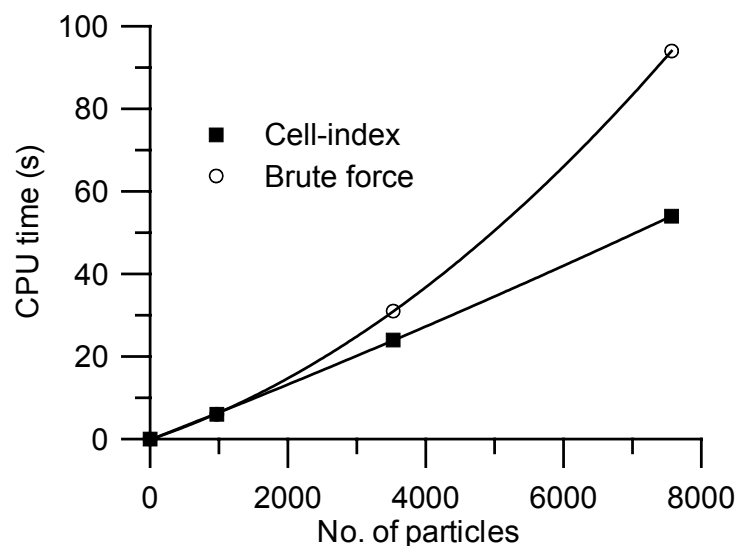


Fig. 4.3. Relative execution speeds of sputtering simulations that use the cell-index and brute-force neighbour search algorithms respectively, as a function of the number of particles (N) in the system. For $N < 2000$, the performance of both algorithms is similar. For $N \sim 8000$, the brute force algorithm is 75% slower than the cell-index algorithm. For $N < 1000$, the brute force method may be slightly faster than the cell-index method.

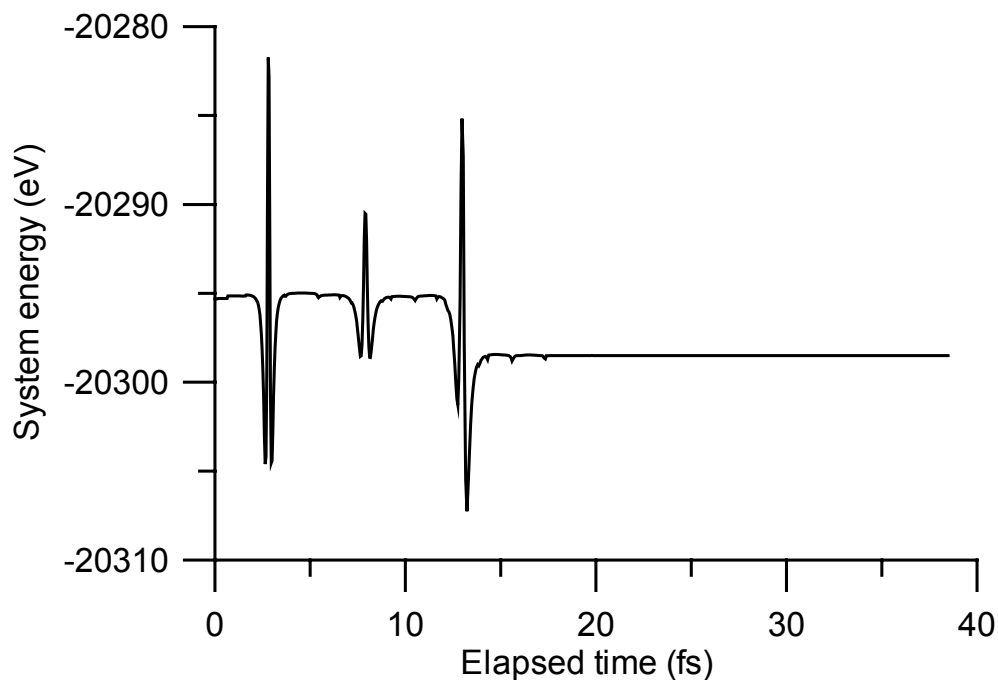


Fig. 4.4. Variation of system energy with elapsed time (5 keV Ar-Cu(111) system, with 0.04 fs initial timestep). The net energy leakage was about 3 eV, but the instantaneous energy leakage during hard collisions (indicated by energy spikes) near the start of the simulation (for $t < 15$ fs) was almost 40 eV. Energy leakage after 15 fs was negligible.

4.2.7. Multiple impact (MI) simulations (notes)

In MI simulations, the atomic positions and velocities are not re-initialised prior to the start of successive runs. Thus structural damage caused by prior ion bombardment accumulates over time and its effects can be included in simulations.

In a MI simulation, the *current* projectile variables (position, velocity etc.) always occupy the same space in memory. The variables associated with previous projectiles (scattered, or embedded in the target) are copied into new memory locations at the end of each run and define so-called **replica atoms**. The original memory locations are then reinitialised.

The number of replica atoms required is obtained as the product: (number of atoms in the Impact file)×(number of atoms in the projectile species). If the number of replica atoms is underestimated, the simulation will terminate prematurely (when all available replica atoms have been consumed). There is no harm in setting the value too high.

The manner in which the replication is carried out depends on the type of projectile involved (inert atoms or metallic atoms). This complication reflects a limitation in the logic used by *Kalypso*, not an intentional design goal. There are two schemes.

1. For **metallic atom projectiles** (e.g. Cu, Cu₄, NiCu) the row number index(es) (rw) of the replica atom(s) succeed those of the **last atom in the Target file**. Example: consider a MI simulation involving Ni bombardment of a 1000-atom Cu target. At the start of run 1, rw = 1 for the projectile, and rw = 2-1001 for the target atoms. At the start of run 2, rw = 1 for the new projectile, and rw = 2-1002 for the target atoms, which now include the embedded projectile from run 1 at rw = 1002. Likewise, at the start of run 3, rw = 2-1003 for the target, with rw = 1003 being associated with the projectile from run 2.
2. For **inert atom projectiles** (e.g. Ar) the replica atoms succeed those of the **last projectile atom**. Example: consider a MI simulation of a 1000-atom Cu target with an Ar projectile. At the start of run 1, rw = 1 for the projectile, and rw = 2-1001 for the target atoms. At the start of run 2, rw = 1 for the new projectile, rw = 2 for the old projectile, and rw = 3-1002 for the target atoms.

Figs. 4.5 and 4.6 illustrate the relationships associated with each indexing scheme.

rw =	1	2	3	4	5	6	7	8	...	N+1	N+2	N+3
run 1	P1	T1	T2	T3	T4	T5	T6	T7	...	TN		
run 2	P2	T1	T2	T3	T4	T5	T6	T7	...	TN	P1	
run 3	P3	T1	T2	T3	T4	T5	T6	T7	...	TN	P1	P2

Fig. 4.5. Row number (rw) indexing scheme for a multiple impact simulation in which the projectile species consists of a **metal atom** (e.g. Cu). Replica atoms are indicated in **bold** font. PN (TN) refers to the Nth projectile (target) atom.

rw =	1	2	3	4	5	6	7	...	N+1	N+2	N+3
run 1	P1	T1	T2	T3	T4	T5	T6	...	TN		
run 2	P2	P1	T1	T2	T3	T4	T5	...	T(N-1)	TN	
run 3	P3	P2	P1	T1	T2	T3	T4	...	T(N-2)	T(N-1)	TN

Fig. 4.6. Row number (rw) indexing scheme for a multiple impact simulation in which the projectile species consists of an **inert atom** (e.g. Ar). Replica atoms are indicated in **bold** font. PN (TN) refers to the N^{th} projectile (target) atom.

The indexing schemes must be properly understood when using *Winnow* to carry out data analysis. In the second scheme (inert projectile), the identity of an atom depends on both its row number (rw) and the run number (rn) (see Chapter 9 for an explanation of these terms). With reference to Fig. 4.6, the target atoms are now identified by the filter condition: `[rw-rn >= 1]`, instead of the condition: `[rn >= 2]` as in Fig. 4.5. Alternatively, in scheme 2, you can use a filter condition based on mass to distinguish between projectile and target atoms. Another workaround to the indexing problem for inert projectile atoms is discussed in section 9.17 of Chapter 9.

In a multiple impact simulation, the validity of the initial position of the projectile with respect to gas phase atoms produced in a previous impact (e.g. sputtered species) is not checked. It is possible that the point at which the fresh projectile will ‘appear’ is already occupied (or very close to) the point occupied by a sputtered atom ejected by a preceding impact event. This accidental situation will give rise typically to a mutual repulsion and a very large energy leak in the range 1 keV- 1 MeV. A program crash, with errors E003 or E004 may also be reported. Since the error is due to wrong positioning of the projectile, the only remedy is to change this aspect of the simulation. The problem is more likely to arise when the projectile starts near the surface, or when many projectile impacts are involved. It is important to detect artefacts of this type, so it is recommended that you always monitor energy conservation when carrying out a multiple impact simulation. This monitoring should be done even if you are applying a thermostat to the target: in this case, you can check that energy leakage due to cooling in each run does not exceed (on average) the energy input by the projectile. Do not expect exact agreement, due to lattice relaxation, as well as integration errors. Anomalous events will show up as much larger leakages.

Momentum imparted to the target during MI simulations causes it to drift noticeably over time in the direction of projectile approach (the amount of drift depends on the nature of the projectile-target collision). This can only be remedied by anchoring the target atoms in the bottom 1-2 layers of the target (e.g. assign a very large mass, or set the `ofFixedAtom` flag: see section 3.4 of the User Guide).

4.2.8. Periodic boundary conditions (notes)

Setting up the Target file for a simulation with periodic boundary conditions requires some care. Several of the example projects that accompany the *Kalypso* package use periodic boundaries. You can study their documentation and Target files for further information and hints. The following discussion refers to a one-dimensional problem, but the generalisation to two dimensions is not difficult.

Consider a target crystallite that consists (typically) of atomic rows with atoms at x -axis locations: ..., $-3d$, $-2d$, $-d$, 0 , d , $2d$, $3d$, ... (layers 1, 3, 5, ...) and ..., $-3d/2$, $-d/2$, $d/2$, $3d/2$, ... (layers 2, 4, 6, ...) where d is the row interatomic spacing.

For a periodic simulation, one could specify (in the Run file) $L_x = 2d, 4d, 6d$ etc. depending on the type of simulation. Suppose, for example, that you choose to use $L_x = 6d$. Then the periodic cell would run from $x = -3d$ to $x = 3d$, and so the points $x = -3d$ and $x = 3d$ would represent the **same location** in the x -dimension. It is therefore necessary to specify only the following atomic coordinates in the Target file: $-2d, -d, 0, d, 2d, 3d$, (layers 1, 3, 5, ...) and $-5/2d, -3d/2, -d/2, d/2, 3d/2, 5/2d$ (layers 2, 4, 6, ...). Note that the coordinates $x = -5/2d$ and $x = 5/2d$ do not represent the same point in the periodic target (both points lie inside the boundaries), so atoms with both coordinates should be included in the Target file.

To check that the periodic boundaries have been set up correctly (i.e. that the raw Target file has been trimmed at the correct places), use *Kalypso* to check the binding energy of atoms at one or more lattice sites located at the Target file boundaries (see footnote to section 8.2 of the User Guide). You should obtain the same value as for atoms in equivalent sites that are not at the boundaries.

The basic rules for creating a Target file for use with boundary conditions are (for the x and y axes):

1. Create a Target file that is symmetric about the axis origin.
2. Place the periodic boundaries equidistant from the axis origin (but on opposite sides of it) in a position such that the atomic locations in an infinite lattice would be symmetric on either side of the boundary (i.e. at lattice sites, or between lattice sites, or both).
3. If the boundaries cut through any lattice sites (they normally do, if the Target has more than one atomic layer), then remove the atoms that lie at the boundary **on one side of the origin only**.

Failure to set the periodic boundaries correctly may result in *Kalypso* crashing, since two atoms can then appear at the same location in the lattice (or close together, if vibrational effects are included), which results in a very large mutual repulsion. A typical error is to confuse L_x , the length of the periodic lattice, with $L_x/2$, the distance that the lattice extends on either side of the origin.

Periodic boundaries in one, rather than two, dimensions can be achieved by setting a very large value for one of the periods. This will effectively produce a target with one periodic boundary dimension.

References for Chapter 4

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

5. THE MODEL FILE

5.1. Function of the Model file

The Model input file (*.MDL) stores the input data that specify the interatomic potentials used for the simulations. *Kalypso* simulations involve targets of (at most) two metallic elemental types (designated as **M** (either element) or **M(0)** and **M(1)** respectively in *Spider* and in this chapter), and a projectile species that may be comprised of one (or both) of these two elements, or an inert gas element designated **I**. These possibilities give rise to several parameters associated with the various permutations of interacting atom pairs I-I, I-M(0), M(0)-M(0) and so on.

The Model file input dialog box presented by *Spider* consists of three pages, shown in Figs. 5.1-5.3. The data input required on each page will be described in the following sections. Chapter 2 discusses the interatomic potentials used by *Kalypso* in detail.

Screened Coulombic potential: I = inert atom, M(0), M(1) = metallic atoms

Set system type [disables unneeded items]

- ☐ Mono-metallic with inert atom(s)
- ☐ Mono-metallic without inert atom(s)
- ☒ Bi-metallic with inert atom(s)
- ☐ Bimetallic without inert atoms

Constants (b, c) for I-M and M-M potential

	Inert - Metal	Metal - Metal
c1	<input type="text" value="0.028171"/>	<input type="text" value="0.028171"/>
c2	<input type="text" value="0.28022"/>	<input type="text" value="0.28022"/>
c3	<input type="text" value="0.50986"/>	<input type="text" value="0.50986"/>
c4	<input type="text" value="0.18175"/>	<input type="text" value="0.18175"/>
b1	<input type="text" value="0.20162"/>	<input type="text" value="0.20162"/>
b2	<input type="text" value="0.40290"/>	<input type="text" value="0.40290"/>
b3	<input type="text" value="0.94229"/>	<input type="text" value="0.94229"/>
b4	<input type="text" value="3.1998"/>	<input type="text" value="3.1998"/>

Set

- ☒ ZBL
- ☐ Molière
- ☐ (no action)

- ☒ ZBL
- ☐ Molière
- ☐ (no action)

$$V(r_{ij}) = (Z_i Z_j e^2 / 4\pi \epsilon_0 r_{ij}) \sum_{k=1}^4 c_k \exp(-b_k r_{ij} / a)$$

Screening lengths (a) for I-M and M-M interactions

Atomic numbers

I M(0) M(1)

Set screening length

- ☒ ZBL
- ☐ Molière-Lindhard
- ☐ Molière-Firsov
- ☐ (no action)

	Screening length (Å)	Correction
I - I	<input type="text" value="0.1204948666"/>	<input type="text" value="1.0"/>
I - M(0)	<input type="text" value="0.1138927914"/>	<input type="text" value="1.0"/>
I - M(1)	<input type="text" value="0.1143776935"/>	<input type="text" value="1.0"/>
M(0) - M(0)	<input type="text" value="0.1079766084"/>	<input type="text" value="1.0"/>
M(0) - M(1)	<input type="text" value="0.1084123458"/>	<input type="text" value="1.0"/>
M(1) - M(1)	<input type="text" value="0.1088516143"/>	<input type="text" value="1.0"/>

Fig. 5.1. Model file input dialog box (page 1 of 3: Screened Coulombic Potentials).

(i - j) pair	(0 - 0)	(0 - 1)	(1 - 0)	(1 - 1)
2A (eV)	0.156525	0.134760	= 0.134760	0.112995
r0 (Å)	2.5560	2.5239	= 2.5239	2.4918
p	11.18320	12.63493	= 12.63493	14.08666
2q	4.63941	3.92837	= 3.92837	3.21733
ξ (eV)	1.23552	1.31789	= 1.31789	1.40054
b	0.0	0.0	= 0.0	0.0
	File...	File...	File...	File...

Fig. 5.2. Model file input dialog box (page 2 of 3: Tight-Binding Potentials).

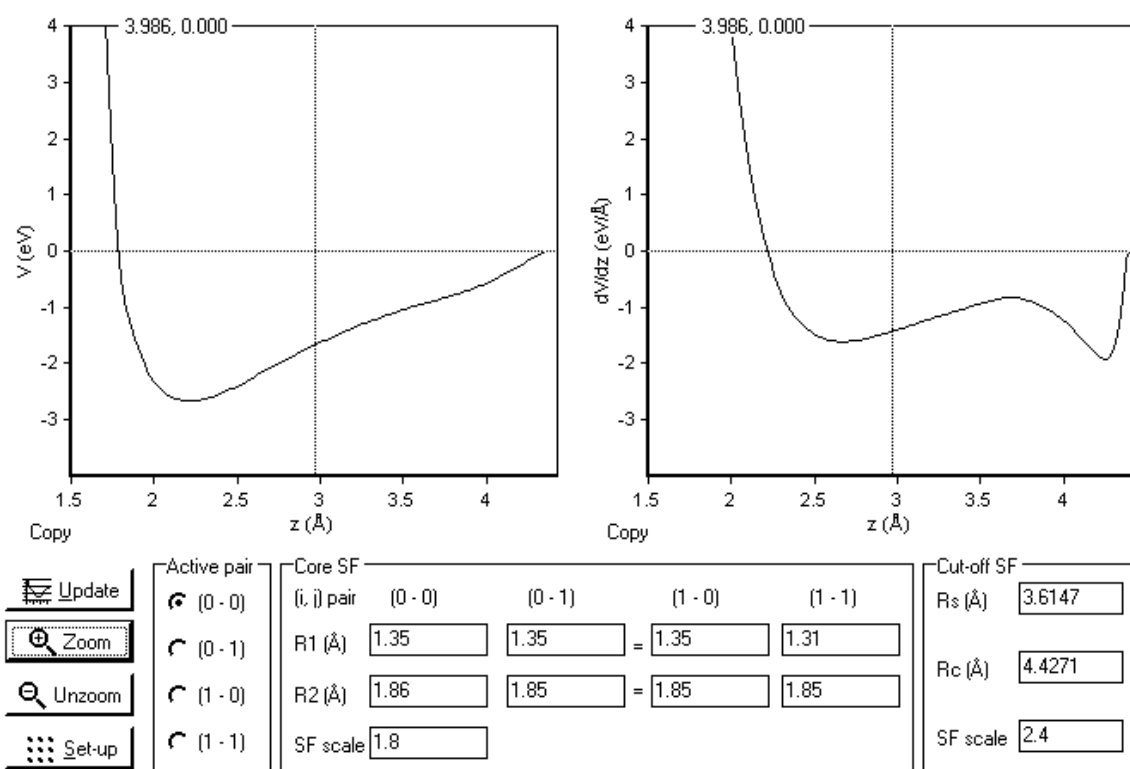


Fig. 5.3. Model file input dialog box (page 3 of 3: Switching Functions).

