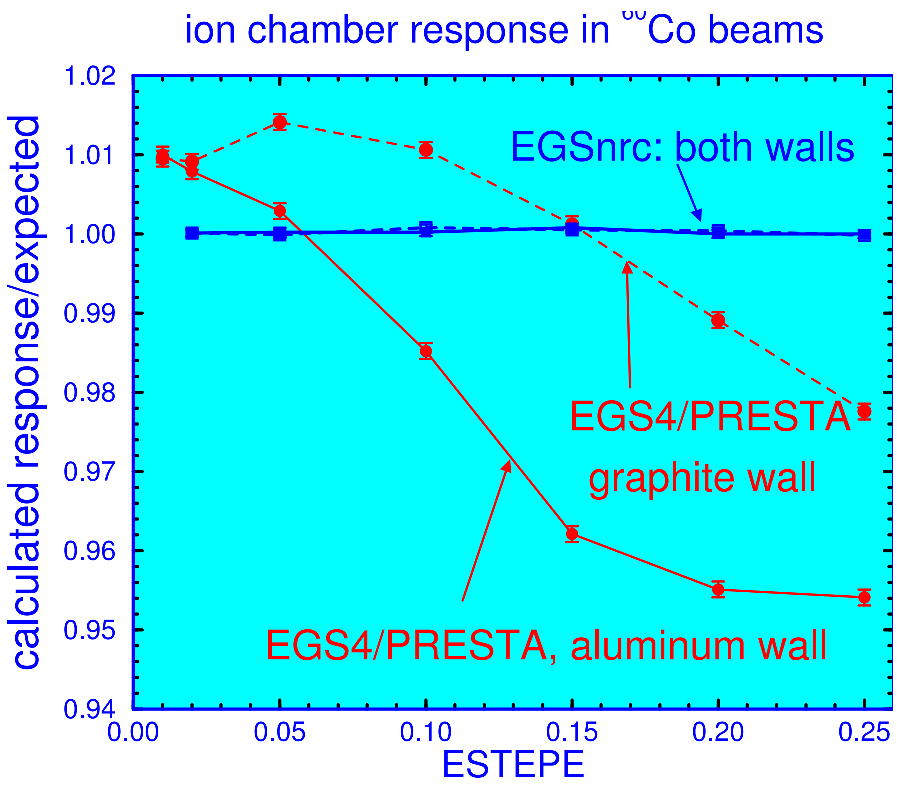
The EGSnrc Code System:
Monte Carlo Simulation of Electron and Photon Transport

I. Kawrakow
E. Mainegra-Hing, D.W.O. Rogers, F. Tessier and B.R.B. Walters
Ionizing Radiation Standards,
National Research Council Canada,
Ottawa, Canada

January 13, 2025

NRCC Report PIRS-701



Preface

Sixth printing: May 2011
A long time has passed since the last update to this document. Although no fundamental changes have been made to the system, there have been a number of important additions and improvements to the system since 2009. Numerous bugs have been corrected and a new user code for free-air chamber (FAC) correction calculations was added to the distribution.

- Use of arbitrary electron impact ionization (EII) cross section compilations. Added a data base of EII cross sections based on the DWBA/PWBA theory by Bote and Salvat.
- For backward compatibility with previous calculations, users can now request the use of PEGS4 photon data.
- Inclusion of C++ `egs_fac` application with example. This user-code implements a self-consistent algorithm for the fast calculation of FAC correction factors.
- EII cross sections printout for materials in the simulation if user requests output of the photon cross-sections.
- Updated several GUI's to include most of the latest additions and corrected several bugs.

Fifth printing: July 2009
There has been a huge number of changes in the system since the last printing in November 2003 that have not been properly added to the documentation. This printing is an attempt to update this report to better reflect the state of EGSnrc, although it is far from complete. The development of the EGSnrc C++ class library, which includes a general purpose geometry package, and its first public release in 2005 has been a major step forward. The C++ class library is described in a separate report (PIRS–898). Major changes and additions to the EGSnrc physics include: option to simulate electron impact ionization, an improved bremsstrahlung data base that includes an exact evaluation of electron-electron bremsstrahlung in the first Born approximation, an improved differential pair production cross section tabulation based on exact PWA calculations that takes into account the asymmetry of the energy distribution at energies close to the threshold, the ability to explicitely simulate triplet interactions (*i.e.*, pair production in the electron field), the ability to take into account radiative corrections for Compton scattering in the one-loop approximation, the ability to use user-supplied atomic and molecular form factors for Rayleigh scattering, and the ability to use total photon cross sections from EPDL97, XCOM, or any other user-supplied tabulation in addition to the default Storm & Israel tabulations. New options added for Compton scattering called `simple` and `norej`. When using `simple` binding is taken into account via an incoherent scattering function ignoring Doppler broadening. The incoherent scattering function is still obtained from the impulse approximation. The `norej` option uses actual bound Compton cross section when initializing the photon cross sections and rejections in subroutine COMPT lead to re-sampling rather than rejecting the entire interaction. An `alias sampling` algorithm is now used to select the photon angle after a Rayleigh scattering to avoid an undersampling at large angles observed in the original EGS4

implementation. Particle track scoring object added to the `egspp` library allowing visualization of particle tracks with the C++ geometry viewer `egs_view`. See example input file tracks1.egsinp for the tutor7pp C++ user-code.

Finally, to better reflect their contributions to the development and maintenence of the system, Ernesto Mainegra-Hing, Blake Walters and Frederic Tessier have been added as co-authors.

Fourth printing: November 2003
Added references to EGSnrcMP and Report PIRS-877. A few minor changes related to the major change in the operating system to make it Windows compliant. There is no associated change in the physics of the system. Table 7 re timing of random numbers has changed substantially.

Third printing: April 2002
Minor changes reflecting code changes.

Second printing: May 2001
The second printing contains a description of the use of `RHOF` (section 3.4.2, page 126) and `$SET-RHOF` (section 3.4.1.v, page 122). There is a brief new discussion of `combine_egsnrc`, a script for automatic analysis of the many files created by p̂process in parallel runs ((section 9.6, page 302)). There is a new section about terminating histories with WT=0.0 (section 3.8, page 143).

First printing: May 2000
In the decade and a half since the original version of EGS4 was released there have been well over 1000 papers published which cite the original SLAC-265 Report. The code itself has been improved in many different ways by a large number of people. For a detailed history of much of this up to 1994, the reader is referred to a report titled "History, overview and recent improvements of EGS4" by Bielajew et al.

In the last few years there have been significant advances in several aspects of electron transport. For example, improvements in multiple scattering theory have been developed by Kawrakow and Bielajew[1, 2, 3] which get over most of the shortcomings of the Moliere theory used in EGS4. Perhaps more important has been the development by Kawrakow and Bielajew[4] of a new electron transport algorithm, sometimes called PRESTA-II, which makes a significant advance in the science of electron transport. In addition to these advances, Kawrakow has implemented several other improvements in the electron transport algorithm of EGS which make it capable of accurately calculating ion chamber response at the 0.1% level (relative to its own cross sections)[5, 6].

EGSnrc also has implemented a variety of additional features, many of which have previously been extensively developed as additions to EGS4 by Namito, Hirayama and Ban at KEK as well[7, 8, 9, 10]. The EGSnrc approach differs from that of the KEK group, partially because once we were making fundamental changes to the code, we carried it through in a consistent manner. However, the KEK group have implemented several options which are not yet in EGSnrc (e.g. polarized photon scattering and electron impact ionization).

This report is meant to document the many changes that have occurred going from EGS4 to EGSnrc. Although this report is written by two people, the EGS system is obviously the child of many parents who have made a wide variety of contributions over the years. This goes right back to Richard Ford, then at SLAC, who was a major contributor to EGS3. Hideo Hirayama, S Ban and Yosh Namito of KEK have made innumerable contributions to EGS, especially concerning the low energy photon physics. Alex Bielajew worked on EGS at NRC from the early 80's to late 90's and his name is linked to a huge number of important contributions to EGS, perhaps most importantly the PRESTA algorithms, but also many other specific improvements to the physics, the Unix based scripts and the NRC user codes. His name appears very extensively in the reference lists. The name of Walter Ralph Nelson is practically synonymous with EGS and all users of any version of the EGS system will forever be in Ralph's debt. It has been his enthusiasm and willingness to help others and share this resource so selflessly which has made it the great success it is. To all of these people who have contributed so extensively to the EGS system, and to the countless others who have played a variety of roles, we all owe a huge debt of gratitude.

It is worth noting that NRC and SLAC have drawn up a formal agreement which recognizes that both have rights associated with EGS4 and EGSnrc. Thus, in this report there are sections which are taken verbatim from SLAC-265 (in particular the PEGS4 manual and the User's guide to Mortran3) and we wish to thank SLAC for permission to reproduce them. We also draw attention to the copyright and licensing arrangements associated with EGSnrc which are similar to those for EGS4, but which are becoming more tightly controlled in this changing world we live in. Neither EGS4 nor EGSnrc are public domain software. They are both copyright protected by NRC and/or SLAC. The formal license statement is more precise and part of the package, but the general meaning is that individuals are granted a without cost license to use it for non-commercial purposes but that a license from NRC is needed for any commercial application, and by definition someone working for a for-profit organization or working on a contract for such an organization is working on a commercial application.

What is next?
In the section of the Preface to SLAC-265, there were 7 areas identified as needing more work. The work on EGS is not complete, and at least 2 of the 7 are still open, viz:

- development of an efficient, general purpose geometry package tailored to the EGS structure
- implementation of a general purpose energy loss straggling algorithm which properly handles energy cutoffs

There are other issues which are still undone within EGSnrc:

- modeling of electron impact ionization
- some critical feature for your next application!!

We encourage users to contribute their improvements to the code. We will happily add those which are of general interest and make available on the distribution site those additions which are of special interest. We will also appreciate receiving bug reports. Although we have done extensive QA on the system, there have been many changes and not all parts of the code are as carefully checked as we would like. However, the pressure to release the code is forcing us to proceed at this point.

We wish to thank our many colleagues at NRC who have helped with this work. In particular Michel Proulx for his excellent help keeping the computer systems going smoothly,

Jan Seuntjens for his help with the user codes, Joanne Treurniet for her help with the most recent version of the EGS_Windows system and Blake Walters for his work on QA of the system.

I.K and D.W.O.R. Feb 2000

Contents

List of Figures

List of Tables

1 Introduction

1.1 Intent of this report

The EGS (**E**lectron–**G**amma–**S**hower) system of computer codes is a general purpose package for the Monte Carlo simulation of the coupled transport of electrons and photons in an arbitrary geometry for particles with energies above a few keV up to several hundreds of GeV. This report introduces a new, enhanced version called EGSnrc. In addition to explaining and documenting the various enhancements and changes to the previous version (EGS4[11]), this document includes several introductory and advanced tutorials on the use of EGSnrc (section 4) and also contains the EGSnrc Reference Manual(section 3), the PEGS4 User Manual (section 6), and an EGS User Guide to Mortran3 (section 8). Our intention has been to make this document wholly self-contained so that the user need not refer to the original EGS4 User Manual[11] although it is on-line and available at http://www.slac.stanford.edu/pubs/slacreports/slac-r-265.html. The heart of the present report is Section 2 which documents the physics in EGSnrc. This has changed substantially from the EGS4 comparable Chapter 2 because of the many changes in EGSnrc. However, we have chosen not to repeat the general introduction to sampling and probability theory that was in Chapter 2 of SLAC-265.

For a basic introduction to the code, see the reference manual, section 3 (page 107).

We have not presented any comparisons with experiment in this document since it has become such an extensive field that we have no hope of reproducing a fraction of the data. Instead, we have prepared a separate report on QA which presents extensive comparisons between EGS4 and EGSnrc[12]. There are significant differences in many situations because of the improved physics in EGSnrc. Two papers[5, 6] discuss many of the details of the new physics, especially as related to ion chamber calculations (which are perhaps the toughest test of any electron-photon Monte Carlo transport code). These papers provide analytic models which explain many of the shortcomings of the EGS4/PRESTA system in this very difficult problem. The cover of this report shows a comparison of the two codes run in their standard default modes.

1.2 History of the EGS system

The history has already been outlined in the Preface. For a detailed history up to 1994, the reader is referred to a report titled "History, overview and recent improvements of EGS4" by Bielajew et al. That report draws heavily on the history sections of the SLAC-210[13] and SLAC-265[11] reports with an update to 1994.

As stated in the Preface, EGSnrc is the child of many parents who have made a wide variety of contributions over the years. We will not repeat the preface here except to note that Walter "Ralph" Nelson has been the key player in the development of the system over the years and we all owe him a debt of gratitude.

1.3 Summary of EGSnrc Capabilities and Features

The following is a summary of the main features of the EGSnrc Code System, including statements about the physics that has been put into it and what can be realistically simulated.

- The radiation transport of electrons (+ or −) or photons can be simulated in any element, compound, or mixture. The data preparation package, PEGS4, creates data to be used by EGSnrc, using cross section tables for elements 1 through 100. In addition there are other data files which must be read in to implement many of the new options.
- Both photons and charged particles are transported in steps of random length rather than in discrete steps.
- The dynamic range of charged particle kinetic energies goes from a few tens of keV up to a few hundred GeV. Conceivably the upper limit can be extended higher, but the validity of the physics remains to be checked.
- The dynamic range of photon energies lies between 1 keV and several hundred GeV (see above statement).
- The following physics processes are taken into account by the EGSnrc Code System:
  - Bremsstrahlung production using either Bethe-Heitler cross sections or the NIST cross sections.
  - Positron annihilation in flight and at rest (the annihilation quanta are followed to completion).
  - Multiple scattering of charged particles by coulomb scattering from nuclei is handled using a new multiple scattering theory which overcomes the shortcomings of Molière multiple scattering theory. It allows for steps of any size and moves seamlessly from a single scattering model for short steps to an accurate multiple scattering model at large steps. The user has the option of scattering based on Rutherford scattering or scattering accounting for relativistic and spin effects.
  - Møller (e^-e^-) and Bhabha (e^+e^-) scattering. Exact rather than asymptotic formulae are used.
  - Continuous energy loss applied to charged particle tracks between discrete interactions.
    - Total restricted charged particle stopping power consists of soft bremsstrahlung and collision loss terms.
    - Collision loss determined by the restricted Bethe-Bloch stopping power with Sternheimer treatment of the density effect in the general case but with provision of using an arbitrary density effect correction and data supplied to use the density effect recommended by the ICRU in Report 37.
  - Pair production.
  - Compton scattering, either Klein-Nishina or bound Compton.

  - Coherent (Rayleigh) scattering can be included by means of an option.
  - Photoelectric effect.
  - Relaxation of excited atoms after vacancies are created (eg after photoelectric or Compton scattering events) to create fluorescent photons (K, L, M shells) and Auger and Coster-Kronig electrons may be produced and tracked if requested.
  - Electron impact ionization can be modeled using arbitrary theories for generating cross-sections. Five such cross-section compilations are provided in the EGSnrc distribution (Kawrakow, Casnati, Kolbenstvedt, Gryzinski, and Bote and Salvat).

- PEGS4 is a stand-alone data preprocessing code consisting of 12 subroutines and 85 functions. The output is in a form for direct use by EGS4.
  - PEGS4 constructs piecewise-linear fits over a large number of energy intervals of the cross section and branching ratio data.
  - In general, the user need only use PEGS4 *once* to obtain the media data files required by EGSnrc.
  - PEGS4 control input uses the NAMELIST read facility of the FORTRAN language (in Mortran3 form).
  - In addition to the options needed to produce data for EGSnrc, PEGS4 contains options to plot any of the physical quantities used by EGSnrc.
  - In addition to the material specific data files produced by PEGS4, EGSnrc uses a variety of other data files as input for the calculations.

- EGSnrc is a package of subroutines plus block data with a flexible user interface.
  - This allows for greater flexibility without requiring one to be overly familiar with the internal details of the code.
  - Together with the macro facility capabilities of the Mortran3 language, this reduces the likelihood that user edits will introduce bugs into the code.
  - EGSnrc uses material cross section and branching ratio data created and fit by the companion code, PEGS4. However, photon cross-section data are re-calculated on-the-fly using a logarithmic energy grid with an user-specific number of interpolation points. This behaviour can now be reverted if the user so desires.
  - The geometry for any given problem is specified by a *user-written* subroutine called HOWFAR which, in turn, can make use of auxiliary subprograms.
  - Auxiliary geometry routines for planes, cylinders, cones, spheres, etc., are provided with the EGSnrc Code System for those who do not wish to write their own.
  - Macro versions of these routines are also provided in the set of defining macros (*i.e.*, in the `egsnrc.macros` file) which, if used, generally result in a faster running simulation.
  - Transport can take place in a magnetic field by writing a specially designed HOWFAR subprogram, or in a more general manner (*eg.*, including electric field) by

making use of Mortran3 macro templates that have been appropriately placed for that purpose in subroutine ELECTR.The file `emf_macros.mortran` contains Bielajew's macros to implement this.

- The user scores and outputs information in the *user-written* subroutine called AUSGAB.
  - By setting various `AUSFLG` flags, the user can arrange to have access to the simulation parameters under many different situations to allow scoring of almost any parameter of interest with out delving into the code itself.
  - Auxiliary subprogram WATCH is provided in order to allow an event-by-event or step-by-step tracking of the simulation, either to the terminal or for 3-D graphics display using the program EGS_Windows.
- EGSnrc allows for the implementation of *importance sampling* and other variance reduction techniques (*eg.*, leading particle biasing, splitting, path length biasing, Russian roulette, etc.).
  - EGSnrc introduces options to allow for efficient bremsstrahlung splitting and Russian Roulette of secondary charged-particles, but only if "turned on" by the user.
  - EGSnrc calculates the range and distance of the particle to the nearest boundary on every step as part of the electron transport algorithm and there is an option to do range rejection on any particle that cannot get out of the current region.
- Initiation of the radiation transport:
  - An option exists for initiating a shower with two photons from pi-zero decay (*i.e.*, use $IQI = 2$ in the `CALL SHOWER` statement).
  - The user has the choice of initiating the transport by means of a monoenergetic particle, or by sampling from a known distribution (*eg.*, a synchrotron radiation spectrum).
  - Transport can also be initiated from sources that have spatial and/or angular distributions.

1.4 Summary of changes from EGS4

This is a brief listing of these changes which are discussed more fully in section 2 and in section 5.2.

1.4.1 Physics changes

- A completely new electron transport algorithm is used which removes all known shortcomings of the EGS4/PRESTA algorithm. If the geometry permits, the new algorithm can take much larger steps with better accuracy than previously. As it crosses a boundary, it goes into single scattering mode to ensure an accurate boundary crossing. The EGS4/PRESTA algorithm is still available as an option.

- A new multiple scattering theory is used which gets around the shortcomings of Moliere multiple scattering theory. It seamlessly goes from a single scattering mode for short steps to a multiple scatter mode for long steps.

- Within the new multiple scattering theory an option has been added to include relativistic spin effects in the cross section instead of just the Rutherford cross section which underlies Moliere theory.

- If desired, it is possible to do the entire calculation modeling elastic scattering in a single scattering mode. This is at the cost of a great deal of computing time and also does not model the inelastic energy losses in a single scattering model.

- A relaxation simulation feature has been added which allows creation and following of fluorescent photons from K, L, M shells, Auger electrons and Coster-Kronig electrons. Currently this can be called after photo-electric and Compton scattering events.

- If relaxation is not being modeled, then a photo-electron in EGSnrc carries the entire energy of the incident photon. This is a better approximation in most cases than dumping the binding energy locally and subtracting the binding energy from the photo-electron's energy (as done in EGS4).

- Sampling the angular distribution of the photo-electron is available as an option.

- Bound Compton scattering can be simulated as well as Klein-Nishina Compton scattering.

- Bremsstrahlung angular sampling has been changed from a fixed angle approximation to allowing the angular distribution to be sampled in one of two ways.

- A bug was fixed in the bremsstrahlung photon energy sampling routine which affected simulations for which AP was not small relative to the electron energy. Doing this led to a complete rewrite of the sampling routine which also increased its efficiency.

- A second bremsstrahlung photon energy sampling option was added which uses the more accurate NIST differential cross sections.

- PEGS4 has been modified to pick up the data files which scale the radiative cross sections to produce the NIST/ICRU 37 radiative stopping powers.
- A variety of variance reduction techniques which were commonly used with EGS4 have been "built in" with EGSnrc to improve the efficiency
  - bremsstrahlung splitting is done within the routine BREMS, thereby avoiding repeatedly calculating several constants
  - Russian Roulette of secondary charged particles is done in a manner which sometimes avoids sampling the particles phase space unless it survives.
  - range rejection, viz the termination of a particle history, when it cannot escape the local region, is implemented naturally and very efficiently since the particle range and distance to the nearest boundary are already calculated on every step.
- Subroutine HATCH has been modified considerably to allow initialization for the many new options.
- The Moller sampling routine was corrected as first done in the 1997 release of EGS4.
- The efficiency of the annihilation sampling routine has been improved.
- The sampling of the azimuthal angle has been recoded and saves a noticeable amount of time in a real calculation (2% in one example).
- Various changes have been made in the COMIN blocks to accommodate the above changes. Also LATCH and LATCHI are now a default part of STACK.
- Several more AUSGAB calls are available to score Auger and Coster-Kronig electrons and fluorescent x-rays.

1.4.2 System changes pre-EGSnrcMP(see ref[14] for the MP changes)

- The various source files have been rationalized and various add-on features to EGS4 have been made part of EGSnrc.
- Two options for random number generator are available. The default is the RANLUX generator which allows various "luxury levels" of generator to be used and the RANMAR generator, which had become the standard for the Unix distribution of EGS4, is also available, although re-coded. Both generators have the ability to generate sequences which are known to be independent and thus can be used for parallel processing. At the default luxury level of 1, the RANLUX generator is slightly slower than the recoded RANMAR generator, but the difference has a negligible impact on overall computing time.
- The default for calculating sines is now a function call because modern machines do this very rapidly and the table lookup method is known to be inaccurate for very small angles.

- The EGS_Windows code for generating 3-D interactive displays has been ported to run on any X-windows platform using non-proprietary software.
- The entire code has been written using `IMPLICIT NONE`. Further, all declarations have been done using `$REAL` and `$INTEGER` constructs which allow conversion to running double precision by redefining 2 macros, as long as the user codes do the same thing!
- Subroutine WATCH has been modified to accommodate the changes in the physics.

1.4.3 User code changes

- The tutor codes have been rewritten to work with EGSnrc and a new version of tutor6 has been written to demonstrate control of all variables available to EGSnrc users.
- Four of the standard NRC user codes for cylindrical geometry problems are now distributed with the system, DOSRZnrc, FLURZnrc, CAVRZnrc and SPRRZnrc.
- The above user codes have been extensively reworked to use a new generalized input package which makes it much easier for the user to generate the input files since the inputs are text oriented. Also, the geometry and physics transport inputs are common for all codes.
- The output routines have been reworked to avoid the use of VAX extensions to Fortran which were not available with many Unix compilers.
- The user codes have been cleaned up to some extent although not as much as desirable! The main user codes systematically use `$IMPLICIT-NONE` and `$REAL, $INTEGER` constructs to allow compatibility with EGSnrc and the ability to change to double precision at will.
- a bug in the energy sampling routine which caused problems in some cases has been removed. An entirely new code which is faster and more accurate is used now.

1.5 Summary of changes since 2005 edition of this report

1.5.1 Physics changes since 2005

- Electron impact ionization can be modeled using four different theories for generating cross-sections (Kawrakow, Casnati, Kolbenstvedt, or Gryzinski).
- The user has the option of supplying custom molecular form factors when including Rayleigh scattering in a simulation.
- New `alias sampling` algorithm used to select the photon angle after a Rayleigh scattering.
- New options added for Compton scattering called `simple` and `norej`.
- The user has the option to include radiative corrections for Compton scattering.

- The user has the option to use exact PWA cross sections for pair production
- The user has the option to explicitely simulate triplet production
- A new bremsstrahlung data base has been added that incorporates a much improved evaluation of electron-electron bremsstrahlung
- The ability to use total photon cross sections from EPDL-97, XCOM, or any other user-supplied tabulation has been added

1.5.2 EGSnrc snapshots

Since 2008 development snapshots of the EGSnrc system have been made available on the EGSnrc web page. These snapshots are updated much more regularly (typically every 2–8 weeks) but for now they can only be installed via a significantly simplified script on Linux systems with the GNU compilers.

1.5.3 User code changes since 2005

- Two tutor codes, `tutor2pp` and `tutor7pp`, have been added which make use of the C++ class library. These codes are the C++ equivalents to `tutor2` and `tutor7`.
- Four new user codes which use the EGSnrc C++ class library have also been added: 1) `egs_cbct` for simulating cone beam CT scans, 2) `egs_chamber` for efficient chamber simulations, 3) `egs_fac` for simulating transport in a free-air chamber, and 4) `egs_pet` for simulating PET scans, in addition to `cavity`, the first major C++ code for EGSnrc released in 2005. Note that `egs_cbct` and `egs_pet` are not included in this public release.

1.6 Summary of changes since 2009 edition of this report

There have been several additions to the physics and to the user codes available since the 2009 edition of PIRS-701. These are outlined in this section.

1.6.1 Physics changes since 2009

- A new electron impact ionization (EII) cross-section data base, based on the DWBA/PWBA model by Bote and Salvat has been added to the distribution.
- The ability to use different EII cross section compilations has been generalized to allow the use of any user-supplied tabulation.
- The user has the option of forcing the use of photon cross-sections from the PEGS4 data file.

1.6.2 EGSnrc snapshots

Since 2010 no snapshots of the EGSnrc system have been made available on the EGSnrc web page. The number of users making use of this option is so small that it doesn't justify the effort put into creating them.

1.6.3 User code changes since 2009

The C++ user-code `egs_fac` has been finally included in the distribution. This user-code simulates the transport in a free-air chamber and the calculations of correction factors in a self-consistent manner. The user-codes `egs_cbct` and `egs_pet` are not included since they are in a very experimental stage and not ready for public release.

1.7 Outline of report

In the remainder of this report there are 7 sections.

Section 2 (page 27) presents a detailed description of the physics in EGSnrc. While this section provides very important documentation of what the code is doing, it is not essential reading in order to get the code working. Arguably it is essential reading before you can use the code really well!

Section 3 (page 107) provides a detailed reference manual which tells you what must be done to write your own user code.

Section 4 (page 159) presents a series of short tutorial programs which demonstrate the essential elements of EGSnrc user codes. These are designed for those who learn by seeing examples (like DWOR).

Section 5 (page 198) presents a summary of the changes compared to EGS4 and a step by step procedure for upgrading an EGS4 user code to work with EGSnrc.

Section 6 (page 219) is the PEGS4 Users manual taken directly from SLAC-265 along with a few additional pieces of documentation, mostly about the upgrades since the original PEGS4 was released.

Section 7 (page 274) is an EGS user guide to Mortran3, again taken directly from SLAC-265.

Section 8 (page 294) outlines various system considerations associated with running EGSnrc in a Unix environment. It also discusses installation and distribution of the code.

We also draw your attention to the index which may help find things.

1.8 Associated documents

There are a variety of papers which have been written about EGSnrc and several NRC reports which are part of the distribution. These are listed below. There are also a large number of papers which have been written about EGS4.

1.8.1 Refereed Papers

- **Accurate condensed history Monte Carlo simulation of electron transport. I. EGSnrc, the new EGS4 version:**
  I. Kawrakow, Medical Physics, **27** (2000) 485 – 498.
  Describes the overall implementation of the new electron transport physics in EGSnrc.
- **Accurate condensed history Monte Carlo simulation of electron transport. II. Application to ion chamber response simulations:**
  I. Kawrakow, Medical Physics, **27** (2000) 499 – 513.
  Quantifies which problems in EGS4 led to the difficulties simulating ion chamber response accurately and demonstrates that EGSnrc does not have these problems.
- **Monte Carlo study of Spencer-Attix cavity theory at low photon energies:**
  J. Borg, I. Kawrakow, D.W.O. Rogers and J.P. Seuntjens, Med. Phys. **27** (2000) 1804 – 1813.
  Uses code to explore the accuracy of Spencer-Attix cavity theory. Paper demonstrates the accuracy of the EGSnrc code system for calculations related to real ion chambers.
- **On the representation of electron multiple elastic-scattering distributions for Monte Carlo calculations:**
  I. Kawrakow and A. F. Bielajew, Nucl. Inst. Meth. B 134, (1998) 325 – 336
  Describes the multiple scattering theory used in EGSnrc.
- **On the condensed history technique for electron transport:**
  I. Kawrakow and A. F. Bielajew, Nuclear Instruments and Methods B 142 (1998) 253 – 280.
  Describes the electron transport algorithm used in EGSnrc and demonstrates its improved accuracy compared to all other published algorithms.

1.8.2 Internal Reports

- **Monte Carlo calculated wall and axial non-uniformity corrections for primary standards of air kerma**,
  D. W. O. Rogers and J. Treurniet, NRC Report PIRS–663, May 1999.
  Describes extensive EGSnrc calculations of ion chamber response and comparison to experimental data from standards labs of response vs wall thickness. Also notes that results for these calculations, which are or correction factors, are the same as for EGS4/PRESTA calculations.

1.8.3 Manuals etc

- **NRC User Codes for EGSnrc**
  D.W.O. Rogers, I. Kawrakow, J.P. Seuntjens and B.R.B. Walters, NRC Report PIRS–702, May 2009.

Describes the EGSnrc user codes DOSRZnrc, CAVRZnrc, FLURZnrc and SPRRZnrc as well as the generalized input routine developed for use with EGSnrc user codes.

- **EGS_Windows4.0 User's Manual**,
  J. A. Treurniet and D. W. O. Rogers, NRC Report PIRS–669, Oct 1999.
  Describes the latest version of EGS_Windows which works on any X-windows based system and can be used to display EGSnrc histories in 3-D.

- **QA tests and comparisons of the EGSnrc system with EGS4**,
  B. R. B. Walters, J. Treurniet, D. W. O. Rogers and I. Kawrakow, NRC Report PIRS-703, March 2000.
  Summarizes a large number of comparisons between EGS4 and EGSnrc to highlight some differences and similarities.

- **EGSnrcMP: the multi-platform environment for EGSnrc**,
  I. Kawrakow, E. Mainegra-Hing and D. W. O. Rogers NRC Report PIRS-877, Sept 2006.
  Describes the changes in the system of scripts used to run the EGSnrc system. These major changes mean that EGSnrc now works under the Windows OS and there are GUI's for installing and running the system.

- **egs_inprz, a GUI for the NRC RZ user-codes**,
  E. Mainegra-Hing, NRC Report PIRS-801, May 2005.
  Manual for GUI for RZ user codes.

- **EGSnrc C++ class library**,
  I. Kawrakow, NRC Report PIRS-898, Apr 2006.
  Manual describing the geometry and source modules available with the EGSnrc C++ class library. Also describes how to build a user code using this library. Includes descriptions of the C++ user codes, `cavity`, `egs_chamber`, `egs_cbct`, `egs_fac` and `egs_fac`.

2 Radiation transport in EGSnrc

2.1 Introduction

Photons interact with surrounding matter via four basic processes: materialization into an electron/positron pair in the electromagnetic field of the nuclei and surrounding atomic electrons, incoherent (Compton) scattering with atomic electrons, photo-electric absorption and coherent (Rayleigh) scattering with the molecules (or atoms) of the medium. The first three collision types transfer energy from the photon radiation field to electrons[1], one of them dominate depending on energy and the medium in which the transport takes place. The pair production process[2] dominates at high energies. At some intermediate energies incoherent scattering is the most important process, at low energies the photo-electric process dominates.

Electrons, as they traverse matter, lose energy by two basic processes: inelastic collisions with atomic electrons and radiation. Radiative energy loss, which occurs in form of bremsstrahlung and positron annihilation, transfers energy back to photons and leads to the coupling of the electron and photon radiation fields. The bremsstrahlung process is the dominant mechanism of electron energy loss at high energies, inelastic collisions are more important at low energies. In addition, electrons participate in elastic collisions with atomic nuclei which occur at a high rate and lead to frequent changes in the electron direction.

Inelastic electron collisions and photon interactions with atomic electrons lead to excitations and ionizations of the atoms along the paths of the particles. Highly excited atoms, with vacancies in inner shells, relax via the emission of photons and electrons with characteristic energies.

The coupled integro-differential equations that describe the electromagnetic shower development are prohibitively complicated to allow for an analytical treatment except under severe approximations. The Monte Carlo (MC) technique is the only known solution method that can be applied for any energy range of interest.

Monte Carlo simulations of particle transport processes are a faithful simulation of physical reality: particles are “born” according to distributions describing the source, they travel certain distances, determined by a probability distribution depending on the total interaction cross section, to the site of a collision and scatter into another energy and/or direction according to the corresponding differential cross section, possibly producing new particles that have to be transported as well. This procedure is continued until all particles are absorbed or leave the geometry under consideration. Quantities of interest can be calculated by averaging over a given set of MC particle “histories” (also refereed to as “showers” or “cases”). From mathematical points of view each particle “history” is one point in a d-dimensional space (the dimensionality depends on the number of interactions) and the averaging procedure corresponds to a d-dimensional Monte Carlo integration. As such, the Monte Carlo estimate of quantities of interest is subject to a statistical uncertainty which depends on N,

[1] In this report, we often refer to both positrons and electrons as simply *electrons.* Distinguishing features will be brought out in the context.

[2] Occasionally the materialization into an e^+e^- pair takes place with the participation of an atomic electron which, after receiving sufficient energy, is set free. Such processes are known as triplet production.

the number of particle histories simulated, and usually decreases as $N^{-1/2}$. Depending on the problem under investigation and the desired statistical accuracy, very long calculation times may be necessary.

An additional difficulty occurs in the case of the Monte Carlo simulation of electron transport. In the process of slowing down, a typical fast electron and the secondary particles it creates undergo hundreds of thousands of interactions with surrounding matter. Because of this large number of collisions, an event-by-event simulation of electron transport is often not possible due to limitations in computing power. To circumvent this difficulty, Berger [15] developed the "condensed history" (CH) technique for the simulation of charged particle transport. In this method, large numbers of subsequent transport and collision processes are "condensed" to a single "step" The cumulative effect of the individual interactions is taken into account by sampling the change of the particle's energy, direction of motion, and position, at the end of the step from appropriate multiple scattering distributions. The CH technique, motivated by the fact that single collisions with the atoms cause, in most cases, only minor changes in the particle's energy and direction of flight, made the MC simulation of charged particle transport possible but introduced an artificial parameter, the step-length. The dependence of the calculated result on the step-length has become known as a step-size artifact [16].

