# • The CMEE library for numerical modeling of • electron effects

Peter Stoltz and Seth Veitzer*
Tech-X Corp.
pstoltz@txcorp.com

Ron Cohen and Art Molvik
LLNL

Miguel Furman and Jean-Luc Vay
LBNL

*The work of Tech-X on this project is supported by the DoE OFES through the SBIR program

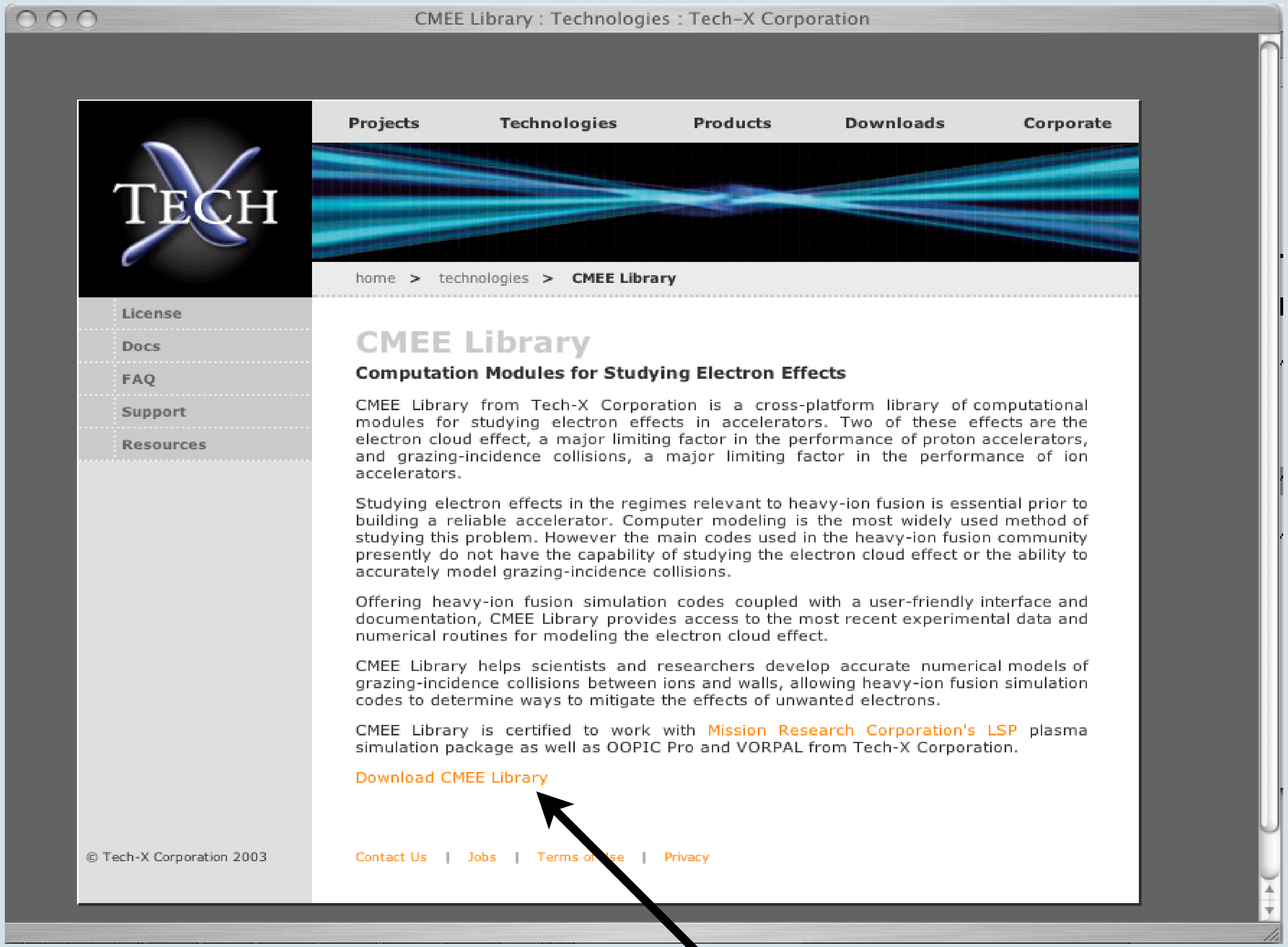
## The CMEE library is a collection of numerical routines for modeling electron effects

- CMEE = Computational Modules of Electron Effects
- The goal is to provide routines for modeling electron effects (SEY, ionization rates, etc)
- The approach is to use tested routines from the community and make them available to other codes

## CMEE presently contains secondary electron yield modules from POSINST

- CMEE right now contains the secondary electron routines from POSINST
- Next models:
  - Ion-induced electrons
  - Neutral desorption
  - Impact ionization
  - Ion scattering
  - Energy loss

# Users download CMEE and build library locally


CMEE Library : Technologies : Tech-X Corporation
Projects
Technologies
Products
Downloads
Corporate
TECH
home > technologies > CMEE Library
License
Docs
FAQ
Support
Resources
© Tech-X Corporation 2003
Contact Us | Jobs | Terms of Use | Privacy

CMEE Library

Computation Modules for Studying Electron Effects

CMEE Library from Tech-X Corporation is a cross-platform library of computational modules for studying electron effects in accelerators. Two of these effects are the electron cloud effect, a major limiting factor in the performance of proton accelerators, and grazing-incidence collisions, a major limiting factor in the performance of ion accelerators.