5.2. Model file options

5.2.1. Screened Coulombic potentials

The functional form of the screened Coulombic potential was discussed in section 2.3 of Chapter 2:

$$V(r_{ij}) = \frac{Z_1 Z_2 e^2}{4\pi\epsilon_0 r_{ij}} \sum_{k=1}^N c_k \exp(-b_k r_{ij} / a) \quad (2.3)$$

1. **Set system type:** this setting will enable/disable input items in the Model file input dialog box according to the requirements of the simulation type that you select. Examples of the different system types are: (a) Mono-metallic with inert atoms [Ar → Cu(100)]; (b) Mono-metallic without inert atoms [Cu → Cu(100)]; (c) bi-metallic with inert atoms [Ar → Cu/Ni(100)]; bi-metallic without inert atoms [Ni → Cu(100)]. Parameters in edit boxes that are 'greyed out' (disabled) will have no effect on your simulation, so their values are unimportant.
2. **Constants (b , c) for I-M and M-M potential:** these constants refer to the terms b_i and c_i that appear in Eq. 2.3. These terms take standard values for the ZBL and Molière potentials respectively. If required, two different types of screened Coulombic potential may be used for the I-M and M-M interactions respectively. However, all M-M interactions must use the same type of potential (e.g. ZBL potential). This does not present a significant restriction. Values for the potentials can be entered by selecting the appropriate radio-buttons in the box labelled **Set** [⊙]. For other potentials (e.g. Bohr potential) you must enter the values for each term by hand (enter 0.0 for any terms that are unallocated in your potential).
3. **Screening lengths (a) for I-M and M-M interactions:** these constants refer to the screening length term a that appears in Eq. 2.3. The screening length depends on the **atomic numbers** (Z) of the interacting atoms, which must be specified for the I, M(0) and M(1) species, as applicable. For example, I = 18 (Ar), M(0) = 29 (Cu) and M(1) = 28 (Ni) would be appropriate for the Ar → Cu/Ni system. Atom types are identified on the basis of their atomic numbers in *Spider* and *Kalypso*, so: **do not duplicate atomic numbers** (the programs will ignore the parameters associated with the second duplicated atom). Standard choices for the screening lengths can be obtained by selecting the appropriate radio-buttons in the box labelled **Set Screening Length** [⊙] (note: when you open a previously created Model file, the 'no action' radio-button is selected). The standard screening lengths are often modified by scaling with a correction factor that has a value typically in the range 0.7-1.0. If you wish to use such a screening length correction, enter the scaling factor in the appropriate edit box in the column designated **Correction**.

When a previously created Model file is reopened, the radio button items on page 1 of the Model file dialog box switch (Fig. 5.1, bottom left) to the **no action** setting. Manual selection of any other setting, ZBL or Molière, will cause the updating of the associated dialog boxes (e.g. any parameters that have been edited by hand will be modified).

5.2.2. Tight-binding potentials

The functional form of the tight-binding potentials was given in section 2.5 of Chapter 2. The potential parameters are A , ξ , p , q , r_0 , as defined in Eqs. 2.4b and 2.7 of the same chapter:

$$U_{ij}(r_{ij}) = A \exp(-p(r_{ij} / r_0 - 1)) \quad (2.4b)$$

$$\phi(r_{ij}) = \xi^2 \exp(-2q(r_{ij} / r_0 - 1)) \quad (2.7)$$

The inputs required for the Model file dialog box are $2A, \xi, p, 2q, r_0$ (the parameter b that also appears in the dialog box should be set to 0.0, since standard tight-binding potentials do not use it).

One set of parameters is required for each distinct interaction type (e.g. Cu-Cu, Cu-Ni, Ni-Cu, Ni-Ni). The heteronuclear interaction parameters $2q$ and ξ that appear in the attractive part of the potential, Eq. 2.7, do not have to be symmetric with respect to interchange of the atomic indices.¹ However, in most potentials found in the literature they *are* symmetric. The author has found that lifting this restriction may improve the transferability of the potentials. The repulsive (pairwise) parts of the potentials are always symmetric with respect to interchange of the atomic indices, and this requirement is enforced by *Spider* (by disabling incompatible input). The labelling convention used by *Spider* is similar to the used for the embedded atom method potentials: the parameter $\xi(0,1)$ is used to compute the interaction term $\phi(r_{01})$ for a central atom of type 0 when it is in proximity to a neighbouring atom of type 1.

To reduce labour, you can store potential parameters that you use frequently in a text file, and load them into the dialog box by clicking the button labelled **File**. One such file, `potentials.dat`, is supplied with the *Kalypso* distribution (in the `\bin` directory).

5.2.3. Switching functions

Switching functions are discussed in section 2.6 of Chapter 2. The parameters that the user must specify are: the upper and lower bounds to the core switching function (R_1, R_2), and the upper and lower bounds to the cut-off switching function (R_S, R_C - where R_C is the potential cut-off). Each switching function includes a scale factor that can be adjusted as necessary in order to improve the smoothness of the composite potential.

Choosing the optimum values is largely a matter of trial and error. The potential and force functions may be viewed immediately in the graphs provided in the dialog box (click the **Update** button). **To do this efficiently**: click the edit box whose value you wish to adjust, then press the arrow up \uparrow or arrow down \downarrow keys: this will step through a range of values, and will update the graphs automatically.

1. **Active pair**: the radio-button selection [\odot] in this box determines what potential and force functions are displayed on the Model file dialog box graphs, but it has no effect on the simulation.
2. **Core switching function (SF)**: (R_1, R_2) and the switching function scale are specified in this box for each type of interaction term that applies to your simulation. Terms that so not apply to your simulation are geyed out (disabled). Make sure that $R_1 < R_2$.
3. **Cut-off switching function (SF)**: (R_1, R_2) and the switching function scale are specified in this box. Make sure that $R_S < R_C$ and that $R_2 < R_S$.

The graphs in the Model file dialog box display, by default, the total potential and force functions for an interacting pair of atoms, (n, m), based on the parameters that are currently entered in the Model file dialog box, where n and m represent the atom

¹ Exception: *Kalypso* does require that the heteronuclear interaction parameters be symmetric if $b \neq 0.0$.

types specified in the Model file ($= 0$ or 1). The variable z in this case refers to the separation between the atoms. *Spider* uses all of the parameters currently specified in the Model file to do this calculation. For a heteronuclear pair of atoms ($A \neq B$), the calculations involve two tight-binding and switching function parameter sets $[(0,1)$ and $(1,0)]$.

In order to do the potential and forces calculations, *Spider* builds a minimal simulation project using atom A as the projectile and atom B as target, and runs it quietly (using the *Kalypso* simulation engine code). The scratch files used for this purpose can be found in the `\bin\temp` directory of the *Kalypso* distribution (it is safe to delete these files). If you wish, you can substitute your own Target file in place of atom B . For example, instead of an **atom-atom interaction**, you might wish to examine the potential associated with the interaction of atom A with a lattice composed of B atoms (i.e. an **atom-lattice interaction**). The Target file that defines the lattice of interest can be specified via the **Set-Up** button (the file into which the potential data are dumped can also be specified).

5.3. Model files for ion-scattering spectroscopy (ISS) simulations

Molecular dynamics (MD) is not the most efficient method for doing ISS simulations, but because of increased hardware speed, MD is now feasible. There may even be some advantages in using MD for this purpose when scattering cannot be described by binary collision processes, or when the surface has a complex structure.

Most ISS simulations do not require modelling of an attractive potential between the projectile and the target, or between target atoms. It is therefore recommended to reduce the cut-off distance to very small values (less than the nearest-neighbour distance of the lattice) which will speed up the simulation. In this type of simulation, it is normal to ignore all target-target interactions (the option for this is found in the General Specifications section of the Run file dialog box). This means that the parameters specified for the tight-binding attractive potential and the associated switching functions in the Model file are irrelevant because they will never be used (a) because of the short cut-off distance (below the range that is relevant for the tight-binding potential), (b) because all target-target interactions are ignored. It may be helpful to emphasise this by setting the tight-binding potential parameters $2A = \xi = b = 0$.

To give a specific example: for the He-Cu scattering system, you can set for the Cu-Cu interaction $2A = \xi = b = 0$, R_1, R_2 (core switching function range) = 1.2-1.7 Å, and R_s, R_c (cut-off switching function range) = 1.8-2.0 Å (the nearest neighbour distance in Cu is 2.56 Å). The 2.0 Å cut-off specified here will also apply to the He-Cu interaction, and is the only parameter that has any effect. The other parameters are chosen arbitrarily. They have no effect if the target-target interactions have been disabled (in the Run file). Although the parameters $2A, b, \xi, R_1, R_2, R_s$ are arbitrary, you should choose reasonable values that will not cause problems in the interpolation routines. For example, do not set $R_1 = 1.0$ Å, $R_2 = 1.000001$ Å.

The example projects supplied with the *Kalypso* package include a number of ISS simulations. You should study these example input files before attempting your own simulations.

6. THE IMPACT FILE

6.1. Function of the Impact file

The Impact file consists of rows of Cartesian coordinates (b_x , b_y , b_z , expressed in Å). The z coordinates, (b_z) in the Impact file always refer to the vertical starting position of the projectile relative to the z coordinate of the anchor atom (row #1) of the Target file. The significance of the x and y coordinates (b_x , b_y) depends on the **projectile mode** parameter that is specified in the Run file (section 4.2.3 of Chapter 4).

1. In **impinging** mode (the most common case) (b_x , b_y) represent the (x , y) **impact parameters of the projectile**, expressed relative to the coordinates of the anchor atom.
2. In **recoiling** and **mixed** mode, (b_x , b_y) are the **starting (x , y) coordinates of the projectile**, expressed relative to the (x , y) coordinates of the anchor atom.

The coordinates used for the anchor atom are based on those written in the Target file. Fig. 6.1 illustrates these distinctions, which are further discussed in sections 6.2-6.4.

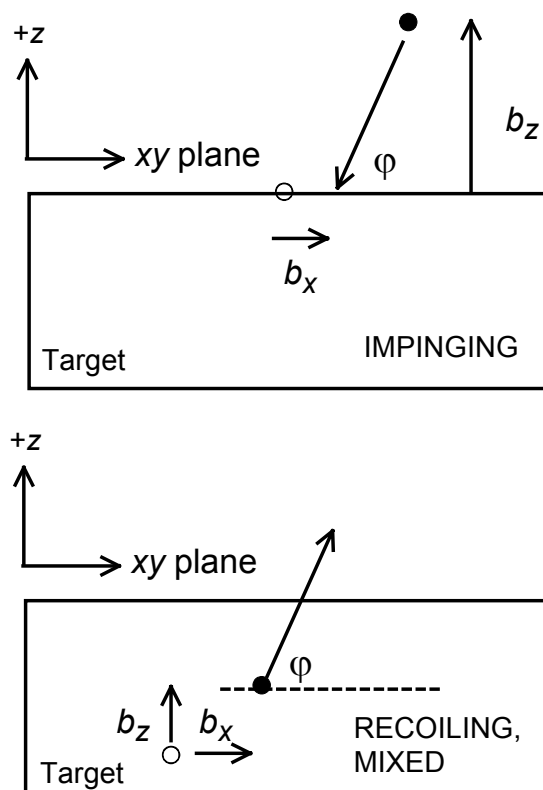


Fig. 6.1. Significance of the Impact file parameters for impinging, recoiling and mixed projectile modes. Symbols: ○ = anchor atom, ● = projectile atom.

6.2. Impinging mode

The **impinging** projectile mode is used for projectiles that approach the target from outside the target, e.g. simulations of sputtering, ion scattering, or thin-film deposition. If the simulation Run file specifies an azimuthal angle ϕ and the altitudinal angle φ , the projectile starting position in laboratory coordinates (x_0, y_0, z_0) is expressed relative to the coordinates of the anchor atom (x_1, y_1, z_1) as follows:

$$\begin{aligned} x_0 &= x_1 + b_z \cos \phi / \tan \varphi + b_x \\ y_0 &= y_1 + b_z \sin \phi / \tan \varphi + b_y \\ z_0 &= z_1 + b_z \end{aligned} \quad (6.1)$$

Eqs. 6.1 assume that the projectile positional coordinates (x_p, y_p, z_p) defined in the Projectile file (section 3.9) are (0, 0, 0) (default values). If this is not the case (e.g. for a cluster projectile) then $x_0 = x_1 + b_z \cos \phi / \tan \varphi + b_x + x_p$ and so on, i.e. the projectile atom starting position is displaced according to the coordinates read from the Projectile file.

For the special case of normal projectile incidence ($\varphi = 90^\circ$) the relationship (6.1) becomes:

$$\begin{aligned} x_0 &= x_1 + b_x \\ y_0 &= y_1 + b_y \\ z_0 &= z_1 + b_z \end{aligned} \quad (6.2)$$

It is important to grasp that the coordinates listed in the Impact file refer to a projectile position *relative* to the location of the anchor atom. This is not usually obvious from inspection of the Impact file, since the anchor atom is typically placed at (0, 0, 0) by default.

For completeness, the initial values of the projectile velocity components (Chapter 4, Fig. 4.2) are repeated here (v_{in} is the projectile incident velocity):

$$\begin{aligned} v_{0x} &= -v_{in} \cos \varphi \cos \phi \\ v_{0y} &= -v_{in} \cos \varphi \sin \phi \\ v_{0z} &= -v_{in} \sin \varphi \end{aligned} \quad (6.3)$$

6.3. Recoiling mode

The **recoiling** projectile mode is used for projectiles that originate from within the target and travel in the +z direction, e.g. simulations of a ‘fan’ of backscattered projectiles, or desorption processes. The projectile starting position (x_0, y_0, z_0) does **not** depend on the azimuthal angle ϕ and the altitudinal angle φ specified in the Run file, but is expressed relative to the coordinates of the anchor atom (x_1, y_1, z_1) as follows (cf. Eq. 6.1):

$$\begin{aligned}
x_0 &= x_1 + b_x \\
y_0 &= y_1 + b_y \\
z_0 &= z_1 + b_z
\end{aligned}
\tag{6.4}$$

However, the initial values of the projectile velocity components **do** depend on the azimuthal angle ϕ , and an altitudinal angle φ specified in the Run file:

$$\begin{aligned}
v_{0x} &= v_{in} \cos \varphi \cos \phi \\
v_{0y} &= v_{in} \cos \varphi \sin \phi \\
v_{0z} &= v_{in} \sin \varphi
\end{aligned}
\tag{6.5}$$

6.4. Mixed mode

The **mixed** projectile mode is used for projectiles that originate from within the target and travel in a variety of directions, e.g. simulations of Doppler-induced gamma ray recoils. The projectile starting position (x_0, y_0, z_0) in the mixed mode is identical to that for the recoiling mode (Eq. 6.4). The initial values of the projectile velocity components depend on the azimuthal angle ϕ and the altitudinal angle φ specified in the Run file:

$$\begin{aligned}
v_{0x} &= \pm v_{in} \cos \varphi \cos \phi \\
v_{0y} &= \pm v_{in} \cos \varphi \sin \phi \\
v_{0z} &= \pm v_{in} \sin \varphi
\end{aligned}
\tag{6.6}$$

(The \pm signs in Eq. 6.6 are selected randomly for each direction of motion.)

6.5. Impact file format

The first line of a .IMP file is formatted as follows:

```

0.000 0.000 4.5000 0.0000 2.5560 0.0000 1.2778 90.0 13 8 4.50
bx    by    bz    x1    x2    y1    y2    angle Nx Ny bz

```

The first three numbers represent (b_x, b_y, b_z) (in Å). The next five numbers are the inputs from the dialog box which are required for reading an Impact file from disk. The last three numbers (N_x, N_y, b_z) are used internally by *Spider*. In order to run a simulation, only the first three numbers are required, as found on subsequent lines of the Impact file.

The numbers in the Impact file must be delimited by one or more space characters, and may be written in a variety of conventional floating point formats (1.04, 1.04e0, 1.04E0 and so on). The format is very simple, and if you prefer, you could create your own Impact files without the help of *Spider*. (It is not necessary to include the extra inputs, cols. 4-11, on the first line in this case.)

6.6. Creating Impact files

The creation of Impact files is one of the most difficult aspects of a simulation project. The statistics of sampling is something which an experimenter never has to

think about, but it is important to get this right in a simulation model. The determination of the shape and size of the sampling area based on surface symmetry requires a familiarity with surface structure. This knowledge is assumed in the following sections. There is usually more than one way to generate a Target file and Impact file combination, so the procedures outlined here should be regarded as descriptive rather than prescriptive.

The Impact file must define a set of statistically representative points on the target surface. These represent a sample of points within a zone of irreducible symmetry, which will be called the primary impact zone (PIZ) here. The PIZ depends on both surface symmetry, and collision geometry (see Harrison *et al.*[1] for a discussion).

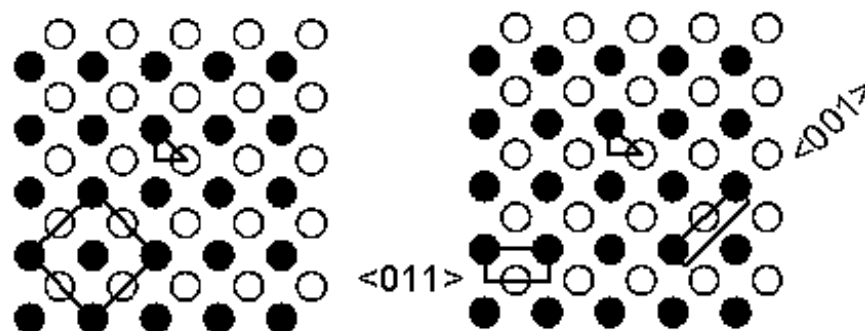


Fig. 6.2(a). Primary impact zones for the fcc (100) surface (normal and oblique incidence). Arrows (\rightarrow) indicate projectile direction of incidence. \bullet = surface layer atoms; \circ = 2nd layer atoms.

Fig. 6.2(a) illustrates the surface of a fcc (100) target as generated by Spider. On the left of the figure, the PIZ for a normally incident projectile ($\phi = 90^\circ$) is represented by the small triangle, while the square region (bottom left) represents one face of the bulk crystallographic unit cell. For off-normal projectile incidence, the PIZ shape depends on the azimuthal angle, ϕ . The right panel of Fig. 6.2(a) compares the PIZ shapes for normal, $\langle 011 \rangle$ and $\langle 001 \rangle$ directions of incidence. The area of the PIZ is the same in the last two cases, but the shapes are different.

For an arbitrary azimuthal angle with no special symmetry properties, the PIZ is equivalent to the area occupied by a face of the bulk unit cell. The unit cell could, in fact, be used for the other cases, but this is normally avoided because it is regarded as being statistically inefficient. Likewise, any integral multiple of the true PIZ is acceptable, if inefficient.

Spider generates impact points on a grid contained within the PIZ. Another approach (not available via *Spider*) would be to generate a set of points randomly within the PIZ.

There are two stages to the creation of an Impact file. **First**, you must identify the PIZ for your experimental system and configuration. **Second**, you must decide how to represent this PIZ using *Spider*.

Tables 6.1-6.6 summarise the procedures for creating some common PIZs for cubic (100) and (110) surfaces for non-normal and normal projectile incidence.¹ These are the easiest cases: (111) surfaces require a more complicated procedure (section 6.4.3). The tables employ the following abbreviations:

¹ Some serious errors discovered in these tables for bcc targets have been corrected in this version of the User Guide (Dec. 2005).