EGSnrc is a general purpose package for the Monte Carlo simulation of coupled electron and photon transport that employs the CH technique. It is based on the popular EGS4 system [11] but includes a variety of enhancements in the CH implementation and in some of the underlying cross sections. We recognize that many of the modifications that we have made to the original EGS4 implementation are not important for high energy applications, initially EGS4' primary target. On the other side, the energy range of application of the EGS4 system has shifted over the years to lower and lower energies. To facilitate this transition many enhancements to the original EGS4 implementation has been developed, *e.g.* the PRESTA algorithm [17], the inclusion of angular distribution of bremsstrahlung photons [18], the low energy photon cross section enhancements by the group at KEK/Japan [7], to mention only some of them. The availability of these improvements, recent advances in the theoretical understanding of the condensed history technique [4, 5] and multiple elastic scattering [1], as well as unpublished results of our recent research have motivated us to undertake a major re-work of the EGS4 system the result of which is EGSnrc.

It is the purpose of this report to summarize the current stage of the EGSnrc system. We have attempted a self-consistent presentation and so, some of the material contained in this report is not new. In particular, various parts come from the EGS4 manual, SLAC-265 by Nelson et al[11].

This report does not attempt to provide a complete treatment of Monte Carlo methods or probability and sampling theory. Readers not familiar with the Monte Carlo technique are encouraged to read one of the many excellent reviews available.

2.2 Photon interactions

2.2.1 Pair and triplet production

2.2.1.i Cross section

The Feynman diagram for the production of electron-positron pairs in the nuclear field is given in Fig. 1. The triplet production process is similar but the interaction takes place with

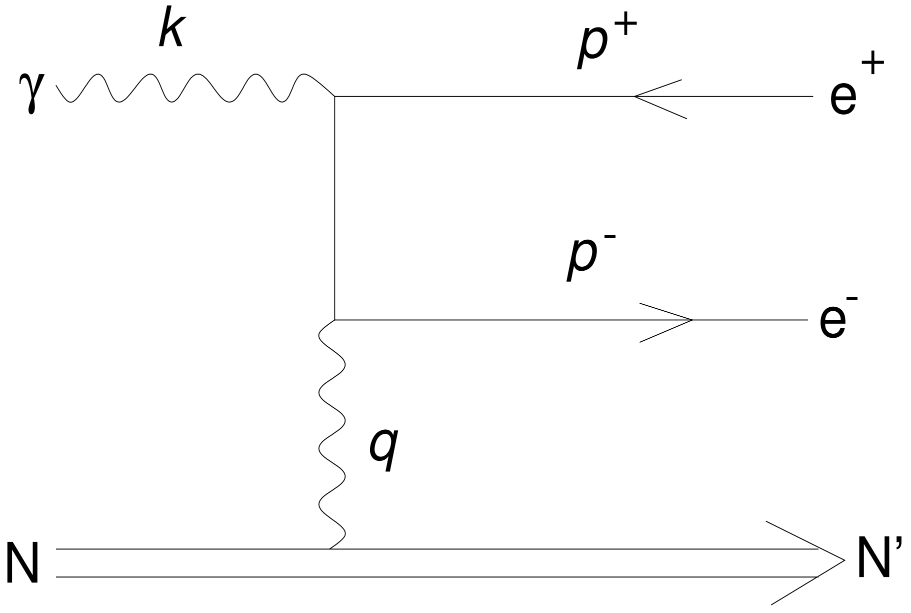


Figure 1: Feynman diagram for the pair production process

one of the atomic electrons which receives sufficient energy to be set free (so that there are 3 secondary electrons produced in the interaction). By default the triplet production process is not simulated explicitly but taken into account in an approximate way by using the total pair+triplet cross section to sample distances to subsequent pair production collisions. By default EGSnrc adopts the cross sections used in EGS4, *i.e.* extreme relativistic first Born approximation (Coulomb corrected above 50 MeV) differential cross sections as formulated in the article by Motz, Olsen and Koch [19]. For a photon energy k incident on the nucleus with the atomic number Z, the differential pair production cross section is

$$\frac{\mathrm{d}\sigma_{\mathrm{pair}}(Z,k,E_+)}{\mathrm{d}E_+} = \frac{A'_{\mathrm{p}}(Z,k) r_0^2 \alpha Z (Z+\xi(Z))}{k} \tag{2.1.1}$$
$$\left\{ \left(E_+^2 + E_-^2\right) \left[\phi_1(\delta) - \frac{4}{3}\ln Z - 4\tilde{f}_c(k,Z)\right] + \frac{2}{3}E_+E_- \left[\phi_2(\delta) - \frac{4}{3}\ln Z - 4\tilde{f}_c(k,Z)\right]\right\}$$

where E_+ and E_- are the total energies of the positron and electron,

$$\delta = 136 Z^{-1/3} 2\Delta \ , \qquad \Delta = \frac{km}{2E_+E_-} \tag{2.1.2}$$

and $\tilde{f}_c(Z)$ is the Coulomb correction,

$$\tilde{f}_c(Z) = \begin{cases} f_c(Z), & k \geq 50 \text{ MeV} \\ 0, & \text{else} \end{cases} \tag{2.1.3}$$

where $f_c(Z)$ was derived by Davies, Bethe and Maximon [20],

$$f_c(Z) = a^2 \sum_{\nu=1}^{\infty} \frac{1}{\nu(\nu^2 + a^2)}, \quad a = \alpha Z \ . \tag{2.1.4}$$

The empirical correction factor $A'_\text{p}(k, Z)$ is introduced in order to improve the total pair production cross section at lower energies and is defined as "The best estimate of the total cross section available divided by the total cross section resulting from the integration of Eq. (2.1.1) with $A'_\text{p}(k, Z) = 1$". For energies above 50 MeV A'_p is taken to be unity, for energies below 50 MeV the total pair+triplet cross sections compiled by Storm and Israel [21] are used. The replacement $Z^2 \to Z(Z + \xi(Z))$ takes into account the triplet production process, where $\xi(Z)$, evaluated by the PEGS4 function `XSIF`, is given by

$$\xi(Z) = \frac{L'_\text{rad}(Z)}{L_\text{rad}(Z) - f_c(Z)} \tag{2.1.5}$$

where $f_c(Z)$ is defined in Eq. (2.1.3) and L, L' are Tsai's radiation logarithms [22],

$$\begin{aligned} L'_\text{rad}(Z) &= \begin{cases} \ln(1194\, Z^{-2/3}) & , \quad \text{if} \quad Z > 4 \\ 6.144 & , \quad \text{if} \quad Z = 1 \\ 5.621 & , \quad \text{if} \quad Z = 2 \\ 5.805 & , \quad \text{if} \quad Z = 3 \\ 5.924 & , \quad \text{if} \quad Z = 4 \end{cases} \\ L_\text{rad} &= \begin{cases} \ln(184.15\, Z^{-1/3}) & , \quad \text{if} \quad Z > 4 \\ 5.310 & , \quad \text{if} \quad Z = 1 \\ 4.790 & , \quad \text{if} \quad Z = 2 \\ 4.740 & , \quad \text{if} \quad Z = 3 \\ 4.710 & , \quad \text{if} \quad Z = 4 \end{cases} \end{aligned} \tag{2.1.6}$$

The functions $\phi_1(\delta)$ and $\phi_2(\delta)$, which account for screening effects, are given by

$$\begin{aligned} \phi_1(\delta) &= 4 \int_\Delta^1 \frac{\mathrm{d}q}{q^3} (q - \Delta)^2 \Big[1 - F(q, Z) \Big]^2 + 4 + \frac{4}{3} \ln Z \ , \\ \phi_2(\delta) &= 4 \int_\Delta^1 \frac{\mathrm{d}q}{q^4} \left[q^3 - 6\Delta^2 q \ln\left(\frac{q}{\Delta}\right) + 3\Delta^2 q - 4\Delta^3 \right] \Big[1 - F(q, Z) \Big]^2 + \frac{10}{3} + \frac{4}{3} \ln Z \end{aligned} \tag{2.1.7}$$

where q is the momentum transfer and $F(q, Z)$ the corresponding atomic form factor for an atom with atomic number Z. For a Thomas-Fermi potential $\phi_1(\delta)$ and $\phi_2(\delta)$ are independent of Z and Butcher and Messel have approximated them [23] as

$$\phi_1(\delta) = \begin{cases} 20.867 - 3.242\delta + 0.625\delta^2 & , \quad \delta \leq 1 \\ 21.12 - 4.184 \ln(\delta + 0.952) & , \quad \delta > 1 \end{cases} \tag{2.1.8}$$

$$\phi_2(\delta) = \begin{cases} 20.029 - 1.93\delta - 0.086\delta^2 & , \quad \delta \leq 1 \\ \phi_1(\delta) & , \quad \delta > 1 \end{cases} \tag{2.1.9}$$

The differential pair production cross section for compounds and mixture is derived from the independent atom approximation and can be approximately written in the same form as Eq. (2.1.1) but replacing

$$\begin{aligned} Z(Z+\xi(Z)) \quad &\text{with} \quad Z_{\rm eff}^2 \equiv \sum p_i Z_i (Z_i + \xi(Z_i)) \\ \frac{1}{3}\ln Z + \tilde{f}_c(k,Z) \quad &\text{with} \quad Z_V \equiv \sum p_i Z_i(Z_i+\xi(Z_i))\left[\frac{1}{3}\ln Z_i + \tilde{f}_c(k,Z_i)\right] \\ \delta \quad &\text{with} \quad \delta_C 2\Delta \;, \quad \delta_C \equiv \frac{136}{Z_{\rm eff}^2}\sum p_i Z_i(Z_i+\xi(Z_i))Z_i^{-1/3} \end{aligned} \qquad (2.1.10)$$

where p_i is the normalized fraction of atoms of type i in the molecule.

It is worth noticing that, due to the use of the extreme relativistic approximation, the differential cross section as defined in Eq. (2.1.1) becomes inaccurate for energies close to the threshold energy for pair production (2 m). In the EGS4 implementation, the entire photon energy was given to one of the pair particles for $k \leq 2.1$ MeV. We have defined a macro `$SELECT-LOW-ENERGY-PAIR-PRODUCTION` which, in its default replacement, samples E_+ uniformly in the allowed range $m \cdots k/2$. If the user is aware of a better approach, this simplistic treatment can be modified by the appropriate replacement of this macro.

2.2.1.ii NRC pair cross sections

In a more recent addition to EGSnrc, the user has the option to specify use of a library of differential pair cross sections created at the NRC. These cross sections are based on the exact partial-wave-analysis calculations by Øverbø, Mork and Olsen (OMO) [24] for the unscreened nuclear potential modified by a multiplicative screening correction. The main difficulty in creating this data library, which provides cross sections up to 85 MeV for all elements between 1 and 100, consisted in finding numerically stable approaches for performing the PWA summation for energies above a few MeV (the OMO paper [24] only contains results up to 5.1 MeV). Note that these cross sections take into account the asymmetry in the positron-electron energy distribution and eliminate the need for the `$SELECT-LOW-ENERGY-PAIR-PRODUCTION` macro mentioned above.

In order to make use of the NRC pair cross sections, the user must set the variable `pair_nrc`=1. Note that `pair_nrc` is part of the `COMIN/BREMPR` common block (see Section 3.3 of this report).

2.2.1.iii Explicit simulation of triplet production

In a more recent addition to EGSnrc, the user has the option to explicitely simulate triplet production events according to the first Born approximation result first derived by Votruba [25] and later by Mork [26]. Due to the 3 particle final state the expression for the differential triplet production cross section is very complicated and so not reproduced here. It is worth noting that the published expressions most likely contain typos because their direct implementation in a computer program lead to meaningless results (*e.g.* negative cross section within the kinematically allowed range of energies and directions). The triplet production cross section was therefore re-derived using the CompHEP package, its results were manipulated using Mathematica and were then output directly to Fortran code.

In order to turn on explicit simulation of triplet events, the user mist set the variable `itriplet`=1. Note that `itriplet` is part of the `COMIN/BREMPR` common block (see Section 3.3 of this report).

The technique for sampling random directions and energies from the differential triplet cross section is very involved. Its detailed description awaits a future version of this report.

2.2.1.iv Simulation of pair production, particle energies

When the NRC pair differential cross section tabulations are used, an alias table is employed for picking the positron energy. For the default Bethe-Heitler cross sections, the sampling algorithm implemented in EGS4 becomes extremely inefficient as the incident photon energy approaches the threshold energy. This is due to the following two facts: (i) The electron and positron energies, E_- and E_+, are sampled in the range $0 \cdots k/2$, the allowed range becomes a small fraction of the above interval for $k \to 2m$. (ii) The rejection functions used are normalized to their maximum at $\delta = 0$. For photon energies that are not much larger than $2m$ the actual possible maximum is much smaller.

We have therefore slightly modified the EGS4 pair production sampling algorithm to improve its efficiency. If we define the functions

$$\begin{aligned} B(\delta) &= 3\Big[\phi_1(\delta) - 4Z_V\Big] - \Big[\phi_2(\delta) - 4Z_V\Big] \\ C(\delta) &= 3\Big[\phi_1(\delta) - 4Z_V\Big] + \Big[\phi_2(\delta) - 4Z_V\Big] \end{aligned} \tag{2.1.11}$$

which will serve as rejection functions, and make a change of variables,

$$\varepsilon = \frac{E_+ - m}{k - 2m} , \tag{2.1.12}$$

the differential pair production cross section can be rewritten as

$$\frac{\mathrm{d}\sigma_{\rm pair}}{\mathrm{d}\varepsilon} = N\left\{\frac{B(\delta)}{B_{\max}} + \left(1 - \frac{2m}{k}\right)^2 \frac{A_{\max}}{3B_{\max}} A(\delta)\left[12\left(\varepsilon - \frac{1}{2}\right)^2\right]\right\} \tag{2.1.13}$$

where N combines all constant factors that are irrelevant for the sampling algorithm and $A_{\max}$ and $B_{\max}$ are the maxima of the rejection functions $A(\delta)$ and $B(\delta)$,

$$A_{\max} = A\left(\frac{4\delta_C m}{k}\right) , \qquad B_{\max} = B\left(\frac{4\delta_C m}{k}\right) . \tag{2.1.14}$$

The sampling algorithm, which determines the energy of the lower energy “electron” is then as follows:

1. Calculate $A_{\max}$, $B_{\max}$ and α,

$$\alpha = \frac{1}{1 + (1 - 2m/k)^2 A_{\max}/3/B_{\max}} \tag{2.1.15}$$

To save time at high energies, use $A_{\max} = A(0)$ and $B_{\max} = B(0)$ for $k \geq 50$ MeV.

2. Draw a random number r_1
3. If $r_1 > \alpha$, then sample ε from $12(\varepsilon - 1/2)^2$, *i.e.*
$$\varepsilon = \frac{1}{2}\left(1 - \mathrm{Max}\{r_2, r_3, r_4\}\right) \tag{2.1.16}$$
and use $A(\delta)/A_{\rm max}$ as a rejection function in step 5
4. Else, sample ε uniformly, *i.e.*
$$\varepsilon = \frac{1}{2} r_2 \tag{2.1.17}$$
and use $B(\delta)/B_{\rm max}$ as a rejection function in step 5
5. Calculate δ and the rejection function $R = A(\delta)/A_{\rm max}$ or $B(\delta)/B_{\rm max}$
6. If $r_5 < R$, accept ε, else go to step 2.

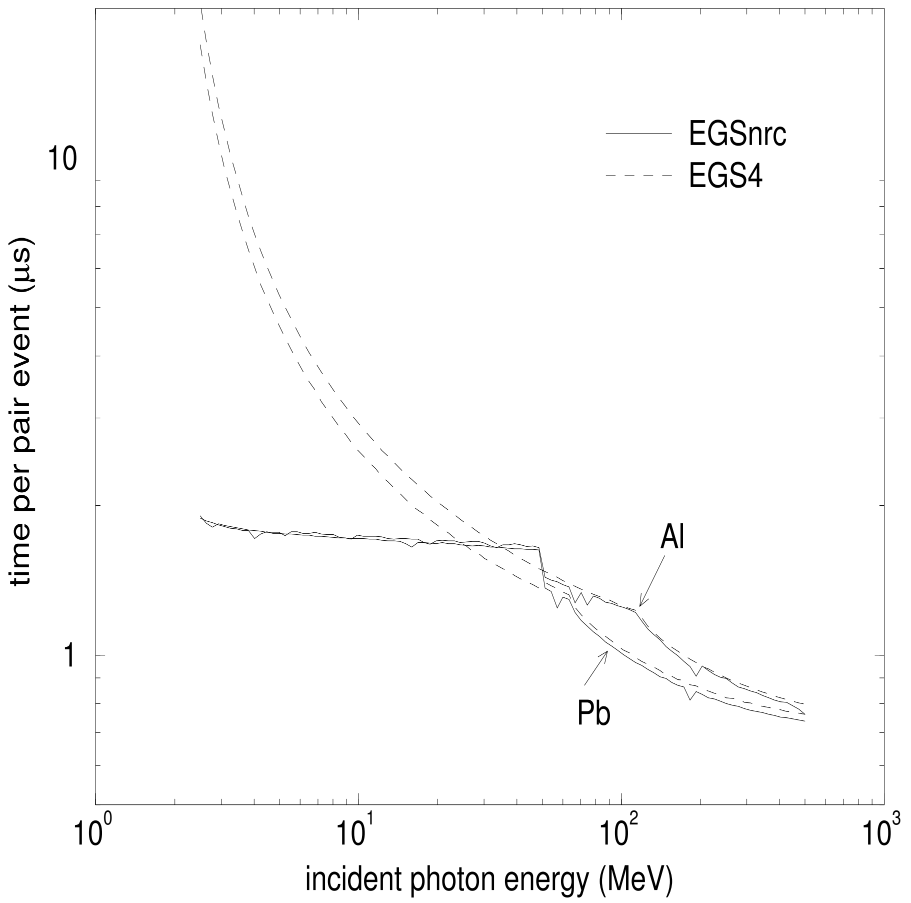


Figure 2: CPU time in μs on a 500 MHz PIII computer to sample a pair energy.

Fig. 2 shows the CPU time in μs on a 500 MHz PIII computer running Linux necessary to sample a pair energy using the algorithms discussed here (solid lines) and the original EGS4 algorithm (dashed lines) for aluminum and lead as a function of the incident photon energy (this is just the time for energy sampling, excluding angle sampling and rotations). Note the logarithmic scale and the dramatic increase in CPU time for the EGS4 algorithm and a photon energy less then 20 or 30 MeV. The discontinuity in the EGSnrc algorithm around 50 MeV is due change in the approach to calculate $A_{\rm max}$ and $B_{\rm max}$ (see item 1).

2.2.1.v Simulation of pair production, particle angles

In the original EGS4 version electrons and positrons were produced at a fixed polar angle $\theta_\pm$ with respect to the direction of the incoming photon given by $\theta_\pm = m/k$. This approach was subsequently improved as discussed in PIRS Report 0287 [27] which introduced the `$SET-PAIR-ANGLE` macro as an NRC extension to the EGS4 system. This macro is now included in the EGSnrc system. The angle selection procedure is controlled by the variable `IPRDST` (part of the `COMIN/EDGE` common block– see Section 3.3) which can assume the following values:

`IPRDST=0`: The original EGS4 approach is used, *i.e.* $\theta_\pm = m/k$

`IPRDST=1`: The leading order term of the angular distribution is employed, *i.e.*

$$\frac{{\rm d}\sigma}{{\rm d}\Omega_\pm} = N \frac{1}{(1-\beta_\pm \cos\theta_\pm)^2} \tag{2.1.18}$$

where N is again a normalization constant and $\beta_\pm$ denotes the velocity of the positron or electron in units of the speed of light.

`IPRDST=2`: The formula 3D-2003 of the article by Motz *et al.* [19] is used, which is the cross section, differential in electron/positron energy and angle:

$$\begin{gathered}\frac{{\rm d}\sigma}{{\rm d}E_\pm {\rm d}\Omega_\pm} = \frac{N}{(u^2+1)^2}\left\{-(E_+-E_-)^2 - \frac{16u^2E_+E_-}{(u^2+1)^2} + \left[E_+^2+E_-^2+\frac{4u^2E_+E_-}{(u^2+1)^2}\right]\ln M(k,E_\pm,u)\right\} \\ u = E_\pm\theta_\pm \ , \quad \frac{1}{M(k,E_\pm,u)} = \left(\frac{k}{2E_+E_-}\right)^2 + \left(\frac{Z_{\rm eff}^{1/3}}{111(u^2+1)^2}\right)^2\end{gathered} \tag{2.1.19}$$

where all energies are measured in units of m. Note that Eq. (2.1.19) is based on an extreme relativistic, small angle approximation where

$$\begin{aligned}(1-\beta_\pm\cos\theta_\pm)^2 &\approx \left[1-\beta_\pm\left(1-\frac{\theta_\pm^2}{2}\right)\right]^2 = (1-\beta_\pm)^2\left[1-\frac{\beta_\pm}{1-\beta_\pm}\frac{\theta_\pm^2}{2}\right]^2 \\ &\approx (1-\beta_\pm)^2(1+u^2)^2 \ .\end{aligned} \tag{2.1.20}$$

Perhaps, it would be a good idea to replace $1+u^2$ with $1-\beta_\pm\cos\theta_\pm$ in the denominator outside of the curled brackets of Eq. (2.1.19), but we have not undertaken this modification.

The sampling algorithm for Eq. (2.1.19) is discussed extensively in Ref. [27]. It should be noted that this algorithm becomes progressively more inefficient with increasing energies. Given this fact and the approximations involved which make its use questionable at low energies, we have chosen `IPRDST=1` as the default pair angle selection scheme in EGSnrc. The generation of electron and positron polar angles from the distribution (2.1.18) is trivial, it is accomplished by

$$\cos\theta_\pm = \frac{2r - 1 + \beta_\pm}{\beta_\pm(2r-1) + 1} \tag{2.1.21}$$

where r is an uniformly distributed random number between zero and unity. As pair-production is a three-body process, separate polar angles for the electron and positron are needed. The two azimuthal angles are chosen to be opposite. This is, strictly speaking, not correct, but due to lack of a better alternative, adopted from the original EGS4 version.

2.2.1.vi Russian Roulette for pair production events

It is wasteful to simulate all pair production events if the user intends to play Russian Roulette with electrons set in motion in photon interactions. We have therefore implemented an EGSnrc internal Russian Roulette scheme which is turned on by setting the flag `i_play_RR` which is in `COMIN/EGS-VARIANCE-REDUCTION/` to 1. The survival probability for the electrons is `prob_RR`, also in `COMIN/EGS-VARIANCE-REDUCTION/`. If `i_play_RR` is set, the following actions are taken at the beginning of subroutine `PAIR`:

1. Pick a random number r
2. If $r >$ `prob_RR`, reduce the stack size by one, return to `PHOTON` (*i.e.* save the simulation of the pair event). If the stack becomes empty, a zero weight, zero energy photon is left on the stack so that the `PHOTON` routine can exit properly.
3. If $r <$ `prob_RR`, increase the weight of the current photon by 1/prob_RR and simulate the pair event as usual.

For more discussion of Russian Roulette see sections 3.11.3 and 3.4.6.

2.2.2 Incoherent (Compton) scattering

2.2.2.i Cross section

The Feynman diagram for the Compton scattering process is shown in Fig. 3. The circle in the line of the incoming atom A indicates that the electron is initially bound to the atom and represents the probability that an atomic electron with a four-momentum $p = (E, \vec{p})$ interacts with the incoming photon with a four-momentum $k = (k, \vec{k})$ into a final $e^-\gamma'$ state given by $k' = (k', \vec{k'})$ and $p' = (E', \vec{p'})$. To simplify the notation, all energies will be measured in units of the electron's rest energy m and all momenta in units of m/c in the following equations of this section.

If the binding to the atom is neglected and the electron is considered to be initially at rest (*i.e* $p = (1, 0, 0, 0)$), the cross section for the process is given by the Klein-Nishina formula

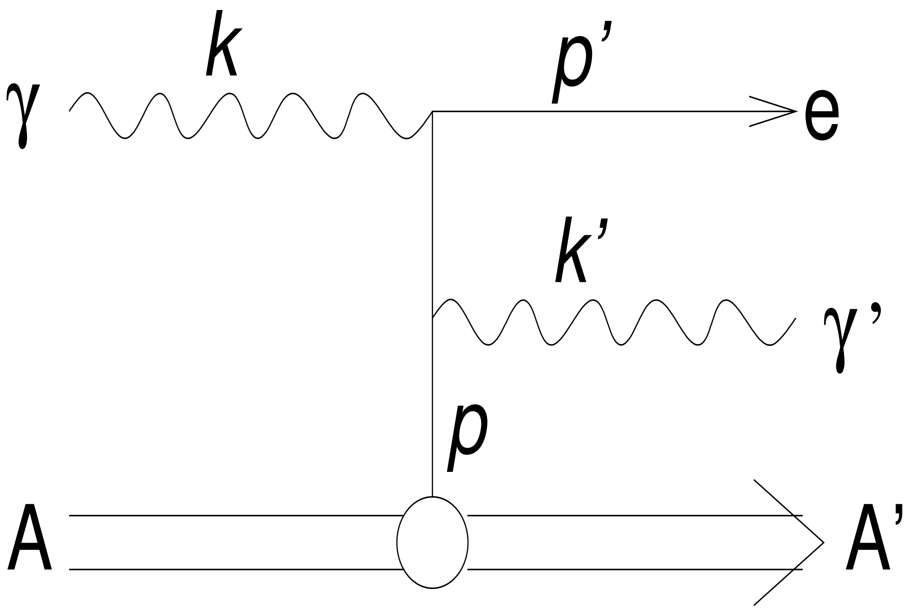


Figure 3: Feynman diagram for the Compton process

[28],

$$\frac{\mathrm{d}\sigma_{\mathrm{KN}}}{\mathrm{d}\cos\theta} = \pi r_0^2 Z\ X_{\mathrm{KN}}\ , \qquad X_{\mathrm{KN}} = \left(\frac{k_c}{k}\right)^2 \left[\frac{k_c}{k} + \frac{k}{k_c} - \sin^2\theta\right] \tag{2.2.1}$$

where θ is the polar angle of the scattered photon with respect to the initial direction and k_c is the energy of a photon scattered at an angle θ by free electrons at rest,

$$k_c = \frac{k}{1 + k(1 - \cos\theta)} \tag{2.2.2}$$

The treatment of the Compton process in EGS4 is based on these equations with $k' = k_c$. In EGSnrc we have included binding effects and Doppler broadening according to the impulse approximation (IA) [29]. The IA assumes that the potential in which the target electrons move is constant so that their states can be represented by plane waves. The double differential cross section for photon scattering into the final state $k' = (k', k'\sin\theta\cos\phi, k'\sin\theta\sin\phi, k'\cos\theta)$ is given by (Eq. (15) of Ref. [30])

$$\frac{\mathrm{d}^2\sigma_{\mathrm{comp}}}{\mathrm{d}k'\mathrm{d}\Omega} = \frac{r_0^2}{2}\frac{k'}{kq}\left[1 + p_z^2\right]^{-1/2} X J(p_z) \tag{2.2.3}$$

where Ω is the solid angle (θ, ϕ) and where

- q is the modulus of the momentum transfer vector $\vec{q} = \vec{k}' - \vec{k}$,

$$q = \sqrt{k^2 + k'^2 - 2kk'\cos\theta} \tag{2.2.4}$$

- p_z is the projection of the initial electron momentum on the direction of $\vec{q}$,

$$p_z = \frac{\vec{p}\cdot\vec{q}}{q} = \frac{kk'(1-\cos\theta) - k + k'}{q} \tag{2.2.5}$$

Note that the above equation is derived from a non-relativistic approximation and requires $|p_z| \leq 1$.

- X is defined as

$$\begin{aligned} X &= \frac{R}{R'} + \frac{R'}{R} + 2\left(\frac{1}{R} - \frac{1}{R'}\right) + \left(\frac{1}{R} - \frac{1}{R'}\right)^2 \\ R &= k\left[\sqrt{1+p_z^2} + \frac{k - k'\cos\theta}{q} p_z\right] \\ R' &= R - kk'(1-\cos\theta) \ . \end{aligned} \tag{2.2.6}$$

Note that R and R' simplify to

$$R \approx k\,(1 + O(p_z)) \ , \quad R' \approx k\Big[1 - k_c(1-\cos\theta)\Big]\,(1 + O(p_z)) \tag{2.2.7}$$

for $p_z \ll 1$. In this limit

$$X = X_{\rm KN}\,(1 + O(p_z)) \tag{2.2.8}$$

- The function $J(p_z)$ is the Compton profile,

$$J(p_z) = \int {\rm d}p_x {\rm d}p_y |\psi(\vec{p})|^2 \ , \tag{2.2.9}$$

where $\psi(\vec{p})$ is the wave function of the bound electrons. Extensive tables of atomic and shell-wise Hartree-Fock Compton profiles for all elements have been published by Biggs *et al* [31]. Following Brusa *et al* [32], contributions from different electron shells are considered separately, so that the atomic or molecular Compton profile is the sum of one-electron shell Compton profiles $J_i(p_z)$, and binding effects are taken into account by rejecting interactions that transfer less energy to the electron than the binding energy U_i, *i.e.*

$$J(p_z) = \sum Z_i J_i(p_z) \Theta(k - k' - U_i) \ . \tag{2.2.10}$$

Here, Z_i is the occupation number of shell i and the J_i have the normalization

$$\int\limits_{-\infty}^{\infty} {\rm d}p_z J_i(p_z) = 1 \tag{2.2.11}$$

With all this, and after changing the cross section from differential in k' to differential in p_z, Eq. (2.2.3) can be written as

$$\frac{{\rm d}^2\sigma_{\rm comp}}{{\rm d}p_z {\rm d}\Omega} = \frac{r_0^2}{2} X_{\rm KN} \left(\sum Z_i J_i(p_z)\Theta(k-k'-U_i)\right) F(k,\cos\theta,p_z) \tag{2.2.12}$$

where the function $F(k,\cos\theta,p_z)$ combines all remaining factors times ${\rm d}k'/{\rm d}p_z$,

$$F(k,\cos\theta,p_z) = \frac{k'}{k_c}\Big[1+p_z^2\Big]^{-1/2} \frac{X}{X_{\rm KN}} \left(1 + \frac{k_c}{k}\frac{k\cos\theta - k'}{q} p_z\right)^{-1} \tag{2.2.13}$$

Here, k' is a function of $k, \cos\theta$ and p_z and follows from solving Eq. (2.2.5) with respect to k', *i.e.*

$$k' = \frac{k_c}{1 - p_z^2 \varepsilon^2} \left[1 - p_z^2 \varepsilon \cos\theta + p_z \sqrt{1 - 2\varepsilon \cos\theta + \varepsilon^2 \left(1 - p_z^2 \sin^2\theta\right)} \right] , \qquad \varepsilon = \frac{k_c}{k} \tag{2.2.14}$$

The incoherent scattering cross section, differential in the photon scattering angle, is

$$\frac{\mathrm{d}\sigma_{\mathrm{comp}}}{\mathrm{d}\Omega} = \frac{r_0^2}{2} X_{\mathrm{KN}} S(k, \cos\theta) \tag{2.2.15}$$

where

$$S(k, \cos\theta) = \sum Z_i \Theta(k - U_i) S_i, \qquad S_i = \int_{-\infty}^{p_i} \mathrm{d}p_z J_i(p_z) F(k, \cos\theta, p_z) \tag{2.2.16}$$

can be identified with the incoherent scattering function. The upper limit of the p_z integration for the i'th shell, p_i,

$$p_i = \frac{k(k - U_i)(1 - \cos\theta) - U_i}{\sqrt{2k(k - U_i)(1 - \cos\theta) + U_i^2}} , \tag{2.2.17}$$

follows from Eq. (2.2.5) with $k' = k - U_i$ and assures that sufficient energy is transferred to the electron to set it free. To first order, $S(k, \cos\theta)$ depends only on $k\sqrt{(1 - \cos\theta)/2}$ and so, Namito *et al* [9] use tabulated incoherent scattering functions in their extension of the EGS4 system to include binding effects and Doppler broadening, when sampling the photon scattering angle. This approach introduces a slight inconsistency in their treatment of the Compton process.

As the Compton profiles are strongly peaked around $p_z = 0$, the main contributions to the integral come from small p_z values where the function $F(k, \cos\theta, p_z)$ is very close to unity and so, Brusa *et al* [32] approximate $S(k, \cos\theta)$ with

$$S(k, \cos\theta) \approx S_A((k, \cos\theta) \equiv \sum Z_i \int_{-\infty}^{p_i} \mathrm{d}p_z J_i(p_z) \Theta(k - U_i) \tag{2.2.18}$$

for their implementation of binding effects and Doppler broadening in the PENELOPE system [33][3] This approach introduces a small error at low energies for high Z materials as can be seen in Fig. 4 which shows $S(k, \cos\theta)$ for 50 keV photons in lead calculated with or without taking into account $F(k, \cos\theta, p_z)$.

To minimize the amount of data necessary for the simulation of the Compton process, and to make the calculation of the incoherent scattering function "on the fly" possible, we use the analytical approximations for the Compton profiles $J_i(p_z)$ proposed by Brusa *et al* [32]:

$$J_i(p_z) = J_{i,0}(1 + 2J_{i,0}|p_z|) \exp\left[\frac{1}{2} - \frac{1}{2}\left(1 + 2J_{i,0}|p_z|\right)^2\right] \tag{2.2.19}$$

[3]They take into account $F(k, \cos\theta, p_z)$, using its Taylor expansion up to $O(p_z)$, for the sampling of p_z.