Studying electron effects in the regimes relevant to heavy-ion fusion is essential prior to building a reliable accelerator. Computer modeling is the most widely used method of studying this problem. However the main codes used in the heavy-ion fusion community presently do not have the capability of studying the electron cloud effect or the ability to accurately model grazing-incidence collisions.

Offering heavy-ion fusion simulation codes coupled with a user-friendly interface and documentation, CMEE Library provides access to the most recent experimental data and numerical routines for modeling the electron cloud effect.

CMEE Library helps scientists and researchers develop accurate numerical models of grazing-incidence collisions between ions and walls, allowing heavy-ion fusion simulation codes to determine ways to mitigate the effects of unwanted electrons.

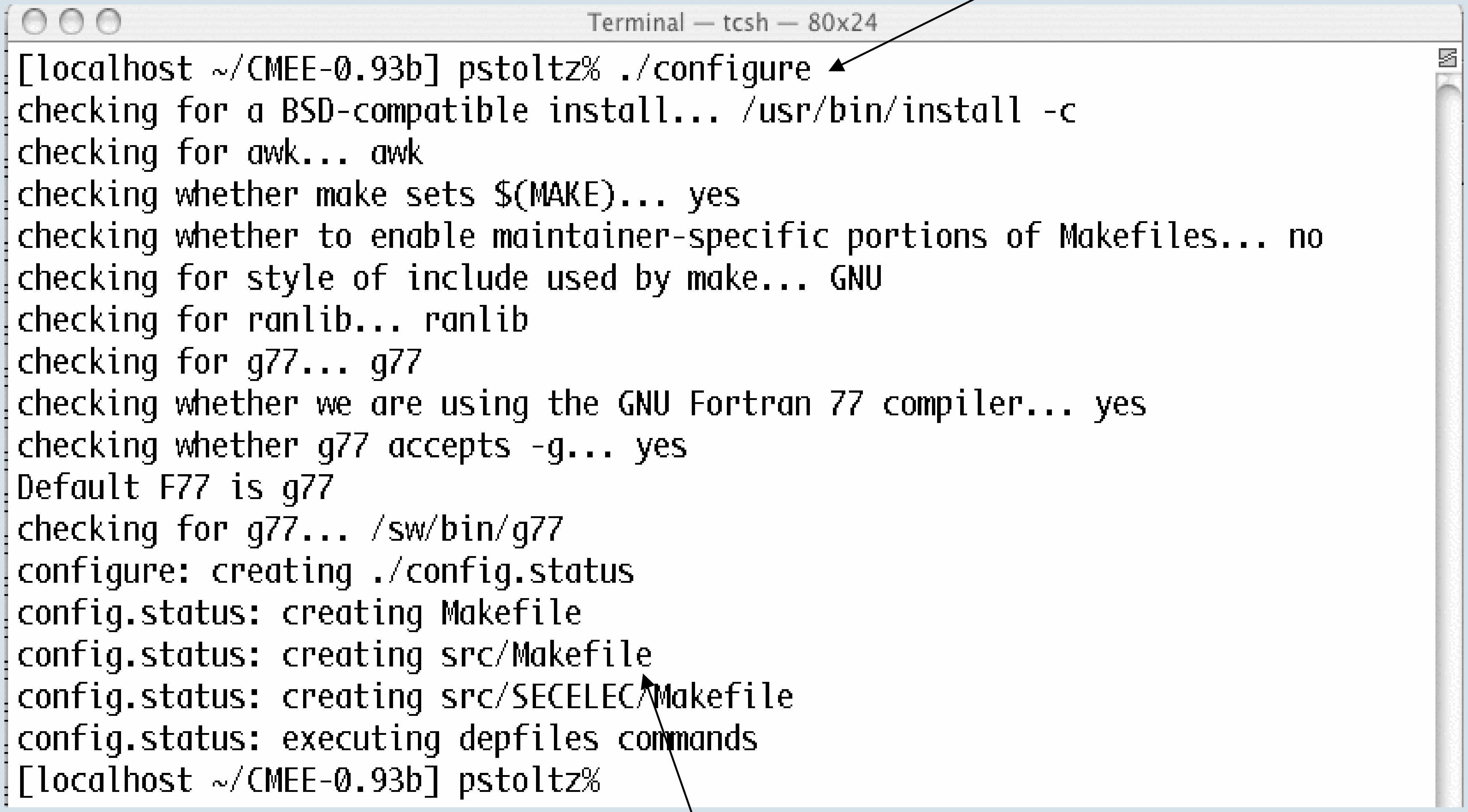
CMEE Library is certified to work with Mission Research Corporation's LSP plasma simulation package as well as OOPIC Pro and VORPAL from Tech-X Corporation.