- N/A: Not applicable (parameter is irrelevant)
- a : bulk unit cell lattice parameter
- d_0 , d : See definitions in footnotes to individual tables.
- ϕ : Projectile azimuthal direction of incidence.
- R_{xy} column [with values $R45$, $R00$ etc.]: The azimuthal rotation ($^\circ$) which must be applied to the Target file *as generated by Spider* in order to align ϕ with the specified axial direction. ($R00$ means no rotation is required.) Use the Visualiser tool in *Spider* to apply these rotations. The tables assume that the projectile azimuthal angle of incidence specified in the Run file is zero (0.0°).
- $b_x(\text{min})$ etc.: refer to values entered for the corresponding fields in the Impact file dialog box.
- (x, y) angle: another field in the Impact file dialog box.
- **Normal** incidence means $\phi = 90^\circ$; **Non-normal** incidence means $\phi \neq 90^\circ$

Table 6.1. Fcc and diamond lattices: **Non-normal** projectile incidence.^a

Surface	ϕ	R_{xy}	$b_x(\text{min})$	$b_x(\text{max})$	$b_y(\text{min})$	$b_y(\text{max})$	(x, y) angle
(100)	$\langle 110 \rangle$	$R00$	0.0	d_0	0.0	$d_0/2.0$	90
(100)	$\langle 100 \rangle$	$R45$	0.0	$a/2.0$	0.0	$a/2.0$	90
(110)	$\langle 100 \rangle$	$R00$	0.0	a	0.0	$d_0/2.0$	90
(110)	$\langle 110 \rangle$	$R90$	0.0	d_0	0.0	$a/2.0$	90

^a a is the fcc or diamond unit cell parameter, while $d_0 = a/\sqrt{2}$.

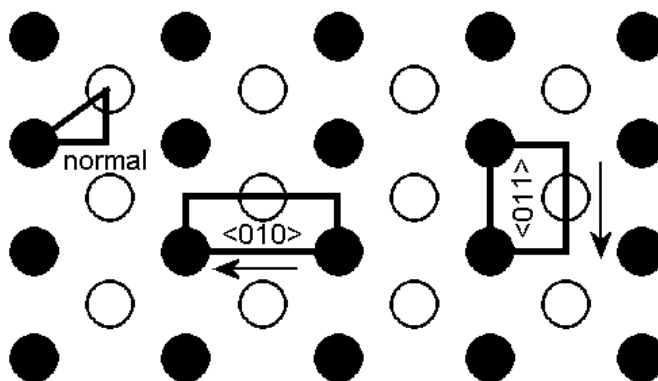


Fig. 6.2(b). Primary impact zones for the **fcc (110)** surface (normal and oblique incidence). Arrows (\rightarrow) indicate projectile direction of incidence. \bullet = surface layer atoms; \circ = 2nd layer atoms.

Table 6.2. Fcc and diamond lattices: **Normal** projectile incidence.^a

Surface	ϕ	R_{xy}	$b_x(\text{min})$	$b_x(\text{max})$	$b_y(\text{min})$	$b_y(\text{max})$	(x, y) angle
(100)	N/A	$R00$	0.0	$d_0/2.0$	0.0	$d_0/2.0$	45
(110)	N/A	$R00$	0.0	$a/2.0$	0.0	d_0	35.2644

^a a is the fcc or diamond unit cell parameter, while $d_0 = a/\sqrt{2}$.

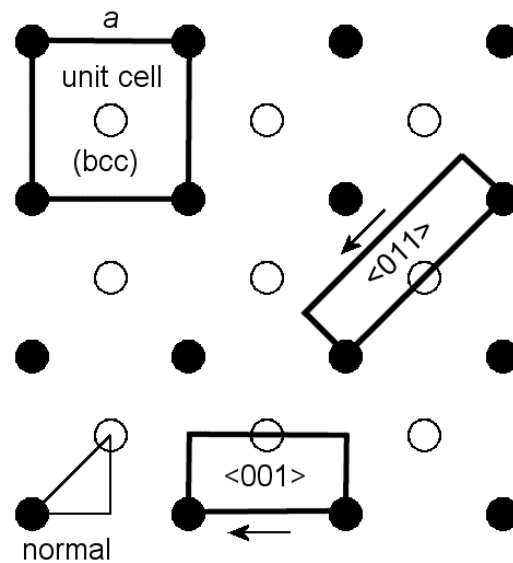


Fig. 6.2(b). Primary impact zones for the **bcc (100)** surface (normal and oblique incidence). The bulk bcc unit cell is also indicated. Arrows (\rightarrow) indicate projectile direction of incidence. \bullet = surface layer atoms; \circ = 2nd layer atoms.

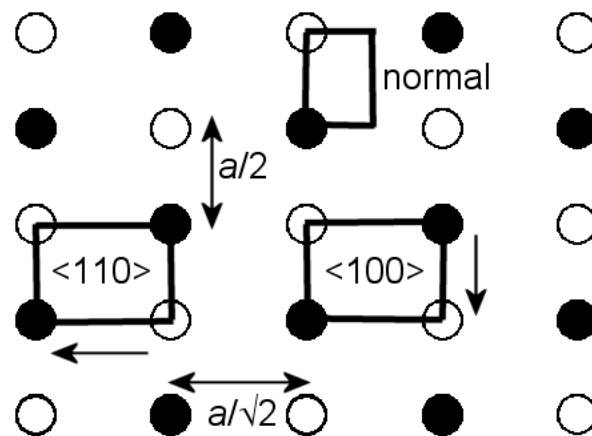


Fig. 6.2(c). Primary impact zones for the **bcc (110)** surface (normal and oblique incidence). Arrows (\rightarrow) indicate projectile direction of incidence. \bullet = surface layer atoms; \circ = 2nd layer atoms.

Table 6.3. Bcc lattice: **Non-normal** projectile incidence.^a

Surface	ϕ	R_{xy}	$b_x(\text{min})$	$b_x(\text{max})$	$b_y(\text{min})$	$b_y(\text{max})$	(x, y) angle
(100)	$\langle 100 \rangle$	R_{00}	0.0	a	0.0	$0.5a$	90
(100)	$\langle 110 \rangle$	R_{45}	0.0	$\sqrt{2} \times a$	0.0	$a/(2\sqrt{2})$	90
(110)	$\langle 110 \rangle$	R_{00}	0.0	$a/\sqrt{2}$	0.0	$0.5a$	90
(110)	$\langle 100 \rangle$	R_{90}	0.0	$0.5a$	0.0	$a/\sqrt{2}$	90

^a a is the bcc unit cell parameter.Table 6.4. Bcc lattice: **Normal** projectile incidence.^a

Surface	ϕ	R_{xy}	$b_x(\text{min})$	$b_x(\text{max})$	$b_y(\text{min})$	$b_y(\text{max})$	(x, y) angle
(100)	N/A	R_{00}	0.0	$a/2.0$	0.0	$a/2.0$	45
(110)	N/A	R_{00}	0.0	$a/(2\sqrt{2})$	0.0	$a/2.0$	90

^a a is the bcc unit cell parameter.Table 6.5 Rocksalt lattice: **Non-normal** projectile incidence^a

Surface	ϕ	R_{xy}	$b_x(\text{min})$	$b_x(\text{max})$	$b_y(\text{min})$	$b_y(\text{max})$	(x, y) angle
(100)	$\langle 100 \rangle$	R_{00}	0.0	a	0.0	$a/4.0$	90
(100)	$\langle 110 \rangle$	R_{45}	0.0	$a/\sqrt{2}$	0.0	$a/(2\sqrt{2})$	90

^a a is the rocksalt unit cell parameter.Table 6.6 Rocksalt lattice: **Normal** projectile incidence^a

Surface	ϕ	R_{xy}	$b_x(\text{min})$	$b_x(\text{max})$	$b_y(\text{min})$	$b_y(\text{max})$	(x, y) angle
(100)	N/A	R_{45}	0.0	$d/2.0$	0.0	$d/2.0$	45

^a a is the rocksalt unit cell parameter, while $d = a/\sqrt{2}$.

6.7. Impact file examples

6.7.1. Normal projectile incidence on Cu(110) surface

An Impact file that covers the PIZ for normal incidence on a cubic lattice (100) or (110) surface can be generated relatively easily. For fcc, bcc and diamond lattices, the

PIZ can be used immediately with the corresponding Target file generated by *Spider*; in other words, no azimuthal rotation of the target is required ($R_{xy} = 0.0$, represented as $R00$ in Tables 6.4 and 6.5).

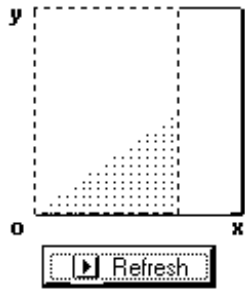
Before you begin, you must know the lattice parameter (a) for Cu ($= 3.6147 \text{ \AA}$) and compute the parameter d_0 defined in Table 6.4 as \sqrt{a} (i.e. $d_0 = 2.55598 \text{ \AA}$). Select the Impact|New menu item in *Spider*. A dialog box comes up. With reference to Table 6.4, enter the values shown in Fig. 6.3, then click the Refresh button to see the PIZ points represented in the Impact file (you should see a triangular region, as in the figure).

Impact file

Impact parameters

bx (min) (Å)	bx (max) (Å)	No. impacts
0.0	1.80735	18
by (min) (Å)	by (max) (Å)	No. impacts
0.0	2.55598	26
(x,y) angle (deg.)		35.2644
Projectile-anchor Δz (Å)		4.5

120 impact points.
 $x = 0.1789186$
 $y = 2.5304202$



Save Save As Close Help

Fig. 6.3. Impact file dialog box that defines the PIZ for normal projectile incidence on Cu(110).

Note the following points:

- The No. Impacts ("Number of Impacts") values determine the mesh widths of the impact points in the x - and y -directions respectively. In this case, the values reflect the ratio $b_x(\text{max}):b_y(\text{max})$. The number of impact points required in the Impact file depends on the goal of the simulation. For a sputtering simulation, 500-1000 impact points would be typical, whereas for an angle-resolved ion scattering spectrum simulation, you might need 10^5 impact points if backscattering is weak.
- The Projectile z_0 parameter indicates the starting projectile z -coordinate relative to the anchor atom. in this case, the projectile z -coordinate will be initialised as $z_{\text{anchor atom}} + 4.5 \text{ \AA}$.
- Since $b_x(\text{min})$ and $b_y(\text{min})$ are both zero, the first projectile trajectory will start off directly above the anchor atom (see Eq. 6.2).
- The (x, y) angle parameter refers to the bottom, left angle of the triangle (dotted region).

Click the OK button and save the file as 'cu110.imp'. It is always wise to check the dimensions of the PIZ.

- To do this, go to Target|New|Face-centred cubic, and generate a 2-layer Cu(110) lattice.
- Examine this Target file with the Visualiser tool that is located on the Target menu.
- Next, select **New Imposition File** on the menu associated with the Visualiser window, and load the Impact file 'cu110.imp' for display.
- You should now see the relationship between the surface symmetry and the PIZ which you generated, as in Fig. 6.4 (click Zoom + Impose to expand the Visualiser display).

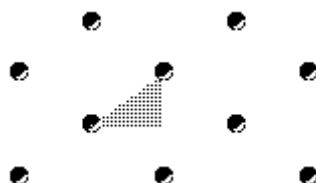


Fig. 6.4. Primary impact zone (PIZ) for normal projectile incidence on a Cu(110) surface.

6.7.2. Non-normal projectile incidence on Cu(100) surface: $\phi = \langle 001 \rangle$

If you refer to Table 6.1, you will see that this configuration requires a Target file which is rotated azimuthally by 45° ($R_{xy} = R45$). The reason for this is that the Cu(100) Target file is created, by default, with $\langle 011 \rangle$ edges parallel to the x - and y -directions. However, we need to match the Target file with a PIZ which has edges parallel to $\langle 001 \rangle$. [2] We will thus first create the Impact file, then briefly show how to rotate the Target file.

Impact file ✕

Impact parameters

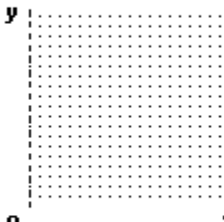
bx (min) (Å)	bx (max) (Å)	No. impacts	400 impact points. x = y = 
<input type="text" value="0.0"/>	<input type="text" value="1.80735"/>	<input type="text" value="20"/>	
by (min) (Å)	by (max) (Å)	No. impacts	
<input type="text" value="0.0"/>	<input type="text" value="1.80735"/>	<input type="text" value="20"/>	
(x,y) angle (deg.)		<input type="text" value="90"/>	
Projectile-anchor Δz (Å)		<input type="text" value="4.5"/>	
			<input type="button" value="Refresh"/>

Fig. 6.4. Impact file dialog box that defines the PIZ for $\langle 001 \rangle$ azimuthal projectile incidence on Cu(100).

The Impact file is created using the inputs shown in Fig. 6.4 (with reference to Table 6.1, using values of a and d_0 appropriate to Cu from the previous example).

The PIZ is a square region. Next use the Target|New|Fcc|(100) command to generate a Cu(100) Target file, and save it as cu100a.trg. The coordinates in this Target file must be rotated azimuthally by 45° before the file can be used in conjunction with the Impact file.

- The first step is to display cu100a.trg using the Target|Visualiser command.
- Enter 45 into the z-axis rotation field in this dialog box (top right – see Fig. 6.5) and click the following buttons in this order: Test, Reload, Apply. The last command will allow you to save the rotated file to disk (e.g. as cu100b.trg).
- Before doing so, you may wish to superimpose the PIZ defined in your Impact file, as explained in the preceding section. If so, click Commands|New Imposition file file, and select cu100.imp.
- Click the Impose button whenever you need to refresh the display of the PIZ. You will see that the PIZ reflects the symmetry of the rotated file (i.e. after you click Test), but not that of the original file (cu100a.trg).

Operate on data

Rotations (degrees)	
Z-axis rotation, R(x,y)	<input type="text" value="45"/>
X-axis rotation, R(y,z)	<input type="text" value="0"/>
Y-axis rotation, R(z,x)	<input type="text" value="0"/>

Fig. 6.5. Target file azimuthal rotation specification in the Visualiser tool.

6.7.3. Impact file for fcc (111) surface, normal incidence

In this case, it is necessary to trim the default PIZ generated by the Impact|New command to the desired shape (equilateral triangle) using the Visualiser tool. The difficulty in generating Impact files for the more unfamiliar cases is not in using the tools, but in knowing the crystallography of your surface. It is easy to make mistakes, so you should always check that the final results are reasonable using *Spider's* graphical tools, and by any other means available to you.

Fig. 6.6 illustrates the goal of the procedure: i.e. a suitable distribution of impact points for normal projectile incidence on a (111) surface. The surface atoms of Al(111) form equilateral triangles. In a simulation we direct projectiles into a symmetry-reduced triangular area. The apexes of this triangle are located at the positions of atoms in 3 different layers of the (111) surface. *Spider* does not generate the symmetry-reduced Impact file directly: it has to be trimmed and aligned according to the procedures now described.

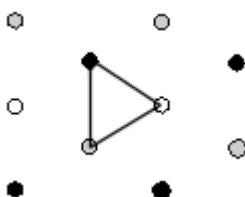


Fig. 6.6. Shape of the PIZ for a fcc(111) surface. Atoms from the 1st, 2nd and 3rd layers are represented by white, grey and black symbols respectively.

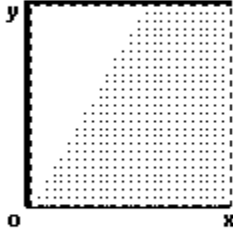
- Use the following data for Al: lattice const. (a) = 4.0495 Å, nearest neighbour distance (d_{NN}) = 2.8634 Å.
- Select New on the Impact menu. Name the file (*.IMP) as you like.
- Enter the data shown in Fig. 6.7 into the Impact file dialog box (replace 1.653184 by $d_{NN}/\sqrt{3}$ for other fcc crystals).

Impact file

Impact parameters

bx (min) (Å)	bx (max) (Å)	No. impacts
0.0	1.653184	26
by (min) (Å)	by (max) (Å)	No. impacts
0.0	1.653184	26
(x,y) angle (deg.)		60
Projectile-anchor Δz (Å)		4.5

476 impact points.
x =
y =



o x

Refresh

Save Save As Close Help

Fig. 6.7. Impact file dialog box used as **first step** in construction of the PIZ for normal azimuthal projectile incidence on Al(111).

- The number of impact points this data generates is 476. Some of these will be trimmed off later. You may of course select any number other than 26, according to your preferences. Choose the same number for both x and y dimensions to ensure a uniform distribution of points (advisable rather than mandatory).
- Now select the Impact|Visualiser menu command and load the Impact file you created in the previous steps. You need to reorient and cut the mesh of points in this file so that they match the dimensions of the PIZ. This involves two operations.
- Enter 60 for the z -axis rotation ($^\circ$), and 1.4317 (or $d_{NN}/2$ for other fcc crystals) for the maximum y -axis limit. Leave the other limits at their default values. Click the Test button. This will rotate the displayed mesh by 60 degrees, and clip its edge above $y = 1.4317$.
- The Impact file resulting from this operation has 306 impact point coordinates (compared with 476 originally). If you are satisfied with the test, you can then click the Reload button, followed by the Apply button, to carry out the same operation and save the resulting data to disk under a new name.
- Close the Visualiser, then open it again, now loading the new Impact file.
- Enter -30 (minus 30) for the z -axis rotation, and reset the maximum y -axis value to its default value (1000 Å). Click Test. This will rotate the mesh into the final

alignment shown in Fig. 6.6. Finally, click Reload then Apply to save the oriented Impact file to disk.

- The PIZ covered in the above exercise is only valid for normal primary ion incidence on a (111) surface. For oblique angles of incidence on a you will need to use a larger zone, whose dimensions depend on the primary ion azimuthal direction.

6.7.4. Impact files for multiple impact simulations

The coordinates in the Impact file are generated systematically, which is generally not suitable for simulating random projectile impacts in a multiple impact simulation. For this purpose, Spider provides the **Impact|Randomise order** menu command which will randomly re-order the coordinates in an Impact file. You can also change the order of an Impact file by loading it into a spreadsheet, adding a column of random numbers, and performing a sort operation using the random numbers as keys.

References and notes for Chapter 6

[1] D.E. Harrison, C.E. Carlston, G.D. Magnuson, Phys. Rev. 139 (1965) A737.

[2] Of course, it is also possible to rotate the PIZ instead of the target, but we choose not to for this example.

7. THE INELASTIC FILE

7.1. Introduction

Energetic particles can lose energy through excitation of valence or core electrons, or lattice vibrations. These energy losses are termed **inelastic losses** [1]. The term **electronic stopping** is also used to describe this type of energy loss. The parameters for loss processes of this type are specified in the Inelastic file. The Inelastic file is an optional part of a simulation project, in the sense that you only have to include it if you plan to include non-elastic effects in your simulation. A number of other non-elastic energy transfer effects can also be introduced into the simulation via parameters that are set in the Inelastic file: **temperature control** (heating and cooling), **image potential effects**, and **lattice site ‘springs’**.