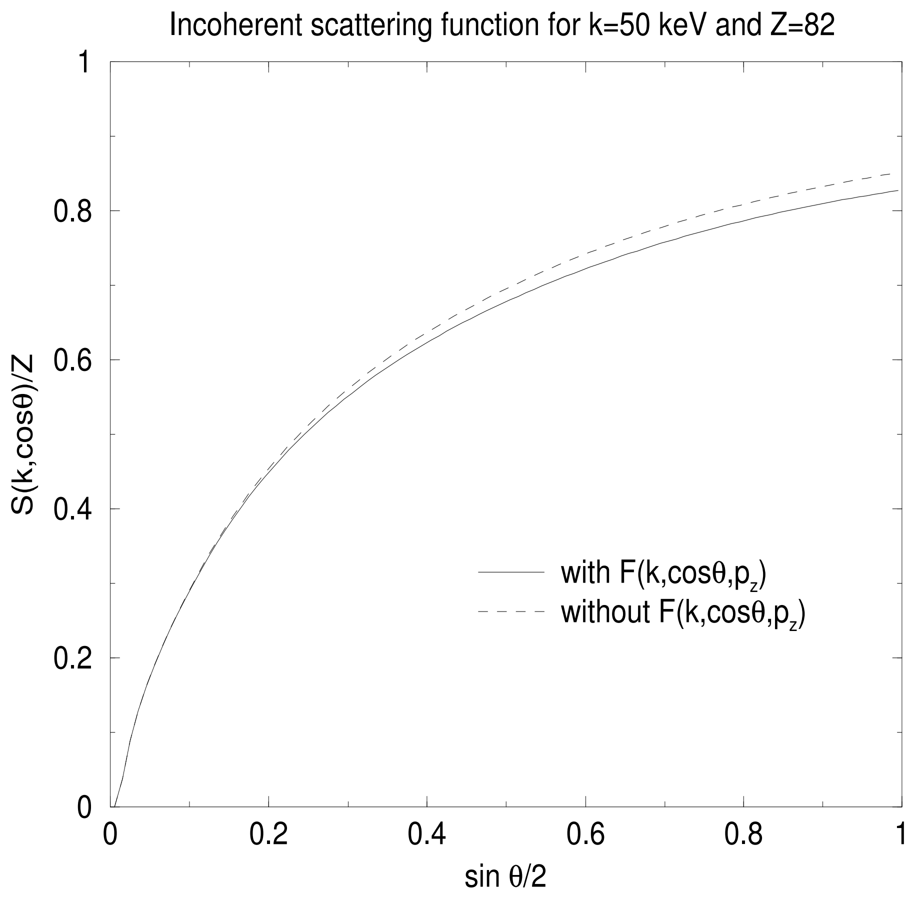


Figure 4: The incoherent scattering function for lead and $k = 50$ keV calculated from Eq. (2.2.16) (solid line) and from Eq. (2.2.18) (dashed line) by numerical integration
.

where $J_{i,0} \equiv J_i(0)$ is the value of the profile at $p_z = 0$ obtained from the Hartree-Fock orbital [31]. In addition, we approximate $F(k, \cos\theta, p_z)$ by

$$F((k, \cos\theta, p_z) = \begin{cases} 1 - \alpha p & , \quad p_z \leq -p \\ 1 + \alpha p_z & , \quad |p_z| < p \\ 1 + \alpha p & , \quad p_z \geq p \end{cases} \tag{2.2.20}$$

where

$$\begin{aligned} \alpha &= \frac{q_c}{k}\left(1 + \frac{k_c(k_c - k\cos\theta)}{q_c^2}\right) \\ q_c &= \sqrt{k^2 + k_c^2 - 2kk_c\cos\theta}\ . \end{aligned} \tag{2.2.21}$$

Equation (2.2.20) results from a Taylor series expansion of $F(k, \cos\theta, p_z)$ up to $O(p_z)$. As this approximation becomes inaccurate for large p_z values it is applied only for $|p_z| < p$, else the function values at $\pm p$ are used. We have checked by numerical integration of Eq. (2.2.16) that the incoherent scattering function calculated with the approximation (2.2.20) agrees to better than 0.3% with the incoherent scattering function calculated using the exact expression for $F(k, \cos\theta, p_z)$ if $p = 0.15$ is used.

Combining now Eq. (2.2.16), (2.2.19) and (2.2.20), we obtain for S_i

$$\begin{aligned}
& S_i(k,\cos\theta) \\
= \;& (1-\alpha p)\frac{e^{-b}}{2}\,, \quad \text{if } \; p_i \le -p \\
= \;& (1+\alpha p_i)\frac{e^{-b}}{2} - \frac{\alpha}{4J_{i,0}}\sqrt{\frac{\pi}{2}}e^{1/2}\left[\mathrm{Erf}\left(\frac{1+2J_{i,0}p}{\sqrt{2}}\right) - \mathrm{Erf}\left(\frac{1+2J_{i,0}|p_i|}{\sqrt{2}}\right)\right]\,, \quad \text{else if } \; p_i \le 0 \\
= \;& 1-(1+\alpha p_i)\frac{e^{-b}}{2} - \frac{\alpha}{4J_{i,0}}\sqrt{\frac{\pi}{2}}e^{1/2}\left[\mathrm{Erf}\left(\frac{1+2J_{i,0}p}{\sqrt{2}}\right) - \mathrm{Erf}\left(\frac{1+2J_{i,0}|p_i|}{\sqrt{2}}\right)\right]\,, \quad \text{else if } \; p_i < p \\
= \;& 1-(1+\alpha p)\frac{e^{-b}}{2}\,, \quad \text{else}
\end{aligned} \tag{2.2.22}$$

where

$$b = \frac{1}{2}\left(1+2J_{i,0}|p_i|\right)^2 - \frac{1}{2} \tag{2.2.23}$$

and Erf is the error function. It is worth noticing that S_i is always less or equal to unity. This fact allows for using it as a rejection function in the sampling algorithm discussed in the next section.

The total incoherent scattering cross section $\sigma^{(\mathrm{tot})}_{\mathrm{comp}}$ can be obtained by a numerical integration over all scattering angles from Eq. (2.2.15), (2.2.16) and (2.2.22). To avoid substantial changes of the data preparation program PEGS4, we use instead the total Klein-Nishina cross section, $\sigma^{(\mathrm{tot})}_{\mathrm{KN}}$, while tracking the photons through the geometry, and reject Compton interactions with the probability $1-\sigma^{(\mathrm{tot})}_{\mathrm{comp}}/\sigma^{(\mathrm{tot})}_{\mathrm{KN}}$ once at the interaction site (fictitious cross section method). This rejection probability results automatically (without calculating $\sigma^{(\mathrm{tot})}_{\mathrm{comp}}$) from the sampling algorithm, as we shall see in the next section.

Another advantage of this approach is that the user can turn on or off binding effects and Doppler broadening using the switch `IBCMP` (included in the common block `COMIN/COMPTON-DATA`– see Section 3.3), without preparing two separate material data files. The only additional data needed to simulate incoherent scattering in the impulse approximation are the Compton profile parameters $J_{i,0}$, taken from the tabulations by Biggs *et al* [31], and the occupation numbers Z_i and binding energies U_i for all elements, taken from Lederer and Shirley [34]. These data are read in in subroutine `init_compton` which is called from `HATCH`.

2.2.2.ii Simulation of incoherent scattering events

The incoherent scattering cross section, differential in the photon scattering angle $\Omega = (\theta,\phi)$ and the Doppler broadening parameter p_z, per interaction site sampled from the total Klein-Nishina cross section is

$$\begin{aligned}
\frac{\mathrm{d}^2\sigma_{\mathrm{comp}}}{\sigma^{(\mathrm{tot})}_{\mathrm{KN}}} &= \sum \frac{Z_i}{Z}\Theta(k-U_i)\;\sigma_i\;\mathrm{d}\Omega\;\mathrm{d}p_z & (2.2.24) \\
\sigma_i\;\mathrm{d}\Omega\;\mathrm{d}p_z &= S_i\,\frac{\mathrm{d}\phi}{2\pi}\,\frac{X_{\mathrm{KN}}(\cos\theta)\;\mathrm{d}\cos\theta}{\int\limits_{-1}^{1} X_{\mathrm{KN}}(\cos\theta)\mathrm{d}\cos\theta}\;\frac{J_i(p_z)F(k,\cos\theta,p_z)\Theta(p_i-p_z)\mathrm{d}p_z}{\int\limits_{-\infty}^{p_i}\mathrm{d}p_z J_i(p_z)F(k,\cos\theta,p_z)}
\end{aligned}$$

It is then clear that the following algorithm produces the required number of rejections and samples the differential cross section correctly, if the interaction is accepted:

1. Sample the shell number i using the probabilities Z_i/Z
2. If the incident photon energy k is smaller than the binding energy U_i, reject the interaction, else, sample $\cos\theta, \phi$ and p_z using the differential cross section σ_i of the selected shell as follows:
3. Sample the photon polar scattering angle θ using
$$P_1(\cos\theta) = \frac{X_{\rm KN}(\cos\theta)\ {\rm d}\cos\theta}{\int\limits_{-1}^{1} X_{\rm KN}(\cos\theta){\rm d}\cos\theta}$$
which is a normalized probability distribution function (PDF). The method for sampling $\cos\theta$ will be explained below.
4. Calculate the maximum possible value of p_z, p_i, from Eq. (2.2.17), and S_i from Eq. (2.2.22).
5. If a uniformly distributed random number is greater then S_i, reject the interaction
6. Sample p_z from
$$P_2(p_z) = \frac{J_i(p_z)F(k,\cos\theta,p_z)\Theta(p_i-p_z){\rm d}p_z}{\int\limits_{-\infty}^{p_i} {\rm d}p_z J_i(p_z)F(k,\cos\theta,p_z)}$$
which is a normalized PDF for p_z. The sampling technique employed for this distribution function is explained below.
7. Sample the azimuthal scattering angle from ${\rm d}\phi/(2\pi)$
8. Calculate the energy k' of the scattered photon from Eq. (2.2.14)
9. The electron set in motion has then a kinetic energy of $k - k' - U_i$, a polar scattering angle θ_e given by
$$\cos\theta_e = \frac{k - k'\cos\theta}{\sqrt{k^2 + k'^2 - 2kk'\cos\theta}} \qquad (2.2.25)$$
and an azimuth opposite to the photon's azimuth.

Note that if the user does not want to take into account binding effects and Doppler broadening (the switch `IBCMP` is set to zero) the sampling algorithm consists of steps 3, 7 and 9 with $k' = k_c$ and $U_i = 0$.

As the shell with which the interaction takes place is explicitly determined in step 1, at the end of the sampling algorithm the vacancy created by the Compton process is known. The relaxation of shell vacancies with binding energies above the specified transport threshold energies is performed in subroutine `relax` (see section 2.3) and may lead to the creation of additional fluorescent photons and Auger and Coster-Kronig electrons on the stack. Many of the EGS4 based user codes assume that the outcome of an incoherent scattering process is a scattered photon and a Compton electron. This assumption is obviously not satisfied for EGSnrc, the outcome of an incoherent event may by any one of the following:

1. The original photon, if the interaction was rejected due to the one of the rejection criteria
2. A scattered photon and a Compton electron, if all relaxation particles had energies below the specified transport threshold energies (`ECUT` and `PCUT`)
3. A scattered photon, a Compton electron plus n relaxation particles, else.

See section 3.7.2 (page 139) for more information.

Finally, the portion of the binding energy that resulted in the creation of sub-threshold relaxation particles is made known to the user via a call to the scoring routine `AUSGAB` with the argument `IARG=4`.

We conclude this section with some details about steps 3, 4 and 6 of the sampling algorithm.

Step 3, sampling of $\cos\theta$: Following the EGS4 manual[11] we rewrite the PDF P_1 in terms of $\varepsilon = k_c/k$,

$$P_1(\varepsilon) = P_1(\cos\theta)\frac{\mathrm{d}\varepsilon}{\mathrm{d}\cos\theta} = N\left(\frac{1}{\varepsilon} + \varepsilon - \sin^2\theta\right) \tag{2.2.26}$$

where N is a normalization constant that is irrelevant for the sampling algorithm. The minimum and maximum possible values for ε, $\varepsilon_{\rm min}$ and $\varepsilon_{\rm max}$, follow from $\cos\theta = -1$ and $\cos\theta = 1$, respectively, and are given by

$$\varepsilon_{\rm min} = \frac{1}{1+2k}\;, \qquad \varepsilon_{\rm max} = 1\;. \tag{2.2.27}$$

Equation (2.2.26) can be further rewritten as

$$P_1(\varepsilon) = N(\alpha_1+\alpha_2)\left\{\frac{\alpha_1}{\alpha_1+\alpha_2}\left(\frac{1}{\varepsilon\alpha_1}\right) + \frac{\alpha_2}{\alpha_1+\alpha_2}\left(\frac{\varepsilon}{\alpha_2}\right)\right\}\left[1 - \frac{\varepsilon\sin^2\theta}{1+\varepsilon^2}\right] \tag{2.2.28}$$

with

$$\alpha_1 = \ln(1+2k)\;, \qquad \alpha_2 = \frac{1-\varepsilon_{\rm min}^2}{2} \tag{2.2.29}$$

$1/\varepsilon/\alpha_1$ and ε/α_2 are normalized PDFs, they can be used to sample ε with the probability $\alpha_1/(\alpha_1+\alpha_2)$ and $\alpha_2/(\alpha_1+\alpha_2)$, respectively. The expression of the square brackets has a maximum of unity for $\varepsilon = 1$ ($\sin\theta = 0$) and is thus a valid rejection function. The sampling algorithm is then as follows:

3.1 Calculate $\varepsilon_{\rm min}, \alpha_1, \alpha_2$ and $w = \alpha_1/(\alpha_1+\alpha_2)$

3.2 Pick three random numbers r_1, r_2 and r_3

3.3 If $r_1 \leq w$,

$$\varepsilon = \varepsilon_{\rm min} \exp(r_2 \alpha_1) \; , \tag{2.2.30}$$

else,

$$\varepsilon = \sqrt{\varepsilon_{\rm min}^2 + 2 r_2 \alpha_2} \tag{2.2.31}$$

3.4 Calculate the rejection function (g), if $r_3 > g$ go to step 3.2

3.5 Deliver ε (and $\cos\theta$)

The efficiency of this algorithm goes to unity for $k \to \infty$ and therefore this is the most efficient algorithm at high incident photon energies. For $k \to 0$, the efficiency approaches 2/3. In addition, the necessity to calculate a logarithm and an exponential function, two CPU intensive calculations, makes this algorithm slower than a simple rejection technique with a uniform sampling of ε. The rejection function in this case is given by

$$g = \frac{1}{g_{\rm max}} \left(\frac{1}{\varepsilon} + \varepsilon - \sin^2\theta \right) \; , \qquad g_{\rm max} = \frac{1}{\varepsilon_{\rm min}} + \varepsilon_{\rm min} \tag{2.2.32}$$

and the algorithm is as follows

3.1 Calculate $\varepsilon_{\rm min}$ and $g_{\rm max}$

3.2 Pick two random numbers r_1 and r_2

3.3 Set

$$\varepsilon = r_1 + (1 - r_1)\varepsilon_{\rm min} \tag{2.2.33}$$

and calculate g

3.4 If $r_2 > g$, go to step 3.2

3.5 Deliver ε (and $\cos\theta$)

The efficiency of this algorithm is 2/3 for low energies and decreases with increasing k. Because there are no CPU intensive calculations involved, it is faster up to $k \sim 2$ and used by EGSnrc in this energy range.

Step 4, calculation of S_i: The calculation of S_i requires two numerically intensive operations, the computation of $\exp(-b)$ and the calculation of the error function that depends on p_i. The former is necessary for the sampling of p_z in step 6, the latter can be avoided by using an approximate formula for Erf (Eq. 7.1.25 of Abramowitz and Stegun [35]):

$$e^{1/2} {\rm Erf} \left(\frac{1 + 2J_{i,0}|p_i|}{\sqrt{2}} \right) = e^{1/2} - e^{-b} t (a_1 + a_2 t + a_3 t^2) \tag{2.2.34}$$

$$t = \frac{1}{1 + 0.332673(1 + 2J_{i,0}|p_i|)} \; , \quad a_1 = 0.34802 \; , \quad a_2 = -0.0958798 \; , \quad a_3 = 0.7478556$$

which is accurate enough for our purposes.

Step 6, sampling p_z: p_z can be sampled using $J_i(p_z)$ as a PDF and F as a rejection function. To determine the maximum of F we note that α (given by Eq. (2.2.21)) is always positive except for a small $\cos\theta$-range for $k > 3$. For such high incident photon energies the influence of F is negligible and so we ignore it if $\alpha < 0$ (*i.e.* we set $\alpha = 0$). The maximum of the rejection function, $F_{\max}$, is then

$$F_{\max} = \begin{cases} 1-\alpha p & , \quad p_z \le -p \\ 1+\alpha p_i & , \quad |p_i| < p \\ 1+\alpha p & , \quad p_i \ge p \end{cases} \tag{2.2.35}$$

and the algorithm for sampling p_z as follows:

6.1 Calculate $F_{\max}$ from Eq. (2.2.35)

6.2 Pick two random numbers, r_1 and r_2, calculate $r' = r_1 e^{-b}$ (e^{-b} is known from step 4 of the main algorithm)

6.3 Set

$$p_z = \begin{cases} \frac{1-\sqrt{1-2\ln[2r']}}{2J_{i,0}} & , \quad r' < 1/2 \\ \frac{\sqrt{1-2\ln[2(1-r')]}-1}{2J_{i,0}} & , \quad r' \ge 1/2 \end{cases} \tag{2.2.36}$$

6.4 Calculate $F(p_z)$, if $r_2 > F(p_z)/F_{\max}$, go to step 6.2

6.5 Deliver p_z

It is easy to see that the sampling of incoherent scattering events with binding effects and Doppler broadening taken into account requires substantially more numerical work compared to the free electron case (using the Klein-Nishina cross section). Fig. 5 shows the CPU time per incoherent scattering event for EGS4 (solid line), EGSnrc without (dotted line) and with binding effects (dashed line) as a function of the incident photon energy. The long dashed line would result if the function $F(k, \cos\theta, p_z)$ was ignored. EGSnrc without binding effects is faster than EGS4 for low photon energies due to the use of the uniform sampling technique. The inclusion of binding effects increases the CPU time per Compton event by not more than a factor of two and so has only a minor effect on the overall simulation time when electron transport is included. Binding effects and Doppler broadening for coherent scattering are therefore turned on by default in the `block data` sub-program (the switch `IBCMP` is set to 3, `norej`).

Since the 2009 release of EGSnrc, the user has the option to change the EGSnrc behavior with respect to Compton interactions by setting `ibcmp=2` or `ibcmp=3`. When `ibcmp=2`, binding effects are taken into account via an incoherent scattering function, but there is no Doppler broadening. This option was mainly added for the sake of being able to study the effect of Doppler broadening by comparing simulations with `ibcmp=2` to `ibcmp=1`. To select this option the user must set `Bound Compton scattering` to `simple` in the `MC TRANSPORT PARAMETER` input block. The `ibcmp=3` option is the same as `ibcmp=1`, but now the actual total bound Compton cross section is used and there are no rejections at run time. The addition of this option was motivated by the complications associated with the initial bound Compton scattering approach when using an un-weighting technique to compute ion chamber

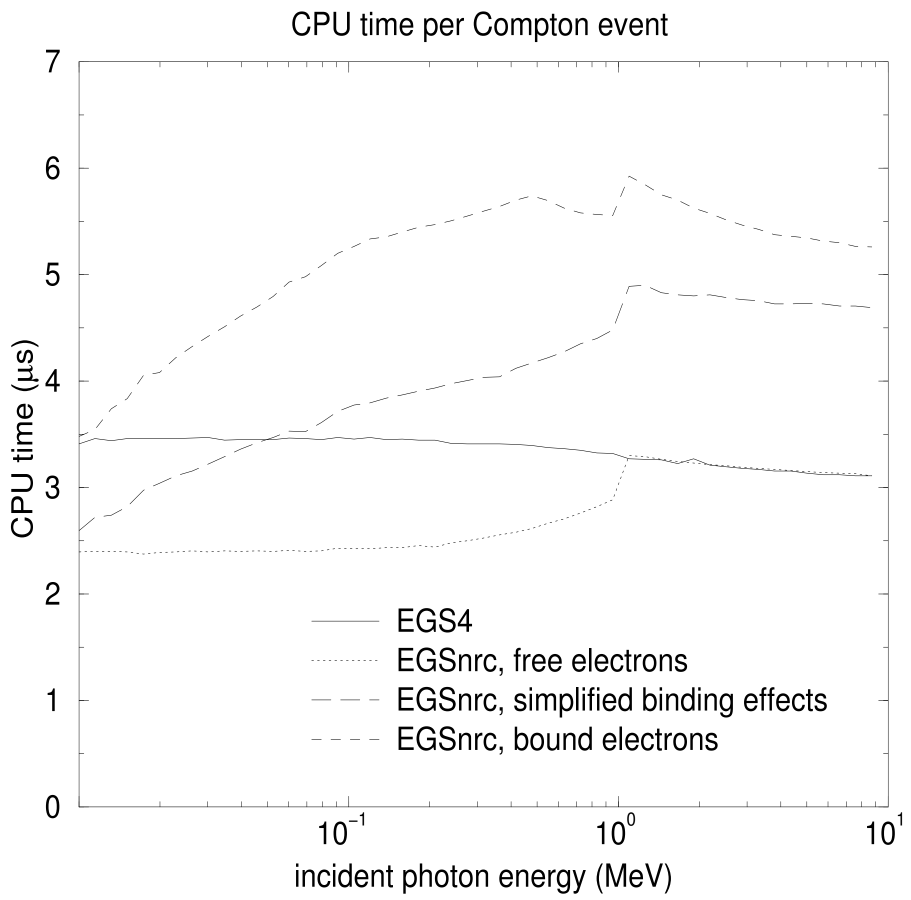


Figure 5: CPU time in μs on a 500 MHz PIII computer to sample an incoherent scattering event (including necessary rotations of particle directions).

correction factors for photon attenuation. To select this option the user must set `Bound Compton scattering` to `norej` in the `MC TRANSPORT PARAMETER` input block.

If the Russian Roulette option is turned on `i_play_RR` set to 1, electrons produced as a result of the Compton interaction (the Compton electron and electrons from the relaxation of the shell vacancy) are removed from the stack with the probability `prob_RR`. If they survive, their weight is increased by 1/`prob_RR`.

2.2.2.iii Radiative Compton corrections

Recently, the code has been modified to also allow the user to include radiative corrections for Compton scattering in the one-loop approximation. These corrections are based on the original Brown & Feynman equations [36]. The effect of loop corrections is to reduce the cross section at large photon scattering angles. This is partially offset by the addition of the double-Compton scattering process (two photons in the final state). Devising an efficient technique for sampling energies and directions from the double-Compton cross section is the most difficult part in the implementation. The method is quite involved and its detailed description awaits a future version of this report.

In order to turn on radiative corrections, the input variable `radc_flag` (in the common block `COMIN/COMPTON-DATA`) must be set to 1. This option can be enabled in applications which can read the `MC transport parameter` input block from an input file (`*.egsinp`) via the input key:

```
Radiative Compton corrections = on
```

Moreover, one must include the file `$(EGS_SOURCEDIR)rad_compton1.mortran` to the list of MORTRAN sources to be built just before `$(EGS_SOURCEDIR)get_inputs.mortran`. Depending on the application type this is accomplished in different ways:

- For all C++ applications one must edit the file `$HEN_HOUSE/specs/egspp1.spec` and include the above file in the definition of the `C_ADVANCED_SOURCES` variable if the application is derived from the `EGS_AdvancedApplication` class or the `C_SIMPLE_SOURCES` variable for any application derived from the `EGS_SimpleApplication` class.
- For a specific C++ application one could define the variable `CPP_SOURCES` in the application's `Makefile` in the same manner as done above for the `C_ADVANCED_SOURCES`.
- For a `BEAMnrc` application one must edit the file `sources.make` located in the application's folder and include the above file in the definition of the `SOURCES` variable or the `LIB_SOURCES` variable if this `BEAMnrc` application will be used as a particle source for another application.
- For a MORTRAN application named for instance `app` one must edit the corresponding file app.make located in the application's folder `$EGS_HOME/app` and include the above file in the definition of the `SOURCES` variable.

2.2.2.iv Total Compton cross sections

As mentioned above, the total Compton cross section used in EGSnrc is obtained from the theoretical expressions by integration of the differential cross sections. Another recent addition to the EGSnrc code allows the user to supply their own cross sections for bound Compton scattering. This option is only available if the user is simulating bound Compton scattering (`IBCMP`=1,2,3). The cross sections must be contained in a file `$HEN_HOUSE/data/comp_xsections_compton.data`, where `comp_xsections` is the variable (in common block `COMIN/MEDIA`) holding the name as supplied by the user.

2.2.3 Photo-electric absorption

In the photo-electric absorption process a photon is absorbed by an atom and an electron is emitted with an energy given by the incident photon energy minus its binding energy. The atom, left in an excited state with a vacancy in the ionized shell, relaxes via the emission of fluorescent photons and Auger and Coster-Kronig electrons.

In the original default EGS4 implementation the emission of relaxation particles following photo-electric absorption events was ignored. This approach was later modified to include the production of K_α and K_β fluorescent radiation. However, for incident photon energies below the K-shell binding energy, the entire photon energy is deposited locally. Another shortcoming of the EGS4 approach is that the K-shell binding energy is always subtracted from the energy of the electron set in motion, even though there is a certain probability that the photo-absorption process takes place with a shell other than the K-shell (for high-Z materials this probability is of the order of 20%). Finally, the use of the fluorescent option in EGS4 requires the user to select an "effective" atomic number for each material. The photo-absorption then always takes place with this atomic number. The meaningful selection of an "effective" Z proves to be a difficult task for mixtures, especially when only a small fraction of a high-Z element is present.

Although this release of EGSnrc uses the total photo-absorption cross sections from PEGS (which are taken from the compilation by Storm and Israel [21]) the simulation of the photo-absorption process is completely changed and is controlled by the flag `IEDGFL` (in the `COMIN/EDGE` common block-see Section 3.3). If `IEDGFL` of the region is non-zero, a detailed simulation is performed, otherwise a simplified treatment of the photo-absorption process is undertaken. The default setting of `IEDGFL` is 1.

2.2.3.i Detailed simulation of photo-electric absorption

1. For compounds and mixtures, the first step is to sample the atomic number of the element the photon is interacting with. If Z_i denotes the atomic number of the i'th element in the molecule, p_i its stoichiometric index, and $\sigma_{\rm ph}(k, Z)$ the photo-electric absorption cross section for a photon of energy k by an element with atomic number Z, then the probability $w_i(k)$ that the photon is absorbed by the element Z_i is

$$w_i = \frac{p_i \sigma_{\rm ph}(k, Z_i)}{\sum p_i \sigma_{\rm ph}(k, Z_i)} \ . \qquad (2.3.1)$$

The sampling of the element therefore requires the knowledge of all elemental photo-absorption cross sections at run time, not just the total photo-absorption cross sections that comes from the PEGS data set. To minimize the amount of additional data required, we use fit formulas for the $\sigma_{\rm ph}(k, Z_i)$'s which are accurate to within 1-2% and have the form

$$\begin{aligned} \sigma_{\rm ph}(k, Z) &= \frac{A_K(Z)}{k} + \frac{B_K(Z)}{k^2} + \frac{C_K(Z)}{k^{7/2}} + \frac{D_K(Z)}{k^4} \ , \quad \text{if } k \geq U_K(Z) \qquad (2.3.2) \\ &= \exp\left[A_j(Z) + B_j(Z)t + C_j(Z)t^2 + D_j(Z)t^3\right] \ , \quad \text{else if } k \geq U_j(Z) \end{aligned}$$

where $t = \ln k$ and where $U_K(Z)$ is the K-shell binding energy and $U_j(Z)$ binding energies for shells other then the K-shell. We have obtained The coefficients $A_K, B_K, ...$ and $A_j, B_j, ...$ by fitting the photo-absorption cross sections from the `XCOM` program [37] and are found in the file `photo_cs.data` (only shells with a binding energy above 1 keV are included). The algorithm to select the element that absorbs the incident photon is then:

1.1 Calculate all $\sigma_{\rm ph}(k, Z_i)$ and their sum

1.2 Pick a random number r_1

1.3 In a loop over the number of elements, calculate $r_1 = r_1 - w_i$, if $r_1 \leq 0$ exit the loop and take Z_i as the element interacting with the photon.

Note that, at least in principle,

$$\sum p_i \sigma_{\rm ph}(k, Z_i) = \sigma_{\rm ph}(k)$$

where $\sigma_{\rm ph}(k)$ is the photo-absorption cross section for the material under consideration. This cross section is interpolated using the PEGS supplied data in the photon transport routine in order to determine the interaction type. One could make the element selection algorithm more efficient by employing

1.1' Pick a random number r_1

1.2' Set $i = 1$

1.3' Calculate $\sigma_{\rm ph}(k, Z_i)$ and $r_1 = r_1 - p_i \sigma_{\rm ph}(k, Z_i)/\sigma_{\rm ph}(k)$.

1.4' If $r_1 > 0$ and i less then the number of elements, then $i = i + 1$, go to step 1.3'.

1.5' Deliver i.

as this saves the evaluation of one or more $\sigma_{\rm ph}(k, Z_i)$ (especially if elements are ordered by decreasing probability for photo-absorption prior to the actual simulation). We have not implemented this more efficient algorithm as the photo-absorption cross section interpolated using the PEGS supplied data becomes inaccurate around absorption edges. This potential improvement is left for future releases of the system (which are anticipated to not make use of PEGS data sets).

2. Once the absorbing element is determined, the shell with which the interaction takes place has to be sampled. The probability ν_j that the photon is absorbed by the j'th shell is close to energy independent for the K-shell but depends on the incident photon energy for other shells. This means that shell-wise photo-absorption cross sections would be required to be available at run time in order to sample the shell if one wanted to perform a complete modeling of the photo-absorption process. In this release of the EGSnrc system we use instead energy independent interaction probabilities ν_j which are determined as follows. If $\phi(k)$ is the fluence of the photon radiation field, the number of photo-absorption events by the element Z per unit volume, N, is given by

$$N = n(Z) \int {\rm d}k \phi(k) \sigma_{\rm ph}(k, Z) \tag{2.3.3}$$

where n is the density of scattering centers. The number of photo-absorptions by the shell j of this element per unit volume is

$$N_j = n(Z) \int {\rm d}k \phi(k) \sigma_{{\rm ph},j}(k, Z) \tag{2.3.4}$$

where $\sigma_{{\rm ph},j}(k, Z)$ is the photo-absorption cross section of the j'th shell. The ratio of N_j to N is the average interaction probability for absorption by the j'th shell, ν'_j, for the radiation field described by $\phi(k)$,

$$\nu'_j = \frac{N_j}{N} = \frac{\int {\rm d}k \phi(k) \sigma_{{\rm ph},j}(k, Z)}{\int {\rm d}k \phi(k) \sigma_{\rm ph}(k, Z)} \tag{2.3.5}$$

The actual quantity used in the simulation of photo-electric absorption is the probability ν_j that the photon is absorbed by the shell j if it was not absorbed by one of the shells $1 \cdots j-1$, which is given by

$$\nu_j = \frac{\int \mathrm{d}k \phi(k) \sigma_{\mathrm{ph},j}(k,Z)}{\sum_{i=j}^{N_{\mathrm{sh}}} \int \mathrm{d}k \phi(k) \sigma_{\mathrm{ph},i}(k,Z)} \tag{2.3.6}$$

where N_{sh} is the total number of shells. To calculate ν_j one needs $\phi(k)$, a quantity that is not known (but intended to be calculated by the Monte Carlo simulation). Nevertheless, one could calculate ν_j by making a guess about the photon fluence, this approach is the basis for the generation of group interaction coefficients for discrete ordinate methods (see *e.g* the book by Lewis and Miller [38]). Figure 6 shows the

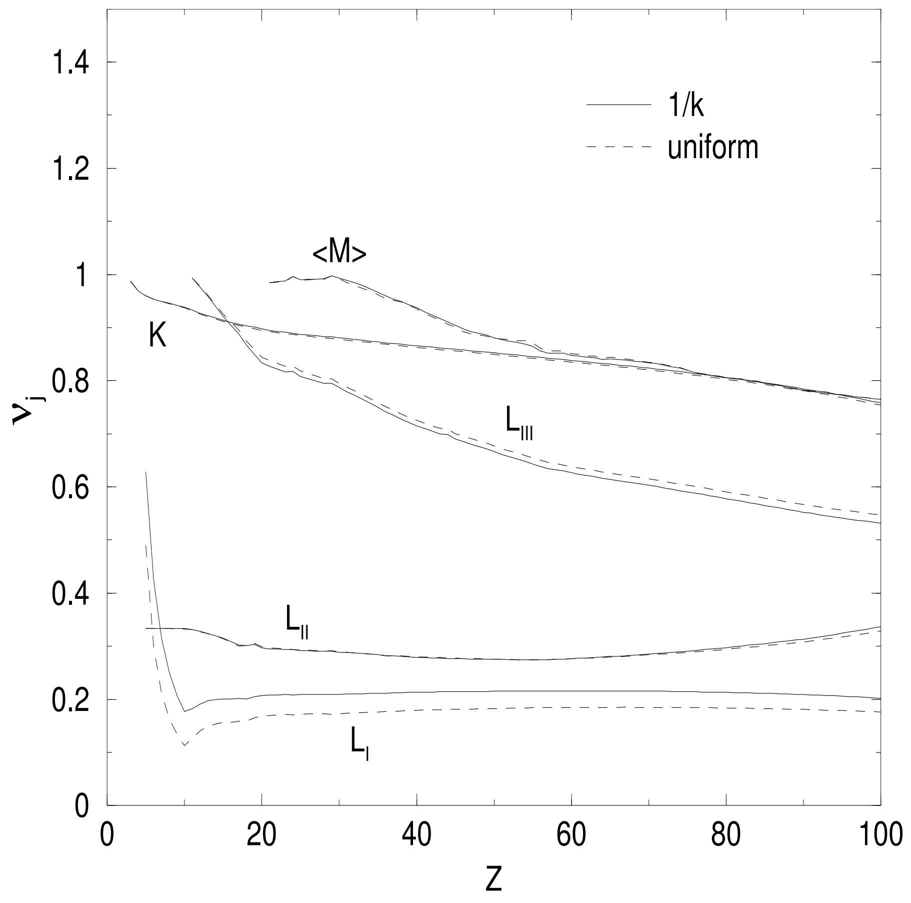


Figure 6: Interaction probabilities ν_j for different shells as calculated from Eq. (2.3.6) using $\phi(k) = 1/k$ (solid lines) or $\phi(k) = \mathrm{const}$ (dashed lines).

interaction probabilities ν_j of the K, L_I, L_{II} and L_{III} shells and an "average" M shell (see next paragraph) as a function of the atomic number Z for $\phi(k) = \mathrm{const}$ (dashed line) and $\phi(k) = \mathrm{const}/k$ (solid line). Shell-wise photo-absorption cross section from the Evaluated Photon Data Library (EPDL) [39] were used to generate this figure, the upper limit of the k-integration was set to 1 MeV. The dependence of the ν_js on the weighting function $\phi(k)$ is negligible except for the L_I shell where the difference is of the order of 10%. This fact was the motivation for using the energy independent interaction probabilities ν_j, a better approach is scheduled for a future release of the system.