Download CMEE Library

Tech-X website has link to tarball

## Building library is automated for different platforms

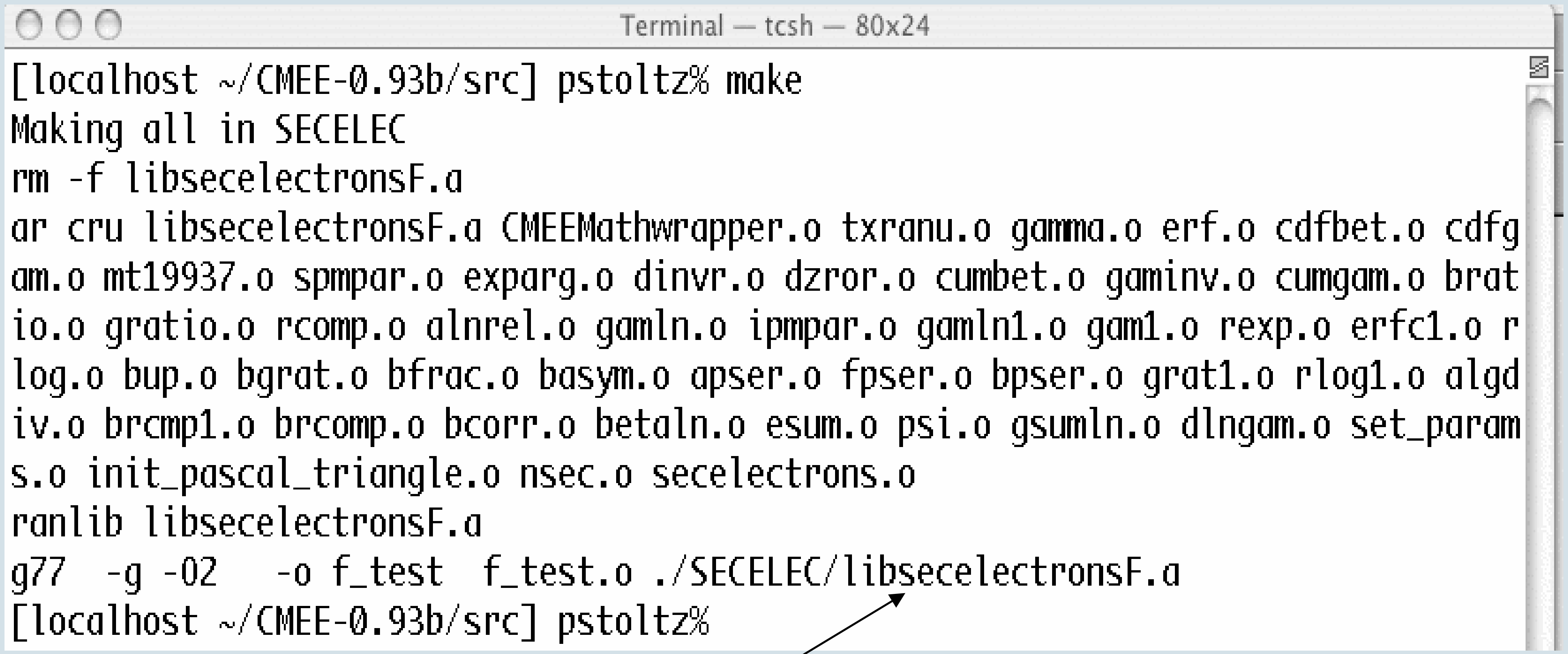
You type 'configure'



```
[localhost ~/CMEE-0.93b] pstoltz% ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking for awk... awk
checking whether make sets $(MAKE)... yes
checking whether to enable maintainer-specific portions of Makefiles... no
checking for style of include used by make... GNU
checking for ranlib... ranlib
checking for g77... g77
checking whether we are using the GNU Fortran 77 compiler... yes
checking whether g77 accepts -g... yes
Default F77 is g77
checking for g77... /sw/bin/g77
configure: creating ./config.status
config.status: creating Makefile
config.status: creating src/Makefile
config.status: creating src/SECELEC/Makefile
config.status: executing depfiles commands
[localhost ~/CMEE-0.93b] pstoltz%
```

Makefiles created for you!

