EGSnrc

Iwan Kawrakow
Ionizing Radiation Standards
National Research Council of Canada

NPL Workshop
March 17 – 18, 2004

Abstract

This lectures provides an overview of the EGSnrc Monte Carlo system discussing its capabilities, important features, recent developments and future plans

General overview

- EGSnrc is a package for the Monte Carlo (MC) simulation of coupled electron-photon transport
- Dynamic energy range is 1 keV – $\sim$ 10 GeV
- All elements with Z $= 1 \cdots 100$, arbitrary mixtures using the independent atom approximation
- Derived from the popular EGS4 package
- Many improvements in electron transport physics and low energy photon cross sections compared to EGS4
- Web page:
  http://www.irs.inms.nrc.ca/inms/irs/EGSnrc/EGSnrc.html

History

- 1996–1997, Kawrakow and Bielajew
  - New exact multiple scattering theory
  - New accurate electron-step algorithm
- 1997–1999, Kawrakow
  - New electron transport routine
  - Improved energy loss evaluation
  - Correct fictitious cross section implementation
  - Extend multiple scattering theory to elastic scattering with spin effects taken into account
  - Compton scattering with binding and Doppler broadening
  - Improved photo-absorption
  - Atomic relaxations
  - All EGS4 extensions included by default
  - Many bug fixes, optimizations and code clean-up

History (cont'd)

- 2000 Kawrakow and Rogers
  - New 300 pages manual with detailed description of physics (PIRS–701)
  - EGSnrc V1 released in May 2000.
- 2003 Kawrakow, Mainegra-Hing and Rogers
  - Completely new run-time environment, works on Unix/Linux, Windows NT/2000/XP and Mac OSX
  - Using different compiler/OS combinations within the same installation
  - Cleaned-up code to not assume local variables to be static
  - Graphical user interfaces
  - Graphical installer
  - C/C++ interface
  - New parallel processing
  - EGSnrc V4 (a.k.a. EGSnrcMP) released December 2003

The EGS approach

- EGSnrc provides subroutines and functions for cross section data initialization, sampling of the various processes and electron and photon transport routines.
- A complete EGSnrc applications requires a "user code" that must provide a main function, a scoring routine and 2 geometry related functions.
- User codes can be written in Mortran, Fortran, C or C++.
- EGSnrc comes with a series of NRC user codes for calculating quantities of interest for ion chamber dosimetry, detector response and energy deposition calculations in XYZ and RZ geometries.
- The BEAMnrc package (distributed separately) can be used to simulate the treatment head of medical linear accelerators, ^{60}Co and X-ray units.

The EGS approach

- Is scary for the novice user (unless there is already a user code available)
- Is very flexible and powerful:
  - Taylor-made user codes to calculate exactly what is needed
  - Custom variance reduction techniques can be implemented via the user scoring routine `ausgab` or the various macros
- Has played a major role in the wide adoption of EGS

Photon interactions and cross sections

- Incoherent (Compton) scattering
- Coherent (Rayleigh) scattering
- Pair/triplet production
- Photo-absorption

Compton scattering

Theoretical total and differential cross sections: the user has the choice between

- Scattering with free electrons at rest using the Klein-Nishina formula. This is the same as in EGS4 but the sampling algorithm has been optimized
- Binding effects and Doppler broadening in the relativistic impulse approximation (default).
  - The approach used is similar to PENELOPE's but a few approximations have been removed and the sampling algorithm is more efficient.
  - The necessary Compton profiles are taken from [Biggs *et al*, Atomic Data and Nucl. Data Tables **16** (1975) 201.]
  - The relaxation cascade of inner shell vacancies created in Compton scattering events is taken into account

Rayleigh scattering

Same as in EGS4:

- Elements: empirical total cross sections from Storm & Israel
- Elements: differential cross sections based on atomic form factors from [Hubbell and Øverbø, J. Phys. Chem. Ref. Data (1979) 69.]
- Mixtures: independent atom approximation (known to be not very accurate!)