The use of an interaction probability for an "average" M-shell is motivated by the fact that relaxation transitions from and to M-shells are treated in an average way, see section 2.3. Given the definition of the ν_j's, $\nu_{\langle M\rangle}$ is

$$\nu_{\langle M\rangle} = 1 - \prod(1 - \nu_{M_i}) \tag{2.3.7}$$

where the product runs over the number of M-sub-shells available for the element Z (up to 5). The interaction probabilities $\nu_K, \nu_{L_I}, \nu_{L_{II}}, \nu_{L_{III}}$ and $\nu_{\langle M\rangle}$, calculated with the $1/k$ weighting function, are stored in the file `photo_relax.data` and read in by the subroutine `edgset` which is called from `HATCH`.

With all this, the algorithm for selecting the shell that absorbs the incident photon is as follows:

2.1 Determine the inner-most shell j that has a binding energy lower than the incident photon energy and pick a random number r_2

2.2 If $r_2 < \nu_j$ or $j > \langle M\rangle$ [4], then deliver j

2.3 Set $r_2 = (1 - r_2)/(1 - \nu_j)$, $j = j + 1$, go to step 2.2

3. Once the element and its shell absorbing the photon is determined, a photo-electron with the kinetic energy $k - U_j(Z)$ is set-up, where $U_j(Z)$ is the binding energy of the selected shell. The vacancy created is treated in the routine (`relax`, see section 2.3), this significant change compared to EGS4 is motivated by the fact that shell vacancies are created also in other processes, *e.g.* Compton scattering (see section 2.2.2). The sampling of the photo-electron direction is discussed in section 2.2.3.iii.

2.2.3.ii Simplified simulation of photo-electric absorption

If the flag `IEDGFL` for the current region is set to zero, a simplified simulation of the photo-absorption process is undertaken. This simplified treatment consists of one step:

1. Set-up a photo-electron with a kinetic energy k

There are several reasons which motivated us to change the logic compared to the current EGS4 version (where the K-shell binding energy is always subtracted):

- The treatment is greatly simplified
- If the flag `IEDGFL` is set to zero, it is reasonable to assume that the detailed calculation of the spread of energy released in photo-absorption events is not important for the situation under investigation

[4] The outer-most shell treated is the M shell, if the photon was not absorbed by it, it is assumed that it is absorbed by the N shell.

- The effect of the production of relaxation particles in the de-excitation cascade following photo-electric absorption is to spread out the binding energy around the point of interaction. By giving the photo-electron the entire incident photon energy, this effect is at least partially simulated.

2.2.3.iii Photo-electron direction

The behavior of the sampling of the photo-electron direction is controlled by the switch `IPHTER` (included in the `COMIN/EDGE` common block) which set on a region by region basis. If set to zero, the photo-electron "inherits" the direction of the incident photon. If set to non-zero (the default selection), the direction is sampled from the Sauter distribution [40]. The implementation as discussed in detail in Ref. [41] is adopted. For completeness, we give a brief summary here.

Sauter's distribution in the polar angle $\mu = \cos\theta$ with respect to the incident photon direction may be cast in the form [41]

$$f(\mu)\mathrm{d}\mu = \frac{1-\mu^2}{(1-\beta\mu)^4}\left[1+\kappa(1-\beta\mu)\right]\mathrm{d}\mu \tag{2.3.8}$$

where β is the electron's velocity in units of the speed of light and

$$\kappa = \frac{\gamma}{2}(\gamma-1)(\gamma-2)\ , \quad \gamma = \frac{1}{\sqrt{1-\beta^2}} \tag{2.3.9}$$

This equation can be sampled by generating candidate μ values from

$$g(\mu) = \frac{1}{2\gamma^2(\kappa+\gamma)^2}\,\frac{1+\kappa(1-\beta\mu)}{(1-\beta\mu)^2}\ , \tag{2.3.10}$$

which is a normalized PDF for μ, and using

$$h(\mu) = \frac{\gamma+1}{2\gamma}\,\frac{1-\mu^2}{1-\beta\mu} \tag{2.3.11}$$

which is always positive and has a maximum of unity, as a rejection function. To generate μ values from $g(\mu)$, one uses

$$\mu = \frac{1}{\beta}\left[1-\left(\sqrt{\left(\kappa+\frac{1}{1+\beta}\right)^2+4\beta\gamma^2(\kappa+\gamma^2)r_1}-\kappa\right)^{-1}\right] \tag{2.3.12}$$

where r_1 is an uniformly distributed random number.

It is worth noticing that, strictly speaking, Sauter's distribution is valid only for the K-shell and also derived from an extreme relativistic approximation. The treatment of the photo-electron angular distribution is therefore left as a macro `$SELECT-PHOTOELECTRON-DIRECTION;` and can therefore be replaced, if the user has a better approach.

2.2.4 Coherent (Rayleigh) scattering

Originally, EGSnrc "inherited" the treatment of the coherent photon scattering process from EGS4 [11]. This means that the total coherent scattering cross sections from Storm and Israel [21] and the atomic form factors $F_T(q,Z)$ from Hubbel and Øverbø [42] are used by default. The form factors for molecules are calculated from the independent atom approximation, *i.e.*

$$[F_T(q)]^2 = \sum p_i [F_T(q,Z_i)]^2 \tag{2.4.1}$$

where p_i is the stoichiometric index of the i'th element, Z_i its atomic number and q the momentum transfer when a photon with an energy k is scattered by an angle θ,

$$q = k\sqrt{\frac{1-\cos\theta}{2}} \ . \tag{2.4.2}$$

The coherent scattering cross section, differential in the photon angle $\Omega = (\theta,\phi)$ is

$$\frac{\mathrm{d}\sigma_{\mathrm{R}}}{\mathrm{d}\Omega} = \frac{r_0^2}{2}(1+\cos^2\theta)[F_T(q)]^2 \ . \tag{2.4.3}$$

This equation is sampled by re-writing it in terms of q^2,

$$\frac{\mathrm{d}\sigma_{\mathrm{R}}}{\mathrm{d}q^2} = \frac{4\pi r_0^2}{k^2} A(q^2_{\mathrm{max}}) \frac{1+\cos^2\theta}{2} \frac{[F_T(q)]^2}{A(q^2_{\mathrm{max}})} \tag{2.4.4}$$

where

$$A(q^2) = \int_0^{q^2} \mathrm{d}q'^2 [F_T(q')]^2 \tag{2.4.5}$$

and $q_{\mathrm{max}} = k$ is the maximum possible momentum transfer, and using $[F_T(q)]^2/A(q^2_{\mathrm{max}})$ as a PDF for q^2 and $(1+\cos^2\theta)/2$ as a rejection function.

The actual sampling, accomplished in the macro `$RAYLEIGH-SCATTERING;`, is performed only if the switch `IRAYLR` (included in the `COMIN/MISC` common block–see Section 3.3) is set to non-zero for the current region. By default, `IRAYLR` is set to zero in the `block data` subprogram. The motivation for this choice is the fact that the `$RAYLEIGH-SCATTERING;` macro requires the function $A(q^2)$ to be included in the PEGS data set. This additional data is only included, if the user specifically requested it from PEGS.

We recommend the Rayleigh scattering option to be used for low energy calculations (say, below 1 MeV). It is worth noticing that inclusion of coherent scattering without the use of the bound Compton scattering option (see section 2.2.2) results in too much photon scattering, binding effects for incoherent scattering should therefore always be turned on if the Rayleigh option is used.

2.2.4.i New coherent scattering angular sampling

The original EGS4 sampling algorithm produces a step-like averaging artifact at large angles, which increases with energy. This is shown by the symbols in Figure 7, for 20

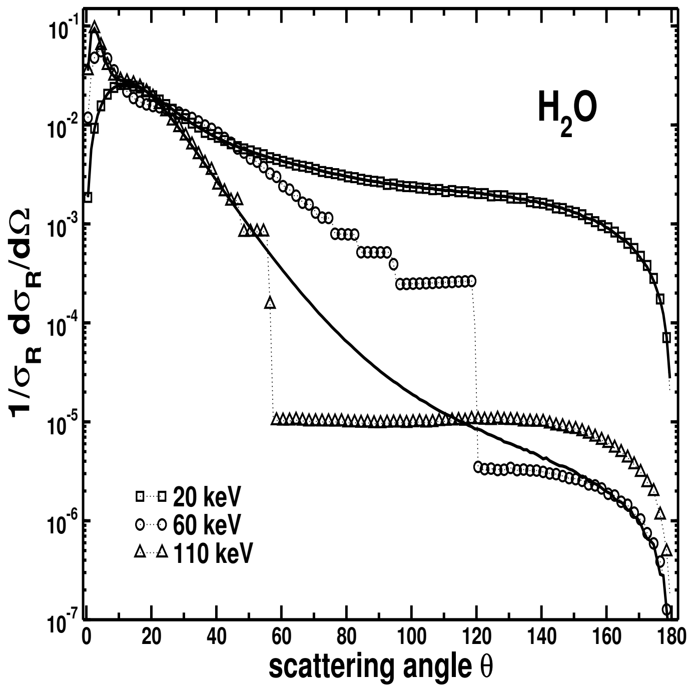


Figure 7: Angular distribution of coherently scattered photons for 20 keV, 60 keV and 110 keV photon beams in water. Symbols represent the original EGS4 sampling algorithm and the solid lines the new alias sampling.

keV, 60 keV and 110 keV photon beams in water. However, this artifact has no practical consequences, unless one is interested in the few photons scattered at large angles.

The default angular sampling algorithm for coherent scattering was replaced in 2007 for an alias-sampling algorithm that properly samples $[F_T(q)]^2/A(q^2_{\max})$ removing this artifact as shown by the solid lines in Figure 7 for 20 keV and 110 keV. The coherent scattering form factors are now read directly either from `$HEN_HOUSE/pegs4/pgs4form.dat` (default atomic form factors, see section 2.2.4 above) or from user-supplied form factor files expected to be in the `$HEN_HOUSE/data/molecular_form_factors/` directory (see section 2.2.4.ii below for more details). This means that one can now switch on coherent (Rayleigh) scattering, even if there is no Rayleigh data in the PEGS4 data set.

The old sampling macro is now named `$OLD_RAYLEIGH-SCATTERING;`. If one is interested in reproducing old calculations or just want to use the old sampling, this macro could be renamed back to `$RAYLEIGH-SCATTERING;`. Please note that in this case a PEGS4 file is required with the needed interpolation coefficients and the input key `Photon cross sections` in the `MC transport parameters input block` must be set to `PEGS4` (see section 6.1.1, page 219).

2.2.4.ii Custom atomic and molecular form factors

The user now has the option of supplying their own custom atomic or molecular form factors for determining Rayleigh cross sections. These are specified through the `iray_ff_media` (medium names) and `iray_ff_file` (files containing form factor data) variables contained in the COMIN/RAYLEIGH_INPUTS common block. A number of measured molecular form factors from the work by Peplow and Verghese [43] are distributed with EGSnrc and can be found in the $HEN_HOUSE/data/molecular_form_factors directory.

To enable this option, one needs to set the input key Rayleigh scattering to custom in the MC transport parameters input block. An example of such an input is shown below:

```
:start MC transport parameter:
  .
  .
  .
  Rayleigh scattering=  custom
  ff media names = BREASTICRU512 \
                   FATICRU512 \
                   MUSCLEICRU512 \
                   KAPTONICRU512
  ff file names = /ff_file_location/mff_breast_tissue.dat \
                  /ff_file_location/mff_pork_fat.dat \
                  /ff_file_location/mff_pork_muscle.dat \
                  /ff_file_location/mff_kapton.dat

:stop MC transport parameter:
```

Note that since there is no check, the user must make sure that the medium name and the form factor file correspond to the same material.

2.2.5 Changing photon cross sections

In the current version of EGSnrc, the user has the option to use photon cross section data other than the default XCOM data. To do this, the user must set the character variable photon_xsections (part of the COMIN/MEDIA common block) to the name of the cross section data to use. Cross sections included with the EGSnrc distribution are: 1) the Evaluated Photon Data Library[44] (set photon_xsections= "epdl") 2) XCOM data from Berger & Hubbell[37] (set photon_xsections= "xcom" – also the default in the absence of any input) and 3) Storm & Israel[21] (set photon_xsections= "si"). For information on how to set the variable photon_xsections, the reader is referred to section 3.4.2 (page 128).

The user can also supply their own photon cross section data. This can be accomplished by placing data files named xxxx_photo.data, xxxx_pair.data, xxxx_triplet.data and xxxx_rayleigh.data in the $HEN_HOUSE/data folder, where xxxx stands for any string different from epdl, xcom or si. These cross sections will then be used if the variable photon_xsections is set to xxxx. The format of the ASCII data files is very simple: for each element between 1 and 100 one has to provide the number N of data points in the tabulation followed by N pairs of the logarithm of the photon energy in MeV and the logarithm of the cross section in barn (see also one of the data files provided with EGSnrc).

Since 2006 EGSnrc was modified to ignore the photon data in the PEGS4 file, re-initializing all interpolation tables to utilize the maximum number of interpolation bins availble. In this way, the user can minimize the inaccuracies around atomic edges by replacing $MXGE with a huge number in their user code. Starting with the 2011 release (V4 release 2.3.2), the use of PEGS4 photon data has been re-enabled for backwards compatibility. To achieve this, the user must set the character variable `photon_xsections` to `PEGS4` (case insensitive). See section 6.1.1 (page 219) for more information on this topic.

It is worth noting that the total cross section for Compton scattering is normally obtained from the theoretical differential cross section being used (Klein-Nishina or RIA, see section 2.2.2) by integrating over the kinematically allowed range instead of being taken from tabulations. This behavior can be changed by the user as described in section 2.2.2).

2.3 Atomic Relaxations

Excited ions, produced by the interaction of photons and charged particles when they travel through matter, relax to their ground state by migration of the initial vacancy to outer shells via the emission of characteristic X-rays and/or Auger or Coster-Kronig electrons (see *e.g.* Ref. [45]).

In the current standard version of EGS4, only K-shell relaxations following photo-electric absorption via the emission of K_α and K_β fluorescence are treated and are intrinsically associated with the `PHOTO` routine. An extension to include L-shell fluorescence was developed by the KEK group [7] and is available for use with the EGS4 system.

In EGSnrc we have extended the treatment of atomic relaxations to include higher shells as well as the production of Auger and Coster-Kronig electrons. With these extensions, the treatment within the `PHOTO` routine has become unpractical. In addition, the relaxation cascade is a separate process, it can be initiated after any photon or electron interaction that has produced an inner shell vacancy. The most general approach for treating excited atoms or ions would have been to define a separate particle type, an excited atom or ion, and to put such particles on the stack whenever they are produced. Such an approach would have been too a large departure from the EGS4 logic and potentially render many user codes unusable. We have therefore abandoned this idea and decided to treat relaxations in a separate routine (`relax`), which is called whenever an inner shell vacancy is created. In this release of EGSnrc, such vacancies can be created in photo-absorption events (see section 2.2.3) and in Compton scattering events (see section 2.2.2). It is anticipated that the next release of the system will include the explicit modeling of inner shell ionizations by electron or positron impact.

The de-excitation cascade is a complex process, there are hundreds of possible transitions for high-Z elements. A complete treatment goes beyond the scope of a general purpose code for the simulation of electron and photon transport such as EGSnrc. In addition, we consider 1 keV to be the lowest limit for the applicability of the code. We have therefore imposed a lower limit of 1 keV on the relaxation process, *i.e.* only vacancies in shells with binding energies above 1 keV are treated.[5] If we then take into account that only elements with $Z \geq 52$ have M-shells with binding energies above the limit of 1 keV and that M-shells have

[5]Another motivation for imposing this limit is the fact that uncertainties on transition probabilities rapidly increase with decreasing binding energies, they are perhaps 10 or 20% even for L-shells

binding energies less than 10 keV for all elements (the M_I binding energy for lead is 3.8 keV and 7.2 keV for Einsteinium [45]), it is a reasonable approach to model transitions from and transitions to M-shells in an "average" way. There is of course no unique procedure to set the binding energies of the "average" M-shells for the elements, $U_{\langle M\rangle}(Z)$, to be used in the de-excitation cascade. Assuming that for most applications K to M transitions are more important than L to M or M to a lower shell, we have defined $U_{\langle M\rangle}(Z)$ to be the weighted average of the binding energies $U_{M_j}(Z)$ of the element Z with weights given by the K to M_j transition probabilities ν_{KM_j},

$$U_{\langle M\rangle}(Z) \equiv \frac{\sum \nu_{KM_j} U_{M_j}}{\sum \nu_{KM_j}} \tag{3.0.6}$$

For instance, $U_{\langle M\rangle}$ determined by this procedure using transition probabilities from the Evaluated Atom Data Library (EADL) [45] for lead is 3.1 keV, the M_I binding energy is 3.8 keV and the M_V binding energy 2.5 keV. A simple averaging with the occupation numbers would result in an average M-shell binding energy of 2.9 keV. If distinguishing between any of the above numbers is important for your application, EGSnrc is most likely not the most appropriate simulation package for your purposes.

Having said all this, it is apparent that N shells are also treated in an average way. The only elements with "average" N-shell binding energies above 1 keV are those with $Z > 95$.

In an implementation consistent with the overall logic of the EGS system, the relaxation algorithm should put all particles produced in the course of the de-excitation cascade on the particle stack, they would then be discarded in the `PHOTON` or `ELECTR` routines if their energies were below the specified transport threshold energies. It is easy to see that such an approach may become extremely wasteful if the transport threshold energies are large compared to the lower limit of 1 keV for the de-excitation cascade. We have therefore decided to stop the relaxation process for vacancies with binding energies less then $E_{\rm min}$,

$$E_{\rm min} = {\rm Max}\{1~{\rm keV},~~{\rm Min}\{\texttt{PCUT}, \texttt{ECUT} - m\}\} \tag{3.0.7}$$

and to score their energy locally. In addition, the energy of photons or electrons that are below the thresholds are also deposited locally, even if they were produced in transitions from vacancies with binding energies above $E_{\rm min}$. The total sub-threshold energy is collected in the variable `EDEP`, which is in the `COMON/EPCONT/`, and made known to the user via an `IARG=4` call to the scoring routine.

To facilitate the handling of the relaxation cascade, we define a shell number n for each of the shells treated. K-shells have $n = 1$, L_I through L_{III} have $n = 2$ to 4, $\langle M\rangle$ corresponds to $n = 5$, $\langle N\rangle$ to $n = 6$, all other shells have $n = 7$. A list of possible transitions, L_n, is associated with each shell,

$$L_n = \{(\nu_1, s_1), (\nu_2, s_2), \cdots, (\nu_{k_n}, s_{k_n})\} \tag{3.0.8}$$

where k_n is the number of possible transitions for the shell of type n and ν_i the transition probabilities for a transition into final state s_i. The final states s_i are defined as follows:

$$s_i = \begin{cases} n_i & , \quad \text{for fluorescent transitions} \\ 10 + n_i & , \quad \text{for Coster-Kronig transitions} \\ 100 n_{i,1} + n_{i,2} & , \quad \text{for Auger transitions} \end{cases} \tag{3.0.9}$$

where n_i or $n_{i,1}$ and $n_{i,2}$ are the shell numbers of the new vacancies created in fluorescent and Coster-Kronig or Auger transitions. Table 1 summarizes all transitions handled in the current version of the relaxation routine.

In addition, we define a "vacancy stack" which holds all vacancies at a given stage of the relaxation cascade.

With these definitions in place, the simulation of the relaxations cascade becomes fairly simple:

1. Put the initial vacancy in the "vacancy stack", set the "vacancy stack" counter m to 1
2. If $m = 0$, return control to the calling routine
3. Take the top vacancy, to be denoted by n_i in the following, from the "vacancy stack", reduce m by 1
4. If $U_{n_i} < E_{\min}$, then `EDEP` = `EDEP` + U_{n_i}, go to step 2
5. Pick a random number r, set $j = 1$
6. If $r \le \nu_j$, go to step 8
7. $r = (1 - r)/(1 - \nu_j)$, $\quad j = j + 1$, go to step 6
8. j is the number of the selected transition,
   - 8.1 if $s_j < 10$, then the new vacancy is in shell $n_j = s_j$, put it on the "vacancy stack", increase m by one, produce a fluorescent photon with energy $E = U_{n_i} - U_{n_j}$. If $E <$ `PCUT`, then `EDEP` = `EDEP` + E, else, select the photon direction uniformly and put it on the EGSnrc particle stack
   - 8.2 else if $s_j < 100$, then the new vacancy is in shell $n_j = s_j - 10$, put it on the "vacancy stack", increase m by one, produce a Coster-Kronig electron with kinetic energy $E = U_{n_i} - U_{n_j}$. If $E <$ `ECUT` $- m$, then `EDEP` = `EDEP` + E, else, select the electron direction uniformly and put it on the EGSnrc particle stack
   - 8.3 else, the two new vacancies are $n_{j,1} = (s_j \bmod 100)$ and $n_{j,2} = s_j - 100\ n_{j,1}$, put them on the "vacancy stack", increase m by two, produce an Auger electron with a kinetic energy $E = U_{n_i} - U_{n_{j,1}} - U_{n_{j,2}}$. If $E <$ `ECUT` $- m$, then `EDEP` = `EDEP` + E, else, select the electron direction uniformly and put it on the EGSnrc particle stack
9. Go to step 2

Finally, we have defined for each of the steps 8.1 to 8.3 new calls to the routine `AUSGAB` with arguments `IARG` = 25 to 27. This gives the possibility for the user to take some actions with the relaxation particles, *e.g.*, set their `LATCH` variable to an appropriate value, or play Russian Roulette with them.

Table 1: Relaxation transitions handled by EGSnrc.

initial vacancy	shell code	transition	final state code
K	1	Fluorescent $K \to L_{II}$	3
		$K \to L_{III}$	4
		$K \to \langle M \rangle$	5
		$K \to \langle N \rangle$	6
		Auger $K \to L_I L_I$	202
		$K \to L_{II} L_I$	302
		$K \to L_{II} L_{II}$	303
		$K \to L_{III} L_I$	402
		$K \to L_{III} L_{II}$	403
		$K \to L_{III} L_{III}$	404
		$K \to \langle M \rangle L_I$	502
		$K \to \langle M \rangle L_{II}$	503
		$K \to \langle M \rangle L_{III}$	504
		$K \to \langle M \rangle \langle M \rangle$	505
		$K \to \langle N \rangle L_I$	602
		$K \to \langle N \rangle L_{II}$	603
		$K \to \langle N \rangle L_{III}$	604
		$K \to \langle N \rangle \langle M \rangle$	605
		$K \to \langle N \rangle \langle N \rangle$	606
L_I	2	Fluorescent $L_I \to \langle M \rangle$	5
		$L_I \to \langle N \rangle$	5
		Coster-Kronig $L_I \to L_{II}$	13
		$L_I \to L_{III}$	14
		Auger $L_I \to \langle M \rangle \langle M \rangle$	505
		$L_I \to \langle N \rangle \langle M \rangle$	605
		$L_I \to \langle N \rangle \langle N \rangle$	606
L_{II}	3	Fluorescent $L_{II} \to \langle M \rangle$	5
		$L_{II} \to \langle N \rangle$	6
		Coster-Kronig $L_{II} \to L_{III}$	14
		Auger $L_{II} \to \langle M \rangle \langle M \rangle$	505
		$L_{II} \to \langle N \rangle \langle M \rangle$	605
		$L_{II} \to \langle N \rangle \langle N \rangle$	606
L_{III}	4	Fluorescent $L_{III} \to \langle M \rangle$	5
		$L_{III} \to \langle N \rangle$	6
		Auger $L_{III} \to \langle M \rangle \langle M \rangle$	505
		$L_{III} \to \langle N \rangle \langle M \rangle$	605
		$L_{III} \to \langle N \rangle \langle N \rangle$	606
$\langle M \rangle$	5	Fluorescent $\langle M \rangle \to \langle N \rangle$	6
		Auger $\langle M \rangle \to \langle N \rangle \langle N \rangle$	606

2.4 Simulation of electron transport

2.4.1 General discussion

The Condensed History (CH) Technique was introduced by M. Berger in the early sixties [15]. In this technique, many track segments of the real electron random walk are grouped into a single "step". The cumulative effect of elastic and inelastic collisions during the step are taken into account by sampling energy and direction changes from appropriate multiple scattering distributions at the end of the step. This approach is justified by the observation that the changes of the electron state in a single collision are usually very small and fails when this condition is not satisfied (at very low energies). Berger also defined two different implementations of the CH technique, which he called Class I and Class II schemes.

EGSnrc uses a Class II CH scheme for the simulation of electron transport. That is to say, bremsstrahlung processes that result in the creation of photons above an energy threshold k_c, and inelastic collisions that set in motion atomic electrons with kinetic energies above T_c, are both simulated explicitly and the secondaries transported. Such interactions are also referred to as "catastrophic" collisions. Sub-threshold inelastic and radiative events and elastic collisions are subject to grouping.

Every CH scheme must provide rules for selecting a path-length Δs_n, an energy loss ΔE_n, a change in direction from Ω_n to Ω_{n+1} and a spatial displacement $\Delta\vec{x}_n$ for each step of the CH random walk. For transport in heterogeneous geometries a boundary crossing algorithm is also required.

In the following we give a brief transport-theoretical treatment of a Class II CH technique which will help to establish the motivation and definitions of various quantities used in the CH implementation.

The transport equation for the electron fluence $\Phi(\vec{x},\vec{\Omega},E,t)$ in a medium with a density of scattering centres (atoms or molecules) $n(\vec{x})$ is given by

$$\frac{\mathrm{d}\Phi(\vec{x},\vec{\Omega},E,t)}{\mathrm{d}t} = S(\vec{x},\vec{\Omega},E,t) + vI[\Phi] \tag{4.1.1}$$

where

- $S(\vec{x},\vec{\Omega},E,t)$ is the number of electrons with energy E and velocity $\vec{v} = (v,\vec{\Omega})$ at a position $\vec{x}$ per unit volume, energy and solid angle interval, imparted per unit time by an external source or by photons interacting with the medium at time t. This latter part of the source term causes the coupling of the electron and photon fluences.

- $\mathrm{d}\Phi/\mathrm{d}t$ is the total time derivative, in the absence of external force fields it is given by

$$\frac{\mathrm{d}\Phi(\vec{x},\vec{\Omega},E,t)}{\mathrm{d}t} = \frac{\partial\Phi(\vec{x},\vec{\Omega},E,t)}{\partial t} + \vec{v}\vec{\nabla}\Phi(\vec{x},\vec{\Omega},E,t) \ . \tag{4.1.2}$$

- The collision term $I[\Phi]$ represents the changes of the particle fluence due to collisions with the atoms or molecules of the surrounding medium,

$$\begin{aligned} I[\Phi] = &- n(\vec{x})\Phi(\vec{x},\vec{\Omega},E,t)\int_0^E \mathrm{d}E' \int_{4\pi} \mathrm{d}\vec{\Omega}'\sigma(E,E',\Omega',\vec{x}) \\ &+ n(\vec{x})\int_E^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\vec{\Omega}'\Phi(\vec{x},\vec{\Omega}',E',t)\sigma(E',E'-E,\vec{\Omega}'\cdot\vec{\Omega},\vec{x}) \end{aligned} \tag{4.1.3}$$

where $\sigma(E,E',\Omega,\vec{x})$ is the microscopic cross section at a position $\vec{x}$ for all possible interactions in which an electron with energy E loses energy E' and changes its direction by $\vec{\Omega}$.

The collision integral $I[\Phi]$ represents the balance between particle losses and gains due to interactions described by the cross section $\sigma(E,E',\Omega,\vec{x})$.

In a Monte Carlo simulation the solution of Eq. (4.1.1) is usually obtained via

$$\Phi(\vec{x},\vec{\Omega},E,t) = \int_0^t \mathrm{d}t_0 \int_E^\infty \mathrm{d}E_0 \int \mathrm{d}\vec{x}_0 \int_{4\pi} \mathrm{d}\vec{\Omega}_0\ S(\vec{x}_0,\vec{\Omega}_0,E_0,t_0)\Phi_0(\vec{x},\vec{\Omega},E,t) \tag{4.1.4}$$

where $\Phi_0(\vec{x},\vec{\Omega},E,t)$ is a short hand notation for $\Phi(\vec{x}_0,\vec{\Omega}_0,E_0,t_0;\vec{x},\vec{\Omega},E,t)$ which is the solution of Eq. (4.1.1) for a source

$$S_0 = \delta^{(3)}(\vec{x}-\vec{x}_0)\delta^{(2)}(\vec{\Omega}-\vec{\Omega}_0)\delta(E-E_0)\delta(t-t_0) \tag{4.1.5}$$

i.e. a single particle set in motion at time t_0 with energy E_0, direction $\vec{\Omega}_0$ at a position $\vec{x}_0$. The integration of Eq. ((4.1.4)) is of course performed by a Monte Carlo technique:

1. Pick particles from the source $S(\vec{x},\vec{\Omega},E,t)$, denote their co-ordinates by $E_0,\vec{x}_0,\vec{\Omega}_0,t_0$.
2. Solve the transport equation for these particles by a Monte Carlo simulation (*i.e.* solve the transport equation for Φ_0)

Usually the Monte Carlo simulation is initiated by a single particle from an external source, which is put on a “particle stack”. In EGSnrc this is accomplished by a call to the subroutine `SHOWER`. Then, at each stage of the Monte Carlo simulation, the source corresponds to all particles currently on the stack.

Sometimes it is customary to use the macroscopic cross section Σ,

$$\Sigma(E,E',\Omega,\vec{x}) = n(\vec{x})\sigma(E,E',\Omega,\vec{x}) \tag{4.1.6}$$

which, when integrated over E' and Ω', represents the number of interactions per unit length for electrons with energy E. The atom or molecule density n is given by

$$n = \rho\frac{N_A}{M_A} = \frac{\rho}{uA} \tag{4.1.7}$$

where $N_A = 6.022045 \times 10^{23}$ mol^{-}1 is Avogadro constant, M_A the molar mass in mol^{-1}, $u = 1.0665655 \times 10^{-24}$ g is the atomic mass unit and A the relative atomic or molecular mass. Macroscopic cross sections Σ are thus obtained from microscopic cross sections σ by multiplication with one of the above factors.

For electrons, σ is the sum of the bremsstrahlung cross section $\sigma_{\rm brem}$, the cross section for inelastic collisions with atomic electrons, $\sigma_{\rm inel}$, and the elastic scattering cross section $\sigma_{\rm el}$:

$$\sigma(E,E',\Omega,\vec{x}) = \sigma_{\rm brem}(E,E',\Omega,\vec{x}) + \sigma_{\rm inel}(E,E',\Omega,\vec{x}) + \sigma_{\rm el}(E,\Omega,\vec{x})\delta(0) \tag{4.1.8}$$

where Dirac's δ function expresses the fact that elastic collisions are zero energy loss events. Positrons interact in addition via the annihilation process described by $\sigma_{\rm annih}$, this cross section enters only the third term of Eq. (4.1.1), however, as annihilation leads merely to particle loss (at least when looking just at the electron fluence, the corresponding "gain" term appears as a source term for the photon fluence).

