

- The CMEE library for computer modeling of ion-material interactions •
- 

Peter Stoltz  
Tech-X Corp.

Seth Veitzer, Andrew Prideaux  
Tech-X Corp.

M. Furman, J. L. Vay  
LBNL

R. Cohen, A. Molivk  
LLNL

Presented by: David Bruhwiler, Tech-X Corp.



HB2004, OCT 2004  
PETER STOLTZ  
TECH-X CORPORATION • BOULDER CO

Problem: researchers would like to use simulation to understand electron cloud, but writing all necessary physics models (SEY, ionization, etc) is difficult

---

- Electron effects limit the performance of many of today's ion accelerators
- Simulation is one of the most widely-used tools in understanding electron cloud effects, but writing numerical models of all the necessary physics is difficult
- Tech-X and collaborators are working to provide in a cross-platform, cross-language way the best models of electron cloud effects like secondary yield, ion-wall interactions, and gas ionization



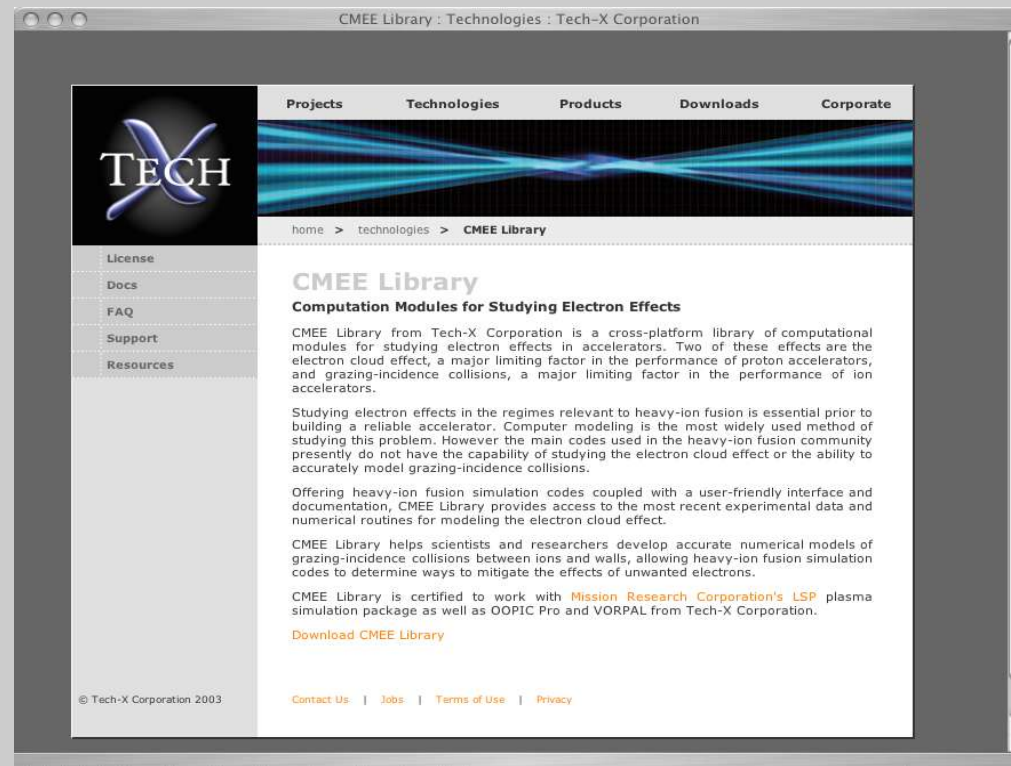
# The CMEE library is a collection of numerical routines for modeling electron effects

---

- CMEE = Computational Modules of Electron Effects
- Latest version of CMEE now provides routines for modeling
  - secondary electron yield
  - Ion stopping, range, and ion-induced electron yield
  - neutral gas ionization by electrons and protons
- The approach is
  - use tested routines from community where possible
  - make them available on any platform or language