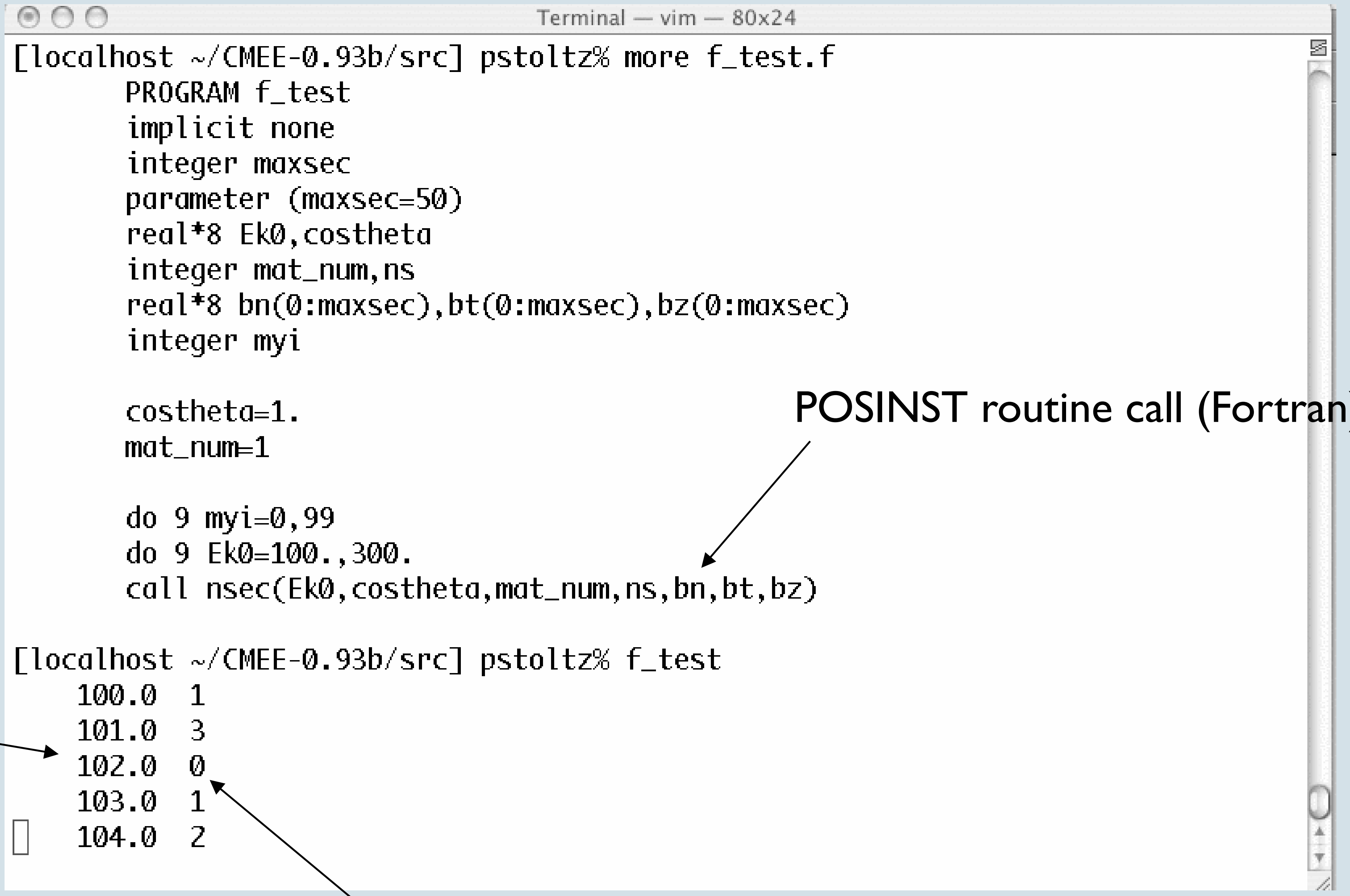
## Building library is automated with GNU tools


Terminal — tcsh — 80x24

```
[localhost ~/CMEE-0.93b/src] pstoltz% make
Making all in SECELEC
rm -f libsecelectronsF.a
ar cru libsecelectronsF.a CMEEMathwrapper.o txranu.o gamma.o erf.o cdfbet.o cdfg
am.o mt19937.o spmpar.o exparg.o dinvr.o dzror.o cumbet.o gaminv.o cumgam.o brat
io.o gratio.o rcomp.o alnrel.o gamln.o ipmpar.o gamln1.o gam1.o rexp.o erfc1.o r
log.o bup.o bgrat.o bfrac.o basym.o apser.o fpser.o bpser.o grat1.o rlog1.o algd
iv.o brcmp1.o brcomp.o bcorr.o betaln.o esum.o psi.o gsumln.o dlngam.o set_param
s.o init_pascal_triangle.o nsec.o secelectrons.o
ranlib libsecelectronsF.a
g77  -g -O2   -o f_test  f_test.o ./SECELEC/libsecelectronsF.a
[localhost ~/CMEE-0.93b/src] pstoltz%
```