Pair/triplet production

- Empirical total cross sections from Storm & Israel
- Extreme relativistic first Born approximation (Coulomb corrected above 50 MeV) for the cross sections differential in energy.
- Triplet production is not explicitely modeled but taken into account by adding the triplet total cross section to the pair total cross section
- The above 3 are the same as in EGS4 except that the pair energy sampling algorithm is more efficient
- The angular distribution of electrons and positrons is selected from a modified version of Eq. 3D-2003 of [Motz *et al*, Rev. Mod. Phys. **41** (1969) 581.] or its leading term.

Photo-absorption

- Empirical total cross sections from Storm & Israel
- For mixtures the interacting element is explicitely sampled
- The interacting shell is explicitely sampled using photo-absorption branching ratios
- The binding energy of the interacting shell is subtracted from the photo-electron energy
- The relaxation cascade of the shell vacancy is taken into account
- The angular distribution of the photo-electrons is picked from the Sauter distribution

Atomic relaxations

- In EGS4 the creation of characteristic X-rays is associated with photo-absorption.
- In EGSnrc the relaxation cascade of inner shell vacancies is an independent process that is initiated each time a vacancy is created (currently in photo-absorption and bound Compton scattering).
- All shells with binding energies above 1 keV
- All radiative and non-radiative transitions to/from K-, LI-, LII- and LIII-shells
- All radiative and non-radiative transitions to/from "average" M- and N-shells.

Note on photon cross sections

- The default total cross sections for pair/triplet, photo-absorption and coherent scattering are from Storm & Israel $\Rightarrow$ outdated
- An alternative data set, courtesy of the group at McGill U in Montreal, is provided with the system. This data set is based on XCOM and can be used by providing a command line argument to PEGS4
- The user may prepare their own total cross section data sets and use them in PEGS4 with the appropriate command line argument

```
pegs4.exe -i ifile [-o ofile] [-a] [-d density_file]
          [-x x_section_data]
```


Electron/positron interactions and cross sections

- Bremsstrahlung
- Inelastic collisions with atomic electrons
- Elastic collisions with nuclei and atomic electrons
- Positron annihilation

Bremsstrahlung

- For the cross section differential in photon energy, the user has the choice between
  - The NIST bremsstrahlung cross section data base (basis for ICRU radiative stopping powers)
  - Extreme relativistic first Born approximation (Coulomb corrected above 50 MeV) with an empirical correction so that ICRU recommended radiative stopping powers are reproduced.
- Total bremsstrahlung cross sections for production of photons with energy greater than a user specified threshold are calculated from the selected differential cross section using numerical integration
- Restricted radiative stopping power for sub-threshold bremsstrahlung
- Angular distribution of bremsstrahlung photons from Eq. (2BS) of Koch & Motz or its leading term

Inelastic collisions

- Møller (e^-) or Bhabha (e^+) cross sections for collisions that result in the creation of δ-particles with energies greater than a user specified threshold
- Continuous-slowing-down approximation using the restricted collision stopping power from the Bethe-Bloch theory for sub-threshold processes
- Density effect corrections from ICRU–37 or the empirical Sternheimer formula

Positron annihilation

- Two-photon in-flight annihilation from the first Born approximation neglecting binding
- Two-photon annihilation at rest
- Single- and n-photon ($n \geq 3$) annihilation ignored because cross sections much smaller

Elastic collisions

User has the choice between

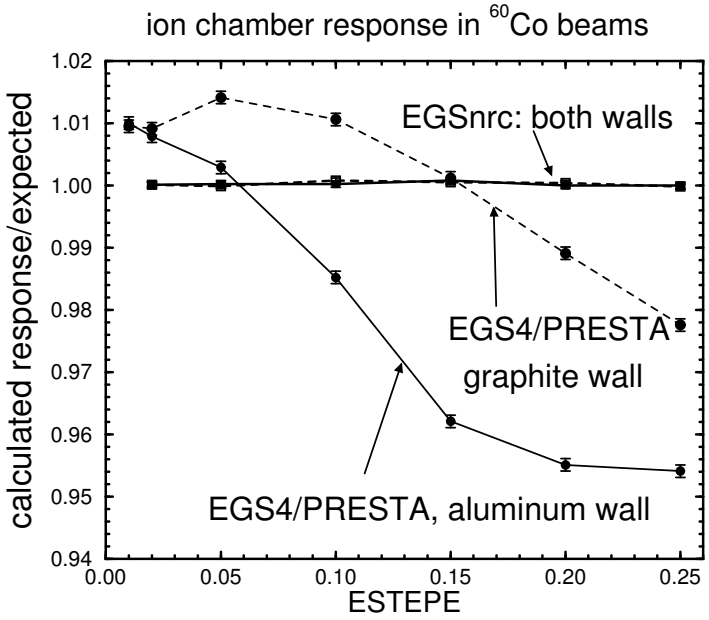
- The screened Rutherford cross section with a screening angle from the single scattering theory of Moliére. This is the single elastic scattering cross section effectively used in EGS4 via the multiple scattering theory of Moliére.
- The screened Rutherford cross section times the Mott spin correction factor (which is different for electrons and positrons) with a screening angle selected so that the first elastic scattering moment from PWA cross sections is reproduced. This is the similar to the cross sections used in ETRAN/ITS/MCNP above 256 keV.