In a Class II CH implementation the collision integral is divided into two parts $I_>[\Phi_0]$ and $I_<[\Phi_0]$. The former includes only interactions with change in energy larger than T_c (inelastic collisions) or k_c (bremsstrahlung), the latter only collision with energy change less than T_c or k_c. Elastic interactions, being a zero energy loss processes, are included in $I_<[\Phi_0]$. For brevity we will drop the $\vec{x}$ and time dependence of the various quantities in the following equations. We have for $I_>[\Phi_0]$

$$\begin{aligned} I_>[\Phi_0] = & \int\limits_{E+T_c}^{\infty} {\rm d}E' \int\limits_{4\pi} {\rm d}\vec{\Omega}' \Phi_0(\vec{\Omega}',E')\Sigma_{\rm inel}(E',E'-E,\vec{\Omega}\cdot\vec{\Omega}') \\ & + \int\limits_{E+k_c}^{\infty} {\rm d}E' \int\limits_{4\pi} {\rm d}\vec{\Omega}' \Phi_0(\vec{\Omega}',E')\Sigma_{\rm brem}(E',E'-E,\vec{\Omega}\cdot\vec{\Omega}') \\ & - \Phi_0(\vec{\Omega},E)\left[\Sigma_{\rm inel}^{\rm (tot)}(E) + \Sigma_{\rm brem}^{\rm (tot)}(E)\right] \end{aligned} \tag{4.1.9}$$

where $\Sigma_{\rm inel}^{\rm (tot)}(E)$ and $\Sigma_{\rm breml}^{\rm (tot)}(E)$ are the total macroscopic cross sections for "catastrophic" inelastic or bremsstrahlung collisions, respectively:

$$\Sigma_{\rm inel,brem}^{\rm (tot)}(E) = \int\limits_{T_c,k_c}^{E} {\rm d}E' \int\limits_{4\pi} {\rm d}\vec{\Omega}' \Sigma_{\rm inel,brem}(E,E',\Omega') \ . \tag{4.1.10}$$

The $I_<[\Phi_0]$ part reads

$$I_<[\Phi_0] = \int\limits_{4\pi} {\rm d}\vec{\Omega}' \left[\Phi_0(\vec{\Omega}',E)\Sigma_{\rm el}(E,\vec{\Omega}\cdot\vec{\Omega}') - \Phi_0(\vec{\Omega},E)\Sigma_{\rm el}(E,\Omega')\right] + I_{\Delta E}[\Phi_0] \tag{4.1.11}$$

where $I_{\Delta E}[\Phi_0]$ is the part of the sub-threshold collision term that is associated with energy loss:

$$I_{\Delta E}[\Phi_0] =$$
$$\int\limits_{E}^{E+T_c} \mathrm{d}E' \int\limits_{4\pi} \mathrm{d}\vec{\Omega}' \Phi_0(\vec{\Omega}', E') \Sigma_{\mathrm{inel}}(E', E'-E, \vec{\Omega}\cdot\vec{\Omega}') - \Phi_0(\vec{\Omega}, E) \int\limits_{0}^{T_c} \mathrm{d}E' \int\limits_{4\pi} \mathrm{d}\vec{\Omega}' \Sigma_{\mathrm{inel}}(E, E', \Omega') +$$
$$\int\limits_{E}^{E+k_c} \mathrm{d}E' \int\limits_{4\pi} \mathrm{d}\vec{\Omega}' \Phi_0(\vec{\Omega}', E') \Sigma_{\mathrm{brem}}(E', E'-E, \vec{\Omega}\cdot\vec{\Omega}') - \Phi_0(\vec{\Omega}, E) \int\limits_{0}^{k_c} \mathrm{d}E' \int\limits_{4\pi} \mathrm{d}\vec{\Omega}' \Sigma_{\mathrm{brem}}(E, E', \Omega') \; . \tag{4.1.12}$$

At this point, the usual approximation made is to assume that angular deflections in small energy loss collisions are either negligible or can be taken into account by modifying the elastic collision part in an appropriate way. To our knowledge, this approximation is made, in one way or another, in all general purpose condensed history codes available. With this approximation and after a change of the integration variables, Eq. (4.1.12) can be re-written as

$$\begin{aligned} I_{\Delta E}[\Phi_0] = & \int\limits_{0}^{T_c} \mathrm{d}E' \left[\Phi_0(\vec{\Omega}, E+E') \Sigma_{\mathrm{inel}}(E+E', E') - \Phi_0(\vec{\Omega}, E) \Sigma_{\mathrm{inel}}(E, E') \right] \\ & + \int\limits_{0}^{k_c} \mathrm{d}E' \left[\Phi_0(\vec{\Omega}, E+E') \Sigma_{\mathrm{brem}}(E+E', E') - \Phi_0(\vec{\Omega}, E) \Sigma_{\mathrm{brem}}(E, E') \right] \end{aligned} \tag{4.1.13}$$

where the cross sections not depending on Ω are integrated over all angles. If we now use a Taylor series expansions for the first terms in the square brackets,

$$\begin{aligned} \Phi_0(\vec{\Omega}, E+E') \Sigma_{\mathrm{inel,brem}}(E+E', E') \approx & \; \Phi_0(\vec{\Omega}, E) \Sigma_{\mathrm{inel,brem}}(E, E') \\ & + \frac{\partial}{\partial E} \left[\Phi_0(\vec{\Omega}, E) \Sigma_{\mathrm{inel,brem}}(E, E') \right] E' + \cdots \end{aligned} \tag{4.1.14}$$

the collision term $I_{\Delta E}[\Phi]$ simplifies to

$$I_{\Delta E}[\Phi_0] \approx \frac{\partial}{\partial E} \left[\Phi_0(\vec{\Omega}, E) L(E, T_c, k_c) \right] \tag{4.1.15}$$

where $L(E, T_c, k_c)$ is the restricted stopping power for threshold energies T_c and k_c,

$$\begin{aligned} L(E, T_c, k_c) &= L_{\mathrm{coll}}(E, T_c) + L_{\mathrm{rad}}(E, k_c) \\ L_{\mathrm{coll}}(E, T_c) &\equiv \int\limits_{0}^{T_c} \mathrm{d}E' \Sigma_{\mathrm{inel}}(E, E') E' \\ L_{\mathrm{radl}}(E, k_c) &\equiv \int\limits_{0}^{k_c} \mathrm{d}E' \Sigma_{\mathrm{brem}}(E, E') E' \end{aligned} \tag{4.1.16}$$

With all this, the transport equation can be written as

$$\begin{aligned}\frac{1}{v}\,\frac{\partial\Phi_0(\vec{x},\vec{\Omega},E,t)}{\partial t} &+ \vec{\Omega}\vec{\nabla}\Phi_0(\vec{x},\vec{\Omega},E,t) = S_0 - \Phi_0(\vec{x},\vec{\Omega},E,t)\left[\Sigma^{(\mathrm{tot})}_{\mathrm{inel}}(\vec{x},E)+\Sigma^{(\mathrm{tot})}_{\mathrm{brem}}(\vec{x},E)\right]\\ &+ \int\limits_{E+T_c}^{\infty}\mathrm{d}E'\int\limits_{4\pi}\mathrm{d}\vec{\Omega}'\Phi_0(\vec{x},\vec{\Omega}',E',t)\Sigma_{\mathrm{inel}}(\vec{x},E',E'-E,\vec{\Omega}\cdot\vec{\Omega}')\\ &+ \int\limits_{E+k_c}^{\infty}\mathrm{d}E'\int\limits_{4\pi}\mathrm{d}\vec{\Omega}'\Phi_0(\vec{x},\vec{\Omega}',E',t)\Sigma_{\mathrm{brem}}(\vec{x},E',E'-E,\vec{\Omega}\cdot\vec{\Omega}')\\ &+ \int\limits_{4\pi}\mathrm{d}\vec{\Omega}'\left[\Phi_0(\vec{x},\vec{\Omega}',E,t)\Sigma'_{\mathrm{el}}(\vec{x},E,\vec{\Omega}\cdot\vec{\Omega}')-\Phi_0(\vec{x},\vec{\Omega},E,t)\Sigma'_{\mathrm{el}}(\vec{x},E,\Omega')\right]\\ &+ \frac{\partial}{\partial E}\left[\Phi_0(\vec{x},\vec{\Omega},E,t)L(\vec{x},E,T_c,k_c)\right]\end{aligned} \tag{4.1.17}$$

where we have put back the position dependence of the fluence, stopping power and cross sections, and the prime on the elastic scattering cross section means that it includes in some way contributions from angular deflections due to sub-threshold energy loss processes.

We now define the variable s, called the path-length, which satisfies

$$\begin{aligned}\frac{\mathrm{d}t}{\mathrm{d}s} &= \frac{1}{v}\\ \frac{\mathrm{d}E}{\mathrm{d}s} &= -L(E,E_c,k_c)\ ,\quad \text{or}\\ s &= \int\limits_{E}^{E_0}\frac{\mathrm{d}E'}{L(E,E_c,k_c)}\end{aligned} \tag{4.1.18}$$

The approximation of Eq. (4.1.14), together with Eq. (4.1.18), is known as continuous-slowing-down approximation (CSDA). CSDA is used in EGSnrc (and also in EGS4) to describe sub-threshold energy loss processes. This is not a necessary approximation, and one could replace it, for instance, by the theory of Vavilov [46]. We have postponed the implementation of Vavilov energy loss straggling into EGSnrc until the implementation of more realistic small energy loss inelastic scattering cross sections (which obviously affect the treatment of the integrals in Eq. (4.1.13)).

Equation (4.1.17) can be solved by the following Monte Carlo algorithm:

1. Pick the next electron from the "particle stack", we denote its energy by E_0, its direction by $\vec{\Omega}_0$ and its position by $\vec{x}_0$.

2. Sample the energy E at which the next "catastrophic" interaction occurs from the probability distribution

$$P(E) = \exp\left(-\int\limits_{E}^{E_0}\mathrm{d}E'\left[\tilde{\Sigma}_{\mathrm{inel}}(E')+\tilde{\Sigma}_{\mathrm{brem}}(E')\right]\right) \tag{4.1.19}$$

where we have defined

$$\tilde{\Sigma}^{(\rm tot)}_{\rm inel,brem}(E) = \frac{\Sigma^{(\rm tot)}_{\rm inel,brem}(E)}{L(E,E_c,k_c)} \tag{4.1.20}$$

which are the total cross sections for bremsstrahlung or inelastic collisions for discrete interactions per unit energy loss.

3. Modify the particles position, direction, and energy, in such a way as to approximate as closely as possible the exact solution of the transport equation

$$\begin{aligned} \frac{\partial \Phi_0(\vec{x},\vec{\Omega},s)}{\partial s} \quad + \quad & \vec{\Omega}\vec{\nabla}\Phi_0(\vec{x},\vec{\Omega},s) = \\ & \int\limits_{4\pi} {\rm d}\vec{\Omega}' \left[\Phi_0(\vec{x},\vec{\Omega}',s)\Sigma'_{\rm el}(\vec{x},s,\vec{\Omega}\cdot\vec{\Omega}') - \Phi_0(\vec{x},\vec{\Omega},s)\Sigma'_{\rm el}(\vec{x},\Omega',s)\right] \end{aligned} \tag{4.1.21}$$

where the s dependence of all quantities is understood in the sense of Eq. (4.1.18).

4. Once at the interaction site, select the interaction type from the total interaction cross sections at the current position and for the current energy, sample energy and direction changes from the appropriate differential cross section, put all resulting particles on the stack and go to step 1.

5. Repeat steps 1-4 until the stack is empty or all energies have fallen below the specified threshold.

The procedure for positrons is similar but involves the annihilation cross section in addition to bremsstrahlung and inelastic collisions with atomic electrons.

After this discussion, it is clear that we need the following quantities and algorithms for a Class II condensed history simulation of electron and positron transport:

1. Restricted stopping powers due to sub-threshold processes. The collision part of the restricted stopping power deserves an extra paragraph, it is discussed in section 2.4.6, the radiative part of the restricted stopping power is discussed in association with the bremsstrahlung cross section (section 2.4.2).

2. Total and differential cross sections, as well as the associated sampling technique, for bremsstrahlung processes with an energy loss greater than k_c (section 2.4.2), inelastic collisions with energy loss larger than T_c (section 2.4.3), and positron annihilation (section 2.4.5)

3. Elastic scattering cross sections that take into account angular deflections due to sub-threshold inelastic collisions (section 2.4.7)

4. A procedure to sample the distance between discrete interactions on the basis of Eq. (4.1.19) (see section 2.4.11)

5. A procedure to calculate the path-length from a given change in energy or vice versa according to Eq. (4.1.18) (section 2.4.11)

6. A procedure for the approximate solution of Eq. (4.1.21) which describes the transport process between subsequent discrete events. This is perhaps the most difficult part of a Class II condensed history algorithm. It involves the construction of a multiple elastic scattering theory, which is necessary for modelling angular deflections, and an "electron-step" algorithm, which relates the spatial displacement to the path-length (and possibly multiple elastic scattering angle). These two aspects of the EGSnrc CH implementation are discussed in sections 2.4.8 and 2.4.9.

2.4.2 Bremsstrahlung

2.4.2.i Cross sections

The bremsstrahlung and pair production processes are cross-symmetric (*i.e.* the Feynman diagram for electron bremsstrahlung is obtained from Fig. 1 by flipping the incoming photon and outgoing positron lines) and therefore their cross sections closely related. In EGSnrc the treatment of the bremsstrahlung process is determined by the parameter `ibr_nist` which is in `COMIN/BREMPR/`. If `ibr_nist = 0` (this is the default), the EGS4 cross sections are employed, *i.e.*

- Coulomb corrected extreme relativistic cross sections above 50 MeV, as formulated in the article by Koch and Motz [47]
- First Born approximation Bethe-Heitler cross sections with an empirical correction factor below 50 MeV [47]

If `ibr_nist = 1`, the bremsstrahlung process is modelled according to the NIST bremsstrahlung cross section data base [48, 49] which is the basis for the radiative stopping powers recommended by the ICRU [50] and which is based on

- Coulomb corrected extreme relativistic cross sections above 50 MeV
- Partial wave analysis calculations by Tseng and Pratt [51] below 2 MeV
- Spline interpolations for 2 to 50 MeV

In addition, a more elaborate procedure for the contribution of atomic electrons to the bremsstrahlung process is employed.

If `ibr_nist = 2`, a new set of tabulations prepared at the NRC is employed. This set uses the nuclear bremsstrahlung cross sections from the NIST data base [48, 49] but replaces the electron-electron part of the interaction with exact calculations in the first Born approximation reported in Ref. [52].

The default bremsstrahlung cross section for an electron with a total energy E incident on an atom with atomic number Z, differential in the photon energy k, is

$$\begin{aligned}\frac{\mathrm{d}\sigma_{\mathrm{brem}}(E,Z)}{\mathrm{d}k} &= \frac{A'(E,Z)r_0^2\alpha Z(Z+\xi(Z))}{k}\left\{\left(1+\frac{E'^2}{E^2}\right)\left[\phi_1(\delta)-\frac{4}{3}\ln Z-4\tilde{f}_c(E,Z)\right]\right.\\ &- \left.\frac{2}{3}\frac{E'}{E}\left[\phi_2(\delta)-\frac{4}{3}\ln Z-4\tilde{f}_c(E,Z)\right]\right\} \qquad (4.2.1)\end{aligned}$$

where $E' = E - k$ is the electron energy after the emission of the photon, r_0 the classical electron radius, α the fine structure constant,

$$\delta = 136 Z^{-1/3} 2\Delta \ , \qquad \Delta = \frac{km}{2EE'} \ , \tag{4.2.2}$$

the functions $\phi_1(\delta), \phi_2(\delta), \xi(Z)$ and $\tilde{f}_c(E, Z)$ have the same definitions as for the pair production process (see section 2.2.1), and $A'(E, Z)$ is an empirical correction factor (see below). For compounds and mixture the cross section can be approximated in the same form with the replacements given by Eq. (2.1.10) in section 2.2.1 (page 29).

Also relevant for the condensed history simulation are the moments M_m,

$$M_m(E, Z; k_{\rm min}, k_{\rm max}) \equiv \int\limits_{k_{\rm min}}^{k_{\rm max}} {\rm d}k k^m \frac{{\rm d}\sigma_{\rm brem}(E, Z)}{{\rm d}k} \ . \tag{4.2.3}$$

$M_0(E, Z; k_c, T)$ is the total cross section for bremsstrahlung interactions by an electron with energy E (kinetic energy $T = E - m$) in a medium Z that produce photons with an energy above the threshold energy k_c. This cross section is required for sampling distances between subsequent "catastrophic" radiative events. $M_1(E, Z; 0, T)$, multiplied with the density of scattering centres (atoms or molecules), n, is the average energy lost to radiation per unit path-length, *i.e.* the radiative stopping power. $M_1(E, Z; 0, k_c)n$ is then the restricted radiative stopping power corresponding to k_c. With these definitions in place we can turn back to the discussion of the empirical correction factor $A'(E, Z)$. In the original EGS4 implementation $A'(E, Z)$ is based on the data provided in the article by Koch and Motz [47]. In Ref. [53] a correction factor $A'(E, Z)$ based on the ICRU-37 radiative stopping powers was implemented into the EGS4 data preparation package PEGS4, it is defined as

$$A'(E, Z) = \frac{M_1(E, Z; 0, T)}{M_1^{\rm NIST}(E, Z; 0, T)} \tag{4.2.4}$$

where $M_m^{\rm NIST}$ is defined in the same way as in Eq. (4.2.3) but the cross section is replaced by the NIST bremsstrahlung cross section. EGSnrc, having "inherited" the use of the PEGS4 package, has both options available. The selection is made via the parameter `IAPRIM` when generating the PEGS4 data set with `IAPRIM=0` corresponding to the original A' method and `IAPRIM=1` to the approach of Ref. [53]. The other two options available in EGSnrc is to use the NIST or NRC cross sections directly, this is accomplished by setting `ibr_nist=1` or `ibr_nist=2`. The result of doing so is

1. The total discrete bremsstrahlung cross section is calculated using 64-point Gauss-Legendre quadrature in subroutine `init_nist_brems` and the cross section interpolation coefficients coming from the PEGS4 data set modified accordingly

2. Alias-sampling tables are prepared from the NIST or NRC cross sections differential in the photon emission energy k (which are available only in a numerical form). These tables are then used at run-time to sample the photon energy

3. The contribution from sub-threshold bremsstrahlung processes to the restricted stopping power is NOT corrected. This introduces a slight inconsistency in the treatment

of of the bremsstrahlung process which is irrelevant if $k_c \ll T$ or if the restricted radiative stopping power is small compared to the restricted collision stopping power, or both. A self-consistent implementation is left for the next release of EGSnrc which will not rely on PEGS4 data sets.

Before we discuss the sampling of the photon energy from Eq. (4.2.1), we compare the default EGSnrc (and also EGS4) differential bremsstrahlung cross section (with `IAPRIM=1` which is default with EGSnrc but was an option with EGS4)) to the NIST data base for gold and incident electron kinetic energies of 10 keV, 100 keV, 1 MeV, 10 MeV, 50 MeV and 100 MeV in Fig. 8. The behaviour for other materials is qualitatively similar. As one can

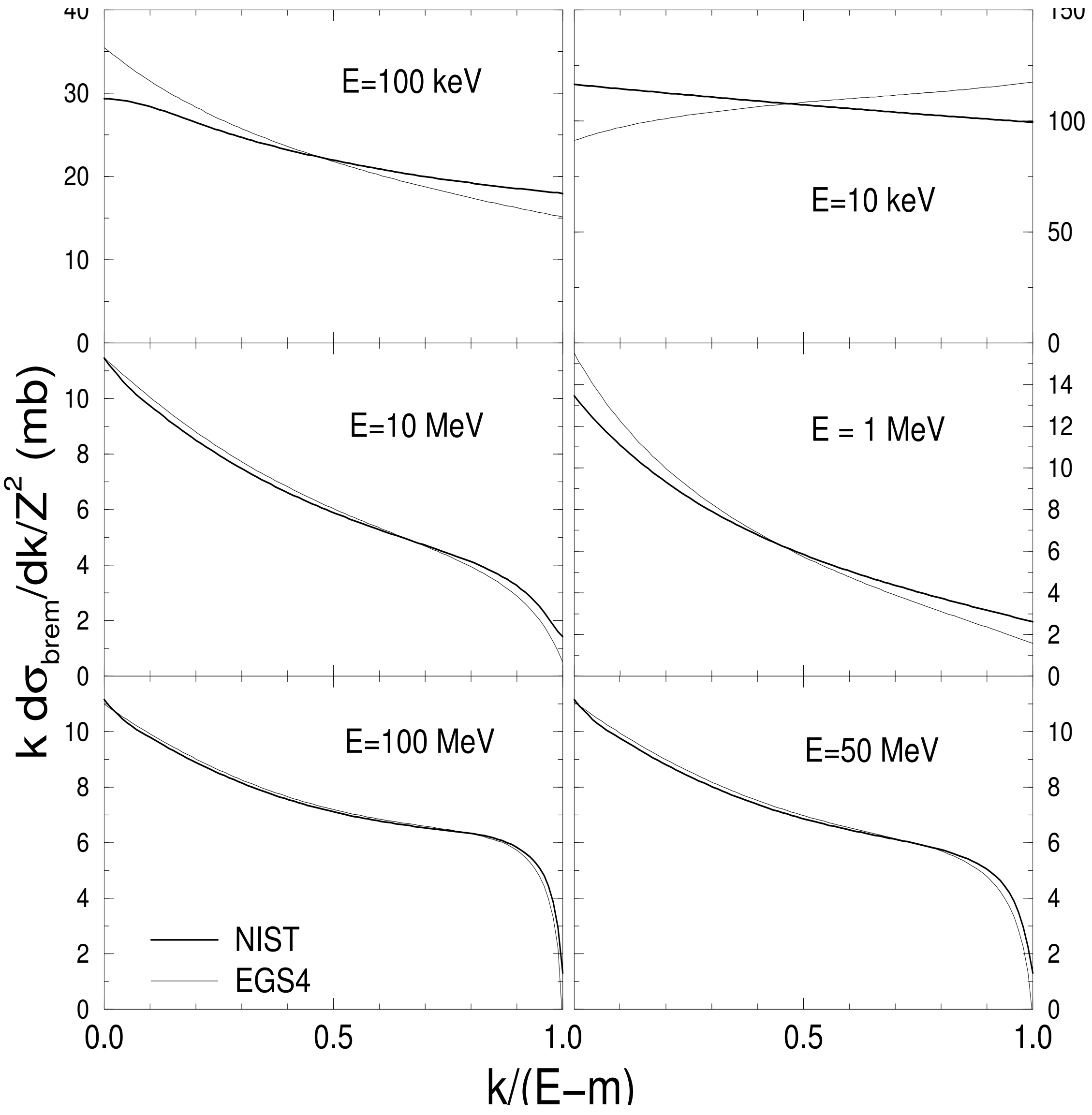


Figure 8: Differential bremsstrahlung cross sections for gold at various incident electron energies from the NIST data base [48, 49] (thick lines) compared to the default EGSnrc (and EGS4) cross sections (thin lines). In both cases `IAPRIM = 1`, which is the default in the EGSnrc system and was an option in the EGS4 system.

expect, the two cross sections are virtually identical at high energies, but there are significant differences at low energies (although the radiative stopping power is the same due to the use of A' from Eq. (4.2.4)).

2.4.2.ii Simulation of discrete bremsstrahlung events, photon energy

In the course of the re-work of the EGS4 sampling routines we have found an error in the sampling algorithm used in EGS4. The error, which was most likely not discovered before because it shows up only if the incident electron energy is not much larger than the threshold energy k_c, is demonstrated in Fig. 9. This figure compares the distribution sampled by the EGS4 routine `BREMS` (points) for 100 keV electrons in aluminum, expressed in terms of x,

$$x = \frac{\ln k/k_c}{\ln T/k_c} \, , \tag{4.2.5}$$

to the theoretically expected result (solid line). The threshold energy k_c was 10 keV (k_c is called `AP` in EGS4 and EGSnrc). This finding was a sufficient motivation to completely

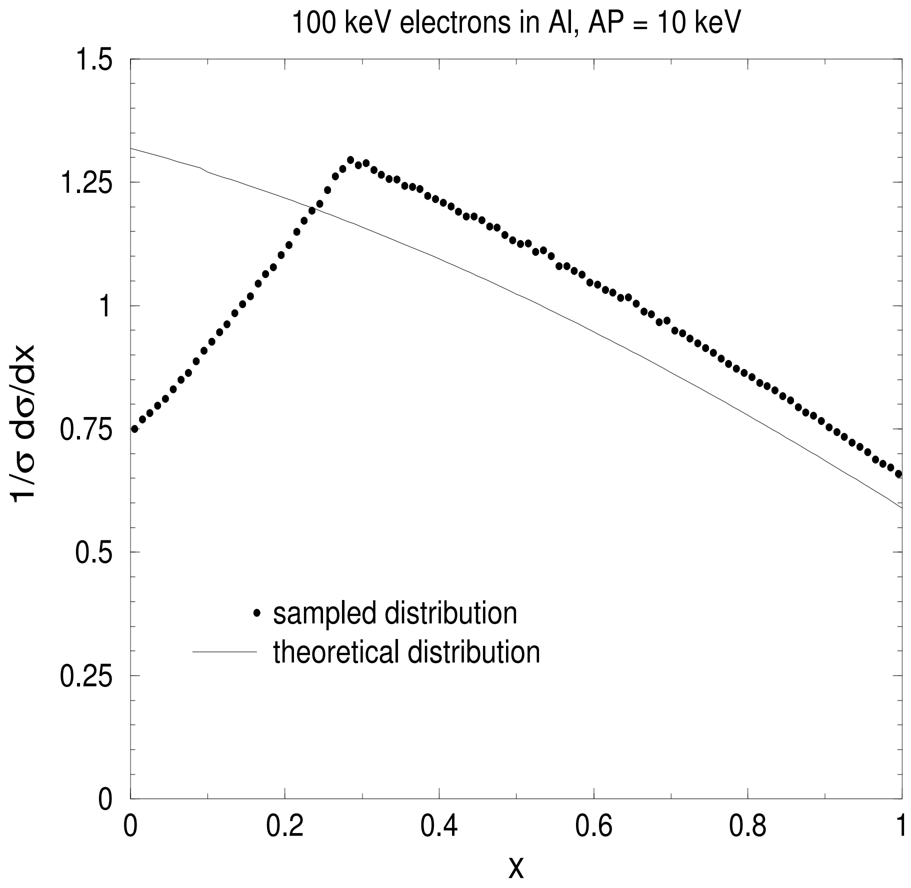


Figure 9: The distribution of photon energies sampled by the EGS4 routine `BREMS` (points) for 100 keV electrons in aluminum, expressed in terms of x, defined in Eq. (4.2.5), compared to the theoretical expectation.

recode the `BREMS` routine.

The most efficient algorithm for sampling photon energies on the basis of Eq. (4.2.1) appears to be the following: after a change of variables from k to x, we have

$$\begin{aligned}\frac{\mathrm{d}\sigma_{\mathrm{brem}}(E,Z)}{\mathrm{d}x} &= C\left\{\left(1+\frac{E'^2}{E^2}\right)\left[\phi_1(\delta)-\frac{4}{3}\ln Z-4\tilde{f}_c(E,Z)\right]\right. \\ &- \left.\frac{2}{3}\,\frac{E'}{E}\left[\phi_2(\delta)-\frac{4}{3}\ln Z-4\tilde{f}_c(E,Z)\right]\right\} \qquad (4.2.6)\end{aligned}$$

where C is a constant combining factors irrelevant for the sampling algorithm. Apart from a normalization constant, the function in the curled brackets, to be denoted in what follows with R, is the quantity $k\mathrm{d}\sigma_{\mathtt{brem}}/\mathrm{d}k/Z^2$, shown with the thin lines in Fig. 8. As can be seen, it is relatively flat and can therefore be employed as a rejection function in conjunction with a uniform sampling of x. R retains its absolute maximum[6], R_{max}, for $k=0$. It is given by

$$R_{\mathrm{max}} = 28.381 - \frac{4}{3}Z_V \qquad (4.2.7)$$

where Z_V is defined in Eq. (2.1.10) in section 2.2.1 and we have made use of Eq. (2.1.8) for the functions $\phi_1(\delta)$ and $\phi_2(\delta)$. The algorithm is then

1. Calculate $b = \ln T/k_c$ [7]
2. Pick two random numbers, r_1 and r_2
3. Set $k = k_c \exp(r_1 b)$, calculate R/R_{max}
4. If $r_2 > R/R_{\mathrm{max}}$, go to step 2
5. Deliver k

Figure 10 shows CPU times in μs necessary to sample one photon energy on a 500 MHz PIII computer using the new algorithm (solid line), the EGS4 algorithm (dotted line) and using the alias sampling technique in the case `ibr_nist` is set to 1. The threshold energy used was $k_c = 10$ keV and the material was aluminum. The precise amount of CPU time spent for sampling the photon energy is somewhat dependent on k_c and Z, but the qualitative behaviour remains the same for other values of k_c. Apart from being more accurate, the new algorithm is also more efficient. The CPU time for the alias sampling technique is energy independent, as one can expect. It is faster at lower energies but slower at high energies and so the use of the `ibr_nist=1` option is not meaningful above 50 MeV (where also the cross sections are identical). The small "waves" in the EGS4 curves are due to the technique employed to sample the distribution $(1-\varepsilon)/\varepsilon$ (see the EGS4 manual, Ref [11]) which is at the same time the reason for the error shown in Fig. 9.

[6]The actual maximum for a threshold energy k_c is slightly smaller and obtained for $k = k_c$, the difference between the two is negligible except at low electron energies.

[7]As the logarithm of the kinetic energy is known prior to the call to the `BREMS` routine (because also used for other purposes), the time consuming evaluation of the logarithm is not necessary if $\ln(k_c)$ is stored in the computer memory for each medium.

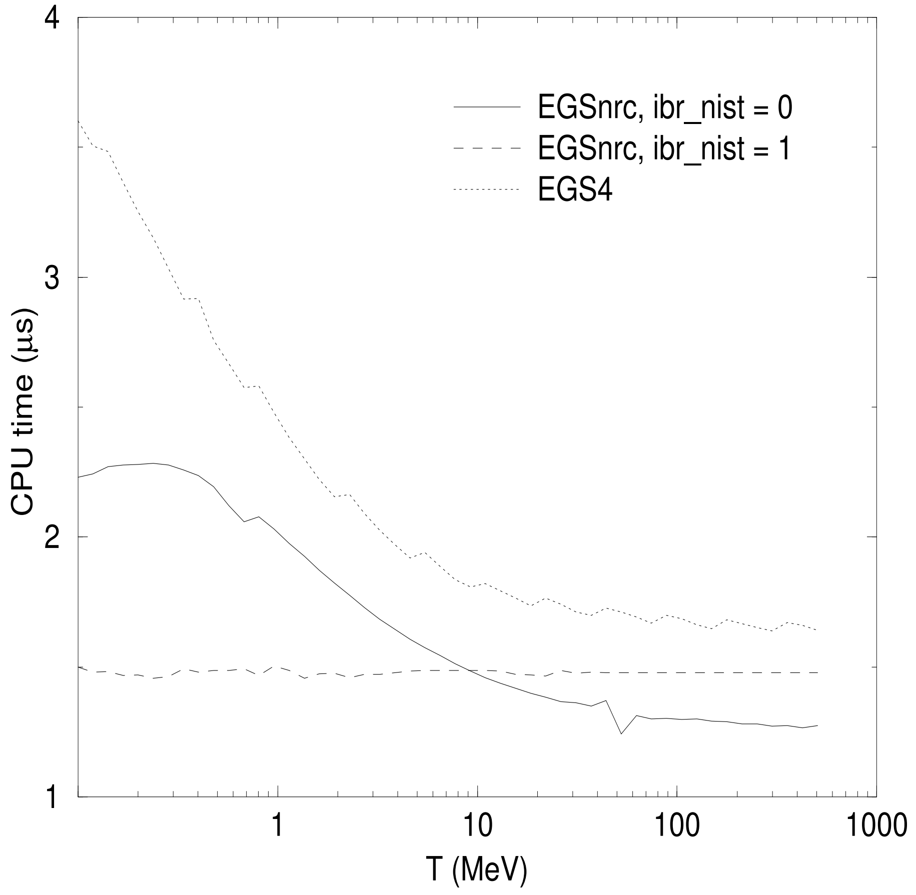


Figure 10: CPU times (in μs), as function of the incident electron kinetic energy T, necessary to sample one photon energy using various algorithms

2.4.2.iii Simulation of discrete bremsstrahlung events, angular distribution

In the original EGS4 implementation, the polar angle of bremsstrahlung emission with respect to the initial electron direction was fixed and given by m/E. In Ref. [18] an improved angle selection scheme based on equation 2BS in the article by Koch and Motz [47] was implemented for use with EGS4. This implementation was adopted in EGSnrc with slight modifications.