# The CMEE library v1.0 is now available (free for research and non-commercial use)



Newly-available CMEE v1.0 includes documentation and examples



HB2004, OCT 2004  
PETER STOLTZ  
TECH-X CORPORATION • BOULDER CO

# Building library is automated with GNU tools

```
Terminal — vim — 80x24
[localhost ~/projects/cmee] pstoltz% ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... no
checking for mawk... no
checking for nawk... no
checking for awk... awk
checking whether make sets $(MAKE)... yes
checking whether to enable maintainer-specific portions of Makefiles... no
checking for gcc... gcc
checking for an ANSI C-conforming const... yes
checking whether ln -s works... yes
checking whether we are using the GNU C++ compiler... (cached) yes
checking whether g++ accepts -g... (cached) yes
checking dependency style of g++... (cached) gcc3
configure: creating ./config.status
config.status: creating Makefile
config.status: creating src/Makefile
config.status: creating Python/Makefile
config.status: creating config.h
config.status: executing depfiles commands
[localhost ~/projects/cmee] pstoltz%
~
```

You type 'configure'

Makefiles created for you!



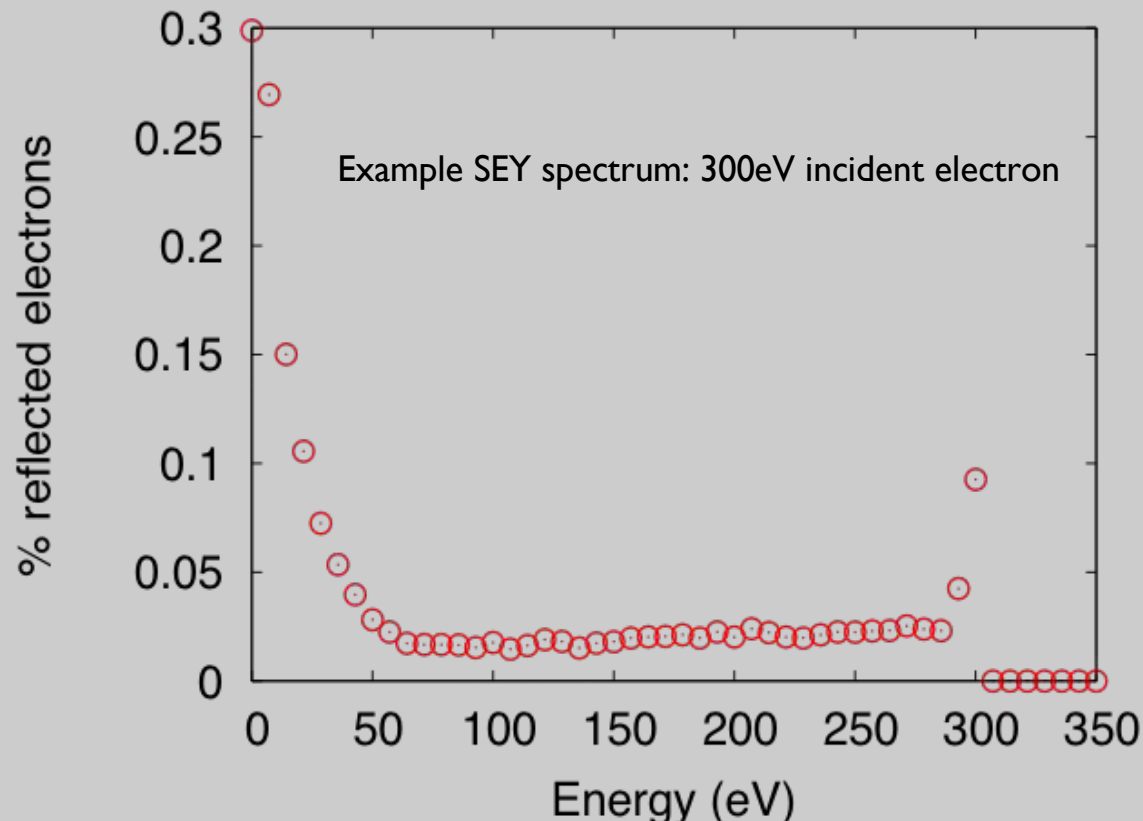
# Building library is automated with GNU tools

```
Terminal - vim - 77x23
[localhost ~/projects/cmee] pstoltz% make
make all-recursive
Making all in src
rm -fr .libs/libcmee.lax/libmi.al
mkdir .libs/libcmee.lax/libmi.al
(cd .libs/libcmee.lax/libmi.al && ar x /Users/pstoltz/projects/cmee/src/../../io
npack/src_impact/.libs/libmi.al)
rm -fr .libs/libcmee.lax/libsecno.al
mkdir .libs/libcmee.lax/libsecno.al
(cd .libs/libcmee.lax/libsecno.al && ar x /Users/pstoltz/projects/cmee/src/..
/posinst/src/.libs/libsecno.al)
rm -fr .libs/libcmee.lax/libcrangeno.al
mkdir .libs/libcmee.lax/libcrangeno.al
(cd .libs/libcmee.lax/libcrangeno.al && ar x /Users/pstoltz/projects/cmee/src
/../../txcrange/src/.libs/libcrangeno.al)
ranlib .libs/libcmee.a
rm -fr .libs/libcmee.lax
creating libcmee.la
(cd .libs && rm -f libcmee.la && ln -s ../libcmee.la libcmee.la)
make[2]: Nothing to be done for `all-am'.
[localhost ~/projects/cmee] pstoltz%
~
```