Condensed history aspects

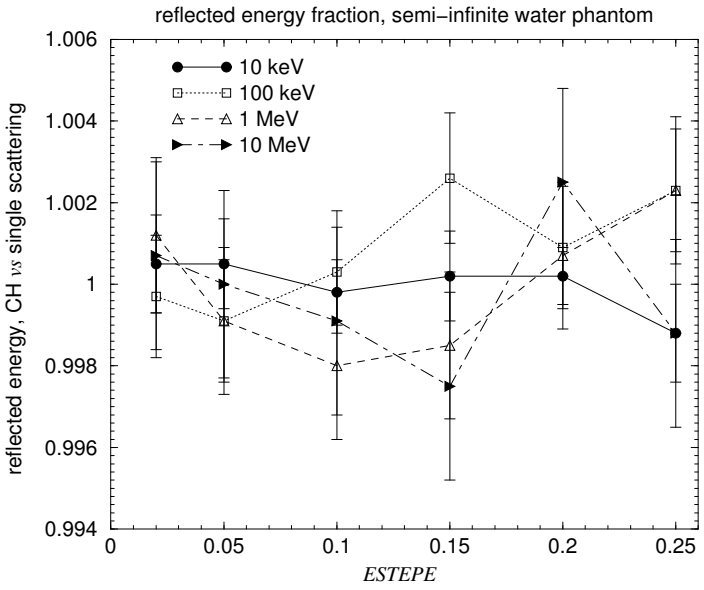
- Multiple elastic scattering: exact theory valid for arbitrary step sizes, both for the screened Rutherford cross section and the cross sections with spin effects taken into account
- Electron-step algorithm: most accurate algorithm known (truncation error is $O(\Delta s^4)$)
- Evaluation of energy dependent quantities: accurate up to $O(\Delta E^4)$
- Fictitious cross section method: unlike EGS4, correct implementation that uses cross sections per unit energy loss.

$\Rightarrow$ Step-size independent and artifact free simulation at the sub 0.1% level.