Users now link their code to this library

## SEY routines come with Fortran and C bindings



```
[localhost ~/CMEE-0.93b/src] pstoltz% more f_test.f
       PROGRAM f_test
       implicit none
       integer maxsec
       parameter (maxsec=50)
       real*8 Ek0,costheta
       integer mat_num,ns
       real*8 bn(0:maxsec),bt(0:maxsec),bz(0:maxsec)
       integer myi

       costheta=1.
       mat_num=1

       do 9 myi=0,99
       do 9 Ek0=100.,300.
       call nsec(Ek0,costheta,mat_num,ns,bn,bt,bz)

[localhost ~/CMEE-0.93b/src] pstoltz% f_test
    100.0  1
    101.0  3
    102.0  0
    103.0  1
    104.0  2
```

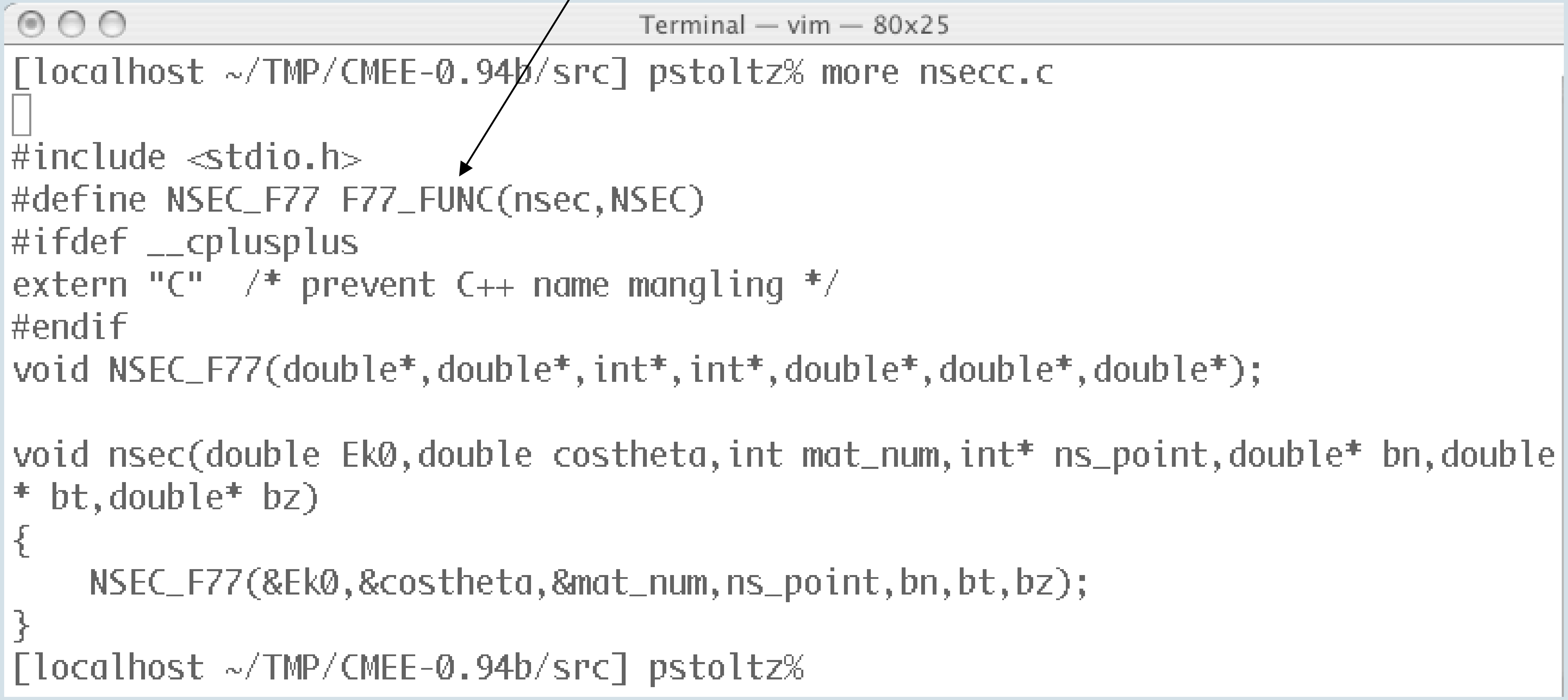
POSINST routine call (Fortran)

Energy (ev)

# of secondaries

# *Automake* provides automated C/Fortran conversions

*Automake* uses C preprocessor variable F77_FUNC to get Fortran routine name



```
[localhost ~/TMP/CMEE-0.94b/src] pstoltz% more nsecc.c

#include <stdio.h>
#define NSEC_F77 F77_FUNC(nsec,NSEC)
#ifdef __cplusplus
extern "C"  /* prevent C++ name mangling */
#endif
void NSEC_F77(double*,double*,int*,int*,double*,double*,double*);

void nsec(double Ek0,double costheta,int mat_num,int* ns_point,double* bn,double
* bt,double* bz)
{
    NSEC_F77(&Ek0,&costheta,&mat_num,ns_point,bn,bt,bz);
}
[localhost ~/TMP/CMEE-0.94b/src] pstoltz%
```