What to include in the simulation

- ☒ Include LSS inelastic losses
- ☒ Include OR inelastic losses
- ☒ Include ST inelastic losses
- ☒ Apply temperature control
- ☐ Include image potential effects
- ☐ Include lattice site springs

Calculate LSS and OR parameters (utility only - not part of input data)

Energy loss model

☒ LSS model

☐ OR model

K(LSS) (eV fs/Å²)

Atomic number of fast atom (Z1)

Atomic number of stopping atoms (Z2)

Atomic weight of stopping atoms (M2)

Density of stopping atoms (g/cm3)

☐ Permute Z1, Z2

$$-\frac{dE}{dx} = 8\pi N_A \left(\frac{e^2}{4\pi\epsilon_0} \right) \alpha_B \frac{Z_1^{7/6} Z_2}{(Z_1^{2/3} + Z_2^{2/3})^{3/2}} v / v_B$$

$-dE/dx = K(LSS) * v$

Lindhard-Scharff-Schiott model

	Z1	K(LSS)	scale
atom 1	18	9.509670	0.5
atom 2	29	13.31823	0.5

Atoms with Z=0 are ignored at runtime

Velocity threshold (m/s)

Location of surface (on z-axis) (Å)

Oen-Robinson model

	Z1*Z2	K(OR)	scale
pair 1	18	0.00112	0.5
pair 2	29	0.00157	0.5

Pairs with Z1*Z2=0 are ignored at runtime

Max. apsidal distance (Å)

Exponential parameter

Fig. 7.1. Inelastic file input dialog box (page 1 of 2).

Kalypso permits the modelling of electronic stopping effects via a combination of three distinct models: (a) the Lindhard-Schiott-Scharff (LSS) model; (b) the Oen-Robinson (OR) model; (c) the Shapiro-Tombrello (ST) model. The parameters for these effects are specified in the Inelastic file. 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. Eckstein's book [2] gives a good introduction to the LSS and OR models.

Fig. 7.1 shows the first of two pages of the Inelastic file dialog box in *Spider* that is used to input the parameters for the electronic stopping models. 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 $\text{Ar} \rightarrow \text{Cu}(100)$ 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 $\text{Ar} \rightarrow \text{Cu}(100)$ 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 omit parameters for the Cu-Cu interaction).

The implementation of each of the electronic stopping models in *Kalypso* takes into account the fact that different individual or pairs of atoms are associated with different inelastic loss parameters. This is achieved by ‘tagging’ each set of parameters with a number, which is used like an index or hashing function at runtime to look up the appropriate set of parameters.

For the LSS model, which represents electronic stopping of an atom by a continuum target material, the tag is the atomic number of the atom which is being stopped.

For the OR and ST models, which predict the inelastic energy loss in a binary collision event, the tag is the *product* of the atomic numbers of the participating atoms, i.e. $Z_1 Z_2$. The tag for an Ar-Ar collision event is thus 324, while for an Ar-Cu event it is 522. Up to 10 different tags may be placed in the same Inelastic file.

For two-dimensional ISS simulations, the only electronic stopping model than can be used is the LSS model (the OR and ST models will produce three-dimensional motion). (Cooling effects can also be applied, but these are not likely to be useful.)

7.2. Lindhard-Schiott-Scharff (LSS) model

7.2.1. Theory

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) [3,4]:

$$-\frac{dE}{dx} = 8\pi N \left(\frac{e^2}{4\pi\epsilon_0} \right) a_B \frac{Z_1^{7/6} Z_2}{(Z_1^{2/3} + Z_2^{2/3})^{3/2}} v / v_B \quad (7.1)$$

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 can be written as:

$$-\frac{dE}{dx} = K(\text{LSS}) \cdot v \quad (7.2)$$

where $K(\text{LSS})$ ($\text{eV fs } \text{\AA}^{-2}$) is a constant that depends on Z_1 and Z_2 . The LSS model thus views the target as a viscous medium, and describes a continuous, friction-like, electronic energy loss process that arises from passage through this medium. The LSS energy loss algorithm does not conserve linear or angular momentum.

To illustrate a rough calculation based on the LSS model: consider an Ar atom with energy 5 keV (velocity $v \sim 1.6 \text{ \AA fs}^{-1}$) moving a distance $\Delta l = 1 \text{ \AA}$ in a target that is characterised by a $K(\text{LSS})$ value of $\sim 10 \text{ eV fs } \text{\AA}^{-2}$ (typical). Then the average inelastic energy loss is predicted to be $\Delta E \sim K(\text{LSS}) \times \Delta l \times v = 16 \text{ eV}$.

Spider allows the definition of $K(\text{LSS})$ given in Eqs. 7.1 and 7.2. to be scaled by an arbitrary factor (e.g. a factor of 0.5 is appropriate if you plan 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 Eq. 7.1 requires some careful thought, that will be guided by the user's physical intuition about the correct 'effective' atomic number of the target. For example, use $Z_2 = 28$ if it is a $\text{Ni}_{0.95}\text{Cu}_{0.05}$ alloy.

7.2.2. LSS parameters

For each atom type that will be affected by LSS energy losses in your simulation, you must specify a set of the following parameters (use the **Add Atom** and **Clear Atom** buttons to add/remove items from the input list):

1. **Z_1** : the atomic number of the moving projectile or target atom which is being stopped by your target (this value is used as a tag only).
2. **$K(\text{LSS})$** : a coefficient (units: eV fs \AA^{-2}) which reflects the stopping power of the target matter for an atom with atomic number Z_1 . K depends on several other parameters, and can be computed most easily by using the **Compute $K(\text{LSS})$, $K(\text{OR})$** gadget in the input dialog box.
3. **Scale**: this is a value by which $K(\text{LSS})$ is scaled at runtime in the simulation (if this value of zero, .
4. **Velocity threshold**: this parameter sets a velocity threshold (in m s^{-1}) for the stopped particle below which LSS energy losses are ignored, i.e. losses are not calculated for slowly moving lattice atoms.
5. **Location of surface**: LSS losses will be applied to a particle only if its z -coordinate is below the value specified here (which should reflect the location of the boundary between the surface electron density and the vacuum, not the nuclear locations - i.e. about half of an interlayer distance). This parameter should be less than the cut-off distance.

7.3. Oen-Robinson (OR) Model

7.3.1. Theory

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 [5]:

$$-\Delta E = 8\pi \left(\frac{e^2}{4\pi\epsilon_0} \right) a_0 \frac{Z_1^{7/6} Z_2}{(Z_1^{2/3} + Z_2^{2/3})^{3/2}} v / v_B \frac{(c/a)^2}{2\pi} e^{-(c/a)R_0} \quad (7.3)$$

In Eq. 7.3, 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 in *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:

$$v = \sqrt{\frac{1}{2}\mu \cdot v(t)^2 + V(r(t))} \quad (7.4)$$

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 *Kalypso* corresponds to the apsidal point); μ is the reduced mass of the system.

The OR model postulates the occurrence 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. 7.1 and 7.3 is brought out if the latter is written in the form:

$$-\Delta E_{OR} = 8\pi \left(\frac{e^2}{4\pi\epsilon_0} \right) a_B \frac{Z_1^{7/6} Z_2}{(Z_1^{2/3} + Z_2^{2/3})^{3/2}} F_{OR} \cdot v / v_B \quad (7.5)$$

$$-\Delta E_{OR} = K(OR) \cdot F_{OR} \cdot v \quad (7.6)$$

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 7.3 and 7.5. $K(OR)$ has 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. 7.3, 7.5 and 7.6.

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. *Kalypso* 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 OR energy loss algorithm conserves linear momentum, but not angular momentum.

The manner in which the computed OR energy loss (ΔE) is implemented in *Kalypso* is similar to the method used for the ST model described in the next section.

7.3.2. OR parameters

The following parameters must be entered for each atomic collision pair that will be affected by OR energy losses in your simulation (use the **Add Pair** and **Clear Pair** buttons to add/remove items from the input list):

1. **$Z_1 * Z_2$** : this number is the product of the atomic numbers of the pair of atoms which are involved in a binary collision (this value is used as a tag only).
2. **$K(OR)$** : a coefficient (units: eV Å²) that reflects the energy loss or electronic stopping effect of the binary collision (see previous section). K depends on several other parameters, and can be computed most easily by using the **Compute $K(LSS)$, $K(OR)$** gadget in the input dialog box.
3. **Maximum apsidal distance**: this parameter (maximum value 3 Å) sets a maximum apsidal distance (distance of closest approach) above which OR energy

losses are ignored, i.e. losses will only be calculated for 'collisions' which involve encounters closer than this value. The parameter should be chosen such that $F(\text{OR})$, when evaluated at this separation, makes a negligible contribution to the total stopping electronic stopping. The best way to set the parameter is by trial and error (observing the reported OR loss from trial simulations). For 5 keV Ar-Cu, $a = 0.116$ Å, R_{\min} at zero impact parameter is 0.36 Å, so with $c = 0.3$, the exponential term in $F(\text{OR})$ is 0.39. For $R_{\min} = 1.0$ Å, the same term decreases to 0.08, and for $R_{\min} = 1.3$, it has a value of 0.03. The latter is not negligible, because the elementary losses have to be weighted by cross-section factors which vary roughly as $(R_{\min})^2$. In general, R_{\min} should be chosen to be as small as possible, because large values may impact on the speed of the calculations.

7.4. Shapiro-Tombrello (ST) model

7.4.1. Theory

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 *Kalypso* will now be given. For a deeper review of the physics involved, the reader is referred to the original literature [6].

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 $\Delta E \cdot N_{\text{MAX}}$). 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 $\sim N/2$ 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:

$$\begin{aligned} \Delta E_T &\leq N\Delta E \\ \Delta E_T &= k\Delta E \text{ (with } k = 1, 2, 3, \dots, N) \end{aligned} \tag{7.7}$$

(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:

$$\langle \Delta E_T \rangle = Np\Delta E \tag{7.8}.$$

The inelastic energy loss corrections for both the ST and OR models are applied at the apsis of the collision.¹ 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 estimate would be exact if the potential declined linearly with separation (r).

Shapiro-Tombrello model

	\AA	eV				
	Z1*Z2	Rcrit	dE	prob.	Nmax	
pair 1	522	0.6	10	0.5	2	Add pair
pair 2	841	0.8	10	0.5	2	Clear pair

Pairs that have Z1 or Z2 = 0 will be ignored at runtime

Temperature control

Cooling period (fs)

Temperature (K)

Cooling start time (fs)

Cooling stop time (fs)

Temperature control effects will be applied to all atoms in the lattice region, except those that have the 'no-cooling' flag set.

Image potential
$$V(z) = \begin{cases} \frac{-e^2}{4\pi\epsilon_0 \sqrt{16(z-z_0)^2 + (e^2 / 4\pi\epsilon_0 V_{\min}^2)^2}}, & z > z_0 \\ -V_{\min}, & z \leq z_0 \end{cases}$$

Vmin (eV) z0 (Å)

The image potential will be experienced by atoms that have the 'image potential' flag set.

Lattice site springs

Type of atom M(0) M(1)

Force constants (eV/Å²)

☐ Snap springs

Snapping vx, vy, vz (m/s)

Springs will be attached to lattice sites whose atoms have the 'springs' flag set

Fig. 7.2. Inelastic file input dialog box (page 2 of 2).

Kalypso uses a similar procedure, but applies 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 *Kalypso* typically computes the correct displacement Δr to within $\sim 10^{-4}$ Å or better, but such is the nature of the repulsive 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 energy loss algorithm conserves linear momentum, but not angular momentum.

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, this correction would make very little difference to the collision dynamics (and no difference at all for direct impact collisions).

7.4.2. ST parameters

The following parameters must be entered for each atomic collision pair that will be affected by ST energy losses in your simulation (use the **Add Pair** and **Clear Pair** buttons to add/remove items from the input list):

1. **Z₁*Z₂**: this number is the product of the atomic numbers of the pair of atoms which is involved in a binary collision (this value is used as a tag only).
2. **R_{crit}**: the critical distance (in Å) for excitation of the level-crossing transition.
3. **dE**: this is the inelastic energy loss (in eV) resulting from the transition (ΔE in Eq. 7.8).
4. **Prob**: the probability of the transition (p in eq. 7.8).
5. **N_{max}**: the maximum number of electrons which can be promoted in the transition.

7.5. Temperature control

Kalypso allows the target to be cooled or heated towards, or maintained at, 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 desired temperature (T_0), the cooling or heating period (i.e. a time constant) (τ), and the time interval (start and end) of the period during which the temperature corrections are to be applied. These data are entered in the Inelastic file dialog box shown in Fig. 7.2. The equation of motion of each particle affected by the temperature correction is written as [7]:

$$m \frac{d^2 \mathbf{r}}{dt^2} = \mathbf{F}(\mathbf{r}) - \frac{m}{2\tau} (1 - T_0 / T) \frac{d\mathbf{r}}{dt} \quad (7.9)$$

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 via: $\langle KE \rangle = 3/2 kT$.

If the instantaneous lattice temperature is absolute zero ($T = 0$), Eq. (7.9) cannot be applied, and the correction 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 basal layer with constant temperature that acts as a thermal sink), you can set the `oFNoCool` option flag (see section 3.4 in Chapter 3) for the particles which are to be ignored (a) when calculating the current temperature, and (b) when applying the temperature correction, as in Eq. 7.9. Regardless of their 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.

The temperature control algorithm, Eq. 7.9, does not conserve energy or momentum. It will not perform well if a large amount of energy is leaving the system through sputtering or if hard energetic collisions are taking place, since these tend to absorb and release kinetic energy sporadically. 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.

7.6. Image potential effects

The model implemented in *Kalypso* for the image potential, $V(z)$, is one of several found in the literature [8]:

$$V(z) = \begin{cases} \frac{-e^2}{4\pi\epsilon_0 \sqrt{16(z-z_0)^2 + (e^2 / 4\pi\epsilon_0 V_{\min}^2)^2}}, & z > z_0 \\ -V_{\min}, & z \leq z_0 \end{cases} \quad (7.10)$$

(see [9] for a variation on this equation). The parameters z_0 and V_{\min} are specified by the user. The image potential is only applied to particles that have the `ofUseImagePotential` flag (= 1) set (see section 3.4 in Chapter 3). If you forget to set this flag, a warning message will be issued.

The most likely reason for using an image potential is to modify the projectile motion. In this case, you should open up the Projectile file and add 1 to the flags field, as shown below:

```
0.0 0.0 0.0 2 4.0026 0 98195.04398 He
[200 eV He Projectile file as generated by Spider]
```

```
0.0 0.0 0.0 2 4.0026 1 98195.04398 He
[200 eV He Projectile file with image potential flag set]
```

See the `\examples\projects\iss\200 eV He-Cu(110)-image pot1` example in the *Kalypso* distribution for a complete project that employs the image potential.

7.7. Lattice site springs

In this model a target atom is attached to its lattice site by a ‘spring’ whose force constant is specified by the user [10]. Thus, the atom will tend to return to its lattice site if it suffers a displacement. In order to be affected by this option, the particle must have its `ofSpungAtom` flag set (section 3.4 of Chapter 3). The object of using such a spring is to improve the stability of a target lattice at its edges, e.g. to prevent reconstruction.

An option is provided to ‘snap’ (disable) the springs for atoms while the following condition is fulfilled for any Cartesian dimension (x , y , or z axis): **the lattice site is located between the atom’s current position and the Target file origin, and the atom is moving back towards the lattice site with a speed that exceeds the specified threshold value** (e.g. based on lattice temperature). For example, suppose the lattice site was located at $z = 20$ Å, and the atom was located at $z = 21$ Å. Then if v_z was negative with a magnitude above the threshold value, the spring would be snapped (but not otherwise). This prevents a fast atom slamming into the lattice under the pull of its spring. Energy is not conserved when a spring is snapped.

7.8. Compute K(LSS), K(OR) tool

This tool, which is illustrated in Fig. 7.1 (bottom left), computes the ‘ K ’ parameters that are needed as inputs to the LSS and OR inelastic loss models. The two models require different input parameters, with different physical dimensions. In particular,

the LSS model treats the target as an homogenous medium, so you may have to define the target atomic number, Z_2 , as some effective or non-integral average value if your target is inhomogenous (e.g. an alloy).

The formulae used to calculate the LSS and OR parameters respectively are discussed in preceding sections (7.2, 7.3). The term $F(\text{OR})$ represents the ‘Oen-Robinson factor’, which is a decaying exponential function of R_{min} , the collision distance of closest approach (apsidal distance).

The LSS asymmetry between (moving) projectile and (stationary) target atom is also reflected in the original formulation of the OR model, where the Z_1 and Z_2 terms appear raised to different powers. The implementation used here allows you to optionally "permute" Z_1 and Z_2 , and thereby remove the asymmetry (an average value of K is then computed by swapping the roles of Z_1 and Z_2).

To use the tool, select an energy loss model (LSS or OR), then click on the **Evaluate** button to compute a $K(\text{LSS})$ or $K(\text{OR})$ value for your simulation. This value is shown in the grey edit box. It can be copied (Ctrl-C) and pasted (Ctrl-V) directly into one of the input cells on the relevant (LSS or OR) tab sheet.

7.9. Inelastic data file

An output option (selected in pane 1 of the Run file dialog box: ‘Output log of inelastic events’) in *Kalypso* is to produce the so-called inelastic data file (*.dat). This file records information about inelastic energy losses associated with the LSS, OR and ST models, including discrete energy loss events for the OR and ST models. Most users will not be interested in this file, so you can usually omit this option.

The inelastic (output) data file is ascii file consisting of 1 line per recorded OR or ST inelastic event with the following format (integer fields in **bold** font):

```
RN, i, j, dE, rlast, rcurr, zapsis, ti, Erel, alt, phi, bx, by, mod, Ne
```

where:

1. **RN** = run number (rn) (1, 2, 3 ...): the simulation run in which the event occurred
2. **i**, **j** = the row numbers (rw) which label the atoms involved (rw = 1 for a single projectile atom, and 2, 3... for target atoms)
3. dE = inelastic energy loss (eV)
4. rlast = separation of atoms **i** and **j** before the inelastic algorithm was applied (in Å); this is an estimate of the distance of closest approach in the collision (apsidal distance)
5. rcurr = separation of atoms **i** and **j** after the inelastic algorithm was applied (in Å); the physical meaning is not well-defined – see above for explanation, or read the literature on the ST algorithm.
6. zapsis = the *z*-coordinate (i.e. depth) at which the inelastic event took place (in Å)
7. ti: time elapsed at the moment of the inelastic event (fs)
8. Erel = relative energy of the collision responsible for the inelastic event (eV) (i.e. energy of the two-particle system, not including the kinetic energy associated with the motion of the centre-of-mass; see a mechanics textbook for explanation.
9. alt, phi = altitudinal (φ) and azimuthal (ϕ) angles of incidence ($^\circ$) for the current run
10. bx, by = impact file (*x*, *y*) coordinates for the current run
11. mod = a label that specifies the inelastic model (= 0 for Oen-Robinson, = 1 for Shapiro-Tombrello)

12. Ne = Number of electrons promoted (only Shapiro-Tombrello model has non-zero values)

No special support is provided for further analysis of inelastic data files. If you need to analyse them, you will have to write your own computer routines. A non-standard application of this file is to generate a list of binary collisions that took place during the simulation, even when inelastic effects are not incorporated in the simulation. To do this, specify the use of the OR model, but set a very small (but non-zero) number for the OR k parameter (e.g. $k = 10^{-12}$ eV Å²) in the Inelastic file. This will result in negligible inelastic losses¹, but will still lead to the generation of the required information for any binary collision whose apsidal distance falls below the maximum value specified for application of the OR model. See the example project: `ion-scattering/4 keV F-NaCl(100)`.