Equation 2BS, which is the bremsstrahlung cross section, differential in the photon energy k and the photon emission angle θ, is [47]

$$\begin{aligned}
\mathrm{d}\sigma_{\mathrm{brem}}(k,\theta) &= 4\alpha Z^2 r_0^2 \frac{\mathrm{d}k}{k}\frac{y\mathrm{d}y}{(y^2+1)^2}\left\{\frac{16y^2r}{(y^2+1)^2} - (1+r)^2 + \left[1+r^2-\frac{4y^2r}{(y^2+1)^2}\right]\ln M(y)\right\} \\
r &= \frac{E'}{E}\ , \quad y = \frac{E}{m}\theta\ , \quad \frac{1}{M(y)} = \Delta^2 + \left(\frac{Z^{1/3}}{111(y^2+1)}\right)^2
\end{aligned} \tag{4.2.8}$$

where all definitions following Eq. (4.2.1) apply. Eq. (4.2.8) is the result of an extreme relativistic, first Born and small angle approximation, but it includes a screening correction based on a Thomas-Fermi potential. The effect of the screening of the nucleus by atomic electrons is contained by the expression in the brackets for $1/M(y)$. This can easily be verified by comparing Eq. (4.2.8) to equation 2BN(a) from the article by Koch and Motz which is derived with the same approximations but using a bare nuclear potential. In order to investigate the performance of Eq. (4.2.8) at low incident electron energies, we can compare it to formula 2BN of the article by Koch and Motz which is for a bare nucleus but does not involve the extreme relativistic and small angle approximations. For a "fair" comparison, screening corrections must be negligible, this is the case if

$$\Delta^2 \gg \left(\frac{Z^{1/3}}{111(y^2+1)}\right)^2 \quad \text{or} \quad \frac{km}{E^2} \gg \frac{2Z^{2/3}}{111^2}\ , \tag{4.2.9}$$

a condition which is satisfied in a wide range of photon/electron energy combinations. Figure 11 shows a typical result of such a comparison. The modification to 2BS that we have undertaken, shown as a dashed line and obviously at a much better agreement with formula 2BN, is fairly simple and allows most of the considerations of Ref. [18] to be applied for the sampling procedure. We note that the leading term of the distribution 2BN is $(1-\beta\cos\theta)^{-2}$ where β is the electron velocity in units of the speed of light. Approximated for small angles and high energies it is equivalent to the leading term of 2BS $(1+y^2)^{-2}$, apart from a normalization constant:

$$\begin{aligned}
(1-\beta\cos\theta)^2 &\approx \left[1-\beta\left(1-\frac{\theta^2}{2}\right)\right]^2 = (1-\beta)^2\left[1+\theta^2\frac{\beta}{1-\beta}\right]^2 \\
&= (1-\beta)^2\left(1+\beta(1+\beta)\theta^2\frac{E}{m}\right)^2 \approx (1-\beta)^2(1+y^2)^2
\end{aligned} \tag{4.2.10}$$

We then use

$$y^2 = \beta(1+\beta)\frac{E^2}{m^2}(1-\cos\theta) \tag{4.2.11}$$

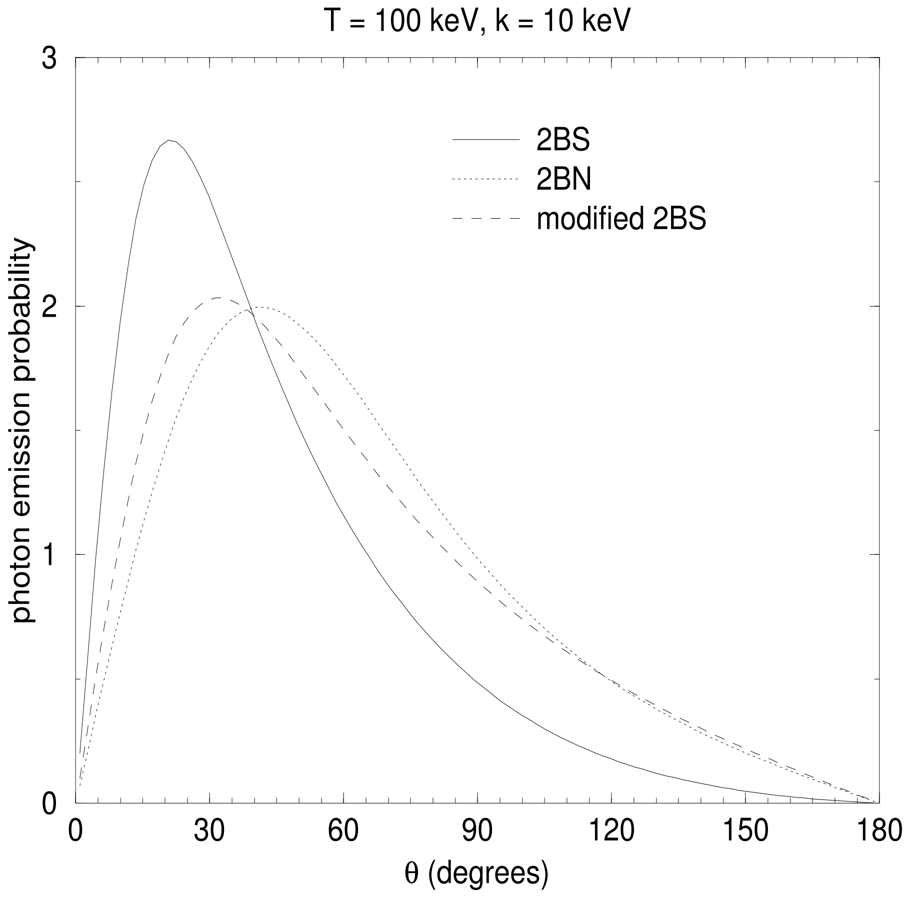


Figure 11: Angular distribution for emission of 10 keV bremsstrahlung photons by 100 keV electrons in aluminum. Solid line represents equation 2BS of Koch and Motz (Eq. (4.2.8) in this report), the dotted line equation 2BN of Koch and Motz, the dashed line is 2BS with modifications as discussed in the text.

in Eq. (4.2.8)[8] and otherwise apply the results of Ref. [18] so that the sampling algorithm is as follows:

1. Calculate the maximum of the function in the curled brackets (to be denoted by $f(y)$), $f_{\rm max}$, which is obtained for $y^2 = 0, y^2 = 1$ or $y^2 = y^2_{\rm max} \equiv 2\beta(1+\beta)(E/m)^2$ for the current E and k
2. Pick two random numbers r_1 and r_2
3. Sample y^2 from $y{\rm d}y/(1+y^2)^2$ using
$$y^2 = \frac{r_1 y^2_{\rm max}}{1+y^2_{\rm max}(1-r_1)} \qquad (4.2.12)$$
4. If $r_2 > f(y)/f_{\rm max}$ go to step 2
5. Deliver $\cos\theta$,
$$\cos\theta = 1 - \frac{y^2m^2}{\beta(1+\beta)E^2} = 1 - \frac{y^2}{2y^2_{\rm max}} \qquad (4.2.13)$$

The efficiency of this algorithm is close to unity for low energies and decreases logarithmically with increasing energy. In addition, it requires several logarithm evaluations (3 in step 1, 1 for each repetition of step 4) and is therefore rather slow. We have therefore implemented a second bremsstrahlung angle selection scheme which uses only the leading term of the angular distribution and can be selected by the user by setting the parameter `IBRDST` in `COMMON/BREMPR/` to zero (the default `IBRDST` value is 1, *i.e.* modified 2BS from Koch and Motz). We have found that the original EGS4 fixed angle approach is not faster than using the leading term of the angular distribution and have therefore removed it.

2.4.2.iv Radiative splitting

If the radiative splitting option is set (`nbr_split` > 1), the result of a bremsstrahlung event will be `nbr_split` photons, each having the fraction 1/`nbr_split` of the weight of the electron, and an electron with an energy given by its initial energy minus the energy of the last bremsstrahlung photon produced. Note that this violates energy conservation on an event-by-event basis, energy is conserved only on average. The motivation to "inline" the bremsstrahlung splitting technique was that various quantities, necessary for the sampling of photon energies, emission angles, and associated rotations, can be calculated only once and then re-used `nbr_split` times.

It is worth noticing that if `nbr_split` is set to zero, the bremsstrahlung event will be skipped. This gives the possibility to study, for instance, the influence of the neglect of bremsstrahlung production on calculated quantities[9], should this be of any use to someone.

[8]One could go one step further and modify terms containing y^2 in the nominator as they obviously come from expressions with $\sin^2\theta$ but this turns out to not improve the agreement to 2BN significantly

[9]A similar result can be achieved by using `IUNRST=4` when generating the PEGS data but in this case the average energy lost to radiation will be still subtracted and deposited locally.

2.4.3 Discrete inelastic collisions

When the binding of atomic electrons is ignored (default EGSnrc behavior), electron-electron scattering can be described by the Møller cross section [54] and positron-electron scattering by the Bhabha cross section [55]. When binding is taken into account, interactions of electrons and positrons with atoms can result in the creation of inner shell vacancies, this process is usually referred to as electron impact ionization (EII).

2.4.3.i Møller scattering

The Møller cross section, which is the cross section for electron-electron scattering differential in the kinetic energy T' of the scattered electron which is initially at rest, is [50]

$$\frac{d\sigma_{\text{inel}}^{-}}{dT'} = \frac{2\pi r_0^2 m}{\beta^2}\,\frac{1}{T'^2}\left[1+\frac{T'^2}{(T-T')^2}+\frac{\tau^2}{(\tau+1)^2}\left(\frac{T'}{T}\right)^2-\frac{2\tau+1}{(\tau+1)^2}\,\frac{T'}{T-T'}\right] \tag{4.3.1}$$

where β is the incident electron velocity in units of the speed of light, T the incident kinetic energy and $\tau = T/m$. Because the two electrons are indistinguishable, Eq. (4.3.1) is symmetric with respect to exchange of the energies of the two scattered particles. Per definition, the electron with the higher energy after the collision is considered to be the primary, so that the total cross section for Møller interactions is obtained via integration of Eq. (4.3.1) from T_c to $T/2$:

$$\sigma_{\text{inel}}^{-} = \int\limits_{T_c}^{T/2} \frac{d\sigma_{\text{inel}}^{-}}{dT'}dT' \tag{4.3.2}$$

In EGS4 and EGSnrc T_c is called `TE` and the corresponding total energy `AE`. The threshold kinetic energy above which Møller events can occur is obviously $2T_c$. The integration of Eq. (4.3.2) is trivial, and is evaluated by the PEGS function `AMOLTM`.

To sample the energy of the scattered electron on the basis of Eq. (4.3.1), one makes a change of variables to $\varepsilon = T'/T$, which can take values between $\varepsilon_0 = T_c/T$ and $1/2$, and after re-arranging obtains [11]

$$\frac{d\sigma_{\text{inel}}^{-}}{d\varepsilon} = C\left(\frac{\varepsilon_0}{1-2\varepsilon_0}\,\frac{1}{\varepsilon^2}\right)g(\varepsilon) \tag{4.3.3}$$

where C is a constant irrelevant for the sampling algorithm, the expression in the brackets is a normalized PDF for ε and $g(\varepsilon)$ will be used as a rejection function. At this point it is worth noticing that there is an error in the EGS4 manual for the rejection function $g(\varepsilon)$ and the resulting error in the `MOLLER` sampling routine was not corrected in EGS4 until 1996 [56]. The proper rejection function reads

$$\begin{aligned} g(\varepsilon) &= \frac{1+g_2\varepsilon^2+r(r-g_3)}{g_{\max}} \\ g_{\max} &= 1+\frac{5}{4}g_2\,, \quad g_2 = \frac{\tau^2}{(\tau+1)^2}\,, \quad g_3 = \frac{2\tau+1}{(\tau+1)^2}\,, \quad r = \frac{\varepsilon}{1-\varepsilon}\,. \end{aligned} \tag{4.3.4}$$

The algorithm used to sample ε is then as follows:

1. Calculate quantities dependent only on T, namely
$$\tau,\ g_2,\ g_3,\ g_{\max}$$
2. Pick two random numbers, r_1 and r_2
3. Sample ε from the expression in the brackets in Eq. (4.3.3) using
$$\varepsilon = \frac{T_c}{T-(T-2T_c)r_1} \tag{4.3.5}$$
4. Calculate $g(\varepsilon)$, if $r_2 > g(\varepsilon)$ go to step 2
5. Deliver ε

The efficiency of this algorithm is close to unity for low incident energies T but goes to 4/9 at high energies. A better approach would be the following: We can rewrite Eq. (4.3.2) as

$$\begin{aligned}\frac{\mathrm{d}\sigma^-_{\mathrm{inel}}}{\mathrm{d}T'} &= \frac{2\pi r_0^2 m}{\beta^2}\ [F(T')+F(T-T')] \\ F(T') &= \frac{1}{T'^2}+\frac{1}{2(T+m)^2}-\frac{2\tau+1}{(\tau+1)^2}\,\frac{1}{TT'}\ .\end{aligned} \tag{4.3.6}$$

If we now extend the range for T' to $T-T_c$, we can drop $F(T-T')$, and sample T' from $F(T')$ only. At the end T' will be set to $\mathrm{Min}(T', T-T')$. There are several possibilities to sample T' from $F(T')$. For instance, one can use $1/T'^2$ as a PDF and

$$g'(T') = \frac{1}{g'_{\max}}\left(1+\frac{T'^2}{2(T+m)^2}-\frac{2\tau+1}{(\tau+1)^2}\,\frac{T'}{T}\right) \tag{4.3.7}$$

as a rejection function. Here,

$$g_{\max} = \begin{cases} 1 & , \quad \text{if } \tau \le 2+\sqrt{6} \\ \frac{3\tau^2}{2(\tau+1)^2} & , \quad \text{else}\end{cases} \tag{4.3.8}$$

The efficiency of such an algorithm is 2/3 at high energies and therefore 1.5 times better than the one used in EGS4 and EGSnrc. Anticipating changes in the treatment of discrete inelastic scattering, in order to take into account electron binding effects, in the near future we have not implemented this more efficient algorithm.

The polar scattering angles θ and θ' in Møller events are uniquely determined by the kinematics. They are given by

$$\begin{aligned}\cos\theta &= \sqrt{\frac{T-T'}{T}\,\frac{T+2m}{T-T'+2m}}\ , &&\text{for the higher energy electron,} \\ \cos\theta' &= \sqrt{\frac{T'}{T}\,\frac{T+2m}{T'+2m}}\ , &&\text{for the lower energy electron.}\end{aligned} \tag{4.3.9}$$

The azimuthal angles are opposite and sample uniformly between zero and 2π.

2.4.3.ii Bhabha scattering

The Bhabha cross section, which is the cross section for positron-electron scattering differential in the kinetic energy T' of the scattered electron which is initially at rest, is given by [11]

$$\frac{\mathrm{d}\sigma^+_{\rm inel}}{\mathrm{d}T'} = \frac{2\pi r_0^2 m}{T^2}\left[\frac{1}{\varepsilon}\left(\frac{1}{\varepsilon\beta^2} - B_1\right) + B_2 + \varepsilon(\varepsilon B_4 - B_3)\right] \tag{4.3.10}$$

where T is the incident positron kinetic energy and the following definitions apply:

$$\begin{array}{l} \varepsilon = \dfrac{T'}{T}, \quad \tau = \dfrac{T}{m}, \quad y = \dfrac{1}{\tau+2}, \quad \beta^2 = \dfrac{\tau(\tau+2)}{(\tau+1)^2} \\ B_1 = 2 - y^2, \quad B_2 = (1-2y)(3+y^2), \quad B_3 = B_4 + (1-2y)^2, \quad B_4 = (1-2y)^3 \end{array} \tag{4.3.11}$$

The range of possible T' values is $T_c \cdots T$ (*i.e.* the positron may have energy less then T_c after Bhabha scattering). The total discrete Bhabha cross section is then obtained by integrating Eq. (4.3.10) in the allowed range, *i.e.*

$$\sigma^+_{\rm inel} = \int\limits_{T_c}^{T} \mathrm{d}T' \frac{\mathrm{d}\sigma^+_{\rm inel}}{\mathrm{d}T'} \; . \tag{4.3.12}$$

The integration is trivial but the result rather lengthy and therefore not given here. The total discrete Bhabha cross section is evaluated by the PEGS function `BHABTM`.

The method employed to sample scattered electron energies on the basis of Eq. (4.3.10) is similar to the Møller method. The electron energy fraction ε is sampled from $1/\varepsilon^2$, the rejection function $g(\varepsilon)$ is

$$g(\varepsilon) = 1 - \beta^2\varepsilon(B_1 - \varepsilon(B_2 - \varepsilon(B_3 - \varepsilon B_4))) \; . \tag{4.3.13}$$

As in the case of Møller scattering, there was an error for the Bhabha scattering rejection function which was not corrected until 1996 [56]. Bhabha polar scattering angles are given by Eq. (4.3.9).

2.4.4 Electron Impact Ionization

Since 2003 the EGSnrc system has the ability to explicitely simulate the creation of inner shell vacancies by electron or positron impact for all K- and L-shells with binding energies above 1 keV. This option can be turned on by setting the parameter `eii_flag` found in `COMIN/EII-DATA` to one. The differential cross sections used are based on a semi-empirical theory described in Ref. [57]. By default, total EII cross sections are obtained by a numerical integration from these differential cross sections. However, the user has the option to use total EII cross sections from Gryziński [58], Casnati [59] Kolbenstvedt [60], or Bote and Salvat [61]. This is accomplished by changing the parameter `eii_xfile` found in `COMIN/MEDIA` to `gryzinski, casnati, kolbenstvedt` or `penelope`. Users can also use their own set of EII cross sections by setting `eii_xfile` to an arbitrary charater string. See section 3.4.2 on page 129 for more detials.

When EII is turned on and there are K- or L-shells with binding energies above `AE` or `AP` in the medium where a discrete inelastic collision takes place, the probability for the collision being with the K- or L-shell is computed. If a random number is less than this probability, the interaction is simulated according to the EII differential cross sections of Ref. [57] and an inner shell vacancy is created that is subsequently relaxed using the subroutine `RELAX` (see section 2.3). Otherwise the interaction is simulated according to the Møller or Bhabha cross sections.

2.4.5 Two Photon Positron-Electron Annihilation

We have adopted the treatment of the two photon positron annihilation process from EGS4 without modifications. For completeness we give below the differential and total annihilation cross sections as used in EGS4 and EGSnrc.

The cross section, differential in the energy k of the one of the annihilation photons, for an incident positron with a total energy of E is [11]

$$\frac{\mathrm{d}\sigma_{\mathrm{annih}}}{\mathrm{d}k} = \frac{\pi r_0^2}{\tau(\tau+2)} \left[S_1(\kappa) + S_1(\tau+2-\kappa)\right] \tag{4.5.1}$$

where τ and κ are the positron kinetic energy and photon energy in units of m and

$$S_1(x) = \frac{1}{x}\left(\tau + 2 + 2\frac{\tau+1}{\tau+2} - \frac{1}{x}\right) - 1 \ . \tag{4.5.2}$$

Equation (4.5.1) is obviously symmetric under exchange of the annihilation photons, the second photon has the energy $E + m - k$.

The polar emission angles of the annihilation photons are uniquely determined by the kinematics and given by [11]

$$k = \frac{m}{1 - a\cos\theta} \ , \quad a = \sqrt{\frac{\tau}{\tau+2}} \tag{4.5.3}$$

so that the minimum and maximum possible photon energies are

$$k_{\mathrm{min}} = \frac{m}{1+a} \ , \quad k_{\mathrm{max}} = \frac{m}{1-a} \ . \tag{4.5.4}$$

The total annihilation cross section is obtained by integrating Eq. (4.5.1) over the allowed k-range and can be written as

$$\sigma_{\mathrm{annih}} = \frac{\pi r_0^2}{\tau+2}\left[\frac{\tau^2+6\tau+6}{\tau(\tau+2)} \ln\left(\tau + 1 + \sqrt{\tau(\tau+2)}\right) - \frac{\tau+4}{\sqrt{\tau(\tau+2)}}\right] \tag{4.5.5}$$

At high energies ($\tau \gg 1$) σ_{annih} decreases as $(\ln\tau)/\tau$, for $\tau \to 0$ the cross section tends to infinity (*i.e.* positrons always annihilate at rest if they have not annihilated before).

The annihilation process is a "catastrophic" event and treated discretely.

To sample the energy of one of the photons from Eq. (4.5.1), one makes a change in variables to $\varepsilon = \kappa/(\tau+2)$, drops the second S_1 because of the symmetry, and after re-arranging obtains

$$\frac{\mathrm{d}\sigma_{\mathrm{annih}}}{\mathrm{d}\varepsilon} = Cf(\varepsilon)g(\varepsilon) \tag{4.5.6}$$

$$f(\varepsilon) = \frac{1}{\ln[(1-\varepsilon_0)/\varepsilon_0]}\,\frac{1}{\varepsilon}\,, \quad g(\varepsilon) = 1 - \frac{[\varepsilon(\tau+2)-1)]^2}{\varepsilon(\tau^2+4\tau+2)} \tag{4.5.7}$$

where C is a constant that contains factors irrelevant for the sampling procedure and ε_0 the minimum possible value for ε,

$$\varepsilon_0 = \frac{1}{(\tau+2)(1+a)}\ . \tag{4.5.8}$$

The function $f(\varepsilon)$ is a normalized PDF, $g(\varepsilon)$ is always positive and has a maximum of 1 for $\epsilon = 1/(\tau+2)$ and thus is a valid rejection function[10]. The sampling algorithm is then as follows:

1. Compute quantities dependent on E, namely

$$\tau,\ A = \tau+2,\ a,\ \varepsilon_0,\ b = \ln\frac{1-\varepsilon_0}{\varepsilon_0}$$

2. Pick two random numbers, r_1 and r_2
3. Set

$$\varepsilon = \varepsilon_0 \exp(r_1 b) \tag{4.5.9}$$

   and calculate $g(\varepsilon)$
4. If $r_2 > g(\varepsilon)$, the go to step 2
5. Deliver ε

At this point one should perhaps mention that a single photon or three or more photon annihilation processes in the nuclear field are also possible. Messel and Crawford [62] point out that the ratio of one to two photon annihilation cross sections is small until higher energies are reached, at which point the absolute value of the cross section is small. Thus, the single photon annihilation process is ignored. Positron annihilation to three or more photons is even less likely than one photon annihilation and therefore also ignored.

If positrons do not annihilate in flight, they annihilate at rest producing two photons. As one can easily verify from Eq. (4.5.3) and (4.5.4), the photon energies go to m as τ goes to zero. In addition, the cross section differential in the photon emission angle becomes uniform in the limit $\tau \to 0$.

If radiative splitting is set (`nbr_split` > 1), the annihilation process will produce 2 `nbr_split` photons, each carrying the fraction 1/`nbr_split` from the positron weight. Simultaneous production of `nbr_split` annihilation events allows the quantity b (see step 1) as well as parameter related to angular rotations to be re-used `nbr_split` times.

[10]Note that our $g(\varepsilon)$ is the $g(\varepsilon)$ defined in Eq (2.12.26) of the EGS4 manual divided by its maximum.

2.4.6 Collision stopping power

EGSnrc “inherits” the treatment of restricted collision stopping powers from EGS4, *i.e.* uses the formulas recommended by Seltzer and Berger [63] which are based on the Bethe-Bloch theory [64, 65, 66]. The standard treatment (see also ICRU report 37 [50] which was used as the source of the formulas below) assumes that there is a certain value for energy transfer to atomic electrons, $T_{\rm med}$, that is (i) large compared to the binding energies (ii) corresponds to an impact parameter that is large compared to the atomic dimensions. Collision processes that are associated with energy loss T' less than $T_{\rm med}$ are treated according to the theory of Bethe, the main result of which is that

$$L^{\pm}_{\rm coll}(T, T' < T_{\rm med}) = \frac{2\pi r_0^2 mn}{\beta^2}\left[\ln\left(\frac{2m\beta^2 T_{\rm med}}{(1-\beta^2)I^2}\right) - \beta^2 - \delta\right] \tag{4.6.1}$$

where β is again the electron velocity in units of the speed of light, I is the mean ionization energy and δ the density effect correction that takes into account the polarization of the medium due to the electron field. As $T_{\rm med}$ is defined being large compared to the binding energies of the atom, collision processes with energy transfer larger than $T_{\rm med}$ can be treated using the Møller [54] (electrons) or Bhabha [55] (positron) cross section (see also section 2.4.3), *i.e.*

$$L^{\pm}_{\rm coll}(T, T' > T_{\rm med}) = \int\limits_{T_{\rm med}}^{T_c} {\rm d}T' T' \frac{{\rm d}\sigma^{\pm}_{\rm inel}}{{\rm d}T'} \tag{4.6.2}$$

Using some additional approximations, $T_{\rm med}$ drops out in the sum of $L^{\pm}_{\rm coll}(E, T' < T_{\rm med})$ and $L^{\pm}_{\rm coll}(E, T' > T_{\rm med})$ and one obtains

$$L^{\pm}_{\rm coll}(T, T_c) = \frac{2\pi r_0^2 mn}{\beta^2}\left[\ln\frac{T^2}{I^2} + \ln(1+\tau/2) + G^{\pm}(\tau) - \delta\right] \tag{4.6.3}$$

where $\tau = T/m$ and the functions $G^{\pm}$ are different for electrons and positrons due to differences in the Møller and Bhabha cross sections and are given by

$$\begin{aligned} G^{-}(\tau) &= -1 - \beta^2 + \ln\left[4\eta(1-\eta)\right] + \frac{1}{1-\eta} + (1-\beta^2)\left[\frac{\tau^2\eta^2}{2} + (2\tau+1)\ln(1-\eta)\right] \\ G^{+}(\tau) &= \ln(4\eta) - \beta^2\left[1 + (2-y^2)\eta - (3+y^2)\frac{y\tau}{2}\eta^2 + (1+y\tau)\frac{y^2\tau^2}{3}\eta^3 - \frac{y^3\tau^3}{4}\eta^4\right] \end{aligned} \tag{4.6.4}$$

where $\eta = T_c/T$ and y is defined in Eq. (4.3.10).

From the above discussion and from the general discussion of a Class II condensed history implementation in section 2.4.1 it is clear that the formalism used to treat inelastic collisions with atomic electrons is only applicable if

$$T_c \gg \quad \text{binding energies of the medium of interest.} \tag{4.6.5}$$

This imposes a rather severe limitation on the use of Class II condensed history codes in high-Z materials (the K-shell binding energy for lead is, for instance, 88 keV). We are therefore currently investigating a more realistic approach for situations when the condition (4.6.5) is not satisfied, but its implementation into EGSnrc is left for the next release of the system. One should probably also mention that the many successful studies carried out with the EGS4 system indicate that the implications of violating the requirement (4.6.5) are perhaps less severe than one might expect from purely theoretical arguments.

The only non-trivial parameters of the restricted stopping power formula are the mean ionization energy I and the density effect correction δ. The default mean ionization energies for elements used in PEGS4, along with atomic numbers, weights, chemical symbols, and mass densities are summarized in Table 2. Mean ionization energies for compounds are derived from

$$\ln I = \sum p_i Z_i \ln I(Z_i) \tag{4.6.6}$$

where p_i is the stoichiometric index of the i'th element which has atomic number Z_i and a mean ionization energy $I(Z_i)$, unless the material belongs to a set of pre-defined materials to be found in Table 2.13.2 of the EGS4 manual [11] or listed in the `BLOCK DATA` section of `pegs4.mortran`.

The density effects correction has been treated extensively in the literature. The default PEGS4 approach is based on the formulation of Sternheimer and Peierls[67] which basically parameterizes the density effect in terms of 6 parameters (`AFACT, SK, X0, X1, CBAR`, and `IEV`). This same parameterized approach was used for the calculations by Berger and Seltzer [68] and by Sternheimer, Berger and Seltzer [69] who fitted the parameters to the density effect as calculated for the ICRU for increasingly larger numbers of materials. The power of PEGS4 is that it will generate a density effect for any arbitrary material, if need be with no recourse to the fitted parameters. In this case it will use the Sternheimer and Peierls[67] general formula. There is also an option in PEGS4 which allows the 6 parameters to be read in directly (using the `ISSB=1` option in PEGS4, see section 6.2 page 256). However, these parameterizations are only fits to the actual density effect data. An option was added to PEGS4 in 1989 which allowed the density effect data to be used directly[70] and the EGSnrc distribution includes a huge data base of all the density effect values calculated by Seltzer and Berger for ICRU Report 37[50]. To turn this option on the user must set the flag `EPSTFL` to 1 in the PEGS4 input file. See section 6.1.3 (page 222) for more details. In this case the density effect correction is calculated by interpolation from pre-computed values stored in a separate file. Selecting this option has the additional effect that the mean ionization energy of the material is taken directly from the density effect correction file. It should be noted that in general the differences in collision stopping powers between using the default Sternheimer and Peierls density effect, or the fitted parameters or the direct ICRU 37 values, are small, of the order of a few percent at most. These differences are only likely of importance for very detailed work.

Table 2: Default atomic numbers, symbols, atomic weights, mass densities, and I values for elements in PEGS4.

Z	Symbol	Atomic weight	Density (g/cm^3)	I(eV)
1	H	1.00797	0.0808	19.2
2	HE	4.00260	0.1900	41.8

continued on next page

continued from previous page

Z	Symbol	Atomic weight	Density (g/cm^3)	I(eV)
3	LI	6.93900	0.5340	40.0
4	BE	9.01220	1.8500	63.7
5	B	10.81100	2.5000	76.0
6	C	12.01115	2.2600	78.0
7	N	14.00670	1.1400	82.0
8	O	15.99940	1.5680	95.0
9	F	18.99840	1.5000	115.0
10	NE	20.18300	1.0000	137.0
11	NA	22.98980	0.9712	149.0
12	MG	24.31200	1.7400	156.0
13	AL	26.98150	2.7020	166.0
14	SI	28.08800	2.4000	173.0
15	P	30.97380	1.8200	173.0
16	S	32.06400	2.0700	180.0
17	CL	35.45300	2.2000	174.0
18	AR	39.94800	1.6500	188.0
19	K	39.10200	0.8600	190.0
20	CA	40.08000	1.5500	191.0
21	SC	44.95600	3.0200	216.0
22	TI	47.90000	4.5400	233.0
23	V	50.94200	5.8700	245.0
24	CR	51.99800	7.1400	257.0
25	MN	54.93800	7.3000	272.0
26	FE	55.84700	7.8600	286.0
27	CO	58.93320	8.7100	297.0
28	NI	58.71000	8.9000	311.0
29	CU	63.54000	8.9333	322.0
30	ZN	65.37000	7.1400	330.0
31	GA	69.72000	5.9100	334.0
32	GE	72.59000	5.3600	350.0
33	AS	74.92160	5.7300	347.0
34	SE	78.96000	4.8000	348.0
35	BR	79.80800	4.2000	357.0
36	KR	83.80000	3.4000	352.0
37	RB	85.47000	1.5300	363.0
38	SR	87.62000	2.6000	366.0
39	Y	88.90500	4.4700	379.0
40	ZR	91.22000	6.4000	393.0
41	NB	92.90600	8.5700	417.0
42	MO	95.94000	9.0100	424.0
43	TC	99.00000	11.5000	428.0
44	RU	101.07000	12.2000	441.0

continued on next page

continued from previous page

Z	Symbol	Atomic weight	Density (g/cm^3)	I(eV)
45	RH	102.90500	12.5000	449.0
46	PD	106.40000	12.0000	470.0
47	AG	107.87000	10.5000	470.0
48	CD	112.40000	8.6500	469.0
49	IN	114.82000	7.3000	488.0
50	SN	118.69000	7.3100	488.0
51	SB	121.75000	6.6840	487.0
52	TE	127.60000	6.2400	485.0
53	I	126.90440	4.9300	491.0
54	XE	131.30000	2.7000	482.0
55	CS	132.90500	1.8730	488.0
56	BA	137.34000	3.5000	491.0
57	LA	138.91000	6.1500	501.0
58	CE	140.12000	6.9000	523.0
59	PR	140.90700	6.7690	535.0
60	ND	144.24001	7.0070	546.0
61	PM	147.00000	1.0000	560.0
62	SM	150.35001	7.5400	574.0
63	EU	151.98000	5.1700	580.0
64	GD	157.25000	7.8700	591.0
65	TB	158.92400	8.2500	614.0
66	DY	162.50000	8.5600	628.0
67	HO	164.92999	8.8000	650.0
68	ER	167.25999	9.0600	658.0
69	TM	168.93401	9.3200	674.0
70	YB	173.03999	6.9600	684.0
71	LU	174.97000	9.8500	694.0
72	HF	178.49001	11.4000	705.0
73	TA	180.94800	16.6000	718.0
74	W	183.85001	19.3000	727.0
75	RE	186.20000	20.5300	736.0
76	OS	190.20000	22.4800	746.0
77	IR	192.20000	22.4200	757.0
78	PT	195.08000	21.4500	790.0
79	AU	196.98700	19.3000	790.0
80	HG	200.59000	14.1900	800.0
81	TL	204.37000	11.8500	810.0
82	PB	207.19000	11.3400	823.0
83	BI	208.98000	9.7800	823.0
84	PO	210.00000	9.3000	830.0
85	AT	210.00000	1.0000	825.0
86	RN	222.00000	4.0000	794.0

continued on next page

continued from previous page

Z	Symbol	Atomic weight	Density (g/cm^3)	I(eV)
87	FR	223.00000	1.0000	827.0
88	RA	226.00000	5.0000	826.0
89	AC	227.00000	1.0000	841.0
90	TH	232.03600	11.0000	847.0
91	PA	231.00000	15.3700	878.0
92	U	238.03000	18.9000	890.0
93	NP	237.00000	20.5000	902.0
94	PU	242.00000	19.7370	921.0
95	AM	243.00000	11.7000	934.0
96	CM	247.00000	7.0000	939.0
97	BK	247.00000	1.0000	952.0
98	CF	248.00000	1.0000	966.0
99	ES	254.00000	1.0000	980.0
100	FM	253.00000	1.0000	994.0

2.4.7 Elastic scattering cross sections

The treatment of electron and positron elastic scattering in EGSnrc is determined by the logical parameter `SPIN_EFFECTS` which is in `common/ET_Control/`. If set to `.true.` (this is the default), elastic scattering cross sections that take into account spin effects are employed, they are discussed in section 2.4.7.ii. If `.false.`, elastic scattering is based on the screened Rutherford cross section. This is consistent with EGS4, although multiple elastic scattering (see section 2.4.8) is based on an exact theory rather than the small-angle theory of Molière[71] which is used in EGS4.

2.4.7.i Screened Rutherford elastic scattering

The screened Rutherford cross section, which is the cross section differential in the cosine μ of the polar scattering angle of electrons or positrons incident on atoms of atomic number Z, is

$$\frac{\mathrm{d}\sigma_{\mathrm{SR}}}{\mathrm{d}\mu} = \frac{2\pi r_0^2 Z^2}{\beta^2\tau(\tau+2)} \frac{1}{(1-\mu+2\eta)^2} \tag{4.7.1}$$

where β is the particle velocity in units of the speed of light, τ the kinetic energy T in units of m and η the screening parameter. The total elastic scattering cross section is obtained from Eq. (4.7.1) by integrating over μ from -1 to 1 and is given by

$$\sigma_{\mathrm{SR}} = \frac{\pi r_0^2 Z^2}{\beta^2\tau(\tau+2)\eta(1+\eta)} \tag{4.7.2}$$

In EGS4 the screening parameter η is based on the single elastic scattering theory of Molière[72]. Molière performed a partial-wave analysis (PWA) expansion of the Klein-Gordon equation (*i.e.* he neglected spin effects) in the nuclear field described by the Thomas-Fermi potential, used a small-angle approximation (*i.e.* replaced the Legendre Polynomials by zeroth order Bessel functions J_0), and employed a WKB-expansion of the resulting radial equation up to zeroth order in $\hbar$ to calculate the phase shifts $\phi(z)$ to arrive at

$$\frac{d\sigma_{\rm M}}{\chi d\chi} = 2\pi a^2 \left| \int\limits_0^\infty dz z J_0\left(z\frac{\chi}{2\sqrt{\eta_0}}\right)\left[\exp\left(-2i\alpha'\phi(z)\right)-1\right]\right|^2 \tag{4.7.3}$$

where χ is the scattering angle, a is the Thomas-Fermi radius, η_0 is defined in Eq. (4.7.5) and α' given by

$$\alpha' = \frac{\alpha Z}{\beta} \tag{4.7.4}$$

($\alpha \approx 1/137$ is the fine structure constant). In addition he approximated the Thomas-Fermi potential by the sum of three exponential functions in which case the phase shifts $\phi(z)$ are given by zeroth order modified Bessel functions. He then required that the average scattering angle squared, calculated from a screened Rutherford cross section is the same as the average scattering angle squared resulting from Eq. (4.7.3) and, after studying the limiting cases $\alpha' \to 0$ and $\alpha' \to \infty$, arrived at the simple formula

$$\eta = \eta_0(1.13 + 3.76\alpha'^2)\ , \quad \eta_0 = \frac{\alpha^2 Z^{2/3}}{4C_{\rm TF}^2\tau(\tau+2)}\ , \quad C_{\rm TF} = \left(\frac{9\pi^2}{128}\right)^{1/3} \tag{4.7.5}$$

for the effective screening parameter η[11].