## CMEE requires dealing with common blocks and non-system library calls

- If a routine uses a common block to pass data, other codes need to know how to access that common data
- For POSINST, we eliminated common blocks and passed all data through subroutine arguments
- If a routine calls non-system libraries (e.g. IMSL math), other codes need to resolve these calls somehow
- For POSINST, we replaced IMSL calls with NETLIB calls

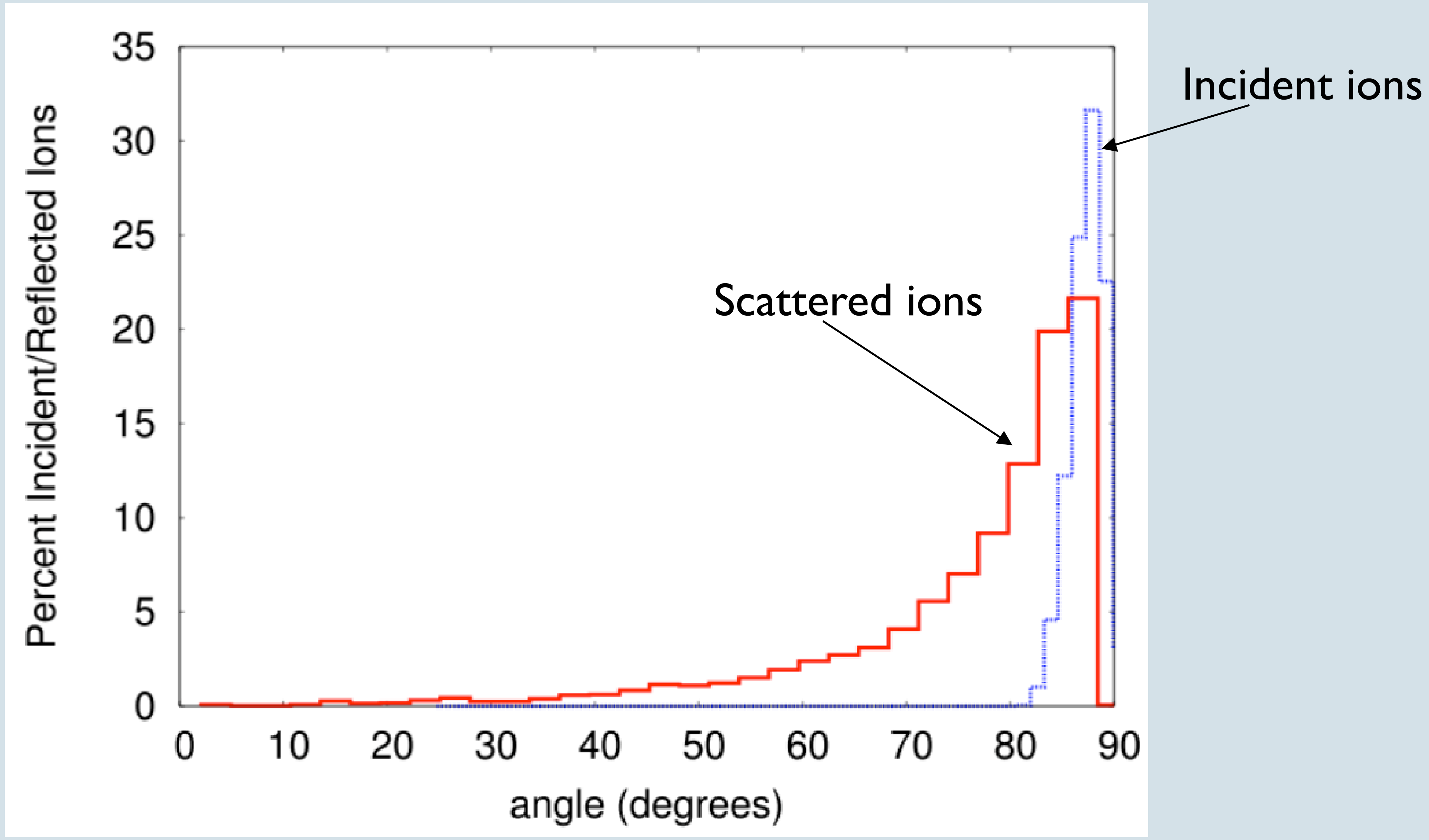
## CMEE also requires dealing with attorneys

- To redistribute routines developed outside Tech-X from the Tech-X website requires a licensing agreement
- For POSINST, there is a non-commercial licensing agreement in place