Users now link their code to this library



# CMEE secondary electron model based on POSINST routines



- Furman and Pivi developed the routines at LBNL

(see: M. A. Furman and M. Pivi, Phys. Rev. ST Accel. Beams 5, 124404 (2002)).

- POSINST models copper and stainless now, with TiN and other materials to come



# Example: Python code for using secondary electrons

---

```
import cmee

ns = cmee.intArray(1)
bn = cmee.doubleArray(10)
bt = cmee.doubleArray(10)
bz = cmee.doubleArray(10)

# cmee expects incident electron energy in eV
inc_energy = 100.

# cmee expects angle of incidence expressed as
# cosine of the angle measured with angle zero
# as normal (costheta = 1. at normal incidence)
costheta = 0.001

# The material the electron hits is described
# by an integer (1=copper, 2=stainless steel)
mat_number = 1

# The call to nsec fills the return arrays with
# the appropriate values
cmee.nsec(inc_energy, costheta, mat_number, ns, bn, bt, bz)
```

SWIG-generated wrapper for POSINST  
routine *nsec*



# Example: Python code for using secondary electrons

---

```
[localhost ~/projects/cmee/Python] pstoltz% python cmee_examples.py
```

```
#####EXAMPLE I#####  
Secondary electron yields  
#####
```