The treatment of elastic scattering in EGS4 is intrinsically associated with Molière's multiple scattering theory[71]. In his treatment of multiple scattering Molière uses a small-angle approximation in which case the moments of the screened Rutherford cross section (see section 2.4.8) are given by first order modified Bessel functions K_1. In addition, a small argument expansion of K_1 is performed so that the elastic scattering cross section for compounds and mixtures can be expressed with two parameters, b_c and χ_{cc}, as follows:

$$\begin{aligned} b_c &= \frac{4\pi r_0^2 C_{\rm TF}^2 \rho}{1.13\alpha^2 u}\frac{Z_S\exp(Z_E/Z_S)}{A\exp(Z_X/Z_S)} = 7821.6\ {\rm cm}^2/{\rm g}\ \rho\frac{Z_S\exp(Z_E/Z_S)}{A\exp(Z_X/Z_S)} \\ \chi_{cc}^2 &= \frac{4\pi r_0^2 m^2}{u}\ \rho\ \frac{Z_S}{A} = 0.1569\ {\rm cm}^2{\rm MeV}^2/{\rm g}\ \rho\frac{Z_S}{A} \end{aligned} \tag{4.7.6}$$

where A is the relative molecular mass and

$$\begin{aligned} Z_S &= \sum p_i Z_i(Z_i + \xi_{\rm MS}) \\ Z_E &= \sum p_i Z_i(Z_i + \xi_{\rm MS}) \ln Z_i^{-2/3} \\ Z_X &= \sum p_i Z_i(Z_i + \xi_{\rm MS}) \ln\left(1 + 3.34\ \alpha^2 Z_i^2\right) \end{aligned} \tag{4.7.7}$$

Note that in the expression for Z_X α' was replaced by αZ_i and we have removed the factor 1.167 in the denominator of the expression for b_c in the EGS4 manual, this factor is not necessary when multiple scattering is treated with the exact formulation. The purpose of the parameter $\xi_{\rm MS}$ is to take into account contributions from sub-threshold inelastic scattering

[11]Note that our η is Molière's $\chi_a^2/4$, $C_{\rm TF}$ is the Thomas-Fermi constant.

with atomic electrons. In PEGS4 a macros `$FUDGEMS` is used for $\xi_{\rm MS}$ with the intent to provide the user with the possibility of implementing a more realistic treatment of sub-threshold inelastic contributions. Experience shows that this capability is rarely used and so, PEGS4 generated data sets usually have $\xi_{\rm MS} = 1$ (the default value). This leads to double counting of angular deflections due to sub-threshold inelastic collisions (see *e.g.* [73]). This problem remains present in EGSnrc if the screened Rutherford cross section is used to model elastic collisions (parameter `SPIN_EFFECTS` is set to `.false.`). We have not attempted a correction in this case as the neglect of spin effects represents a more severe approximation than the double counting of inelastic collisions. A more realistic approach is used when spin effects are turned on, see next section.

In terms of the parameter b_c and χ_{cc}, which are called `BLCC` and `XCC`, the screening parameter is

$$\eta = \frac{\chi_{cc}^2}{4 b_c m^2 \tau(\tau+2)} \tag{4.7.8}$$

and the total macroscopic cross section

$$\Sigma_{\rm SR} = \frac{b_c}{\beta^2} \tag{4.7.9}$$

The latter formula involves the neglect of $1+\eta$ in the denominator of Eq. (4.7.2).

Sampling angular deflections in single elastic scattering events on the basis of Eq. (4.7.1) is trivial, it is accomplished by

$$\mu = 1 - \frac{2\eta r}{1 - r + \eta} \tag{4.7.10}$$

where r denotes a random number between zero and unity. Single elastic scattering deflections are necessary when boundary crossing between different media is modelled exactly, see section 2.4.10.

2.4.7.ii Elastic scattering with spin

Potentially the most accurate elastic scattering cross sections are those obtained from a PWA solution of the Dirac equation in the nuclear field screened by the atomic electrons. The general expression for the cross section, derived by Mott [74], is given by (see also the article by Motz, Olsen and Koch, [75])

$$\begin{aligned} \frac{{\rm d}\sigma_{\rm PWA}}{{\rm d}\Omega} &= \frac{r_0^2}{4\alpha^2\tau(\tau+2)}\left[|f|^2 + |g|^2\right] \\ f &= \sum_{l=0}^{\infty}\left\{(l+1)\left[e^{2i\phi_l} - 1\right] + l\left[e^{2i\phi_{-l-1}} - 1\right]\right\} P_l(\mu) \\ g &= \sum_{l=0}^{\infty}\left\{e^{2i\phi_{-l-1}} - e^{2i\phi_l}\right\} P_l^1(\mu) \end{aligned} \tag{4.7.11}$$

where the definitions from the previous section for τ and α apply, P_l and P_l^m are Legendre and associated Legendre polynomials and ϕ_l denotes the phase shifts. They are obtained from the asymptotic solution of the equation

$$\frac{\mathrm{d}^2 F_l}{\mathrm{d}r^2} + \left[pr + \frac{l(l+1)}{r^2} - U_l\right] F_l = 0 \tag{4.7.12}$$

when written in the form $F_l(r \to \infty) = \sin(pr - l\pi/2 + \phi_l)$. Here, $p = \sqrt{\tau(\tau+2)}$ is the electron momentum in units of m/c and the nuclear and/or atomic charge structure is contained in the effective Dirac potential U_l,

$$\begin{aligned} U_l &= 2(\tau+1)V - V^2 - \frac{l+1}{r^2}\,\frac{D'}{D} + \frac{3}{4}\,\frac{D'^2}{D^2} - \frac{1}{2}\,\frac{D''}{D} \\ D &= \tau + 2 - V, \quad D' = \frac{\mathrm{d}D}{\mathrm{d}r}, \quad D'' = \frac{\mathrm{d}^2 D}{\mathrm{d}r^2} \end{aligned} \tag{4.7.13}$$

where V is a spherically symmetric potential arising from the charge structure of the nucleus and/or the atom. Mott has also presented [76] an analytical solution for the phase shifts for a bare nucleus (*i.e.* $V = \pm Z/r$, where the plus sign is for positrons and the minus for electrons). His result can be written as [75]

$$\frac{\mathrm{d}\sigma_{el}}{\mathrm{d}\Omega} = \frac{r_0^2 Z^2}{\beta^2\tau(\tau+2)}\,\frac{R_{\mathrm{mott}}(Z,\tau,\mu)}{(1-\mu)^2} \tag{4.7.14}$$

where R_{mott} has become known as the Mott correction and is given by

$$\begin{aligned} R_{\mathrm{mott}} &= \frac{1-\mu}{(\tau+1)^2}|F_0 + F_1|^2 + \frac{2\beta^4}{\tau(\tau+2)}\,\frac{(1-\mu)^2}{1+\mu}\,\frac{|G_0+G_1|^2}{\alpha^2 Z^2} \\ F_0 &= \frac{i\Gamma(1-i\alpha')}{2\Gamma(1+i\alpha')}\left(\frac{1-\mu}{2}\right)^{i\alpha'} \\ G_0 &= -i\alpha'\frac{1+\mu}{1-\mu}F_0 \\ F_1 &= \frac{i}{2}\sum_{l=0}^{\infty}\left[l\phi_l - (l+1)\phi_{l+1}\right](-1)^l P_l(\mu) \\ G_1 &= \frac{i}{2}\sum_{l=0}^{\infty}\left[l^2\phi_l - (l+1)^2\phi_{l+1}\right](-1)^l P_l(\mu) \\ \phi_l &= \frac{\exp(-i\pi l)}{l+i\alpha'}\,\frac{\Gamma(l-i\alpha')}{\Gamma(l+i\alpha')} - \frac{\exp(-i\pi\rho_l)}{\rho_l+i\alpha'}\,\frac{\Gamma(\rho_l-i\alpha')}{\Gamma(\rho_l+i\alpha')} \\ \rho_l &= \sqrt{l^2 - \alpha^2 Z^2} \end{aligned} \tag{4.7.15}$$

where Γ is the gamma function. The quantity α' is defined in Eq. (4.7.4) but now the atomic number Z is considered as being $-Z$ for electrons and $+Z$ for positrons. Due to this fact, the Mott correction is different for electrons and positrons. R_{mott} must be evaluated numerically, Fig. 12 gives some representative examples.

Berger and Wang [77] have implemented into ETRAN (see *e.g.* Ref. [78]) the treatment of multiple elastic scattering according to the general Mott PWA cross section, equations (4.7.11) to (4.7.13). They have used the code by Riley [79] for the numerical solution of

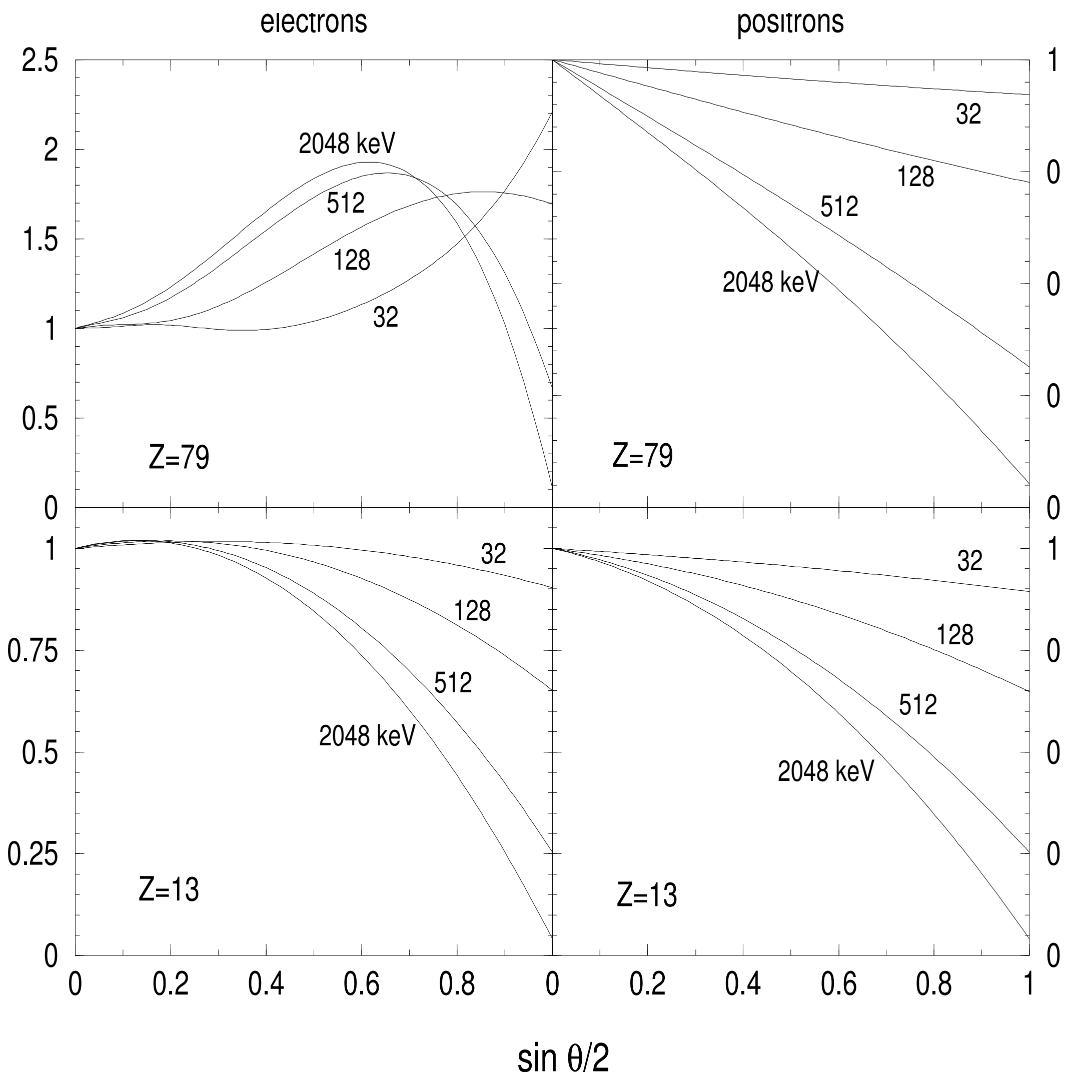


Figure 12: The Mott correction factor $R_{\rm mott}$ for gold and aluminum and various incident electron and positron energies.

Eq. (4.7.12) in a nuclear field screened by an electron density distribution obtained from the multi-configuration Dirac-Fock program by Desclaux [80]. These cross sections were later implemented also in ITS [81]. Note that ETRAN and ITS use a Class I condensed history implementation where electron transport is performed on a pre-determined step-size grid. This allows the calculation of the multiple elastic scattering distribution (see section 2.4.8) for arbitrary complicated cross sections for the step-size grid used with a reasonable amount of pre-calculated data.

The situation in a Class II code is more difficult as step-lengths are stochastic (see section 2.4.1). Using Riley's code and Desclaux' electron density functions, both kindly provided to us by Steve Seltzer of NIST, we have also generated an elastic scattering data base for all elements and energies from 1 keV to 16 MeV[12]. Using this data base, we have studied the construction of a multiple scattering theory but could not find a satisfactory procedure to keep the amount of pre-calculated data required at a reasonable level. We have therefore decided to approximate the elastic scattering cross section as

$$\frac{\mathrm{d}\sigma_{el}}{\mathrm{d}\mu} = \frac{\mathrm{d}\sigma_{SR}}{\mathrm{d}\mu} R_{\mathrm{mott}}(Z,\tau,\mu) \tag{4.7.16}$$

where $\mathrm{d}\sigma_{SR}/\mathrm{d}\mu$ is the screened Rutherford cross section given in Eq. (4.7.1). This approximation is known to be accurate for energies above 1 MeV (high-Z materials) or 100 keV (low-Z materials) as there the spin effect correction decouples from the screening correction [77]. To reproduce the average scattering at low energies we treat the screening parameter η as a free parameter and determine it by numerically solving the equation

$$\int\limits_{-1}^{1} \mathrm{d}\mu \frac{\mathrm{d}\sigma_{SR}}{\mathrm{d}\mu} R_{\mathrm{mott}}(Z,\tau,\mu)(1-\mu) \equiv \int\limits_{-1}^{1} \mathrm{d}\mu \frac{\mathrm{d}\sigma_{\mathrm{PWA}}}{\mathrm{d}\mu}(1-\mu) \tag{4.7.17}$$

The resulting screening parameter for electrons is shown as a function of β for 4 different elements in Fig. 13. In this figure η is expressed in units of η_{M}, where

$$\eta_{\mathrm{M}} = \eta_0(1.13 + 3.76\alpha^2 Z^2) \tag{4.7.18}$$

is the screening parameter used in EGS4.

The elastic scattering cross section for compounds and mixtures is expressed in a similar form as Eq. (4.7.16) but now the Mott correction is

$$R_{\mathrm{mott}}(\tau,\mu) = \frac{\sum p_i Z_i^2 R_{\mathrm{mott}}(Z_i,\tau,\mu)}{\sum p_i Z_i^2} \tag{4.7.19}$$

and the screening parameter is determined from Eq. (4.7.17) where the partial-wave analysis cross section for the compound is calculated from the PWA cross sections of its elements using the independent atom approximation.

[12] The calculation of the elastic scattering cross section from equations (4.7.11) to (4.7.13) becomes increasingly more difficult with increasing energy as more and more phase shifts have to be calculated and numerical round-off errors accumulate in the summations involved. We have modified Riley's code to perform calculations in double precision, use more dense phase shifts calculation grid and more accurate interpolations for the electron density. This allowed calculations up to 16 MeV, for even higher energies the results become numerically unstable.

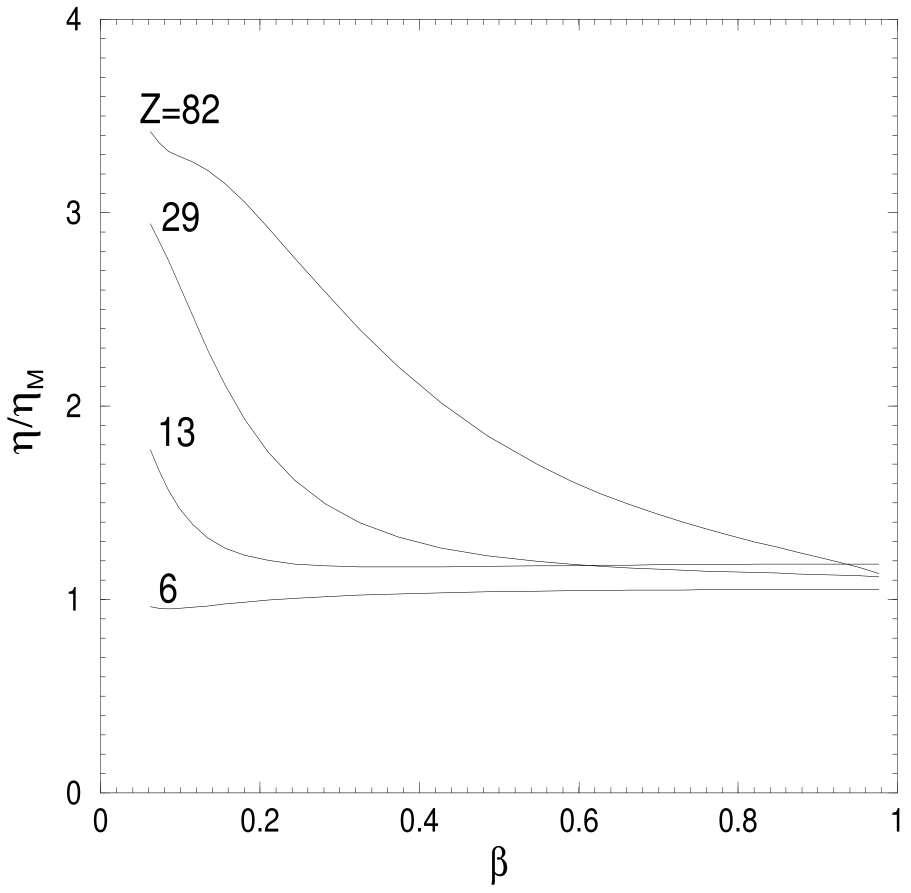


Figure 13: The screening parameter η, determined from Eq. (4.7.17), in units of $\eta_{\rm M}$, defined in Eq. (4.7.18).

We can now turn to the discussion of the modifications necessary to take into account contributions from angular deflections due to sub-threshold processes. Various studies on this subject are available in the literature, see *e.g.* [82, 83, 77]. We have not attempted to implement one of them, instead, we take the simplistic point of view that the contributions to angular deflections for all inelastic collisions, sub-threshold and discrete, can be taken into account by replacing Z^2 with $Z(Z+\xi_0)$ in the screened Rutherford cross section, Eq. (4.7.1). Here ξ_0 is an appropriate parameter, we use by default $\xi_0 = 1$ but this is not a necessary requirement. If now inelastic collisions with energy transfer larger than T_c are simulated explicitly, ξ_0 must be replaced with an energy and cut-off dependent parameter $\xi(T,T_c)$ as follows [84]

$$\xi(T,T_c) = \xi_0 \left(1 - \frac{1}{\bar{Z}+\xi_0}\, \frac{g_{\rm M}(\tau,\tau_c)}{g_{\rm R}(\eta)}\right) \tag{4.7.20}$$

where $\bar{Z}$ is the average atomic number of the compound and

$$\begin{aligned} g_M(\tau,\tau_c) = {} & \ln\frac{\tau}{2\tau_c} + \left[1 + \frac{(\tau+2)^2}{(\tau+1)^2}\right] \ln\frac{2(\tau-\tau_c+2)}{\tau+4} \\ & - \left[\frac{(\tau+2)^2}{4} + \frac{(\tau+2)(\tau+1/2)}{(\tau+1)^2}\right] \ln\frac{(\tau+4)(\tau-\tau_c)}{\tau(\tau-\tau_c+2)} \\ & + \frac{(\tau-2\tau_c)(\tau+2)}{2}\left[\frac{1}{\tau-\tau_c} - \frac{1}{(\tau+1)^2}\right] . \end{aligned} \tag{4.7.21}$$

$(\tau_c = T_c/m)$ and

$$g_{\rm R}(\eta) = (1+2\eta)\left[\ln\left(1+\frac{1}{\eta}\right) - 2\right] \tag{4.7.22}$$

To summarize, when spin effects are turned on in EGSnrc (`SPIN_EFFECTS = .true.`), the elastic scattering cross section used is

$$\frac{{\rm d}\sigma_{\rm el}}{{\rm d}\mu} = \frac{2\pi r_0^2 Z(Z+\xi(T,T_c))}{\beta^2\tau(\tau+2)}\, \frac{R_{\rm mott}(Z,T,\mu)}{(1-\mu+2\eta)^2} \tag{4.7.23}$$

where

- The screening parameter η is determined from the requirement that the above cross section reproduces the average scattering from PWA cross sections obtained via the numerical solution of equations (4.7.11) to (4.7.13) using Hartree-Fock electron densities. This parameter is different for electrons and positrons

- The parameter $\xi(T,T_c)$ takes into account contributions from sub-threshold inelastic collisions, it depends on energy and the threshold energy T_c

- $R_{\rm mott}$ is the Mott correction that is the result of the solution of the Dirac equation in the field of a bare nucleus.

All parameters necessary for the run-time interpolation of η, ξ and $R_{\rm mott}$ are initialized in the subroutine `init_spin` which is called from the subroutine `mscati`, executed at the end of `HATCH`.

Sampling of single elastic scattering events on the basis of Eq. (4.7.23) is performed using a rejection technique. One uses Eq. (4.7.10) to sample the screened Rutherford part, the μ sampled is accepted if a second random number is less than $R_{\rm mott}/R_{\rm mott,max}$. The efficiency of this algorithm is close to unity for low-Z materials but only close to 1/2 for high-Z materials.

2.4.8 Multiple elastic scattering

The multiple elastic scattering distribution for electron transport in an infinite, homogeneous medium for a path-length s, which corresponds to an energy loss $E_0 - E$, can be obtained from Eq. (4.1.21) by integrating it over the position, expanding Φ_0 and the cross sections in Legendre polynomials P_l to arrive at

$$\Phi_0(\mu,\phi,E) = \frac{1}{2\pi}\sum_{l=0}^{\infty}\left(l+\frac{1}{2}\right)\exp(-G_l)P_l(\mu) \tag{4.8.1}$$

where μ is the cosine of the polar angle θ and the process is considered in a frame where the electron is initially moving along the z axis. This expression was first obtained by Goudsmit and Saunderson [85, 86]. The Goudsmit-Saunderson (GS) moments G_l are given by

$$G_l = \int\limits_0^s {\rm d}s'\kappa_l(s') = \int\limits_E^{E_0}\frac{{\rm d}E'}{L(E',E_c,k_c)}\kappa_l(E') \tag{4.8.2}$$

where the κ_l denote the moments of the elastic scattering cross section,

$$\kappa_l(E) = 2\pi\int\limits_{-1}^{1}{\rm d}\mu\Sigma'_{\rm el}(\mu,E)\left[1-P_l(\mu)\right] \ . \tag{4.8.3}$$

The moments κ_l depend on the energy and the material in which the transport takes place, so that the multiple elastic distribution is dependent on the energy, path-length (or corresponding energy loss), material, and threshold energies for discrete interactions. In a Class I condensed history implementation one uses the total stopping power (because discrete interaction are not explicitely modelled). In addition, possible path-lengths are limited to the energy-loss grid which is decided upon prior to the simulation. These two facts allow the MS distributions to be pre-computed and stored in the memory using a relatively small amount of data. In a Class II implementation path-lengths are stochastic and so a straightforward pre-calculation for all possible step-lengths is not possible. In the past this fact has favoured use of small-angle theories for the modelling of multiple elastic scattering, *e.g.* the theory of Molière which is used in EGS4. The limitations of Molière's theory have been discussed extensively in the literature.

In EGSnrc we use an exact formulation, the multiple scattering distributions employed being dependent on the underlying elastic scattering cross sections (see section 2.4.7).

2.4.8.i Multiple elastic scattering from the screened Rutherford cross section

The multiple elastic scattering theory for the screened Rutherford cross section employed in EGSnrc was developed by Kawrakow and Bielajew in Ref. [1] and later refined by Kawrakow in Ref. [5] to better take into account energy loss.

The treatment of Ref. [1] starts with the MS distribution that results from the neglect of energy loss which can be written as

$$2\pi\Phi_0(\mu,\phi,E) = e^{-\lambda}\delta(1-\mu) + \lambda e^{-\lambda}\frac{1}{\sigma_{\rm SR}}\frac{{\rm d}\sigma_{\rm SR}(\mu)}{{\rm d}\mu} + \left(1-e^{-\lambda}-\lambda e^{-\lambda}\right)F_{\rm SR}^{(2+)}(\mu) \tag{4.8.4}$$

where λ is the number of elastic free paths corresponding to the path-length s,

$$\lambda = \Sigma_{SR}s \;, \tag{4.8.5}$$

and the screened Rutherford cross section is given in Eq. (4.7.1). The distribution $F_{\rm SR}^{(2+)}$ is the normalized multiple scattering distribution that results from at least two elastic collisions described by the screened Rutherford cross section,

$$\begin{aligned} F_{\rm SR}^{(2+)}(\mu) &= \sum_{l=0}^{\infty}\left(l+\frac{1}{2}\right)P_l(\mu)j_l^{(2+)} \\ j_l^{(2+)} &= \frac{\exp(-G_{l,{\rm SR}})+[1+\lambda-G_{l,{\rm SR}}]\exp(-\lambda)}{1-\exp(-\lambda)-\lambda\exp(-\lambda)} \end{aligned} \tag{4.8.6}$$

Here, we have put the subscript "SR" on the moments G_l to explicitly state that they are calculated using the screened Rutherford cross section. Equation (4.8.6) can be put in a more tractable form by a variable change,

$$u = (1+a)\left(1-\frac{2a}{1-\mu+2a}\right) \tag{4.8.7}$$

where the parameter a is chosen such as to make $q_{\rm SR}^{(2+)}$,

$$q_{\rm SR}^{(2+)}(u) = F_{\rm SR}^{(2+)}(\mu)\,\frac{{\rm d}\mu}{{\rm d}u} \;, \tag{4.8.8}$$

"as flat as possible". After some straightforward manipulations one obtains [1]

$$a = \kappa + \sqrt{\kappa^2+\kappa} \tag{4.8.9}$$

with the short hand notation

$$\begin{aligned} \kappa &= \frac{\langle 0\rangle - 2\langle 1\rangle + \langle 2\rangle}{4\langle 1\rangle} \\ \langle n\rangle &= \sum_{l=0}^{\infty}\left(l+\frac{1}{2}\right)j_l^{(2+)}\sum_{m=0}^{\infty}\left(m+\frac{1}{2}\right)j_m^{(2+)}\int_{-1}^{1}{\rm d}\mu\;\mu^n P_l(\mu)P_m(\mu) \;, \end{aligned} \tag{4.8.10}$$

The quantity $\omega^2 = a/\eta$ (η is the screening parameter) is a function of λ and has a slight dependence on η. In terms of simulation efficiency, it is better to ignore the η dependence of

ω and use a fit to the data obtained at run time from Eq. (4.8.9,4.8.10) for $\eta \to 0$. We use

$$\frac{\omega^2}{\lambda+4} = \begin{cases} 1.347 + t(0.209364 - t(0.45525 - t(0.50142 - 0.081234t))) & \text{if } \lambda < 10, \\ -2.77164 + t(2.94874 - t(0.1535754 - 0.00552888t)) & \text{else} \end{cases} \qquad (4.8.11)$$
$$t = \ln\lambda$$

The dependence of the distribution $q_{\rm SR}^{(2+)}(u)$ on λ and η is rather weak, as can be seen in

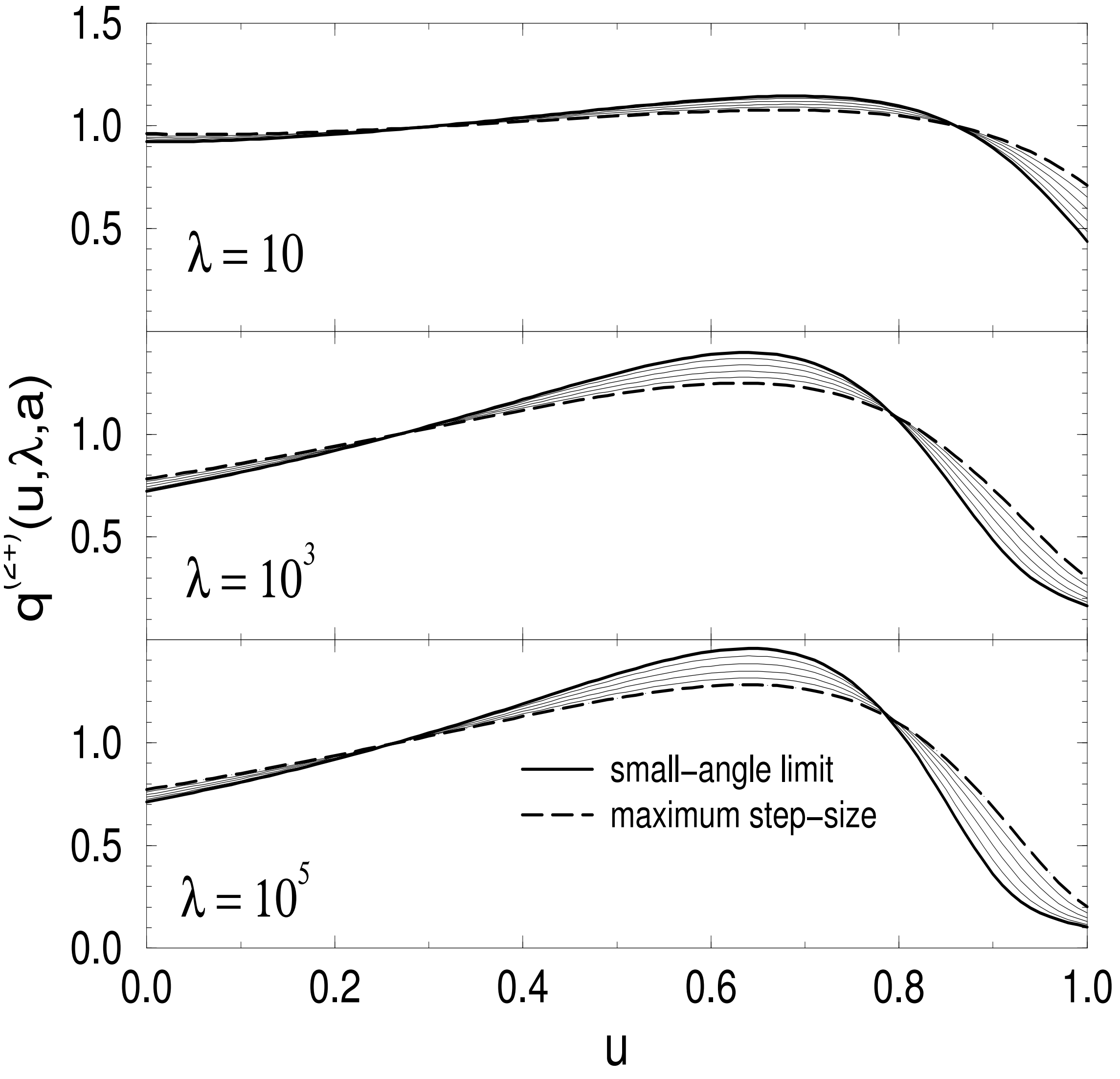


Figure 14: The $q_{\rm SR}^{(2+)}$ distribution for three step-lengths. The curve labelled "small-angle" limit is the distribution for $\eta \to 0$ (infinite energy), the curve for the maximum step-size corresponds to the maximum step-size for condensed history steps for which the data base was generated ($G_1 < 0.5$, see section 2.4.9.)

Fig. 14. $q_{\rm SR}^{(2+)}$ can therefore be interpolated very accurately during run time using linear interpolation in $\ln\lambda$ and $G_{1,{\rm SR}}$ from a pre-computed table. Here $G_{1,{\rm SR}}$ denotes the first GS moment, see Eq. (4.8.2), resulting from screened Rutherford elastic scattering,

$$G_{1,\mathrm{SR}} = 2\lambda\eta \left[(1+\eta) \ln\left(1 + \frac{1}{\eta}\right) - 1 \right] \tag{4.8.12}$$

The $q_{\mathrm{SR}}^{(2+)}$ data which are in a form of a 3 dimensional alias table, are stored in the file `newms.data` and read in by subroutine `init_ms_SR`.

To take into account energy loss, one uses the multiple scattering distribution for an "effective" step energy which is determined from the requirement [5]

$$\frac{G_2(E_0,E)}{G_1(E_0,E)} = \frac{\kappa_2(E_{\mathrm{eff}})}{\kappa_1(E_{\mathrm{eff}})} \tag{4.8.13}$$

for a path-length s_{eff},

$$s_{\mathrm{eff}} = \frac{G_1(E_0,E)}{\kappa_1(E_{\mathrm{eff}})} \ . \tag{4.8.14}$$

These two requirements guarantee that the first two GS moments are exactly reproduced, and gives at the same time a very accurate approximation for higher order GS so that the resulting multiple scattering distribution is virtually identical to the MS distribution calculated via numerical integration of the moments G_l, see Fig. 2 of Ref. [5].

For the screened Rutherford cross section the effective energy and path-length are given by

$$\begin{aligned} E_{\mathrm{eff}} &= E_0 \left[1 - \frac{\epsilon}{2} - \frac{\epsilon^2}{12(2-\epsilon)} \left(\frac{5\tilde{\tau}^2 + 10\tilde{\tau} + 6}{(\tilde{\tau}+1)(\tilde{\tau}+2)} + 2b(\tilde{E}) \right) + O(\epsilon^3) \right] \\ s_{\mathrm{eff}} &= s \left(1 - \frac{\epsilon^2}{3(2-\epsilon)} \, \frac{\tilde{\tau}^4 + 4\tilde{\tau}^3 + 7\tilde{\tau}^2 + 6\tilde{\tau} + 4}{(\tilde{\tau}+1)^2(\tilde{\tau}+2)^2} + O(\epsilon^4) \right) \end{aligned} \tag{4.8.15}$$

where we have defined

$$b(E) = \frac{E}{C(E)} \, \frac{\mathrm{d}C(E)}{\mathrm{d}E} \ , \qquad C(E) = L(E)\beta^2 \ , \tag{4.8.16}$$

β denoting the particle velocity in units of the speed of light. In the above equations $\epsilon = (E_0 - E)/E$ and $\tilde{\tau} = 1/2(E_0 + E)/m$.