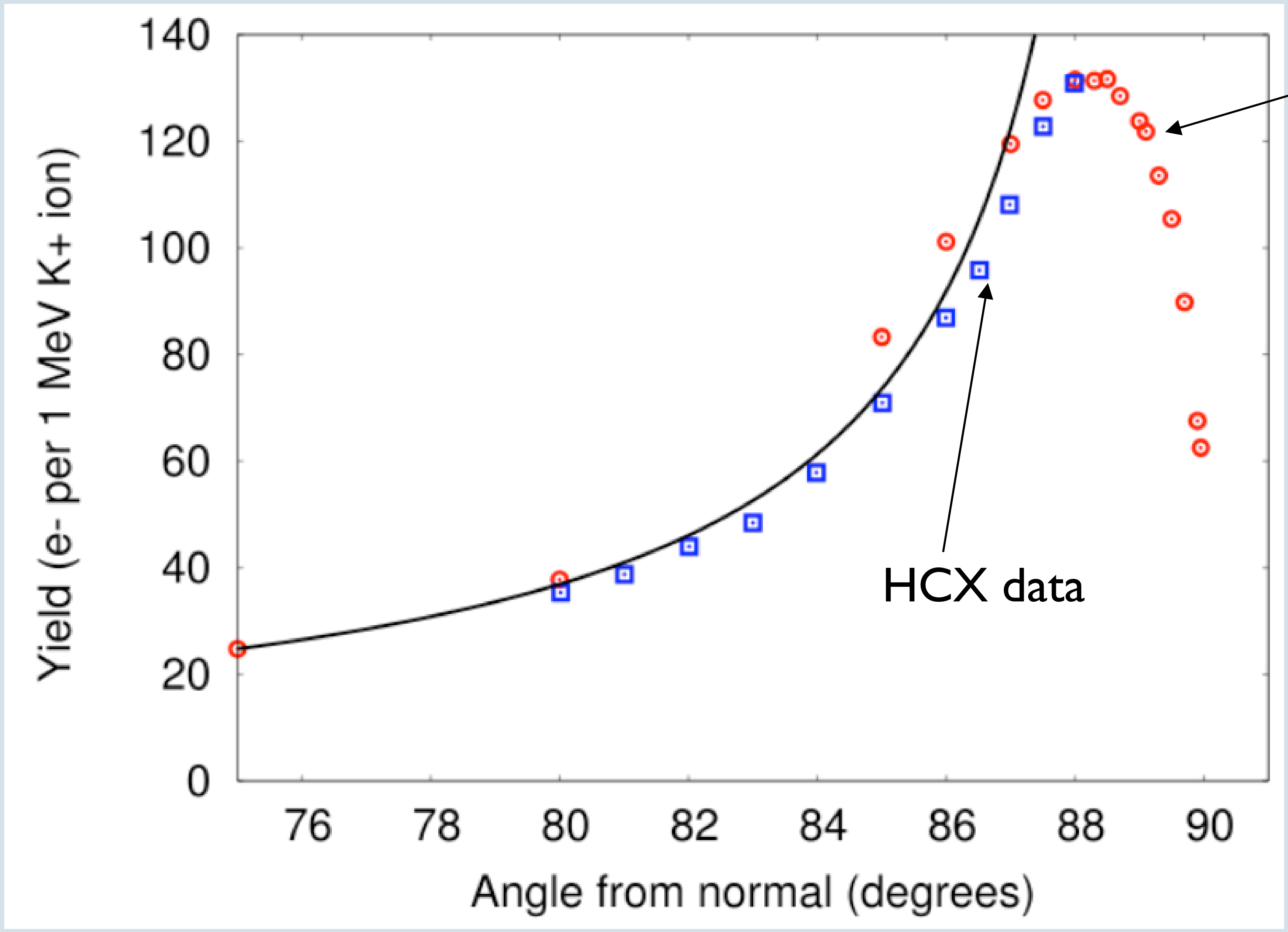
## Recently, we've worked on two aspects of ion-surface interactions: scattering and electron yield

- For scattering, we wrote Python routines to access tables of SRIM data
- For ion-induced electron yield, we used an open-source electronic energy loss code called CRANGE