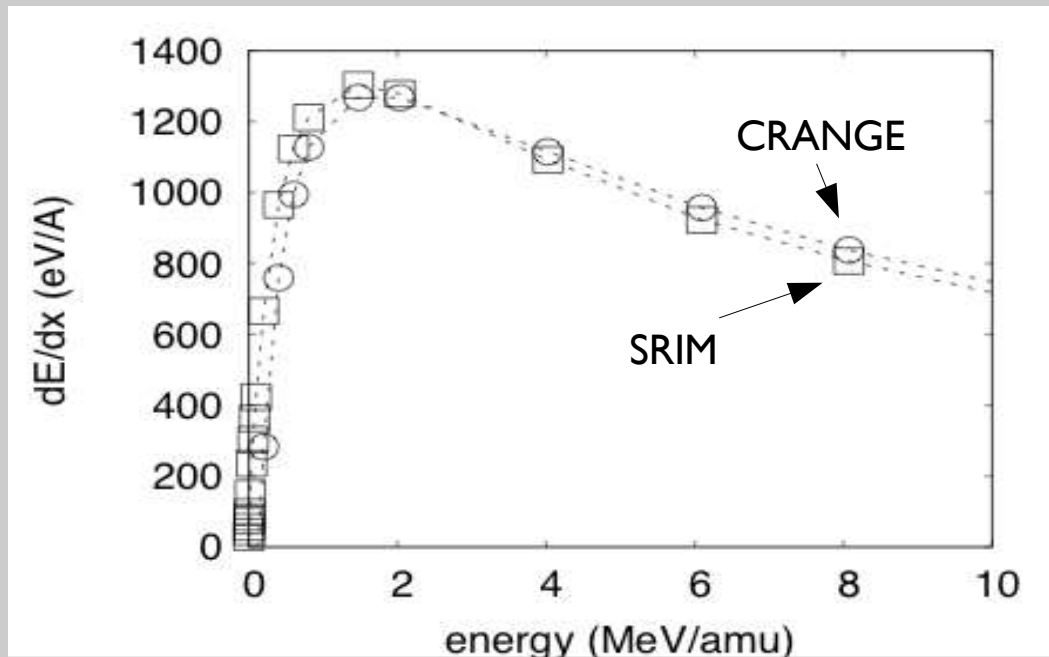
```
Number of secondaries: 4  
beta of electron 0 : 0.00575132547057  
beta of electron 1 : 0.00512703369113  
beta of electron 2 : 0.00645169479077  
beta of electron 3 : 0.00511909873649
```



Results of Python call to *nsec*



# CMEE ion-material routines are based on CRANGE code



Comparison of  $dE/dx$  from CRANGE and SRIM for 1.0 MeV  $K^+$  hitting stainless steel

- B.A. Weaver developed the routines at Space Science Lab  
(see: B. A. Weaver and A. J. Westphal, Nucl. Inst. Meth. B 187 (2002) 285–301).

- CRANGE compares well with SRIM, but is open-source and runs on Linux and Mac



# Example: Python code for using ion-material range and electron yield routines

```
import cmee
import set_crangle_params

# First read a file of parameters defining which options ('switches') are active in
temporary_results=set_crangle_params.set_switches('./txcrangle/src/switch.dat')
FLS,FNS,FBMA,FBA,FSH,FLE,FD,FE,FKIN,FRAD,FPA,FBR=temporary_results

# Set the atomic number and mass of the projectile ion
z_p = 1.
a_p = 1.

# Set the energy of the incident ion in MeV/nucleon
thisenergy = 1.

# Now calculate the range and desorbed electrons.
thisrange=cmee.calc_range(thisenergy,rel0_p,z_p,a_p,
                          FNS,FRAD,FBR,FLS,FBA,FBMA,FD,FKIN,FSH,FLE,FE,FPA,
                          z2,a2,iadj,bind,pla,X0,X1,a,m,d0,etad,rho)
thiselecs=cmee.ion_induced_elecs(thisenergy,rel0_p,z_p,a_p,
                                  FNS,FRAD,FBR,FLS,FBA,FBMA,FD,FKIN,FSH,FLE,FE,FPA,
                                  z2,a2,iadj,bind,pla,X0,X1,a,m,d0,etad,rho)
```

Most of these parameters are set automatically by a helper Python script



HB2004, OCT 2004  
PETER STOLTZ  
TECH-X CORPORATION • BOULDER CO

# Example: Python code for using ion-material range and electron yield routines

```
[localhost ~/projects/cmee/Python] pstoltz% python cmee_examples.py
```

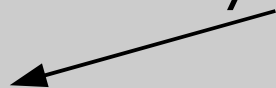
```
#####EXAMPLE 1#####  
Secondary electron yields  
#####
```

```
Number of secondaries: 4  
beta of electron 0 : 0.00575132547057  
beta of electron 1 : 0.00512703369113  
beta of electron 2 : 0.00645169479077  
beta of electron 3 : 0.00511909873649
```