References for Chapter 7

-
- [1] M. Aono and R. Souda, Nucl. Instr. Meth. B27 (1987) 55-64.
 - [2] Wolfgang Eckstein, Computer Simulation of Ion-Solid Interactions, Springer-Verlag, Berlin, 1991.
 - [3] 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).
 - [4] I.S. Tilinin, Phys. Rev. A 51 (1995) 3058.
 - [5] O.S. Oen and M.T. Robinson, Nucl. Instr. and Meth. 132 (1976) 647.
 - [6] 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.
 - [7] R. Smith, M. Jakas, D. Ashworth, B. Owen, M. Bowyer, I. Chakarov and R. Webb, Atomic and Ion Collisions in Solids and at Surfaces, Cambridge University Press, 1997.
 - [8] B.H. Cooper, C.A. DiRubio, G.A. Kimmel, R.L. McEachern, Nucl. Instr. Meth. B 64 (1992) 49.
 - [9] Z. Mišković, J. Vukani, T.E. Madey, Surf. Sci. 141 (1984) 285.
 - [10] This idea was borrowed from the *Camelion* simulation package, which can be obtained from:
<http://www.tm.tudelft.nl/secties/fcm/matphy/software/software.htm>.

¹ In practice, a particle position error of $\sim 10^{-4}$ Å arises from this method of tracking apsidal distances.

8. KALYPSO

8.1. Introduction

Kalypso is the simulation engine of the *Kalypso* package. The program takes its instructions from the project files created using *Spider*, as described in previous chapters. Fig. 8.1 shows the appearance of the main window of *Kalypso*, and indicates the toolbar speed-button functions.

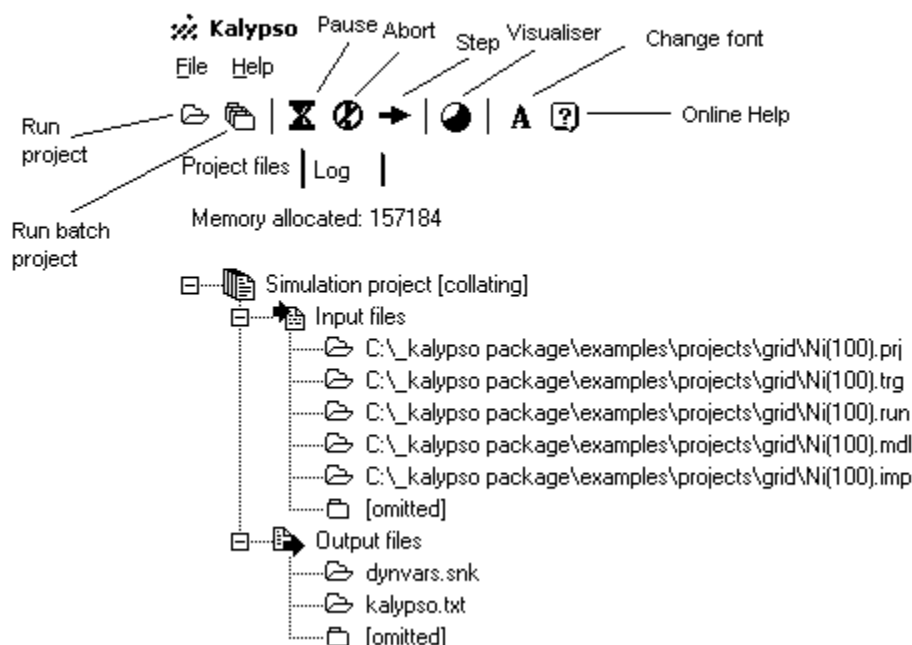


Fig. 8.1. Screenshot of *Kalypso* main window, showing a loaded project in the *Project Files* pane.

8.2. Simulation options

Before attempting to run a simulation, you should always check that the program options meet your needs. Click the Options command on the File menu. The dialog box shown in Fig. 8.2 comes up. *Novices can use the settings shown in Fig. 8.2, except for item 4 below, which can be chosen according to need.*

1. **Resume a previous simulation project:** the dynamics file (*.snk) specified in the project files dialog box (see later) must exist in order to use this option. The simulation project will be resumed at the start of the run prior to the final run listed in that file. For example, if the last output written to the file was at run 100, the project will resume at the start of run 100 (i.e. previous results for run 100 are discarded, and run 100 is re-executed). **This option cannot be used for a multiple-impact simulation.** If your dynamics file is huge (~6 GB), this option may not work (see footnote to section 1.5 for explanation), in which case you have to split up your project..

2. **Append output to log file:** output will be appended to an existing log file, rather than overwriting it. (The Log file is a copy of the screen output.)

Simulation options ✕

Miscellaneous

☐ Resume a previous simulation project

☐ Append output to log file (i.e. don't overwrite)

☐ Enable run skipping button

Screen reporting

☐ Silent (report nothing)

☐ Minimal (report run-by-run progress)

☒ Moderate (report run-by-run progress and energy)

☐ Verbose (report step-by-step system state)

Numeric parameters

Start at impact file line #

Unique ID

Lattice atom

☐ Load these options on start-up

☒ OK ☒ Cancel ☒ Help

Fig. 8.2. Simulation options dialog box for *Kalypso*.

3. **Enable run skipping button:** the button so enabled will enable you to abort runs manually without aborting the entire simulation (useful for debugging).
4. **Screen reporting:** screen reporting and/or energy calculations can slow down some kinds of simulations. Various levels of verbosity can be set, ranging from no output to step-by-step output. At any level of verbosity, you can check progress of the simulation (number of runs executed, elapsed time) by clicking the Pause button on the *Kalypso* main window. Use the **Silent** setting for running fast simulations (i.e. those that involve few particles), and **Minimal** or **Moderate** for slower simulations (e.g. sputtering simulations). The **Verbose** setting is slow, and should only be used for debugging.
5. **Start at Impact file line no. #:** This has value 1 by default, meaning that the first run of the simulation will be based on line 1 of the Impact file. This option allows you to skip a number of lines at the top of the Impact file. If you are **resuming** a simulation, the Impact file line number will automatically be set to the correct value.
6. **Unique ID:** the `ui` field of the output (dynamics) file will be iterated from the starting value entered here (leave the value at 1 unless you have reason to do otherwise). If you are **resuming** a simulation, the `ui` field will automatically be set to the value following that in the current output file.
7. **Lattice atom:** if = 0, this field has no effect. Otherwise, energy information will be returned for the specified lattice (target) atom (useful for debugging).¹ The index entered refers to the row that the atom's coordinates occupy *in the Target file* (if = 1, it refers to the anchor atom).

¹ The energy information is illustrated by the following output: "Atom[n] = 2: E[n](eV) = -3.0892; |F[n]|(eV/Å) = 0.1419; (x, y, z) Å = (0, 0, 0)". The binding energy of atom #2 in the system is -3.089 eV while the magnitude of the force experienced by the atom is 0.142 eV Å⁻¹. In this system, the projectile consists of 1 Ar atom, so atom #2, located at the origin, is the first target atom. This atom is a surface atom, so its binding energy E[n] differs from the value for an atom in the bulk (i.e. the cohesive energy). The atom is also unrelaxed, so the force acting on it is significant.

8. **Load these options on start-up:** the options are stored in the file *Kalypso.cfg* and can be loaded automatically when you start *Kalypso*, if you wish.

8.3. Running a simulation project

8.3.1. Single simulation project

In order to run a simulation project, select the menu command **File|Simulation Project**. this brings up the simulation project files dialog box shown in Fig. 8.3. You must specify the names of the (existing) input files that your project will use, and the names of output files that will be created (or overwritten). Then click Run to start the simulation, or Cancel to close the dialog box.

When the simulation is running you can **pause** it temporarily, or **abort** it prematurely, using the speed-buttons illustrated in Fig. 8.1. If you abort a batch project, you will be asked whether you wish to continue with the next batch job or abort all jobs. The **step** button aborts the current run of a simulation project, and moves to the next run. This is useful mainly for debugging, so the button is disabled by default (you can enable it in the simulation options dialog box, Fig. 8.2).

8.3.2. Batch simulation project

In order to run a batch simulation project, select the menu command **File|BatchProject**. this brings up a file input dialog box in which you can select the batch definition file for your batch project. Click Open to start the processing of the batch job.

Simulation project ✕

Input files		
Projectile data*	Select	C:_kalypso package\examples\projects\grid\Ni(100).prj <input type="checkbox"/> Omit
Target data	Select	C:_kalypso package\examples\projects\grid\Ni(100).trg
Run data	Select	C:_kalypso package\examples\projects\grid\Ni(100).run
Model data	Select	C:_kalypso package\examples\projects\grid\Ni(100).mdl
Impact data*	Select	C:_kalypso package\examples\projects\grid\Ni(100).imp <input type="checkbox"/> Omit
Inelastic data*	Select	<input checked="" type="checkbox"/> Omit

Output files		
Dynamics	Select	dynvars.snk
Log file	Select	kalypso.txt
Inelastic events*	Select	<input checked="" type="checkbox"/> Omit

*Can be omitted for some types of simulation (see Help).

Fig. 8.3. *Kalypso* simulation project files dialog box. The input file names shown here have similar stems, but this is not required. For some projects, certain input files may not be required and can be omitted (this should be indicated by checking the appropriate box(es)). The Inelastic events file is almost never required. No Inelastic file is required if the project does not include inelastic effects, and no Projectile or Impact files are required if the project does not include a projectile.

8.3.3. *Kalypso* screen output

The output shown on screen in the Log pane is the same as that saved to the Log file. The following project, which involves Ni atoms recoiling in a Ni matrix, serves as an example (comments are indicated after the `//` symbols):

```
KALYPSO 2.0
3/6/2004 10:07:35 PM.
Single impact simulation with FREE boundaries.           // type of simulation
C:\_Kalypso package\examples\projects\grid\Ni(100).prj
C:\_Kalypso package\examples\projects\grid\Ni(100).trg
C:\_Kalypso package\examples\projects\grid\Ni(100).run
C:\_Kalypso package\examples\projects\grid\Ni(100).mdl
C:\_Kalypso package\examples\projects\grid\Ni(100).imp
                                                    // no Inelastic file in project
C:\_Kalypso package\examples\projects\grid\dynamics.snk
Output: projectile atom #1 only.                       // type of output chosen

No. of projectile atoms: 1.
No. of projectile atom types: 1.
Atomic No: (0)28    // Atom type 0 [Ni] has atomic number 28
Mass: (0)58.934351  // Atom type 0 has mass 58.934351
Projectile mode: mixed (impinging/recoiling atom).    // projectile mode
Altitudinal angle = -0.001°. [Random values from 0.001 to 90°.]
Azimuthal angle = -360°. [Random values from 0.0 to 360°.]
E(in) = 215.799999942939 eV [projectile 1].    // i.e. E = 215.8 eV

No. of target atoms: 1330.
No. of target atom types: 1.
Atomic No: (0)28
Mass: (0)58.71
Target at 0 K.    // specified target vibrational temperature

Projectile-target potential: TB.
Attractive potential: TB.

Timestep = 0.22 fs (fixed).
Termination: 0 fs (minimum) to 70 fs (maximum).
Z(min) = -17.5 Å.
Z(max) = 0.2 Å.
Neighbour search method: linked cell (cell = 5 Å).
R(switch) = 3.53 Å.
R(cut-off) = 4.3 Å.

// Energy information for the 10th atom in the Target file (or atom with rw =11 in the
// simulation); E is the cohesive energy, F is the net force on the atom located at //
// (x, y, z) [calculations are only meaningful if the atom is in a bulk lattice site]

Atom[n] = 11: E[n](eV) = -3.51332415448804; |F[n]|(eV/Å) = 0.151398876981056; (x, y,
z) Å = (12.45922, -2.49184, 0).

Run 1 begins...
Energy = -5274.74521454 eV.           // total system energy

// the following items are output at the start and end of the simulation when it is
run with Moderate verbosity

// run number, current step, time elapsed, timestep and system energy
run  step  time(fs)  timestep(fs)  energy(eV)
1      0  0.000E+0000  2.200E-0001 -5.2747452E+0003 // total energy at start of run 1
1     319 7.018E+0001  2.200E-0001 -5.2747471E+0003 // total energy at end of run 1

// KE = kinetic energy, PE = potential energy, KE(fastest) = KE of fastest atom
KE(eV)      PE(eV)      KE(fastest)(eV)  fastest
2.1580000E+0002 -5.4905452E+0003  2.1580000E+0002  1 // atom #1 is fastest
1.4004116E+0002 -5.4147883E+0003  5.7970996E-0001  1

// range refers to the neighbour list range, Px, Py, Pz to momentum components
range(A)  np  Px(kgm/s)  Py(kgm/s)  Pz(kgm/s)
4.826E+0000 41  6.1840748E-0022 -2.3542854E-0021 -9.1766920E-0022
4.826E+0000 43  6.1840748E-0022 -2.3542854E-0021 -9.1766920E-0022
```

```
// angular momentum components in units of  $\hbar$ 
Jx(hbar)      Jy(hbar)      Jz(hbar)
-1.4650177E+0004 -3.0488148E+0003 -2.0508471E+0003
-1.4650177E+0004 -3.0488148E+0003 -2.0508471E+0003

Run 1 ends.
Energy = -5274.74711796 eV.    // system energy at termination

// energy conservation error (leak), with average and maximum values over N runs
Energy leak = -0.00190342 eV (average = 0.001903 eV [3.42E-0005%], max.= 0.001903 eV).
3/6/2004 10:07:38 PM.
Execution time = 3 s.
Status: OK.
```

Some points to bear in mind are:

1. Energy conservation should first be checked with all inelastic effects disabled (i.e. omit the Inelastic file from your project).
2. *Kalypso* attempts to track inelastic losses associated with the LSS, OR, ST and lattice springs models (and is reasonably successful for the LSS and lattice springs models). Nevertheless, both ST and OR models produce significant energy leakages (a few eV). These arise when the atoms are shifted by an amount dr at the apsis of the collision. The calculations take into account the potential energy change due to this shift for the ZBL potential in the binary collision, but not the potential energy change due to interactions with other atoms in the system. The problem increases as R_c (critical distance) increases. Note: if spring-snapping is enabled, the energy loss tracking logic will no longer work and item 3 below will apply to this model.
3. *Kalypso* does not attempt to track energy losses associated with the temperature control or image potential models. Energy lost through these processes will show up as (usually large) energy leaks (i.e. inelastic energy losses). This is nothing to be concerned about. The purpose of cooling, in particular, is to extract energy from the system.
4. The verbose screen reporting mode reports the energy etc. after every timestep, while at the other extreme, the silent mode never calculates or reports it. Do not use the verbose mode for serious simulations (it is too slow, and is useful mainly for debugging).
5. The OR and ST inelastic energy loss algorithms are quite time consuming and can increase running time by 5-10%. Use the faster LSS algorithm whenever possible for electronic stopping.

8.3.4. *Kalypso* user interface

As shown in Fig. 8.1, the *Kalypso* main window has speed-buttons that allow a running simulation to be paused, aborted or stepped. Another speed-button on the same toolbar opens the Visualiser window (Fig. 8.4). This displays the running simulation in a variety of views and formats.

Visualisation of the simulation is a slow process, so be sure to close the window during serious calculations (when the window is closed, the visualisation computations are not performed). Information about the visualisation options are available from hints that appear when the mouse cursor hovers over a button, edit box or other user input screen object.

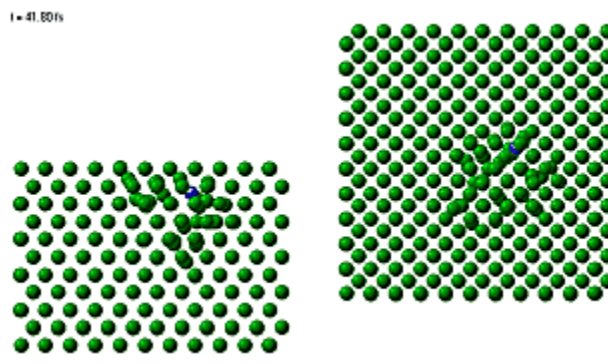


Fig. 8.4. *Kalypso* Visualiser window showing a Ni lattice that is disrupted by a fast recoiling Ni atom. Left: side view of lattice. Right: view from above lattice.

8.3.5. Implementation notes

Kalypso writes its output in buffered chunks that consist (in version 2) of 20,000 records (1.12 MB). The purpose of buffering is to avoid possible wear and tear on the hard disk. If a simulation fails suddenly (e.g. power cut) you will lose all data that are currently stored in the buffer. Dynamics file data that have already been written to disk will be retained, but almost certainly data from the most recent run will be incomplete, so you should filter it out using *Winnnow*, before attempting to restart the simulation. If you get an error message when filtering such a file with *Winnnow*, it means that the last record written to disk is corrupt. However, the filtered data (that exclude this record) should be useable.

Floating point output data are stored as single precision numbers, but calculations are carried out using double precision (to reduce disk storage).

If a run is skipped, or a simulation is aborted, the corresponding output data will still be written to disk (as if the run or simulation had terminated normally), the only difference being that the data are written at a premature time. The program logic for each run is:

```
Repeat
    DoNextTimestep
Until (Termination or aborted)
WriteOutput
```

9. WINNOW

9.1. Introduction

The *Winnow* program is used to process the output dynamics files (*.SNK) produced by *Kalypso*. Processing might typically involve extracting average values for state variables over a number of collision configurations (e.g. sputter coefficients, scattering cross-sections), or it might involve creating trajectory plots destined for graphical representation. This chapter describes the data processing capabilities of *Winnow*.

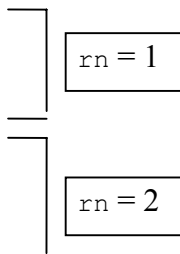
Dynamical variables information characterising your system was written to a dynamics file when you ran *Kalypso*. The dynamics file captures the following dynamical variables and other information (section 4.2.5):

ti, time elapsed since the start of the simulation run;
rw (row number), a particle index that is based upon the position of the particle in the projectile and Target files;
rn (run number), a run index indicating to which run the data refer;
ui (unique identifier), another index that is incremented whenever the output routine of the program is called;
rx, ry, rz, vx, vy, vz: particle position and velocity components;
ms: particle mass;
fl: particle options variable (flags) at time of output;
bx, by: these ‘tag fields’ contain either the projectile impact parameter, or the projectile incident angles (see discussion below).

A dynamics file will generally contain information from more than one run (typically 10^2 - 10^6 runs). The structure of the file is represented in the following diagram (where each line represents one dynamical variables record; the meaning of the **rn** and **rw** parameters is also illustrated):

```
----- atom 1, run 1 (projectile: rw = 1)
----- atom 2, run 1 (1st target atom: rw = 2 )
----- atom 3, run 1 (2nd target atom: rw = 3)
.....
----- atom N, run 1 (last target atom: rw = N)
----- atom 1, run 2
----- atom 2, run 2
.....
----- atom N, run 2
----- atom 1, run 3
----- atom 2, run 3

[ .. and so on, down to the last atom of the last run ]
```



Information from this file can now be extracted by *Winnow* according to your requirements. There are two steps to this process:

1. Filter out all of the irrelevant records from your dynamics file. For example, you might only be interested in sputtered particles, or fast-moving particles, or you

might only be interested in the projectile trajectory. This operation is conceptually similar to the filtering of records in a relational database.

2. Convert your informational requirements into data-processing directives. For example, you may wish to query the mean energy of sputtered particles, or determine the average polar emission angle, or find the projectile range.

Before trying to carry out these operations, you should familiarise yourself with the symbols in *Winnow*'s **query language** (*rx*, *vx*, *ke*, *alt*, *phi* and so on). *Winnow* also provides “Genius” functions that enable you make up queries and expressions, even if you do not know the language.

