Cone

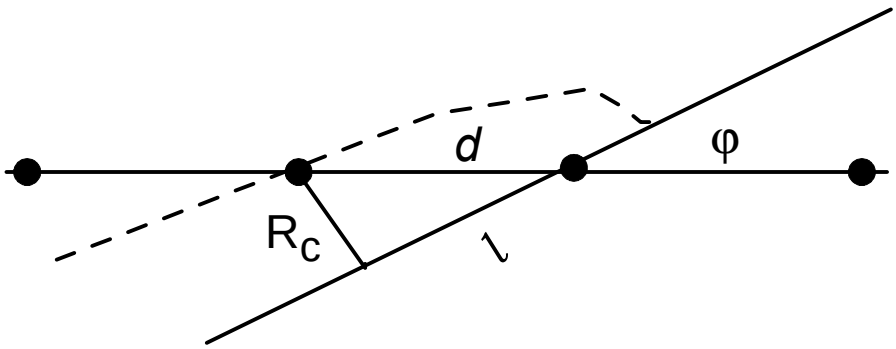
This file briefly describes the utility program *Cone.exe*, which was written by Tan Hean Seng for his final year honours project: "The Determination of Shadow Cone Dimensions for Medium Energy Ar, Kr and Xe Ions." (Physics Department, National University of Singapore, 1992). I am grateful to Hean Seng for allowing me to release the program (with minor modifications) into the public domain

Note: *Cone* is a DOS program. You can run it under DOS, or under Windows 95 in a DOS box or in DOS mode (the latter is recommended, because it is very much faster). To use *Cone*, you need to be conversant with the concepts behind the Impact Collision Ion Scattering Spectrometry (ICISS) experiment.

Version 2 changes to *Cone*:

- The ZBL screening length can be scaled by an arbitrary factor. The user is prompted to enter a screening length correction factor (which would normally be close to 1.0). Enter 1.0 if you don't want to correct the ZBL screening length.
- The Borland Pascal fast CPU bug has been patched (crashed *Cone* on CPUs > 200 MHz)

Cone calculates shadow cones for binary atomic collisions. The problem solved by *Cone* can be stated briefly as follows: given an interatomic separation *d*, what is the angle φ between the interatomic axis and the projectile direction of incidence at the shadowing threshold (see diagram)?



Naturally, it would be nice if you could input *d* and get an estimate of φ. But for various reasons, that is quite complicated to program. Instead, *Cone* demands that you input a related parameter *l* (= *d*cos φ), whereupon the corresponding φ and *d* are computed. (Internally, *Cone* calculates the cone radius, R_C, as a function of *l*.) You may have to input several trial values of *l* until you hit upon the right one (signalled by the correct *d* for your lattice spacing.)

Cone was the basis of the paper: H.S. Tan and M.A. Karolewski, Nucl. Instrum. Methods B73 (1993) 163-171, to which the reader is referred for further details.

Cone gets its atomic data either from the file ELEMENT (if it is located in the working directory) or from the terminal. You can edit ELEMENT to suit your needs (the file currently only lists selected elements, not all).

The equations of motion for the binary collisions are integrated using the Runge-Kutta algorithm, assuming a ZBL potential with unadjusted screening length. All input and output data refer to the Laboratory coordinate frame.

Cone command line options:

- `cone` - interactively output φ, and R_c for specified l .
- `cone /l` - write (l, R_c) output to file 'cone.dat'.
- `cone /d` - write (d, φ) output to file 'cone.dat'.

With the /l option, *Cone* will output the (l, R_c) coordinates which define the shadow cone envelope. With the /d option, *Cone* will output the critical angle corresponding to a specified interatomic separation (d).

Cone uses the ZBL potential to describe the projectile-target interaction. The Runge-Kutta algorithm is used for integration. At the conclusion, *Cone* outputs the energy error which is generally very small as these calculations are rather accurate.

If you want to use *Cone* for interpreting ICISS experiments, bear in mind the limitations of the binary collision model. In particular, (a) If your projectile is He, then *Cone* will be accurate to perhaps 0.5 degrees for shadowing of atoms in the surface layer, and perhaps 1-1.5 degrees for shadowing of atoms in sub-surface layers; (b) For heavy projectiles, *Cone* may be inaccurate for shadowing processes involving sub-surface atoms, or very shallow critical angles (< 10 degrees) . These are guidelines, not guarantees!

Cone is an excellent tool for quickly locating critical angles. However, if you have to include large screening length corrections to force a fit between *Cone* and your experiment you may need to do a full CD simulation.

If the projectile is heavier than the target atom, you may see a message "Shadow cone does not form at this distance" when you try to calculate the cone radius. See the paper referenced above for an explanation.

Cone will abort with a run-time error if it encounters a numeric format error in the input, or occasionally if it is presented with an impossible physical task, e.g. 5 eV Cu projectile on a boron target (no shadow cone is possible at any distance). Apologies for the poor error handling. (But please bear in mind that these errors are committed by the user, not by *Cone*) I have never encountered any true errors in *Cone*, and seriously doubt whether there are any.

Scribe: Marcus Karolewski, University of Alberta (revised April 2001)
e-mail: karolewski@alum.mit.edu