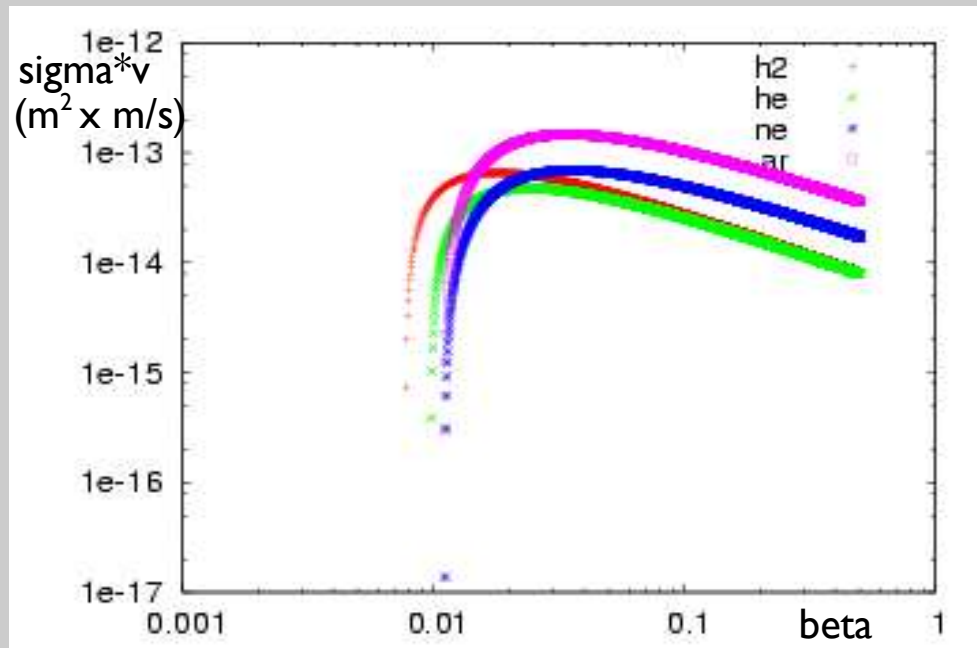
```
#####EXAMPLE 2#####  
Ion energy loss, range, and induced electron yields  
#####
```

```
Energy loss (eV/A): 11.1689156451  
Range (cm): 0.00064079632127  
Average induced electron yield 1.56364819031
```

CMEE results for dE/dx, range and electron yield for a 1.0 MeV proton on Cu



# CMEE impact ionization routines are based on fits from Reiser



CMEE cross sections for electrons ionizing H<sub>2</sub>, He, Ar, Ne

- Impact ionization cross sections come from empirical fits

(see: M. Reiser, Theory and Design of Charged-Particle Beams (Wiley, New York, 1994)).

- CMEE contains fits for electrons and protons ionizing H, H<sub>2</sub>, He, Ar, Ne, N<sub>2</sub>, O<sub>2</sub>, CO, CO<sub>2</sub>