Theoretical benchmark: Fano cavity

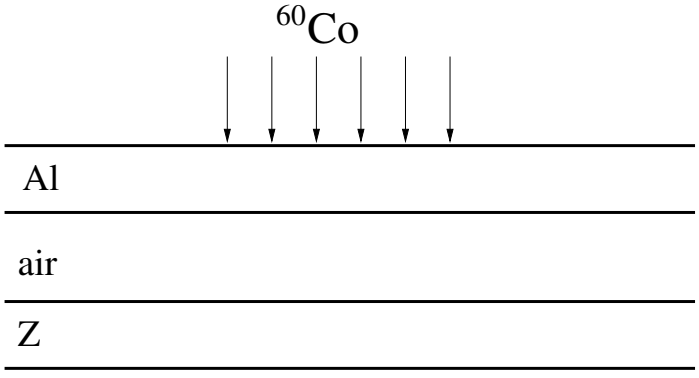


Theoretical benchmark: backscattering



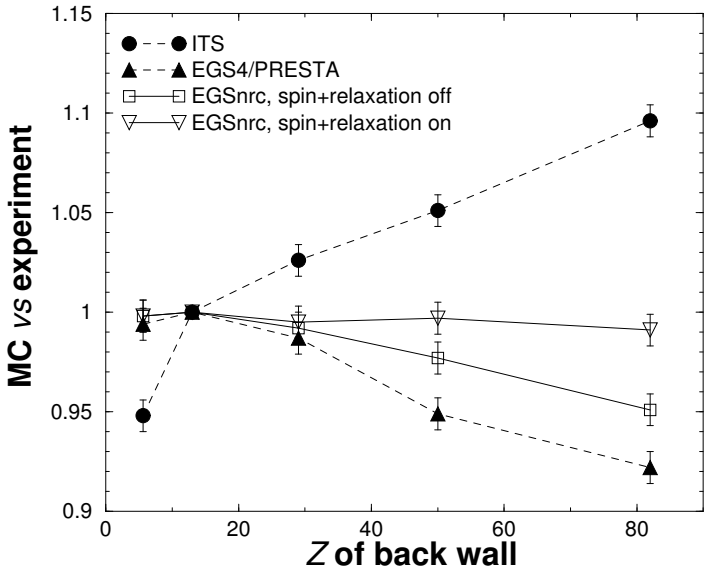
Example 1: ion chamber response

Experiment by B. Nilsson *et al*:

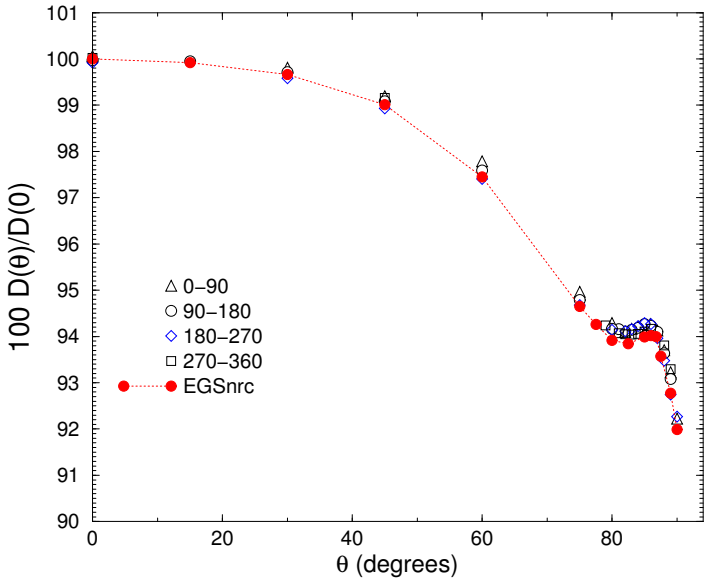


Measured chamber response as a function of the atomic number of the back wall Z

Example 1: ion chamber response

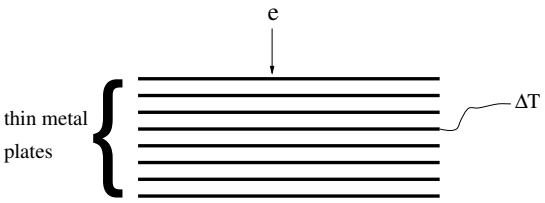


Example 2: pancake chamber



Example 3: depth dose curves

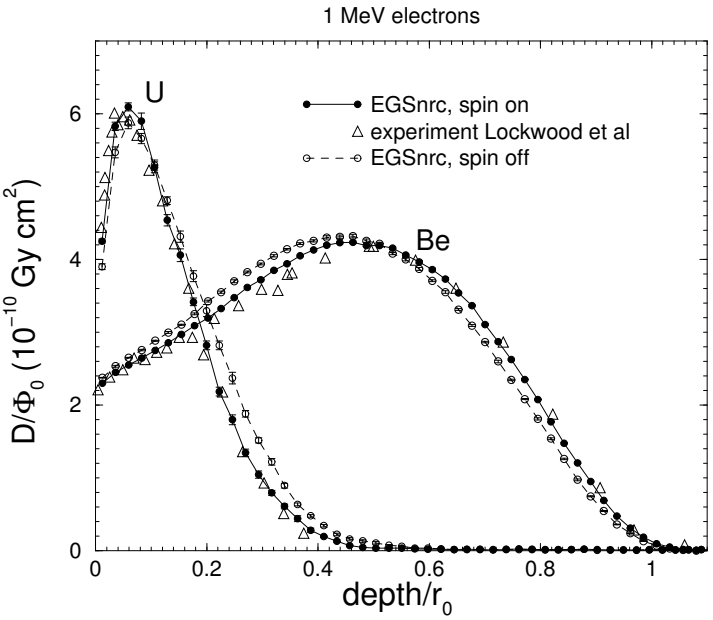
Experiment by Lockwood *et al* at Sandia:



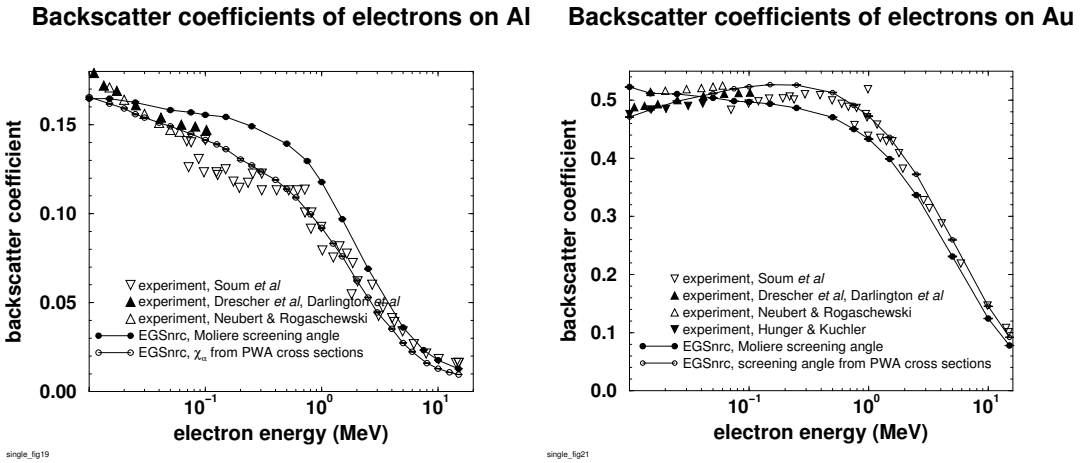
- Incident energy known very precisely
- Incident fluence measured with Faraday cage
- Temperature rise ⇒ energy deposition

⇒ Absolute measurement of dose per incident fluence

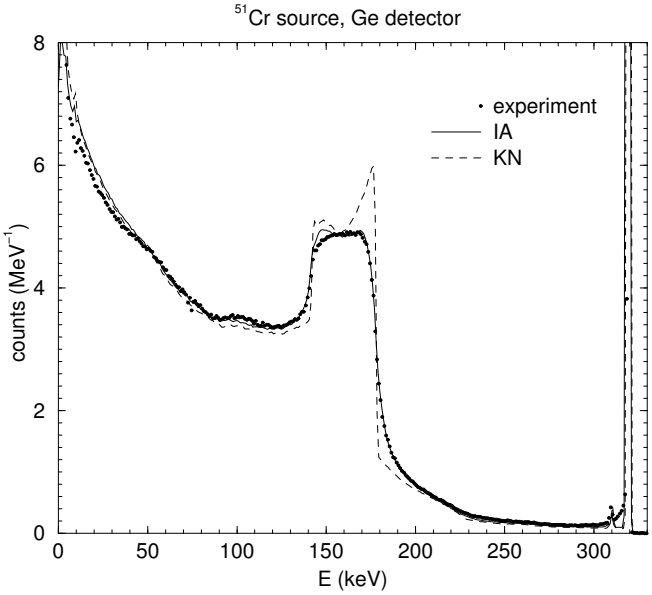
Example 3: depth dose curves



Example 4: backscattering coefficients



Example 5: Ge-detector response function



Parallel processing

- MC simulations frequently need very long simulation time
- MC simulations are easily parallelizable
- There have been parallel versions of EGS based on PVM, but not widely adopted and used
- The old EGSnrc parallel processing approach implemented via a script that
  - Prepares N separate input files from a "master" input file
  - Sends N independent jobs to a queuing system such as NQS or PBS using the N input files.
  - When all jobs are finished, the user must perform a separate short run that combines the results

Old EGSnrc parallel processing

Disadvantages:

- Only Unix/Linux (because of script)
- Not flexible: script must "know" syntax of input file for each user code so that number of histories and initial random number seed can be changed.
- Workload distributed evenly between CPU's even if CPU's have different execution speed $\Rightarrow$ total execution time is determined by the time needed on the slowest (or busiest CPU).
- For relatively short runs, time needed by the queuing system to submit a job becomes significant

Old EGSnrc parallel processing: total time

t_1 - time needed on a single CPU
Δt - time needed by the queuing system to submit a job ($\sim$ 1 second for PBS but $\sim$ 10 seconds for NQS)
N - number of jobs
Total execution time t:

$$t = \frac{t_1}{N} + (N-1)\Delta t$$

The above has a minimum for

$$N = \sqrt{\frac{t_1}{\Delta t}}$$

which is

$$t = 2\sqrt{t_1 \Delta t}\left(1 - \sqrt{\frac{\Delta t}{4t_1}}\right)$$

Example: $t_1 = 20$ min, $\Delta t = 10$ sec. $\Rightarrow N = 11, t = 209.1$ sec.

New EGSnrc parallel processing implementation

Run is controlled via a "job control file" (JCF) placed on a network file system accessible by all jobs:

- Each job "knows" its job number and the number of parallel jobs running via command line arguments. Command line arguments are parsed in the subroutine `egs_init` provided by EGSnrc
- Each job adjusts the RNG initial seed based on its job number so that jobs use independent random number sequences.
- The JCF is created by the first job and contains, among other things, number oh histories remaining ($n_{\rm left}$) and number of parallel jobs currently running ($n_{\rm job}$).
- The entire simulation is divided into calculation "chunks". Each chunk starts Δn particles with $\Delta n \ll n$, where n is the total number of histories requested in the input file.

New EGSnrc parallel processing implementation

- After job one has created the JCF it contains $n_{\rm left} = n - \Delta n,\ n_{\rm job} = 1$.
- Each sub-sequent job that starts running takes a "chunk" and increases the number of running jobs so that $n_{\rm left} \rightarrow n_{\rm left} - \Delta n,\ n_{\rm job} \rightarrow n_{\rm job} + 1$
- When any job finishes a calculation chunk, it reads the JCF and
  - If there are histories remaining to be done ($n_{\rm left} > 0$), the job takes $\mathrm{Min}(n_{\rm left}, \Delta n)$ histories and reduces $n_{\rm left}$ by this amount
  - If there are no histories left, $n_{\rm job} \rightarrow n_{\rm job} - 1$, analyze and output results. If $n_{\rm job} > 0$ exit, else read and combine results from all other jobs.
  - Job submission automation is provided for Unix-like systems via a script but it should not be difficult to do the same for Windows

Advantages

- More flexible because the submission script does not need to "know" anything about the syntax of the input file
- Workload is automatically distributed according to CPU speed
- Delay between the first and last job finishing is not greater than the time needed for a calculation chunk on the slowest machine
- On-the-fly increase or decrease of number of histories is possible
- Much faster for relatively short simulations

New EGSnrc parallel processing implementation: total time

First job runs for t seconds, second job for $t - \Delta t$, third for $t - 2\Delta t$, etc. Each job processes n/t_1 particles per time unit $\Rightarrow$ first job processes tn/t_1, second $(t - \Delta t)n/t_1$, etc. The sum must be $n \Rightarrow$

$$t = \frac{t_1}{N} + \frac{N-1}{2}\Delta t$$

This is shorter by

$$\frac{N-1}{2}\Delta t$$

for the same number of equally fast CPU's compared to the old approach. Optimum number of CPU's and corresponding shortest possible time is

$$N = \sqrt{\frac{2t_1}{\Delta t}}\,, \quad t = \sqrt{2t_1\Delta t}\left(1 - \sqrt{\frac{\Delta t}{8t_1}}\right)$$

i.e. $\sim \sqrt{2}$ times shorter.

Example: $N = 15,\ t = 150$ sec. instead of $N = 11,\ t = 209.1$ sec.