It should be mentioned that *Winnow* offers the option of dumping all of the data in a dynamics file into a file of text format. Such text files can then be loaded directly into spreadsheet programs (e.g. Excel), database programs (e.g. dBase) or statistical programs (like SPSS). If you find *Winnow*'s query language confusing, this would be an alternative way of processing your simulation data.

The most important *Winnow* commands are found on the Process (data-processing) menu. To some extent this menu has become a repository for small algorithms that I have found necessary for my own work over the years. All users of *Winnow* should attempt to master the Filter and Spectrum commands, which are very useful for data analysis. Some of the remaining items can simply be ignored by users, particularly those with simple needs (such as sputter yields or energy spectra).

Please note that the file access modes used by many *Winnow* routines do not operate correctly if the dynamics file (*.snk) has read-only attributes (an error will be reported). *Winnow* never makes changes to the input data file, but this file must still be accessible to the program with both read and write permissions. You should be aware that read-only permissions may be acquired automatically by files if they are copied to/from CD storage.

9.2. Query language

9.2.1. Introduction

This section describes the query language used by *Winnow*.

1. Predefined identifiers allow you to access the values of the variables (and certain functions of the variables) contained in the records of a dynamics file. They are also used to store constants such as the atomic mass unit.
2. *Winnow*'s query language allows you to customise output as required for your particular application. To do this, you need to specify what functions of dynamical variables (in terms of *rx*, *ry*, *rz*, *vx* . . .) you wish *Winnow* to produce.
3. Conditional expressions required by the Filter operation are likewise specified by combining query expressions with relational operators (=, >, >=, etc) .

The query language will be quite transparent to anyone with elementary programming or database experience . Most variables of interest can be accessed via predefined identifiers. For example: use '*rx*py - ry*px*' to represent the z-component of the angular momentum.

9.2.2. Syntax

1. **Case** is ignored. UPPER, lower or MiXeD case letters may be used for any input line symbol (*px* or *Px* or *PX*).

2. **Blank characters** between valid symbols are ignored (`px*px` or `px * px`). The maximum length of the input line is 255 characters.
3. **Comments** may be nested between curly braces `{like this}`. Comments are useful as mnemonics in case you want to re-load the expression from a dialog box history list on a future occasion. Also, the comment will be printed in the output file if you use *Winnow's* Averages or Collate options. Like blank characters, comments are ignored by the parser: `px*px {Here's a comment}`

9.2.3. Predefined identifiers (floating point variables)

The following *predefined identifiers* denote (x , y , z) components of vector dynamical variables (**always expressed in SI units, unless stated otherwise**):

1. `rx`, `ry`, `rz` = position (**m**);
2. `vx`, `vy`, `vz` = velocity (**m s⁻¹**);
3. `px`, `py`, `pz` = momentum (**kg m s⁻¹**)
4. `bx`, `by` = tag fields (the meaning of these variables is explained in section 4.2.5 of Chapter 4).

The tag fields in item 4 (`bx`, `by`) are expressed either in **metres** (when they refer to coordinate data) or in **degrees** (when they refer to angular data). Other predefined identifiers (all in SI units) are:

5. `ti` = time elapsed (**s**; `ti` = 0.0 at the start of a simulated trajectory);
6. `ms` = mass (**kg**);
7. `ke` = kinetic energy (expressed in **eV**);
8. `phi`, `alt`, `phi`, `ph2`, `ph4`, `ph8` = azimuthal and polar angles **expressed in degrees** (see notes below);
9. `rw`, `rn`, `ui`, `fl`: row number, run number, unique identifier, flags (see notes below).

The variables in item 9 take (dimensionless) **integer** values. The other variables take **floating point** (real) values.

Predefined angular variables (**degrees**) are: `phi`, `alt`, `phi`, `ph2`, `ph4`, `ph8`.

The symbol `phi` represents the particle's azimuthal direction of motion (ϕ), defined by:

$$\text{phi} = \arctan(vy/vx)$$

The range of `phi` is 0-360° (unlike the "atan" function, which maps to a value in the range -90 to +90°, the angle `phi` is assigned to the correct quadrant, even if `vx` = 0). Several other azimuthal angle identifiers are defined. All are expressed in degrees. The identifier `ph4` maps the `phi` values to the first quadrant only (0-90°), and is defined as:

$$\text{ph4} = \arctan(\text{abs}(vy/vx))$$

For example, the angles 100°, 260° and 280° are replaced by 80°; the angles 170°, 190° and 350° are replaced by 10°. The `ph4` variable is useful if you want to count a scattering/emission yield which has 4 fold symmetry around the z axis (in order to improve statistics).

The `ph8` identifier is similar to `ph4`. However, it maps azimuthal angles $> 45^\circ$ to the range 0- 45° using the following algorithm:

```
angle = arctan(abs(vy/vx));
if angle > 45.0, then angle = 90.0 - angle;
ph8 = angle;
```

This is a useful way to exploit symmetry if the emission process has 2 sets of 4-fold axes (e.g. sputtering of a (100) cubic lattice by a normally incident projectile).

Finally, `ph2` maps azimuthal angles $> 180^\circ$ into the range 0- 180° . This is useful if the simulation problem has 2-fold symmetry.

The symbol "`alt`" (**alt**itude) represents the particle's altitudinal direction of motion (φ), defined by:

```
alt = arctan(vz/sqrt(vx^2+vy^2))
```

The range of `alt` is from -90° to $+90^\circ$. If $(v_x^2 + v_y^2) = 0$, then `alt` = -90° or $+90^\circ$ (motion parallel to z -axis). If `vz` is negative, then `alt` is also negative.

9.2.4. Integer variables (`rw`, `rn`, `ui`, `fl`)

The following variables take integer values:

- `rw` ('row number') is the index used by *Kalypso* to label the atoms in the simulation. It starts from value 1 (which is assigned to the first projectile atom, if any). Subsequent projectile atoms are labelled with indices 2, 3, ... N_p , where N_p is the number of projectile atoms (usually $N_p = 1$). The remaining atoms in the system (target atoms) are labelled by indexes N_p+1 , N_p+2 , etc. In multiple-impact simulations, the indexes of replica atoms succeed those of the target atoms;
- `rn` ('run number') references a row in the Impact file (`rn` = 1 for the run that used the 1st line of this file to define the impact parameter, and so on);
- `ui` references the 'unique identifier' variable (section 4.2.5 of Chapter 4);
- `fl` references the flags variable (section 3.4.1 of Chapter 3).

Comment: for the common case of an monatomic projectile, the row number assigned to a target atom differs by 1 from its line number in the Target file. Thus, the atom at the top of the Target file is assigned `rw` = 2, while the atom on line 10 of the Target file is assigned `rw` = 11. Understanding this relationship is critical for constructing filters (see section 9.3. Filtering data).

9.2.5. Constants

Constants that can be accessed as predefined identifiers are:

- `pi` = π
- `ep` = e , the proton charge ($1.602176462 \times 10^{-19}$ C)
- `au` = 1 amu, the atomic mass unit ($1.66053873 \times 10^{-27}$ kg)

These constants are expressed in **SI units**. For example, to express an atom's atomic mass in amu, use the expression '`ms/au`'.

9.2.6. Arithmetic Operators and Arithmetic Expressions

The predefined identifiers (see above) may be combined in expressions with decimal numbers in integer or floating point notations (i.e. 35, 35.0, 3.5×10), bracket

symbols `()`, and the usual arithmetic operators `(* / + -)`. The syntax rules for forming expressions conform with those used in many programming languages:

- The elementary arithmetic operators are supported: `+ - * / ()`
- Exponentiation is carried out using the caret `(^)` symbol e.g. `px*px*px = px^3`
- Blank spaces are ignored
- The setting of one of the options flags flag can be tested (in filtering operations only) using the tilde `(~)` operator: the conditional expression `[f1 ~ 16]` is TRUE if the flag 00010000 (2^4 or 16 decimal) is set.
- Mixed expressions are typecast to real number results.
- Division of an integer by another integer typecasts the result of the operation to real.

Here are some examples of valid arithmetic expressions:

```
ke           {kinetic energy in eV }
px*px+py*py+pz*pz)/(2*ms)    { kinetic energy in SI units }
2.3*(rx*rx+2*ry*ry)/(1.3+6.5){arbitrary numerical expression}
rw + 1      { an integer expression; see sect. 2.8 }
```

9.2.7. Functions

Functions are supported, via a Pascal-style syntax: `'function(arg)'`, where the argument in brackets (*arg*) *must resolve to a real number expression*. For example: `'exp(-rx/2E-10)'`. Integer expressions can be typecast, if necessary, by multiplying by 1.0. All trigonometric functions use **degrees** for angular units. As usual, blank spaces will be ignored, and can be used to improve readability.

The following functions are available:

- `sin` = sine of angle e.g. `'sin (phi)'`
- `cos` = cosine
- `tan` = tangent
- `atan` = arctangent (inverse tangent)
- `exp` = exponential e.g. `'exp(-rx/300)'`
- `ln` = logarithm to base *e*
- `lg` = logarithm to base 10
- `abs` = absolute value (modulus) e.g. `'abs(1x)'`
- `sqrt` = square root e.g. `'sqrt(rx*rx+ry*ry+rz*rz)'`
- `tr` = truncate (returns integer value), e.g. `'tr(2.8)'` (return value = 2)

The following examples are physically meaningless, but illustrate how the real number syntax works for `rw` (or `rn`):

- `sin(3*rw)`: invalid - integer argument
- `sin(1.0*3*rw)`: valid - argument is 'typecast' to real expression
- `sqrt(rx + 1)`: valid - mixed expression (`'1'` is integer, `rx` is real) is typecast to real

9.2.8. Conditional expressions and filtering

Conditional expressions arise only in connection with the Filter operation. As with arithmetic expressions, blank spaces are ignored in conditional expressions and may

be used to improve readability. All conditional expressions must be enclosed in square brackets `[]` (which may be nested to any depth, as needed).

Conditional expressions involve comparisons (“relations”) between two arguments A and B, which may be any valid arithmetic expression involving any of the predefined identifiers listed in section 2.3 (`rx`, `ry`, `rz` etc.). Depending on the values of the compared items A and B, and the nature of the comparison, any given conditional expression will resolve to one of two values, either TRUE or FALSE.

A simple example would be the following expression: `[rz > 0.0]`. Here, `rz` is compared with zero: only when the expression is evaluated to a TRUE value (i.e. when `rz` is greater than zero) does the Filter option initiate action (write data to disk). In this example, data are only stored for those particles which are above the target surface (defined by $z = 0.0$). To express this idea in database terminology: the expression enclosed by square brackets `[...]` constitutes a relation or condition which must be satisfied by any record which is written to the output file. What the preceding example has achieved is to filter out (isolate) data for ejected particles from the dynamics file and store it in a new dynamics file. This filtered data can then be processed further, e.g. to generate an energy spectrum of ejected particles.

9.2.9. Logical and relational operators

The relational operators `=`, `<`, `>`, `>=`, `<=`, `<>` are recognised, with the following meanings:

- `[A = B]` TRUE if (expression) A is equal to (expression) B
- `[A > B]` TRUE if A is greater than B
- `[A < B]` TRUE if A is less than B
- `[A >= B]` TRUE if A is greater than or equal to B
- `[A <= B]` TRUE if A is less than or equal to B
- `[A <> B]` TRUE if A is not equal to B

The arithmetic expressions involved in the comparison must either be both real or both integer expressions (see section 2.8). Conditional expressions may be combined using the `&` (logical AND) and `|` (logical OR) operators. The following examples illustrate the syntax, but it would not be sensible to use these expressions for filtering because they always resolve to the same value:

```
[1=1] & [2=2]      {TRUE}
[1=1] & [px > px]   {FALSE}
[1=1] | [2=3]       {TRUE}
[1=2] | [2=3]       {FALSE}
[1=2] | [px = px]   {TRUE}
```

A realistic example of filtering is the following:

```
[ke > 10.0] { particle KE > 10 eV }.
```

An example of a meaningless comparison with invalid syntax is:

```
[ke > rw] { ERROR: rw is an integer! }.
```

Any conditional expression may be negated using the `!` (NOT) operator:

```
[1=1]      {TRUE}
![1 = 2]   {also TRUE}
```

Again, it is stressed that blank spaces are ignored by the parser. The ‘=’ sign or some other relational operator must be present inside the brackets, as the conditional expression always involves a comparison.

Here are more examples of valid conditional expression syntax:

```
[pz*pz/(2*ms) > 10*ep] & [ [rz > 0.0] | [vz > 0.0] ]
```

The above expression is TRUE if the kinetic energy associated with motion normal to surface is > 10 eV and the particle is either outside the surface ($z > 0$) or is travelling away from the surface (in a positive z direction). Note the way brackets are used to nest sub-expressions.

```
[atan(pz/sqrt(py*py+px*px)) > 50*pi/180] & [rz > 0.0]
```

The above expression is TRUE if the particle has left the surface and is travelling at an of-surface angle of $> 50^\circ$. This example shows a typical use of the arctangent function. A better way to express this is:

```
[alt > 50.0] & [rz > 0.0]
```

The Annexe contains more examples of Filter expressions.

9.2.10. Numeric types

Two numeric types are recognised: ‘integer’ and ‘real’ (floating point) types. Enforcing type distinctions in syntax helps to avoid certain kinds of semantic and physical errors which might otherwise go unnoticed. The conventions outlined below are similar to those found in most programming languages.

- The predefined identifiers `rw`, `rn` and all numbers written in integer format (0, 1, -1..) are treated as integer-type numbers.
- All other numbers, variables and constants are of real-type.
- Mixed integer-real expressions (`2*pz`, `1+1.0` etc.) are automatically cast to real-type.
- Arguments passed to functions must be in real format (this can always be enforced by typecasting: multiplying by 1.0):

```
exp(10) {invalid}
exp(1.0*10) valid
```

- Both sides of a conditional expression must be of identical type:

```
[3 > 2]      {TRUE}
[3.0 > 2]     {syntax error: can't compare real with integer}
[3.0 > 2.0]   {TRUE}
[3*1.0 > 2.0] {TRUE - left hand side is typecast to real }
```

- Otherwise the type identity of expressions is the same as that of the constituents, except for integer division (which always results in a real number).

```
rw*2 + 1     {integer}
2*pz + 3     {real}
rw*2 + 1.0   {real}
rw/2 + 1     {real - integer division}
```

9.2.11. Parser errors

The most common parser errors are syntax errors and arithmetic errors (divide by zero etc.). The parser error message will indicate the point (^) in the input expression at which it detected an error. E.g., for the Filter option input:

Example 1:

```
[px > 0]
      ^
```

The pointer indicates character #8 reading from the left. The parser expected a floating point number (such as 0.0) rather than an integer, but only detected an error when it encountered the '[' character. Solution: change '0' to '0.0'.

Example 2:

```
2.0*px > 3e-20]
      ^
```

The parser evaluates both the missing '[' and the real number '2.0' as atomic tokens. It expects to find the pattern BRACKET REAL, rather than REAL, and indicates an error at the end of the illegal REAL token. Solution: add the missing '[' before '2.0'.

To get a feel for the error-response of the parser, you could try feeding it some deliberate errors (e.g. 'px*px + px**px').

Common syntax errors include:

- using x,y,z instead of rx,ry,rz;
- opening parentheses '(' '[' without later closing them;
- using integers in places where real numbers are expected.

If the cause of the error is not obvious, try simplifying the expression until it parses correctly. Remember that no program can protect you against logical errors. **A typical non-syntactical error in this category is forgetting to couch the expression in SI units.**

9.3. Filtering data

The key to getting at the information contained in dynamics files is the Filter option, which discards unwanted data from your raw dynamics file. Put simply, you select the data that you want to examine further, and store it in a new file. At the same time, *Winnnow* reports how many records are stored in the new file. This information can be used to calculate quantities such as sputter yields or projectile reflection coefficients.

There are three filtering commands provided in *Winnnow*: **Filter**, **Batch Filter**, **Predefined Filters**. Functionally, they are similar, but they have different degrees of automation. The Batch Filter is useful for applying the same filter operation to a number of files. The Predefined Filters simplify the construction of some standard filters (for novices).

For example, suppose you are interested in the fraction of sputtered atoms that has energy above 1 eV; let us suppose that a sputtered atom is defined as one that is moving away from the surface [vz > 0.0] and is above the surface beyond interaction range (e.g. 6 Å) [rz > 6.0e-10]. These conditions, plus the 1 eV energy condition [ke > 1.0], can be expressed in the query language as the following conditional expression:

```
[rz > 4.0e-10] & [vz > 0.0] & [ke > 1.0]
```

This expression is translated by the parser into computational directives. The output is sent to a new dynamics file. If you feed the above expression to any of the Filter commands, it will filter out the data that do not satisfy the condition. You can then, for example, proceed to Average or Collate the data written to the new file, or Filter it again according to some other criteria. Note that filtering a large file ($>10^6$ records) can be quite a slow operation, even on a fast computer.

Here are some further examples of the use of conditional expressions (note the nesting of brackets in the composite expressions C and D):

A. Filter out (**discard**) projectile data (assuming a single-atom projectile):

Use filter condition: `'[rw > 1]'`.

B. Filter out atoms (**discard**) with less than 10 eV kinetic energy:

Use filter condition: `'[ke >= 10.0]'`.

C. **Keep** data for ejected atoms ($z > 0$) with $ke > 0.5$ eV only.

Use filter condition: `'[vz > 0.0] & [rz > 0.0] & [ke > 0.5]'`.

D. Combine Filter with Collate to determine origins of ejected atoms ($z > 0$):

(1) Filter with condition: `'[vz > 0.0] & [rz > 0.0]'`.

(2) Subsequently, a 'Collate' operation on the atom-by-atom basis gives the total number of 'hits' satisfying this condition. In other words, we obtain a breakdown of the ejected atom population by 'row number' (the parameter `rw`, which identifies the atom with reference to its position in the Target file: see section 1). The collation function used in this case is irrelevant, because we are simply counting atoms.

9.4. Averaging data

The Averages operation calculates statistical information based on the *entire* dynamics input file. The user defines some function (let's call it `q`) of the system dynamical variables, expressed in terms of the predefined identifiers listed in section 2.3. The Averages operation writes a line of output based on this function, which consists of the mean value ($\langle q \rangle$), root mean square (rms) value ($\sqrt{\langle q^2 \rangle}$), and the standard deviation (σ_q) of `q` respectively:

$\langle q \rangle$ $\sqrt{\langle q^2 \rangle}$ σ_q .

Up to 7 such functions can be specified in one averaging operation. The output is written to a file with `.TXT` extension, e.g. `DYNVARS.SNK` \rightarrow `DYNVARS.TXT`.

For example, you might wish to calculate the mean x -velocity component of in a `SNK` file filled with records of sputtered atoms: simply enter '`vx`' on the input line. Then for this case $\langle q \rangle = \langle vx \rangle$, while $\sqrt{\langle q^2 \rangle}$ corresponds to the rms x velocity component.

If a `.TXT` file of the same name already exists, the output is *appended* to the existing file. This is in contrast to the over-write behaviour of the Collate operation (see next

section). Normally the averaging would be carried out after the `.SNK` file has been filtered (see section 3) to remove unwanted data.

9.5. Collating data

The Collate operation calculates statistical averages on an atom-by-atom, run-by-run, or output-by-output basis, according to the user's specifications. To use it, you need to understand the information content of the dynamics file.