# Example: Python code for using impact ionization routines

---

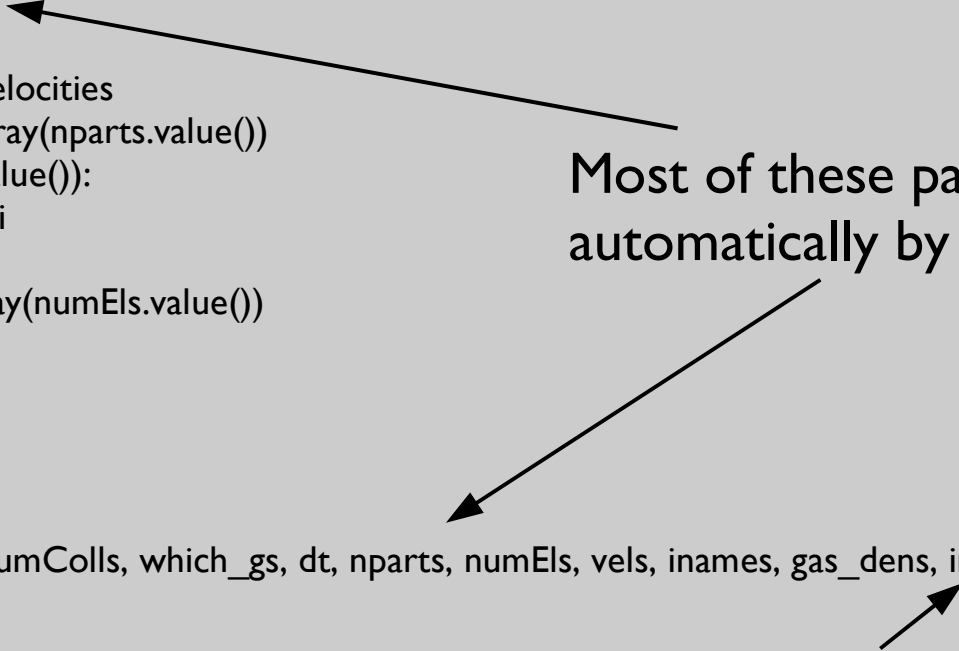
```
import cmee

# num. of incident particles
nparts = cmee.intp()
nparts.assign(10000)

#Initialize values for velocities
vels = cmee.doubleArray(nparts.value())
for i in range(nparts.value()):
    vels[i] = 5.e6 + 1.e4*i
# Initialize gases
inames = cmee.intArray(numEls.value())
inames[0]=1; # H2
inames[1]=2; # He
inames[2]=7; # N2
inames[3]=8; # Ne

cmee.get_Collisions(numColls, which_gs, dt, nparts, numEls, vels, inames, gas_dens, incident)
```

Most of these parameters are set automatically by a helper Python script



Flag for type of incident particle  
(0=electron, 1=proton)



# Example: Python code for using impact ionization routines

---

```
[localhost ~/projects/cmee/Python] pstoltz% python cmee_examples.py
```

```
#####EXAMPLE 3#####  
Impact ionization electron yields  
#####
```

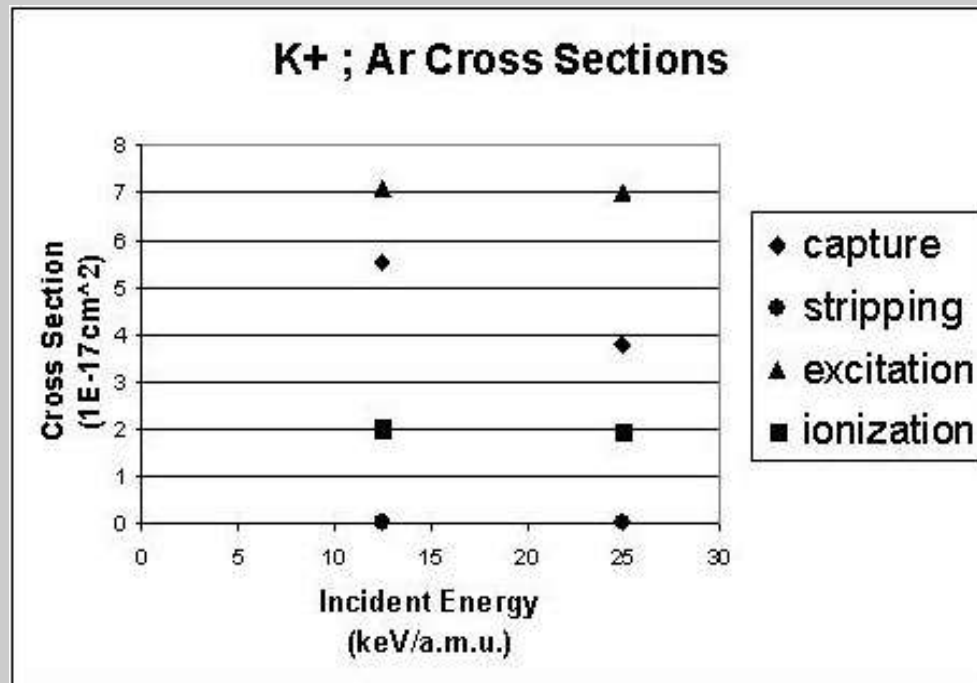
```
The number of colls is: 1599  
For gas 1 : 207 # H2  
For gas 2 : 202 # He  
For gas 7 : 783 # N2  
For gas 8 : 407 # Ne  
Null: 1693
```