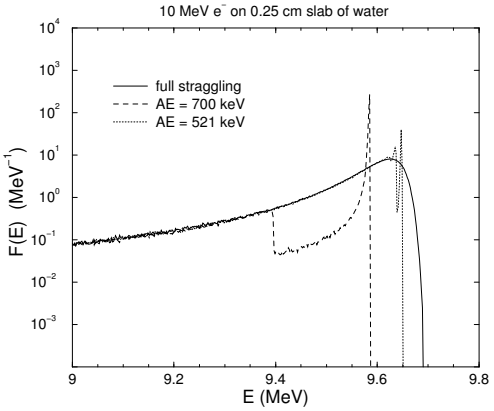
Disadvantages

- JCF must be locked during access. JCF locking is implemented in C $\Rightarrow$ requires C compiler in addition to the Fortran compiler already needed by the rest of EGSnrc.
- Parallel run results are not guaranteed to be reproducible because each separate random number sequence may be used for a different number of histories, depending on the speed of the CPU executing a particular portion of the parallel run.
- Partitioning of phase-space files used as a source in a parallel run is much more complex.

Future development plans

- Energy loss straggling for sub-threshold processes
- Electron impact ionization (EII)
- Extend lower energy limit of applicability
- Provide the user with an easy way to use their own form factors for coherent scattering
- Remove need for PEGS4 by preparing cross section data on-the-fly
- General purpose geometry package

Energy loss straggling



- Use of CSDA not a necessity for class II MC codes
- Implementation of straggling for sub-threshold processes will improve efficiency for certain type of calculations
- Straggling is well described by the Vavilov distribution plus a Gaussian for "soft" collisions.
- The Vavilov distribution converges to the Landau distribution with an appropriate cut-off for $\kappa \to 0$ and to a Gaussian for $\kappa \gg 1$.
- The width of the soft collisions Gaussian depends on the details of the inelastic scattering cross section

$\Rightarrow$ Implementation postponed until EII and realistic low energy inelastic scattering cross sections implemented.

Electron Impact Ionization

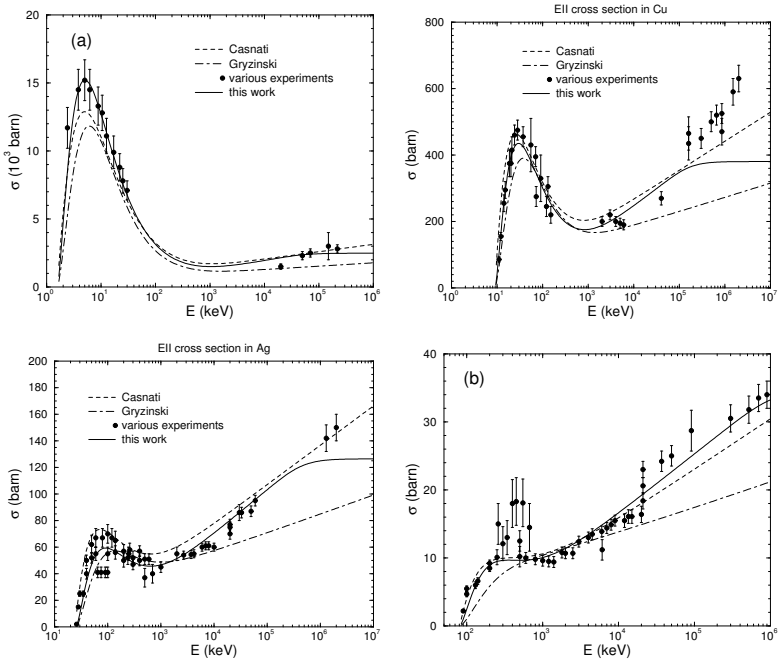
- Inelastic e^-/e^+ collisions with inner-shell electrons can create vacancies that lead to the production of characteristic X-rays, Auger electrons, etc.
- Process is missing in EGSnrc $\Rightarrow$ accurate calculation of X-ray spectra is not possible
- KEK extension for EGS4 provides EII, but
  - Choice of 6 theoretical and empirical K-shell EII cross sections
  - No L- or lower shell EII cross sections
  - Theoretical cross sections don't agree very well with experiment
  - Some empirical cross sections are merely a fit to available data $\Rightarrow$ no guarantee they will work for elements where no data is available
  - No differential cross sections $\Rightarrow$ use Møller/Bhabha and subtract binding energy from δ-particle $\Rightarrow$ inconsistent stopping power
  - At high energies total Møller/Bhabha x-sections converge to a constant whereas EII cross sections known to increase logarithmically $\Rightarrow$ KEK approach breaks down at sufficiently high energies