1. In the atom-by-atom case, the output data is collated according to the 'row number' (rw) index. This represents an *average over particles*.
2. In the run-by-run case, the 'run number' index (rn) from the Impact file is used as the collation key. This represents an *ensemble (configuration) average*.
3. In the output-by-output case, the unique identifier (ui) associated with the output event is used as the collation key. Thus, the average is carried out over all particles that were recorded during the same output 'event'. This option allows averaging operations at different times within the same simulation run.

The collated averages `<.>` are output to an ascii file with a columnar lay-out which is similar to that of an 'Average' file (see previous section):

```
<q>      √<q²>      σq      key      hits
```

where:

- `q` represents the expression input by the user
- `σq` = standard. deviation of `q`,
- `key` = `rw` or `rn` (row number or run number, as selected by user)
- `hits` = number of data used for averages

In addition, for the output-by-output collation option, the time elapsed in the simulation is displayed (in fs):

```
<q>      √<q²>      σq      key      hits      time (in fs)
```

Collated output overwrites the contents of any appropriately named `.DAT` file that may exist. E.g. collating `DYNVARS.SNK` sends output by default to `DYNVARS.DAT`; if the latter already exists, it is overwritten.

The atom-by-atom average could be used to determine (for example) which atoms were sputtered most frequently in a real-life experiment (which is averaged over all impact parameters).

The run-by-run average may be useful in establishing how individual configurations contribute to the overall behaviour. If the dynamics file only contains data from a single run then the run-by-run average is equivalent to the Averages option. (In fact, the Averages option is equivalent to collating by both `rw` and `rn` simultaneously.)

You may specify what function of dynamical variables gets written to the collated file using the query language expressions. For example, `'px*px'` represents the square of the *x*-momentum component.

9.6. Format columns operation

9.6.1. Summary

This command performs a simple conversion based on data in a binary dynamics file (*.snk) to an ascii format (*.dat), laid out in up to 10 columns listing variables or functions specified by the user. In the simplest case, you can use this option to dump your output data to an ascii file for transfer to a spreadsheet program.

You enter the functions you want to dump using the query language. A formatted file can be read by spreadsheet and other data analysis programs, but cannot be processed further by *Winnow*. For example, if you wish to plot particle kinetic energies versus their distance from the (0, 0, 0) point, use the following inputs:

Column 1: `sqrt(rx*rx + ry*ry + rz*rz)` {distance}

Column 2: `ke` {kinetic energy in eV}

This procedure will yield a two-column ascii output file (*.dat) suitable for use by any graphing program.

9.6.2. Example: Create a Target file based on a dynamics (*.snk) file.

It is possible to use the dynamics file (*.snk) output by one simulation to obtain target input data for another simulation (e.g. to restart a multiple impact simulation). This is achieved with the Format Columns command, plus some manual editing, as will now be described.

Consider a simulation involving a CuNi target. Suppose you want to start a new simulation using the output data for run 10 of the simulation (the data for a specific run can be isolated by filtering, e.g. with the expression `[rn = 10]`; see section 9.3). First, invoke the Format Columns command with the input functions shown in Fig. 9.1. This will produce a 5-columns ascii text file (run10.dat) that lists the coordinates (in Å), masses (in amu) and flags for each atom in the system.

Format Columns	
	Source (input) file
Browse	C:\run10.snk
	Destination (output) file <input type="checkbox"/> Display result
Browse	C:\run10.dat
	Functions to be written to destination file (blank lines ignored)
Column 1	rx*1E10
Column 2	ry*1E10
Column 3	rz*1E10
Column 4	ms/au
Column 5	fl

Fig. 9.1. Format Columns dialog box (Process menu of *Winnow*). The text in in the Column 5 input box reads `f1` (mnemonic for flags parameter).

The atomic numbers of the atoms must still be inserted into the output file produced by this procedure for target atoms (and for any implanted projected species, if the file

is derived from a multiple impact simulation). This is easily achieved with a text editor, by replacing each mass term (e.g. **6.35460E+0001** for Cu) with the corresponding atomic number plus mass (e.g. **29 63.546**). The final result, as shown below, is an ascii file which has the Target file format and which can be used in a simulation project in the usual way.

Original ascii file (*.dat = Format Columns output):

```
1.61472E+0001  3.85135E+0000  1.90706E+0000  6.35460E+0001  2
-3.48174E-0002  1.45676E-0004  2.98504E-0001  5.87100E+0001  2
-1.74047E+0001 -1.74428E+0001  1.86811E-0001  5.87100E+0001  2
```

Replace **6.35460E+0001** by **29 63.546** and replace **5.87100E+0001** by **28 58.710**.

Final ascii file (*.trg = Target file):

```
1.61472E+0001  3.85135E+0000  1.90706E+0000  29 63.546  2
-3.48174E-0002  1.45676E-0004  2.98504E-0001  28 58.710  2
-1.74047E+0001 -1.74428E+0001  1.86811E-0001  28 58.710  2
```

Further notes

1. Do not modify the values of the flags output by the Format Columns command! Consult section 3.4 of the User Guide if you want to interpret the meaning of the flags.
2. At present there is no procedure for transferring atomic velocities from the dynamics file to a new simulation, so the latter will start with stationary atoms.
3. Unlike target files created by Spider, the lines of the Target file created by this procedure have no identifying strings like 'Cu' or 'Ni' before the line end (these are not required in order to run a simulation).

9.7. Converting data

The Convert option performs a simple conversion ('dump') of the records in a binary dynamics file to a text (human-readable) ascii format; this file has the values of the dynamical variables in each record laid out in space-delimited columns as follows:

```
rx×1010, ry×1010, rz×1010, vx, vy, vz, ti×1015, ms/amu, bx, by, rw, rn, fl, ui
```

The meanings of the symbols are defined in section 9.1. Thus, effectively:

- **rx ry rz** are expressed in Å
- **ms** (particle mass) is expressed in atomic mass units
- **ti** (elapsed time) is expressed in fs

and other variables are expressed in SI units;

A converted file can be read by commercial spreadsheet programs, but cannot be processed further by *Winnnow*.

9.8. Constructing data spectra (histograms)

The Spectrum command uses information in a dynamics file to create the ‘spectrum’ (frequency histogram) of some function of dynamical variables. The spectrum is output in the form of a .DAT file eg.: TEST.SNK --> TEST.DAT, and can optionally be displayed on screen. This is an important function, because it enables you to look at the distribution of the values of some arbitrary function of dynamical variables.

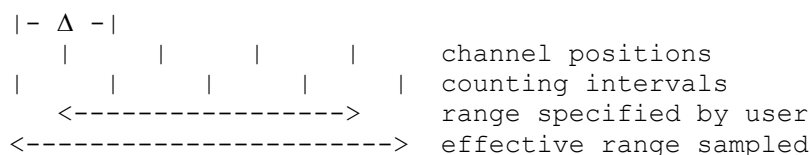
Winnow provides two Spectrum commands which are accessed via different menu items: Spectrum and Weighted Spectrum. The latter command is rarely used; it provides a means of calculating a weighted distribution of some function of dynamical variables. For example, you may wish to calculate the energy distribution weighted by an exponential time decay term e^{-at} , known as the **weighting function**. If needed, the weighting function can be specified using the query language.

In the Spectrum dialog box, the user always needs to specify the **spectrum variable** (F) whose distribution of values is of interest; this is done using *Winnow*’s query language. For example:

- To examine the altitudinal angular distribution, specify F as:
`alt {altitudinal angle, in degrees}`
- For a kinetic energy spectrum (particle energy distribution), specify F as:
`ke {particle kinetic energy}`

Other inputs required from the user are the minimum and maximum limits of the spectrum, and the number of channels to be utilised by the spectrum. The spectrum function is calculated for each record in the dynamics file. If the function falls within the specified spectrum range the appropriate channel count is incremented.

A given channel registers counts for data within a bandwidth $\pm 0.5\Delta$, where the interval Δ is given by: $\Delta = (\text{Max limit} - \text{Min Limit})/(\text{No. channels} - 1)$. The spectrum will thus in practice register counts from $(\text{Max Limit} + \Delta)$ to $(\text{Min Limit} - \Delta)$ i.e. a slightly greater range than that specified by the user, as the figure below shows:



If you need to create spectra that are functions of angular variables, you need to take account of the fact that the result of an expression such as:

```
atan(vy/vx) {azimuthal angle},
```

is ambiguous (since there are two possibilities, separated by π). You might have to specify additional relations involving vy and/or vx .

For expressions involving the azimuthal and altitudinal angles of motion, you can make use of the predefined angular variables “`phi`”, “`alt`”, “`phi4`”, “`phi8`”, “`phid`” and “`altd`” respectively, which always return angular values in the correct quadrant.

A minor difficulty arises with spectra that involve sampling at the maximum and minimum limits of angular variables. These limits are $\pm 90^\circ$ for the “`alt`” variables, and 0° and 360° for the “`phi`” variables. This results in anomalous count rates for the

points at the limits that are ~50% of what you expect. This problem is easy to recognise. One workaround is to reduce the size of the counting interval to avoid sampling beyond the limit.

9.9 Merging and re-merging dynamics files

The Merge operation takes the atomic coordinates listed in the specified Target file, and writes them to the output dynamics file, overwriting the values in the input dynamics file. For example, every record for atom #N in the input file will be substituted with the Target file coordinates for the same atom, but will otherwise remain unchanged.

- Input files
 - SNK file
 - Corresponding TRG file
- Output
 - SNK file

By default, the Merge operation writes the Target file coordinates to the (rx, ry, rz) records in the SNK file. But the user can also choose to overwrite the (vx, vy, vz) records.

The Merge operation is useful if you need to collate the post-simulation behaviour of the system on the basis of the original atomic locations in the Target file.

The Re-Merge operation restores information that was substituted by the Merge command. Suppose you used the Merge command to create a file *A.snk* from *dynvars.snk*. You may wish to do something to *A.snk* - for example, filter out edge atoms from the lattice. This will produce a smaller file, *B.snk*. To restore the original data to the records that remain in *B.snk* after filtering you can Remerge *B.snk* with *dynvars.snk*. The remerged output file, *C.snk*, will have similar records to *dynvars.snk*, except that it will lack those that were removed by the filter operation.

In effect, the Merge/Remerge combination allows you to filter a SNK file on the basis of attributes found in the TRG file.

9.10.

9.10. Find sputtered clusters

This operation examines a dynamics file for the presence of clusters of a specified size (among gas phase sputtered atoms). Clusters are identified on the basis of their spatial proximities. There is no guarantee that the clusters identified are stable in the chemical sense. This routine is based on Stoddard's algorithm [1]. In order to qualify as members of the same cluster, the individual particles must be associated with the same *rw* (row number) and *ui* (unique identifier) indexes. It is the responsibility of the user to ensure that any records that refer to the projectile species have been removed (filtered) from the input dynamics file. Otherwise, the projectile species will be included in the cluster search algorithm, which might distort the results.

- Input file
 - SNK file to be analysed.

- Output files
 - A text file containing information about detected clusters (see below).
 - A SNK file which contains the records of atoms that are members of clusters of the size specified.

The user must specify the clustering radius (typically the same as the potential cut-off distance), the number of atoms in the cluster, and an optional row index (*rw*) sorting value.

If the number of atoms in the cluster is specified as 2 (3,...), information will be reported for dimers (trimers,...). Monomers can also be identified. The row index is used simply to sort atoms into categories of interest (e.g. surface vs. bulk atoms).

The text output file has the following appearance:

```
List of atoms in clusters
Col 1: No. of atoms in cluster
Col 2: rn
Col 3: ui
Cols 4, 5, ...: atom1-rw, atom2-rw, ...

3   1   9   282   338   310
3   4   12  477   533   505
3   6   14  453   454   455
3   7   15  497   554   526
3  13  21  473   502   501
3  15  23  333   388   361
3  15  23  418  1315   ...

No of clusters with 3 atoms: 198
No of *atoms* (not clusters) with rw <= sorting index: 534
No of *atoms* (not clusters) with rw > sort index: 60
Largest cluster in this file has 26 atoms
```

In this example, 198 distinct clusters (trimers) are identified. For example, the first line of output reports that the atoms with *rw* = 282, 338, 310 are emitted as a trimer in run 1. Of the 594 (i.e. 198×3) atoms in the identified clusters, 534 have the *rw* row index parameter less than or equal to the sorting index value. The largest cluster identified in the input file is also reported (in this example it is a 6-atom cluster).

9.11. Sputtering statistics

Simulation statistics for atoms in a dynamics file can be explored using the Sputtering Statistics command. The user must specify how many runs were involved in the simulation in order that the sputter yield and other quantities can be calculated. Some typical output is shown below:

```
Sputtering statistics for ...
Total sputtered atoms = 5431
No. of projectile impacts = 405
Mean sputter yield = 13.40988
Standard deviation of sputter yield = 9.57904
Standard error of sputter yield = 0.47599
```

N	P(N)	N*P(N)	events	sig(P)	sig(N*P)
0	0.02469	0.00000	10	0.00781	0.00000
1	0.01975	0.01975	8	0.00698	0.00698
2	0.03704	0.07407	15	0.00956	0.01913

3	0.03210	0.09630	13	0.00890	0.02671
4	0.04444	0.17778	18	0.01048	0.04190

For example, the table predicts that the probability that 4 atoms will be sputtered after a projectile impact is 0.044 ± 0.010 (this happens in 18 out of 405 simulation runs). A typical presentation of these data would be to plot a histogram of $P(N)$ vs N , with (Poissonian) error bars $\text{sig}(P)$ on the plotted P values (sig refers to the standard deviation, σ). $N \times P(N)$ is also sometimes of interest and can be similarly plotted with (Poissonian) error bars $\text{sig}(N \times P)$.

One use of the statistics is to estimate the sputter yield error. In sputtering simulations, the sputter yield error can be estimated *approximately* using a relation based on Poisson statistics (typically this underestimates the error) as: $\sigma = \sqrt{Y/N}$, where N is the number of runs. This relation gives $\sigma = (13.4/405)^{1/2} = 0.2$. In this simulation there are sputtering events that involve emission of large numbers of atoms ($N > 40$), so that the standard deviation becomes quite large ($\sigma \sim 9.6$ in the output shown above). A better way to estimate the error is via the **standard error**, which represents the uncertainty in the mean sputter yield. The standard error in this example is computed as: $\sigma/\sqrt{N} = 9.56/(405)^{1/2} = 0.48$. Thus, the predicted **sputter yield** with uncertainty (standard error) is 13.4 ± 0.5 .

Winnnow cannot determine the number of runs in the simulation project from the input file alone (some runs with zero yield may not be represented in the file). This is why the user must specify the number of runs (i.e. the number of lines in the Impact file). If the value specified is too small, it may be detected by *Winnnow* (provided it is smaller than the number of runs that are represented in the input file), in which case a warning or error message will be issued. In other cases, *Winnnow* will assume that the value entered is correct.

9.12. Displacements operation

The Displacements operation creates a output dynamics file from 2 input dynamics files. The positions and velocities written to the output file are the differences of the relevant components in the two input files.

Given two input files consisting of records such as:

```
x1  y1  z1  vx1  vy1  vz1  ...  file #1,
x2  y2  z2  vx2  vy2  vz2  ...  file #2,
```

the output file will have a corresponding set of records:

```
x2-x1  y2-y1  z2-z1  vx2-vx1  vy2-vy1  vz2-vz1  ...
```

The remaining records in the output SNK file are reproduced from the second of the input files (file #2). Note: the two input SNK files used in the operation should be of the same size.

The purpose of the operation is to allow you to calculate atomic displacements (in real or momentum space) which have occurred between an initial and a final system state. The output file so created can be processed just like an ordinary dynamics file. The most likely application is in calculating how far atoms have moved from their

lattice sites: in this case, the first input file would contain the initial conditions information (i.e. at $t = 0$), and the appropriate expression to use with the output file would be: `sqrt(x*x+y*y+z*z)`.

9.13. Cross-Reference operation

The rarely-used Cross-Reference operation carries out calculations based on coordinate data stored in the Target file used by your simulation. The user has to supply a file containing pairs of integers, which represent the 'row number' (`rw`) parameters extracted from a SNK file (probably by using the Format Columns command with parameters '`rw`' and '`rw`').

The Cross-Reference operation is used to answer the question: how far away from the origin (and from each other) were these atoms in the Target file? This is mainly useful for locating the origins of sputtered atoms.

The format of the file containing the integer pair is as follows (i.e., pairs of free-format, whitespace-delimited integers):

```
21    3
2    34
5     6
...   ...
```

The integers in each pair can be identical or different. If they are the same, the output in Columns 3 and 4 (see below) will be zero. A list of pairs of different atoms would represent (for example) the constituents of a 2-atom cluster. However, you will have to generate this file yourself (e.g. from the Find Sputtered Clusters command).

The operation reports the following information to the output file:

Column 1 (integer 1 in file)

Column 2 (integer 2 in file)

Column 3: `r0`

Column 4: `r12`

Column 5: `rho0`

Column 6: `rho12`

The meanings of the output items 3-6 are:

- `r0` = distance between centre of mass of atoms 1 and 2 in Target file, and origin of Target file
- `r12` = distance between atoms 1 and 2 in Target file
- `rho0` = lateral separation (in x,y plane) between centre of mass of atoms 1 and 2 in Target file, and origin of Target file respectively
- `rho12` = lateral separation between atoms 1 and 2 in Target file

Symbolically, these quantities may be expressed as:

- $r12 = \sqrt{(\sqrt{x[i]-x[j]} + \sqrt{y[i]-y[j]} + \sqrt{z[i]-z[j]})}$
- $\rho12 = \sqrt{(\sqrt{x[i]-x[j]} + \sqrt{y[i]-y[j]})}$
- $r0 = 0.5 * \sqrt{(\sqrt{x[i]+x[j]} + \sqrt{y[i]+y[j]} + \sqrt{z[i]+z[j]})}$
- $\rho0 = 0.5 * \sqrt{(\sqrt{x[i]+x[j]} + \sqrt{y[i]+y[j]})}$

9.14 Scattering relations

This simple tool that is available in both *Spider* and *Winnnow* calculates useful scattering parameters, based on standard binary collision formulae.

The user enters the projectile and target atomic masses (M_1 , M_2) and a laboratory scattering angle for the projectile (θ_1 Lab).

The symbol E_0 represents the Lab projectile incident energy, while E_1 and E_2 represent the energies of the projectile and target respectively, after the collision.

On pressing the <Enter> key or Evaluate button, the following quantities are calculated:

1. **θ_2 Lab**: This is the scattering (recoil) angle of the target.
2. **θ COM (1), θ COM (2)**: These are the scattering angles in the centre-of-mass coordinate system (identical for both particles). If $M_1 > M_2$, the Lab projectile scattering angle must lie between 0 and a maximum value ($\theta_{\max} = \arcsin(M_1/M_2)$). In this case there are two possibilities (1) and (2) for the COM angle, which still ranges from 0-180°. (See a textbook on classical dynamics for an explanation.) For $M_1 \leq M_2$, there is only one COM angle per Lab angle.
3. **E_1/E_0 , E_2/E_0** : These energy ratios reflect the energy transferred during the collision. If $M_1 > M_2$, there are two possibilities for energy partition. These energy ratios are completely independent of the nature of the interaction potential or primary projectile energy. (The target is assumed to be stationary initially.)