Of the 10,000 particles, the null collision algorithm picked 3300 for possible collision, and 1600 collided

Number of collisions with each gas



The next release of CMEE will include some heavy-ion cross sections (ionization, stripping, capture, excitation)



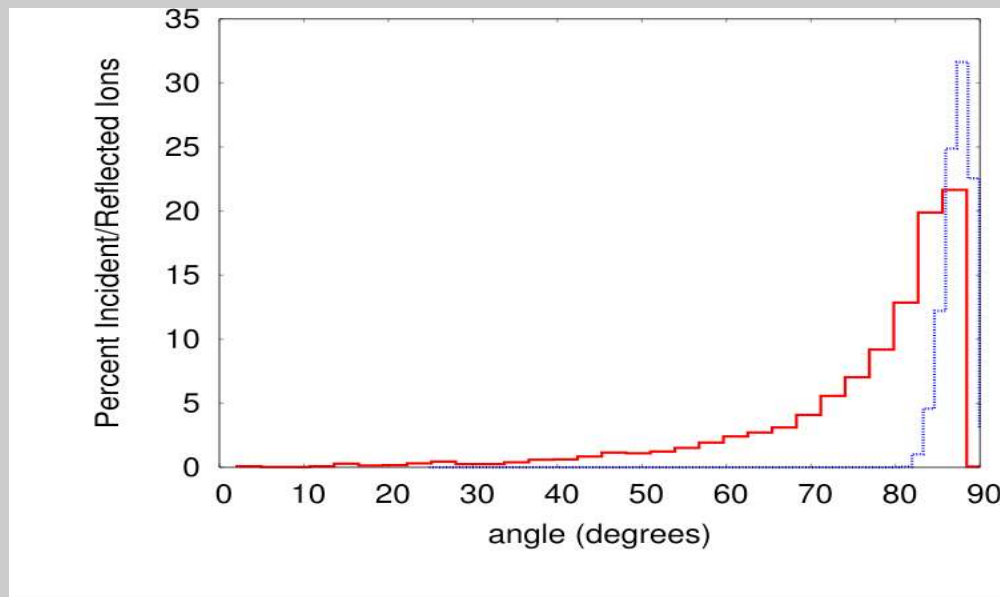
Cross Sections for 1.0 MeV K<sup>+</sup> hitting neutral Ar gas

- Results come from the quantum mechanical close-coupling code of Lin's group  
(see: C. N. Liu, A. T. Le, T. Morishita, B. D. Esry, and C. D. Lin, Phys. Rev. A 67, 051801 (2003) ).
- We plan soon to benchmark this code to Olson's results for Xe hitting N<sub>2</sub>



# The next release of CMEE will also include ion scattering

---



- We are adding access routines to tabulated SRIM results
- We are also adding scattering to CRANGE, possibly using technique of Thieberger, et al, (Phys. Rev. ST Accel. Beams 7, 0932 (2004))

1.0 MeV K<sup>+</sup> scattering from stainless steel (blue is the incident distribution, red is scattered)



HB2004, OCT 2004  
PETER STOLTZ  
TECH-X CORPORATION • BOULDER CO