## Ion scattering is important to electron effects in HIF applications



## Ion scattering is one way to explain grazing electron yields

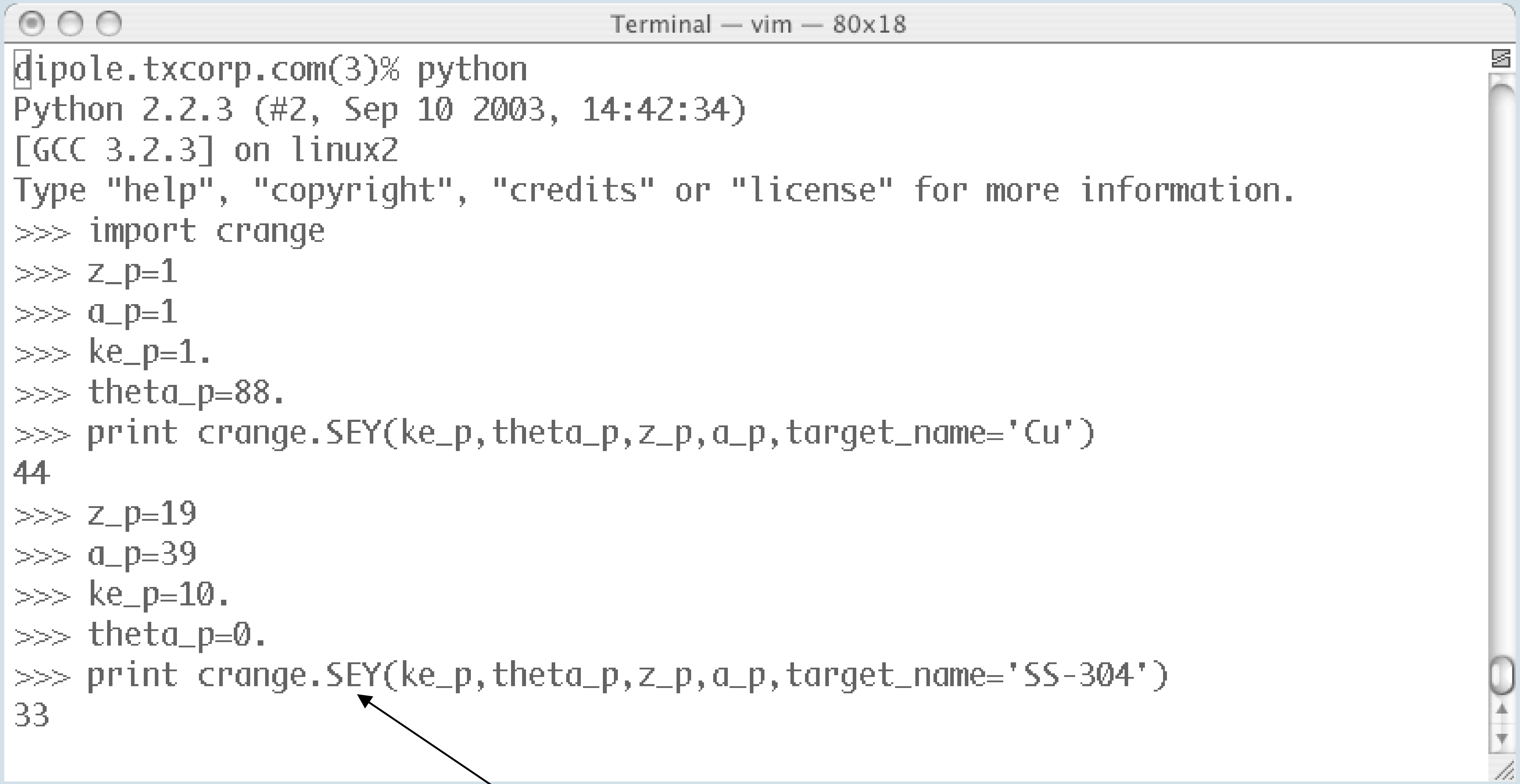


Simulations with scattering

## We’ve also begun porting an ion-material interaction code for CMEE

- SRIM is a standard, but it runs only on Windows and you can’t call it from other codes
- A recently-developed code, CRANGE, from the astrophysics community has similar capability
- We’ve made the steps (removing common blocks and non-system library calls) to add it to CMEE
- It works for high energies (greater than 1.0 MeV/amu), but not yet for lower energies

# CRANGE provides dE/dx, range and an approximate ion-induced electron yield



```
dipole.txcorp.com(3)% python
Python 2.2.3 (#2, Sep 10 2003, 14:42:34)
[GCC 3.2.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import crange
>>> z_p=1
>>> a_p=1
>>> ke_p=1.
>>> theta_p=88.
>>> print crange.SEY(ke_p,theta_p,z_p,a_p,target_name='Cu')
44
>>> z_p=19
>>> a_p=39
>>> ke_p=10.
>>> theta_p=0.
>>> print crange.SEY(ke_p,theta_p,z_p,a_p,target_name='SS-304')
33
```

SEY is based on model by Rothard, *et. al.*, and is proportional to dE/dx

# I'd like feedback where to go next...

- Photoelectrons (from POSINST?)
- Models of SEY for NEG materials (TiN, TiZrV)? Conditioning?
- Models of heavy ion stripping with cross sections ~ $E^{-1/2}$ (Rumolo)?
- Models of ion scattering?
- Simple models of desorption/ionization (could just be constant)?
- Other SEY models (CSEC?  Others?)