9.15. Convert SNK to POV

This command converts *.snk files (dynamics files produced by *Kalypso*) into the *.pov format required by the PovRay ray-tracing program. This facilitates the production of ray-traced images of atomic systems, and simplifies the production of animation sequences from time-lapse snapshots of a simulation.

To produce ray-traced images of simulation targets (based on Target files or other ascii files) you must first convert the ascii file to the dynamics file format using the TRG to SNK utility provided in *Winnnow*.

To produce ray-traced images (or image frames for animations), you must install PovRay, and read its documentation. PovRay is a freeware ray-tracing program that can be downloaded from: <http://www.povray.org>.

To produce animations you will need a utility that can combine individual bitmap frames into a video sequence. There are many freeware tools that can do this. Search the web for 'bmp2avi' or 'bmp to avi'.¹

The conversion routine is limited to *.snk files with 3×10^7 or less atom records (e.g. for a target with 100,000 atoms, up to 3000 runs may be processed from a single *.snk file).

User preferences are entered via the File|Set-Up menu item. The following inputs are required (the defaults will produce reasonable output for exploratory purposes):

1. Names of **input file** (*.snk) and **output (file) stem** (*.pov). The input dynamics file that will be used to produce the ray-traced image is named here. An output file stem is also specified, e.g. 'c:/myjob'. The file extension '.pov' will be added automatically to the output file stem. For an animation job (see Divide data by

¹ The freeware program PJBMP2AVI (by Paul Roberts) is used by the author. You can find it at several places on the Webb via a Google search. Try the following link (working December 2005) http://download.freenet.de/archiv_p/pjbmp2avi_1232.html.

time item below), a numeric sequence will also be appended to the file stem: `myjob1000.pov`, `myjob1001.pov` etc.

2. **Atom Identifier:** This is any name (blanks not allowed) which you choose as a label for the atom types, in order to improve readability of the *.pov file. (The default names can also be used.) Up to 6 atom types may be defined, but not all have to be used.
3. **Colour:** The atom colour can be chosen from the drop-down lists, or specified as a string in the red-green-blue form `rgb<i,j,k>` where i,j and k are ≤ 1.0 . For example: `rgb<0.5,0.2,1.0>`. When creating images that will be published as greyscale figures, it is useful to know that different shades of grey can be created using strings like `rgb<0.4,0.4,0.4>` where the 3 colour indices are the same.
4. **Radius:** Radius of atom. If you set the radius to zero, the atom will be invisible.
5. **Row number:** The rw index you enter here will define the rw value of the first atom of that type (rw = row number of the atom in the simulation project Target file; rw = 1 for the projectile). The values of rw are read from the input dynamics file. Suppose you specify row numbers 1, 2, 1000, 100000, 100000. Then atom #1 is assigned to AtomType1, atoms #2 to #999 are assigned to AtomType2, and atoms #1000 upwards are assigned to AtomType3. This example uses 100000 as an arbitrarily high index which ensures that no instances of AtomType3/4 are ever encountered. By default, the AtomType1 is associated with rw = 1 (the projectile), while all other atom types are considered to be generic target atoms of AtomType2. Even for elemental targets, you may wish to use colours to distinguish between surface and bulk atoms respectively.
6. **Length scale factor:** determines scale of image.
7. **Separate data in *.snk file by time/run (for animation etc.)** option: if selected, this choice will produce multiple output *.pov files, each consisting only of data that refer to the same elapsed simulation time, t (i.e. system snapshots that can be used as frames in an animated sequence). Filename stems will be terminated a numeric index based on the `t_i` parameter in the *.snk file (see item 1 above).
8. **Include file:** this advanced option (rarely needed) lets you specify the name of an include file that will be referenced from the *.pov file(s) that you generate. This allows you to use other features of PovRay that are not incorporated in PovData's output file. See item 12 for a quick way to include custom expressions in the PovRay file. See the PovRay Help for general information about include files.
9. **Camera location:** defines (x, y, z) coordinates of the camera.
10. **Camera look-at:** defines (x, y, z) coordinates of point to which camera is pointed, and the perspective angle. There are two ways to view a target: from a short distance with a large angle, or from a long distance with a small angle. The latter gives less distortion due to perspective effects.
11. **Light sources:** defines (x, y, z) coordinates of your light sources (the checkboxes turn the associated light sources on or off).
12. **Extra options:** Any strings entered here will be written at the top of the PovRay file. For example, the string `background{<1 1 1>}` produces a white background (useful for work that will be published on paper).
13. **Shiny finish:** puts a realistic, mirror-like 'twinkle' on the atoms (otherwise the finish is dull). The appearance of the finish is defined by several finish parameters (see PovRay documentation for explanations of these).

If you want to **Save** the current settings, or **Load** settings previously saved, use the applicable buttons. This is always a good idea, because visualization is an iterative process.

Using PovRay

- (a) The output file (*.pov) generated by the above procedure can be loaded immediately into PovRay [File|Open File].
- (b) The image can be rendered via the command Render|Start Rendering (output image is automatically saved as a bitmap file). For animated sequences use the Queue command (Alt-Q hotkey) to process files as a batch.
- (c) Note that PovRay offers many rendering options, including different image sizes (Render|Edit Settings).

To get the best out of PovRay, you have to study its documentation. However, relatively little knowledge of ray-tracing is involved for this type of application. Note that PovRay input files can be combined (either by splicing together, or by using the #include directive), allowing you to superimpose different scenes. Just make sure that the atom identifiers are unique for each scene. Tip: use the 'AA' render options for best results (Render|Edit Settings menu item in PovRay).

9.16. Convert TRG to SNK

This routine converts a Target file into the format (*.snk) used by dynamics files. The following dynamics file variables are initialised to the values read from the Target file (as usual, in SI units):

rx, ry, rz, ms, fl

Other variables are initialised to meaningless values (either 1 or 0). If the input file has an extension other than TRG, only the rx, ry, and rz fields are read from the file. The routine performs a simple file name conversion: *myfile.trg* is converted to *myfile.snk*. If the output file (A.snk) exists, the user is prompted to confirm the overwrite operation (otherwise the routine aborts).

9.17. Reformat MI data

This operation is designed to compensate for the row indexing problem discussed in section 4.2.7 of Chapter 4 that arises in multiple interaction (MI) simulations involving **inert atom projectiles**. The records in the dynamics file are modified according to the following prescription: $rw \rightarrow rw - rn * nprj + 1$, where *nprj* represents the number of atoms in the projectile species (normally = 1, since it is an inert atom by assumption).

The effect of this operation is that the row number (*rw*) fields of target atom records will now commence from $rw = 2$ regardless of the run number, as they would in a normal simulation. This indexing scheme is more convenient for further processing of the target atom data. For example, any atom with $rw < 2$ in the reformatted file can be

identified as a projectile atom (incident or embedded) and any atom with $r_w > 1$ can be identified as a target atom. This simplifies the construction of filters.

Projectile data will be retained by this operation, but the r_w data will be shifted by this operation to negative values (and also to 0 and 1). Although the new values are systematically related to the original values, they are less easily manipulated than the r_w values in the original file.

Do not apply this operation if the projectile is a metallic atom or cluster.

9.20. Neighbour count

This operation analyses a SNK file in order to count the neighbours of each atom that lie within a certain range of the atom, R . Typically, the routine would be used to determine the coordination numbers of each atom, in which case R would be slightly larger than the 1st neighbour distance in the undisturbed lattice. The output consists of (a) a SNK file that is similar to the input file, except that the r_n field for each atom has been substituted with the number of neighbours of that atom; (b) a numeric ascii file (*.dat) that lists the r_w indexes of the neighbours according to the following format:

```
3 35.59784023 35.62925732 -1.09775612 6 4 32 843 844 872 873
```

The first number (3) is the r_w index of the atom concerned. The next three number are its (x, y, z) coordinates expressed in Å. The next number (6) is the number of neighbours around that atom, and the remaining numbers (4...873) are the r_w indexes of the neighbours (6 in this case).

Annexe: Query Expression Examples

In this annexe, some further examples of query expressions are given. Recall that these arise in two contexts:

- Function specifications (e.g. for construction of spectra or averages)
- Conditional expressions (for filtering operations only)

The conditional expressions combine function specifications with relational and logical operator symbols. Conditional expressions are only used by *Winnow's* Filter option. In the examples below, comments are enclosed in curly braces {thus}: comments can be used freely in *Winnow*, since they are ignored by the parser. The examples below show how to build up conditional expressions by combining several simple relations. If you are uncertain about whether you have set up a filter expression correctly (or whether the program works correctly), you can convert the output file to a text file and check that its records satisfy the intended condition.

Function (expression) specifications

The following are some examples of functions of variables that can be expressed in the query language.

- `ke {particle kinetic energy in eV}`
- `atan(vz/sqrt(sqr(vx)+sqr(vy))) {altitudinal flight angle, ψ }`
- `alt {altitudinal flight angle, ψ - concise version of preceding}`

- `rz*1e10` {z-position in Å}
- `ti*1e15` {elapsed time in fs}
- `rw` {index of the particle}
- `sqrt(rx^2 + ry^2)` {lateral distance of particle from the origin}

Conditional expressions

The following examples show what is possible in terms of conditional expression combination and nesting. However, in practice it may be better to use a sequence of simpler expressions, especially if the query language is unfamiliar.

```
[rw > 1]
{ this expression filters out projectile atoms, i.e. keeps target atoms }
```

```
[rw > 1] & [rz > 7E-10]
{ this keeps sputtered target atoms (those above z = 7 Å ) }
```

```
[rw = 1] & [rz > 7E-10] & [vz > 0.0]
{ this keeps reflected projectile atoms (those above z = 7 Å ) }
```

```
[alt > 50.0]
{any particle travelling with altitudinal angle > 50°}
```

```
[rz > 0.0] & [altd > 50.0]
{any particle that is above z = 0.0 and travelling with altitudinal angles greater than 50°}
```

```
[ke > 10.0] & [rz > 0.0] & [altd > 50.0]
{any particle with energy > 10 eV and that is above z = 0.0 and travelling with altitudinal angles greater than 50°}
```

```
[ti*1e15 < 300.0] & [ke > 10.0] & [rz > 0.0] & [altd > 50.0]
{particles with energy > 10 eV which were emitted at altitudinal angles greater than 50° before 300 fs had elapsed}
```

```
[ti*1e15 < 300.0] & [[ke >= 10.0] & [ke <= 50.0]] & [rz > 0.0] & [altd > 50.0]
{particles with energy in the range 10-50 eV which were emitted at altitudinal angles greater than 50° before 300 fs had elapsed}
```

Suppose we wanted to filter using the following criteria:

1. The record should refer to an elapsed time that is less than 300 fs.
2. The particle energy should be between 10 and 50 eV.
3. The particle's z-coordinate should be greater than 5 Å
4. The particle's altitudinal direction of travel should be between 50 and 90°

This filtering operation could be achieved by joining the following expressions with a series of AND operators:

1. `[ti*1e15 < 300.0]`
2. `[ke >= 10.0] & [ke <= 50.0]`
3. `[rz > 5.0E-10]`
4. `[altd > 50.0]`

The resulting filter expression is:

```
[ti*1e15 < 300.0] & [ke >= 10.0] & [ke <= 50.0] & [rz > 5.0E-10] &
[altd > 50.0]
```

Exercise 1.

What would be the effect of substituting:

```
[ke <= 10.0] & [ke >= 50.0]
```

instead of the kinetic energy items item above?

Answer. You would get no output from your filtering operation, since this expression looks for particles with KE satisfying two mutually exclusive conditions ($KE \leq 10$ eV and $KE \geq 50$ eV).

Exercise 2.

What would be the effect of substituting:

```
[ke <= 10.0] | [ke >= 50.0]
```

instead of the kinetic energy items above?

Answer. The expression can be read as: ($KE \leq 10$ eV or $KE \geq 50$ eV). Your output would capture all particles except those in the energy range 10-50 eV.

References for Chapter 9

[1] S.D. Stoddard, J. Comp. Phys. 27 (1977) 291.

APPENDIX 1: KALYPSO ERRORS AND WARNINGS

Error messages are issued when *Kalypso* encounters conditions that cause a simulation to abort. At the end of the message is a bracketed [hint] about where the error occurred (either in an input file or inside a *Kalypso* routine) or a specific reference to the cause of the error. If (a) the input parameters are sensible, and (b) you have not edited input files by hand, most of these errors should never occur (i.e. if they do occur, they represent programming errors). Rarely, an error could occur because of a surge in the computer power supply. Users should be aware that *Kalypso* does not attempt to check for all errors that can be produced by incorrect file formats: that is the job of Spider.

Warning messages typically draw attention to anomalies in input data (typically redundant input options) that may be due to an oversight on the part of an inexperienced user. However, the simulation *as specified* will still run correctly.

Error conditions can trigger several error messages, some of which may be spurious. Make sure you scroll up to the top of the **Messages** pane in *Kalypso* to find out which message appeared first. For example, an attempt to resume a multiple impact simulation will produce the following messages:

Messages

Error E035: Multiple impact simulations cannot be resumed.

Error E034: Error reading parameters in Target file.

Error E023: Atomic numbers inconsistency [TRG or PRJ data vs. MDL data].

Only the first of these messages represents a real error. The other two are triggered because of actions that failed to complete after the first error condition was detected.

Vaguely defined error messages (such as E001 or E003-E006) may arise from hardware failures, such as disruptions to the power supply. Hardware-related errors are not reproducible, unlike other error types (provided the random seed setting in the Run file is > 0). If an error cannot be reproduced, or if it occurs after several days of computation, you should suspect a hardware problem. If the error is reproducible, but the cause cannot be determined, you can send a description of the problem (including the program version listed on the About dialog box) *with the input files* to the author.

Some types of reproducible error will produce a dialog box that states that an error occurred, but no corresponding error message in the log. You should first check that your Target and Projectile files do not contain atoms with duplicated coordinates.

Error messages

The following is a list of error messages returned by *Kalypso* and/or *Spider*. In some cases, further information about the meaning of the message is provided in *italic* font. The [square brackets] indicate the input data source or code portion that is responsible for the error (where applicable). The phrase *math error* that arises in several of the error messages refers to a non-specific numeric exception (e.g. due to memory corruption). The phrase *internal error* refers to a debugging feature (you should never see this message.)

'Error E001: Math error in timestepping loop [ComputeTraj].';

'Error E002: No. of atoms in system exceeds maximum.'; *Maximum is 1 million atoms (although this is probably not feasible on current hardware).*

'Error E003: Math error in density loop [forces].';

'Error E004: Math error in density loop [energy].';

'Error E005: Math error in forces loop [GetRV].';

'Error E006: Math error in forces loop [GetRVInit].';

'Error E007: Invalid Target file data [Z].'; *Example: three types of atoms are specified.*

'Error E008: Insufficient number of replica atoms [RUN].';

'Error E009: Range-search grid cannot be sized.'; *Example: a huge target.*

'Error E010: Invalid Model file data [$R_c < R_s$].';

'Error E011: Invalid Model file data [Cell size $< R_c$].';

'Error E012: Input file version mismatch [RUN].';

'Error E013: Invalid projectile incident angle [RUN].';

'Error E014: Input file version mismatch [MDL].';

'Error E015: Output expression syntax error [RUN].'; *Example: writing ($rw = 1$) instead of [$rw = 1$].*

'Error E016: Invalid Projectile file data [Z].';

'Error E017: Too many collision partners [neighbour loop]'; *Note: this error can occur if you allow two atoms to approach to an unphysically close distance, e.g. if the timestep is unreasonably large, or if there is an error in the initialisation procedure (such as two target atoms having the same x, y, z coordinates).*

'Error E018: Too many collision partners [ST loop].'; *Similar to E017.*

'Error E019: Impossibly large ST loss [in GetSTLoss].'; *Algorithm error.*

'Error E020: Too many collision partners [OR loop].'; *Similar to E017.*

'Error E021: Image potential parameters [INL].'; *Generated if ($V_{im} \leq 0.0$) or ($z_{im} \leq 0.0$)*

'Error E022: Math error in cooling schedule.';

'Error E023: Atomic numbers inconsistency [TRG or PRJ data vs. MDL data].'; *For example, if the Model file refers to Cu and Ni atoms, but the Target file contains a Ag atom (only 2 target atom types are allowed). This error also arises if there are blank lines (i.e. lines that consist of one or more space characters) inside, or blank characters at the end of, a Target file that has been edited by hand (you should **always** check for this problem if you edit Target files yourself).*

'Error E024: Numeric format error in TRG file.';

'Error E025: Numeric format error in PRJ file.';

'Error E026: Merging and switching function regions cannot overlap [MDL].'; *In the model file (switching functions tab) the following condition must be true: $R_1 < R_2 < R_s < R_c$.*

'Error E027: Pairwise part of potential must be symmetric between particles [MDL].';

'Error E028: Internal error: Anomaly in projectile mass sum [PRJ].';

'Error E028: Error accessing log file.'; *For example, a read-only folder is encountered.*

'Error E029: Cannot terminate by projectile if no projectiles in system [RUN].';

'Error E030: Scan range too wide (reduce it, or increase number of partners)'; *This message can arise when doing layer scans with Spider (it is written to the Spider log file).*

'Error E031: Periodic cell dimensions are too small for the specified potential cut-off [RUN/MDL].'; *R_c must be less than $L_x/2$.*

'Error E032: Lattice atom index out of range [options].';

'Error E033: Cannot compute potential - atomic index out of range.';

'Error E034: Error reading parameters in Target file.';

‘Error E036: Potential parameters: if $b \neq 0$, hetero terms $2q$ must be symmetric [MDL].’; *i.e. if $b \neq 0$, Kalypso requires that $2q[0, 1] = 2q[1, 0]$. See footnote to section 2.4 of this User Guide.*

‘Error E037: Multiple impact simulations cannot be run with pre-implanted projectile species.’ To work around this, see the discussion concerning the `ofPreImplant` flag.

Warning messages

The following is a list of warning messages returned by *Kalypso*.

‘Warning W001: Project uses only 1 of 2 specified potentials.’; *For example, if the target is a Cu crystal, but the Model file specifies potentials for a Cu-Ni target, a warning will be issued.*

‘Warning W002: LSS option disabled [null parameters in INL file].’;

‘Warning W003: Oen-Robinson option disabled [null parameters in INL file].’;

‘Warning W004: ST inelastic option disabled [null parameters in INL file].’;

‘Warning W005: Image potential will never be applied [no atom has flag set].’;

‘Warning W006: Inelastic file has no active models [INL].’;

‘Warning W007: Using adaptive timestep without any projectile may give erratic results.’;

APPENDIX 2: FUNDAMENTAL PHYSICAL CONSTANTS

The values of fundamental physical constants used by *Kalypso* (2.01 onwards) are the so-called 2002 CODATA values (online at URL <http://physics.nist.gov/constants>):

$$\begin{aligned} e &= 1.60217653 \times 10^{-19} \text{ C}; \\ k &= 1.3806505 \times 10^{-23} \text{ J K}^{-1}; \\ \epsilon_0 &= 8.854187817 \times 10^{-12} \text{ F m}^{-1}; \\ 1 \text{ amu} &= 1.6605386 \times 10^{-27} \text{ kg}; \\ \hbar &= 1.05457168 \times 10^{-34} \text{ J s}. \end{aligned}$$

N.B. *Kalypso* 2.00 used the 1998 CODATA values, which differ slightly from the 2002 values.