EII in EGSnrc

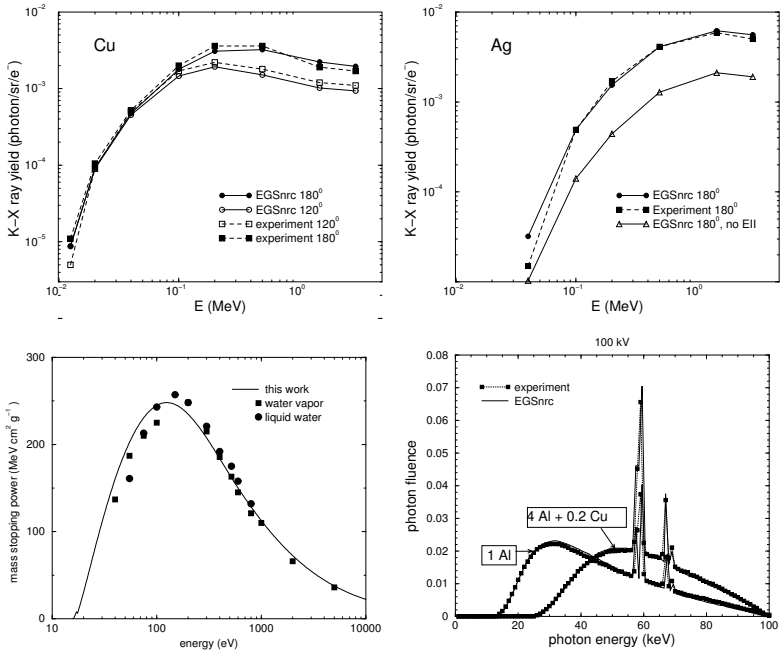
- Theoretical inelastic scattering cross sections with empirical corrections based on a GOS approach
- Provides EII cross sections for all shells, not just the K-shell
- Provides differential inelastic scattering cross sections
- Result in a stopping power that is in perfect agreement with the Bethe-Bloch formula where the Bethe-Bloch theory is applicable
- Permits extension to lower energies
- "Soft" collision cross sections are related to shell-wise photo-absorption cross sections $\Rightarrow$ large amount of data necessary at run time $\Rightarrow$ use linear arrays for storing data and "pointers" for the different media to avoid huge 2D- or 3D- arrays $\Rightarrow$ messy, bug-prone implementation in Mortran/Fortran

$\Rightarrow$ Final implementation postponed until mixing Mortran with C/C++ is easily done within the EGSnrc environment

Total K-shell EII cross sections



K X-ray yields, stopping power, X-ray spectra



Future development plans

- Energy loss straggling for sub-threshold processes
- Electron impact ionization (EII)
- Extend lower energy limit of applicability
- Provide the user with an easy way to use their own form factors for coherent scattering
- Remove need for PEGS4 by preparing cross section data on-the-fly

- General purpose geometry package – use C or C++ ? C++ is more powerful, but
  - There are some complications when linking C++ and Fortran code together
  - Yet another compiler necessary to take full advantage of EGSnrc

Note on compilers and compilation options

Experiment: typical ion chamber response simulation using CAVRZnrc with different compilers/options on an Athlon XP 2400+ CPU running Linux

Compiler	flags	time (s)
GCC 2.95.3	-O2 -fno-automatic -finit-local-zero	720
GCC 2.95.3	-O3 -ffast-math -march=i686	603
GCC 3.3.3	-O3 -ffast-math -march=athlon-xp -mfpmath=sse -m3dnow -mmmx	525
GCC 3.4.0	same as above	405
GCC 3.4.0	same as above + profiling	370
PGI	-fast -Mrecursive	485
Lahey 6.2	-O -tpp	650
Intel 8.0	-O3 -auto -xK -ipo	375
Intel 8.0	same as above + profiling	344

For other codes speed advantage varies between 50% and 100% with GCC 3.4 $\approx$ Intel 8.0