With all this, the algorithm for sampling multiple elastic scattering angles is as follows:

1. Calculate s_{eff} and E_{eff} from Eq. (4.8.15).
2. Calculate λ from Eq. (4.8.5) and (4.7.9), $t = \ln\lambda$ and η from Eq. (4.7.8)
3. Pick a random number number r_1.
4. If $r_1 < e^{-\lambda}$, then the scattering angle is zero, return control to the calling routine
5. Else if $r_1 < e^{-\lambda}(1+\lambda)$, then sample μ from the single scattering distribution according to Eq. (4.7.10), return control to the calling routine

6. Else, we must sample μ from the $q_{\rm SR}^{(2+)}$ distribution. Calculate $G_{1,{\rm SR}}$ from Eq. (4.8.12), ω^2 from Eq. (4.8.11) and $a = \omega^2 \eta$
7. Determine the table look-up indices for $\ln \lambda$ and $G_{1,{\rm SR}}$
8. Sample u from the corresponding $q_{\rm SR}^{(2+)}(u)$ distribution
9. Deliver μ,

$$\mu = \frac{2au}{1-u+a} \tag{4.8.17}$$

2.4.8.ii Multiple elastic scattering with spin effects

In principle, the approach developed in Ref. [1] could be applied to more complicated single elastic scattering cross sections than the screened Rutherford cross section. We have found, however, that the direct use of this approach for the cross section with spin effects (see section 2.4.7.ii) does not lead to a satisfactory interpolation accuracy. We have therefore implemented a rejection technique for the sampling of multiple scattering angles when `SPIN_EFFECTS = .true.`.

We can write the multiple scattering distribution that results from at least 2 elastic scattering processes as

$$F^{(2+)}(\lambda,\eta,Z,\mu) = F_{\rm SR}^{(2+)}(\lambda,\eta,\mu) R(\lambda,\eta,Z,\mu) \tag{4.8.18}$$

where $F_{\rm SR}^{(2+)}$ is the 2+ distribution from a screened Rutherford cross section for a path-length corresponding to λ elastic mean-free paths and a screening angle η and $R(\lambda,\eta,Z,\mu)$ is defined as

$$R(\lambda,\eta,Z,\mu) = \frac{F^{(2+)}(\lambda,\eta,Z,\mu)}{F_{\rm SR}^{(2+)}(\lambda,\eta,\mu)} \tag{4.8.19}$$

(*i.e.* Eq. (4.8.18) is the identity $F^{(2+)} = F^{(2+)}$). The function $R(\lambda,\eta,Z,\mu)$ is a three dimensional surface for each medium. Numerical experiments show that the way which requires the minimum amount of pre-calculated data to interpolate the function $R(\lambda,\eta,Z,\mu)$ between pre-calculated data is to use for each medium

- Linear interpolation in the quantity q,

$$q = \frac{2G_{1,{\rm SR}}(\lambda,\eta)}{1+2G_{1,{\rm SR}}(\lambda,\eta)} \; . \tag{4.8.20}$$

  Obviously q can take only values between zero and unity. For $q \to 0, R(\lambda,\eta,Z,\mu)$ converges to $R_{\rm mott}(E,Z,\mu)$, see section 2.4.7.ii. For $q \to 1, R(\lambda,\eta,Z,\mu)$ goes to unity for all angles μ. The rate at which the function R changes from the one limit to the other depends on the energy.

- Linear interpolation in β^2 for energies greater than 100 keV, linear interpolation in $\ln E$ for energies less then 100 keV.

- Linear interpolation in $\sin\theta/2 = \sqrt{(1-\mu)/2}$.

The pre-calculated data are stored in separate files for all elements in the directory `spinms` and used in subroutine `init_spin` to compute R for the media involved in the actual simulation. `init_spin` is called from `mscati` which is called from `HATCH`.

The sampling algorithm is then similar to the one given at the and of the previous section but involves an additional rejection loop in steps 4 and 8 using R_{mott} or R as a rejection function[13]. In addition, the calculation of the screening parameter η and the number of mean-free-paths λ involve additional correction factors, because both, η/η_0 and the parameter ξ which describe the contribution of sub-threshold inelastic collisions to angular deflections, are not constant, see section 2.4.7.ii.

It is worth showing an example of the influence of the inclusion of spin effects in the treatment of elastic scattering as a conclusion of this section. Figure 15 shows a comparison

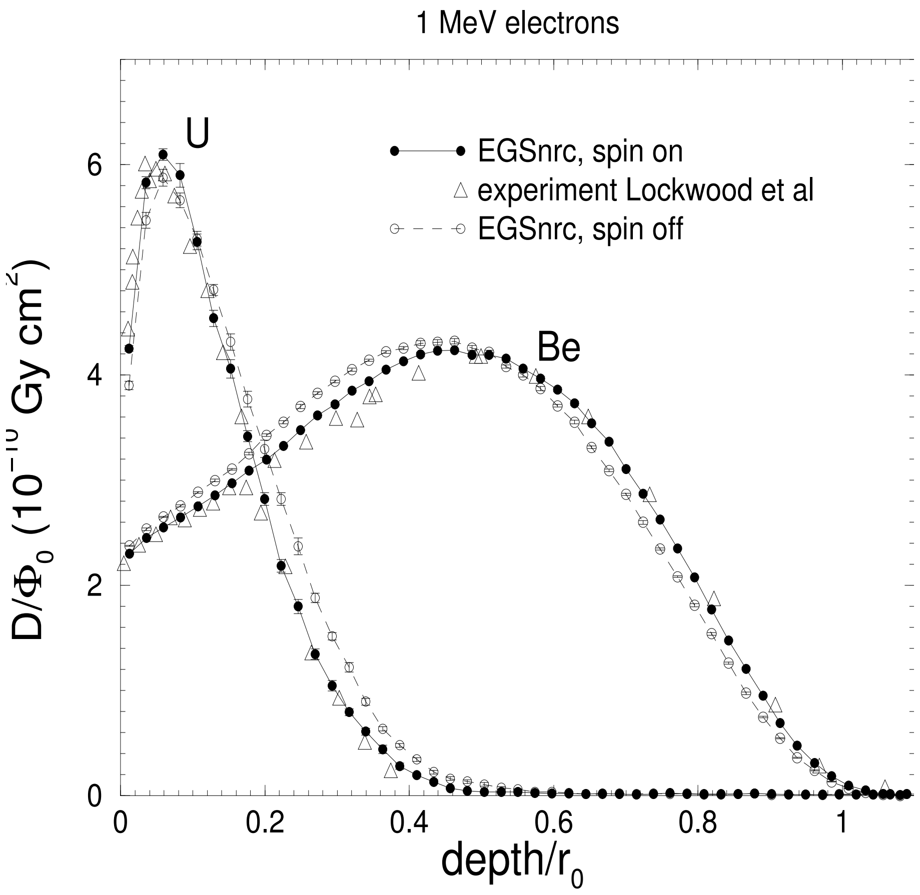


Figure 15: Depth-dose curves for a broad parallel beam of 1 MeV electrons incident on a Beryllium or Uranium target as a function of the CSDA range. The experimental data are from [87].

of calculated depth-dose curves in Beryllium and Uranium to measurements by Lockwood *et al* [87]. Both, the calculations and the measurements are absolute. The calculations with "spin on" are in much better agreement with the experiment. The effect of including spin is to make the effective range of electrons longer for low-Z materials and shorter for high-Z

[13]Both, R_{mott} and R are scaled to their maximum in the routine `init_spin`.

materials. It is also present for the energy range relevant for radiation therapy. The reason for not seeing disagreement between EGS4 calculations and measurements in this energy range is due to the rarity of measurements with precise knowledge of the incident energy. In depth-dose measurements with a known 20 MeV electron beam incident on water, comparisons to EGS4 required using an incident beam energy of 20.3 MeV to get good agreement[88]. In contrast, the calculations with EGSnrc with spin turned on are in good agreement with the measurements when using an incident energy of 20.0 MeV, in agreement with the known energy.

2.4.9 Electron-step algorithm

As mentioned in section 2.4.1, the transport between subsequent "catastrophic" collisions is described by Eq. (4.1.21) with Eq. (4.1.18) providing the link between energy and path-length. An exact solution of this equation is not known and so some approximate methods are required to relate the energy loss, the path-length, and the spatial displacement.

The simplest possible approach one could take is to ignore deflections due to multiple elastic scattering during the condensed history step and to transport the electron on a straight line along the initial direction of motion. In order for this approach to be accurate the CH steps must be sufficiently short so that the straight line approach does not represent to a severe approximation. Larsen has shown [89] that any condensed history algorithm will converge to the correct answer in the limit of sufficiently small step-sizes, provided multiple elastic scattering is faithfully simulated. However, making the step lengths very short may cause the simulation to become extremely inefficient. In addition, if the calculated results show step-size dependencies, one needs to perform a careful step-size study in order to determine when the result has converged.

Many electron-step algorithms that attempt to make corrections for deflections from the straight-line approach have been proposed over the years, *e.g.* [15, 17, 90, 91]. A detailed discussion of these algorithms is given in Ref. [4]. In general none of the algorithms available is accurate enough to allow the condensed history simulation between subsequent discrete events to be always done in a single step. Instead, the distances between discrete collisions is divided into smaller condensed history steps using a procedure to determine maximum acceptable lengths for the steps (step-size restrictions).

The electron-step algorithm employed in EGSnrc depends on the parameter `transport_algorithm` which is in `COMMON/ET_Control/`. If set to one, a slightly modified version of PRESTA's [17] path-length-correction (PLC) and lateral correlation algorithm (LCA) are employed. The PRESTA algorithm is known to underestimate lateral deflections (deflections perpendicular to the initial direction of motion), to underestimate longitudinal straggling and to produce a singularity in the distribution describing the lateral spread of electrons in a single condensed history step [91, 4]. The implications on the simulation results depend on the actual situation under investigation. At high energies, where elastic scattering is weak, PRESTA is perhaps sufficiently accurate. Its use for low energy applications is not recommended. One of us has shown, for instance, that there is up to 8% variation of the calculated ion chamber response subject to a ^{60}Co beam when using PRESTA's transport algorithm [6].

If the parameter `transport_algorithm` is set to zero (it's default value), the electron-step algorithm developed in Ref. [4] and later refined in [5] is employed. This algorithm is constructed in such a way as to reproduce up to second order spatial moments of the fluence $\Phi_0(\vec{x}, \vec{\Omega}, E, t)$, which are known from the theory of Lewis [92] for transport in an infinite, homogeneous medium. This algorithm was shown to produce step-size independent results for ion chamber simulations and backscattering [5, 6], two of the most difficult tasks for condensed history Monte Carlo codes.

For completeness, we give a brief summary of the transport algorithms available in EGSnrc. Final positions are in a frame where the electron is initially at the origin and moving along the positive z-axis. Positions in the actual simulation co-ordinate system are obtained using the appropriate rotations and spatial translations.

- `transport_algorithm = 1` (PRESTA)

$$\begin{aligned} x &= r_\perp \sin\theta \cos\phi \\ y &= r_\perp \sin\theta \sin\phi \\ z &= \langle z \rangle \\ r_\perp &= \mathrm{Min}\left(\frac{s}{2}, \sqrt{\frac{s^2 - \langle z \rangle^2}{\sin^2\theta}}\right) \end{aligned} \tag{4.9.1}$$

  where θ and ϕ are the polar and azimuthal scattering angles and $\langle z \rangle$ the average transport distance in the initial direction of motion for a path-length of s. In the original PRESTA implementation $\langle z \rangle$ was calculated from the theory of Molière, we use in EGSnrc the exact expression of Lewis [92], which is simply

$$\langle z \rangle = s \, \frac{1 - \exp(-G_1)}{G_1} \tag{4.9.2}$$

  if one neglects energy loss. A small correction arises if energy loss is taken into account, it is not included to be consistent with PRESTA where energy loss was also ignored.

- `transport_algorithm = 0` (EGSnrc default)
  The condensed history step is divided into two sub-steps and separate multiple elastic scattering angles θ_1, ϕ_1 and θ_2, ϕ_2 are sampled. The final scattering angle is then determined from θ_1, ϕ_1,θ_2, ϕ_2, *e.g.*

$$\cos\theta = \cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2 \cos(\phi_1 - \phi_2) \ . \tag{4.9.3}$$

The final position is calculated as follows:

$$
\begin{aligned}
x &= s\Big[\eta\delta\sin\theta_1\cos\phi_1 + \eta(1-\delta)\sin\theta_2(\cos\phi_1\cos\phi_2 - \cos\theta_1\sin\phi_1\sin\phi_2) + a_2\sin\theta\cos\phi\Big] \\
y &= s\Big[\eta\delta\sin\theta_1\sin\phi_1 + \eta(1-\delta)\sin\theta_2(\sin\phi_1\cos\phi_2 + \cos\theta_1\cos\phi_1\sin\phi_2) + a_2\sin\theta\sin\phi\Big] \\
z &= s\Big[a_1 + \eta\delta\cos\theta_1 + \eta(1-\delta)\cos\theta_2 + a_2\cos\theta\Big] \\
a_1 &= \frac{1-\eta}{2}(1+\alpha_1) \\
a_2 &= \frac{1-\eta}{2}(1-\alpha_1) \\
\delta &= \frac{1}{2} + \frac{\sqrt{6}}{6} - \left(\frac{1}{4\sqrt{6}} - \gamma\frac{4-\sqrt{6}}{24\sqrt{6}}\right)G_1 + \alpha_2 \\
\gamma &= \frac{G_2}{G_1}
\end{aligned}
\tag{4.9.4}
$$

Here, η is a random number sampled from $2\eta d\eta$ (and not the screening parameter) and α_1 and α_2 are energy loss corrections derived in Ref. [5],

$$
\begin{aligned}
\alpha_1 &= -\frac{\kappa_1'(\tilde{E})}{\kappa_1(\tilde{E})}\,\frac{\Delta E}{2} + O(\Delta E^2) \\
\alpha_2 &= \left(\frac{\kappa_2'(\tilde{E})}{\kappa_2(\tilde{E})} - \frac{\kappa_1'(\tilde{E})}{\kappa_1(\tilde{E})}\right)\frac{\Delta E}{2\sqrt{6}} + O(\Delta E^2)
\end{aligned}
\tag{4.9.5}
$$

where ΔE is the sub-threshold energy loss associated with the step, $\tilde{E}$ the average step energy and κ_1' and κ_2' derivatives of the moments κ_1 and κ_2 (see Eq. (4.8.3)) with respect to E. If elastic scattering is described by the screened Rutherford cross section, the energy loss corrections α_1 and α_2 are given by

$$
\begin{aligned}
\alpha_1 &\approx \left(\frac{2+2\tilde{\tau}+\tilde{\tau}^2}{(1+\tilde{\tau})} - \frac{1+\tilde{\tau}}{\ln(1+1/\tilde{\eta})(1+\tilde{\eta})-1)}\right)\frac{\Delta E/\tilde{E}}{(2+\tilde{\tau})} \\
\alpha_2 &\approx \frac{\Delta E/\tilde{E}}{\sqrt{6}(1+\tilde{\tau})(2+\tilde{\tau})\Big[\ln(1+1/\tilde{\eta})(1+\tilde{\eta})-1)\Big]\Big[\ln(1+1/\tilde{\eta})(1+2\tilde{\eta})-2)\Big]}
\end{aligned}
\tag{4.9.6}
$$

where $\tilde{\tau} = \tilde{E}/m$ and $\tilde{\eta}$ is the screening parameter for the midpoint energy $\tilde{E}$. These equations are used in the subroutine `msdist_pII` which implements condensed history electron transport according to the algorithm given in Eq. (4.9.4). Strictly speaking, when spin effects are "turned on", one should use the energy loss corrections derived from the moments resulting from the corresponding elastic scattering cross section. We have left this refinment for the future because (i) Given the fact that the corrections involve the calculation of derivatives and that the cross sections with spin are available only in numerical form, the implementation is more difficult (ii) Eq. (4.9.6) does take into account the main energy dependence of the corrections, deviations from Eq. (4.9.6) are either higher order in ΔE or have small coefficients.

The algorithm given in Eq. (4.9.4) reproduces first and second order spatial moments to better than 0.1% for $G_1 \leq 0.5$ if energy loss is neglected (G_1 is defined by Eq. (4.8.2) and is a function of pathlength, so this condition imposes a maximum allowed step size). This accuracy is maintained when energy loss is taken into account via the corrections given in (4.9.5) if the maximum fractional energy loss per step is restricted to 25%. These two condensed history step-size restrictions are controlled via the parameters `ximax` (corresponding to the maximum allowed G_1) and `ESTEPE` which are in `COMMON/ET_Control` and set by default to 0.5 and 0.25.

Due to the use of the energy loss corrections derived from the screened Rutherford cross section also for elastic scattering with spin, in this case the deviation from the theoretical expectation is slightly larger and exceeds 0.1% for `ESTEPE` between 0.15 and 0.2. If such a high precision is required for your application, the simplest solution is to use step sizes that don't exceed 20% energy loss per step when spin is on.

2.4.10 Boundary crossing algorithm

The electron-step algorithms discussed in section 2.4.9 are valid only for transport in an infinite, homogeneous medium. In practical situations one has to deal with interfaces between different materials. If an electron is close to an interface with another material, portions of its curved path may be in this different material and so the trajectory different than simulated. In the original EGS4 version, this problem associated with the condensed history simulation of electron transport, which has become known as the interface artifact [93], was entirely ignored. To address the interface artifact, a refined boundary crossing algorithm (BCA) was incorporated into PRESTA. According to this algorithm, the electron is not allowed to take a step longer than $t_\perp$, $t_\perp$ being the perpendicular distance to the closest boundary, unless $t_\perp$ becomes smaller than t_{min}, a user defined minimum step-length for boundary crossing. For $t_\perp < t_{min}$, lateral deflections are switched off and the particle is transported, as in the case of standard EGS4, on a straight line to the boundary.

In a more recent paper [94], Foote and Smyth have demonstrated that the approach of forcing a multiple scattering event at the boundary causes a singularity in the simulated particle fluence. The singularity results from the fact that there is a non-zero probability for scattering parallel to the boundary. Note that in real life, particles moving parallel to the boundary do not cross it and therefore do not contribute to the planar fluence. Although the singularity is present in any situation over a distance of the order of t_{min}, it will be observed, *e.g.* as a dose over-prediction, only if the size of a scoring region is comparable to t_{min} due to averaging. One of us has developed an analytical expression for the effect when (semi)-charged particle equilibrium is present and has demonstrated that in the case of a small air cavity surrounded by a dense material (ionization chamber), the dose over-prediction may be up to 3.5% [6].

To overcome this problem we have implemented into EGSnrc an exact boundary crossing algorithm, *i.e.* the simulation goes over into a single elastic scattering mode whenever an electron comes closer to a boundary than t_{min}. Whereas in the case of PRESTA one of the criteria to determine t_{min} was to assure the applicability of Molière's MS theory, in EGSnrc the only criterion for t_{min} is efficiency (because of the multiple scattering theory employed which is applicable for all step-sizes). It turns out that single scattering simulation becomes more efficient than condensed history simulation at about 3 elastic mean free paths. This is taken as the default value for the parameter `$SKIN_DEPTH_FOR_BCA` which determines t_{min}. Note that `$SKIN_DEPTH_FOR_BCA` is in elastic mean free paths and not in length units. `SKINDEPTH_FOR_BCA` is in `COMMON/ET_Control/`.

The investigation of Ref. [6] shows that the strength of the effect of the fluence singularity due to forcing multiple elastic scattering events exactly at boundaries is proportional to the first GS moment G_1. At high energies G_1 is very small for reasonable step-sizes and so, a single scattering simulation in the vicinity of boundaries is potentially wasteful. Hence we decided to keep the original PRESTA boundary crossing algorithm in EGSnrc. The selection between exact BCA and PRESTA's BCA is made via the parameter `BCA_ALGORITHM` which is in `COMMON/ET_Control/`. If set to 0 (the default), exact boundary crossing is employed, 1 means PRESTA's BCA is used. If PRESTA's BCA is selected and the parameter `SKIN_DEPTH_FOR_BCA` set to 0, EGSnrc will calculate t_{min} according to the procedure in the original PRESTA implementation. It is worth noticing that if `BCA_ALGORITHM` is set to 1 and the parameter `SKINDEPTH_FOR_BCA` to a very large number, the entire simulation will run without lateral deflections in the individual condensed history steps and so mimic the original EGS4 behaviour. If, on the other side, `BCA_ALGORITHM` is set to 0 and `SKINDEPTH_FOR_BCA` to a very large number, the entire simulation will be in a single scattering mode. These are also the two options available, if the geometry under investigation is too complex to allow for the calculation of $t_{\perp}$.

2.4.11 Other condensed history aspects

There are two additional aspects of the implementation of the condensed history technique that deserve some consideration:

1. Calculation of path-lengths corresponding to a given energy loss and vice versa. The two quantities are related via Eq. (4.1.18).
2. Sampling distances between discrete interactions. Since the cross sections are energy dependent and the energy changes via sub-threshold (continuous) energy loss, sampling distances between discrete interactions is slightly more complicated for electrons than for photons.

2.4.11.i Energy loss evaluation

The energy loss ΔE due to sub-threshold processes for a condensed history step of length s is

$$\Delta E = \int_0^s \mathrm{d}s' L(s') \qquad (4.11.1)$$

where the restricted stopping power is a function of s' in the sense of Eq. (4.1.18). Equation (4.11.1) must be evaluated "on the fly" for each condensed history step and so a fast and accurate procedure is needed. In the original EGS4 implementation the above integral was approximated with

$$\Delta E \approx L(E_0)s \qquad (4.11.2)$$

where E_0 is the energy at the beginning of the step. The PRESTA algorithm made the refinement

$$\Delta E \approx L\Big(E_0 - L(E_0)s/2\Big)s \tag{4.11.3}$$

The motivation for this is Euler's integration formula

$$\int\limits_{x-\Delta x/2}^{x+\Delta x/2} \mathrm{d}x' f(x') = f(x)\Delta x + O(\Delta x^3) \tag{4.11.4}$$

and so, one might expect at a first sight an $O(\Delta E^3)$ error due to the use of Eq. (4.11.3). A more careful examination of the integral reveals that Eq. (4.11.3) still has an $O(\Delta E^2)$ error as the original EGS4 approach (although the coefficient of the $O(\Delta E^2)$ term is smaller). A more accurate approach was presented in Ref. [5], but for the official release of the system we decided to use another approach which is simpler but not less accurate.

In EGS4 (and also EGSnrc), a linear interpolation in $\ln E$ is used at run time to calculate various quantities, the restricted stopping power among them, *i.e.*

$$L(E) = a_i + b_i \ln E \quad \text{for } E_i \le E < E_{i+1} \tag{4.11.5}$$

where E_i are the bin edge energies. This approach has proven to be very accurate (and if not accurate enough, the accuracy can always be increased by increasing the number of interpolation bins). We now define R_i,

$$R_i = \int\limits_{E_1}^{E_i} \frac{\mathrm{d}E'}{L(E')} \ , \tag{4.11.6}$$

where E_1 is the first energy in the interpolation table (usually slightly smaller than `TE`). R_i is the path-length that an electron with energy E_i will travel until local absorption if losing energy only via sub-threshold processes (range). In EGSnrc the quantities R_i are stored in the array `range_ep` which is in `COMMON/ELECIN/` for each medium. Using the R_i's we can calculate the range $R(E)$ for arbitrary energies from

$$R(E) = R_i + \int\limits_{E_i}^{E} \frac{\mathrm{d}E'}{a_i + b_i \ln E'} \approx R_i + \frac{E-E_i}{L_i}\left(1 - \frac{b_i}{L_i}\,\frac{\epsilon}{2} + \frac{b_i(2b_i+L_i)}{L_i^2}\,\frac{\epsilon^2}{6} \pm \cdots\right) \tag{4.11.7}$$

where E_i is the lower edge energy of the bin to which E belongs, L_i is a short hand notation for $L(E_i) = a_i + b \ln E_i$ and $\epsilon = E/E_i - 1$ is usually very small[14], so that the expansion up to second order in ϵ is sufficiently accurate. The range of electrons is calculated according to Eq. (4.11.7) at the beginning of each condensed history step in EGSnrc. If now the decision is made to perform a step with length s, the range of the electron at the end of the step is $R - s$ and one can use Eq. (4.11.7) to calculate the corresponding final step energy after finding the bin to which the new energy belongs using the ranges R_i. If, on the other side,

[14] *e.g.* for a data set for the entire energy range of the applicability of EGSnrc, 1 keV to 10 GeV, $\epsilon < 0.107$ using 150 interpolation bins. Normally ϵ is much smaller.

the step-size is determined via a given energy loss ΔE, one calculates $R(E-\Delta E)$ from Eq. (4.11.7) and uses $R(E) - R(E-\Delta E)$ as the corresponding step-length s.

The appeal of the approach described above lies in its simplicity and generality. If one day it is decided that our understanding of restricted stopping powers is incomplete and a change is required, this approach will be still applicable as long as a logarithmic interpolation is used. At this point one should mention that the accurate knowledge of the electron range is absolutely essential for the accurate calculation of energy loss. Under no circumstances the variable `range` (which holds the value of R) should be overwritten by the user!

An additional useful feature that results from the knowledge of the electron range in the current material is that electron range rejection can be applied. If the parameter `i_do_rr` is set to unity for the current region, the electron energy is less than `e_max_rr` (this is total energy, including rest energy!) and the electron range R is less than $t_\perp$ (see section 2.4.10), the simulation of the current electron is terminated and its entire energy deposited locally.

2.4.11.ii Distances between discrete interactions

If one uses the path-length as the variable to measure distances between discrete interactions, the path-length s to the next discrete event must be sampled from

$$\int\limits_0^s \mathrm{d}s' \Sigma^{(\mathrm{tot})}(s') = -\ln r \tag{4.11.8}$$

where r is a uniformly distributed random number between zero and unity. To avoid the numerically intensive solution of the above equation, the so called fictitious cross section method is employed in EGS4: an additional, fictitious interaction is introduced, its total cross section $\Sigma_\mathrm{f}(s')$ is chosen such that

$$\Sigma^{(\mathrm{tot})}(s') + \Sigma_\mathrm{f}(s') \equiv \Sigma_0 = \mathrm{const}\ . \tag{4.11.9}$$

The path-length to the next interaction, real or fictitious, is then simply

$$s = -\frac{\ln r}{\Sigma_0}\ . \tag{4.11.10}$$

Once at the interaction site, the interaction is rejected with the probability $1-\Sigma^{(\mathrm{tot})}(s)/\Sigma_0$ (or, with other words, a fictitious interaction takes place with the probability $1-\Sigma^{(\mathrm{tot})}(s)/\Sigma_0$). In order this approach to work properly, Σ_0 must be greater than $\Sigma^{(\mathrm{tot})}(s)$ for all s. Based on the observation that at high energies the discrete interaction cross section is a monotonic increasing function of energy, the cross section for the initial energy is used in EGS4. This approach fails for threshold energies for delta particle production less than about $m/7$, as pointed out by Rogers in 1984 [95]. Although known for a long time, this problem was never corrected. The modification proposed by Ma and Nahum in 1992 [96] is also biased, as shown in Ref. [5]. If one would attempt to employ the fictitious cross section method using the global cross section maximum as Σ_0, the simulation would become extremely inefficient for low values of T_c and k_c. This can easily be understood from Fig. 16 which shows the total cross section in graphite for two different cutoff energies.

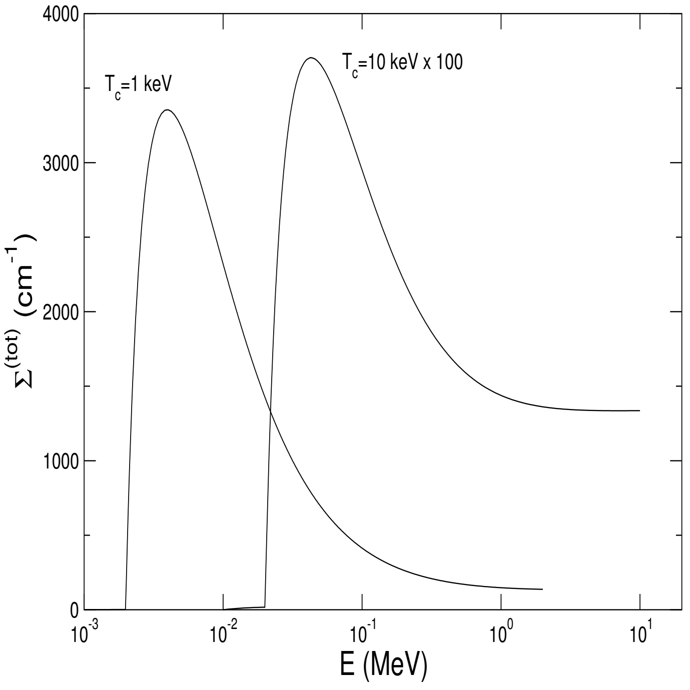


Figure 16: The total cross section for discrete interactions in graphite, calculated with $T_c = k_c = 1$ keV and $T_c = k_c = 10$ keV (scaled by a factor of 100) as a function of kinetic energy.

In EGSnrc the distance between discrete interactions is measured in units of the energy loss due to sub-threshold processes. The relevant total cross section is then $\tilde{\Sigma}^{(\text{tot})}(E) = \Sigma^{(\text{tot})}(E)/L(E)$ (see the general discussion of the transport equation in section 2.4.1). This cross section, shown in Fig. 17 for graphite and gold for two different values of T_c and k_c, is much flatter than $\Sigma^{(\text{tot})}$ (at least for low cutoff energies) and has a single maximum. This maximum is determined after the PEGS data becomes available and is used at run time to sample energy losses between discrete interactions. This saves the necessity of evaluating the cross section at the beginning of each discrete interaction loop (the `TSTEP LOOP`) in subroutine `ELECTR`. The procedure described above is then used to compute the corresponding path-length.

2.4.11.iii Transport in electromagnetic fields

Transport of charged particles through matter under the influence of electric and magnetic fields is now possible using the approach proposed by Bielajew[97]. In this approach, charged particles are forced to take small enough steps so that the external fields do not change significantly, and energy losses as well as angular deflections are negligible. Under these assumptions the effect of the electromagnetic field can be superimposed upon the field-free transport, and the final direction calculated to first order as

$$\Delta\vec{u} = \Delta\vec{u}_{ms,ret} + \Delta\vec{u}_{em} \tag{4.11.11}$$

where $\Delta\vec{u}_{ms,ret}$ is the angular deflection due to elastic and inelastic scattering which can be

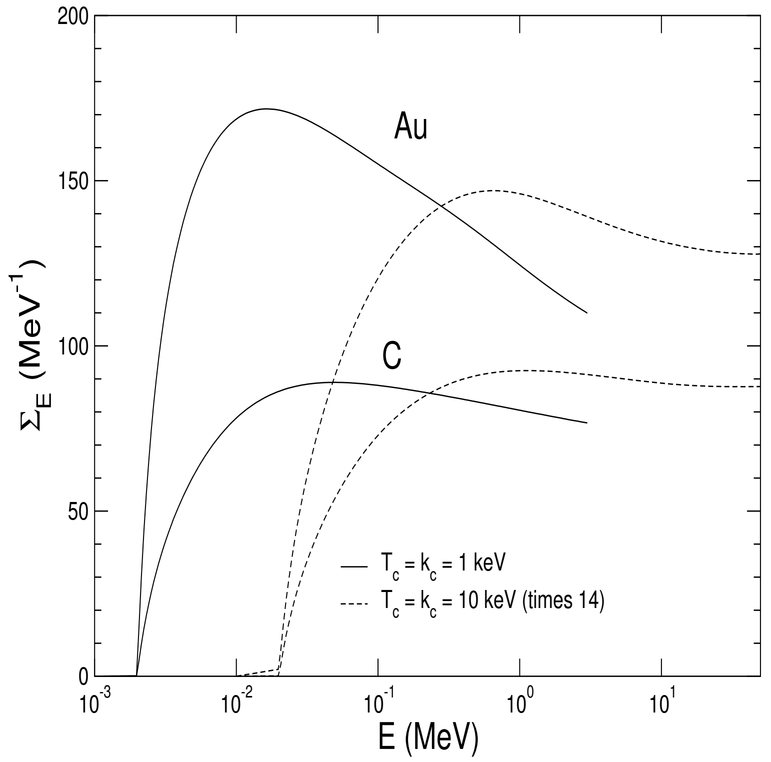


Figure 17: The total cross section per unit energy loss, $\tilde{\Sigma}^{(\mathrm{tot})}(E)$, for discrete interactions in graphite and gold, calculated with $T_c = k_c = 1$ keV and $T_c = k_c = 10$ keV (scaled with a factor of 14 for better visibility) as a function of kinetic energy.

calculated using EGSnrc's electron transport algorithm, and $\Delta\vec{u}_{em}$ is the angular deflection due to the electromagnetic field, which, to first order, can be obtained from

$$\Delta\vec{u}_{em} = \frac{q \cdot s}{m_0 \gamma v_0^2} \left[\vec{E}_0 - \vec{u}_0(\vec{u}_0 \cdot \vec{E}_0) + \vec{v}_0 \times \vec{B}_0 \right] . \tag{4.11.12}$$

The final position can be obtained using

$$\vec{x}_f = \vec{x}_0 + \vec{u}_0 s + \frac{s}{2} \left(\Delta\vec{u}_{ms,ret} + \Delta\vec{u}_{em} \right) . \tag{4.11.13}$$

In the current implementation, lateral displacement due to the Lorentz force is ignored, and only a condensed history (CH) step s is taken along $\vec{u}_0$ using EGSnrc's electron transport algorithm. This simplification relies on the assumption that accounting only for the change in direction due to the Lorentz force is accurate enough when using very small steps.

There are several criteria to restrict the step size based on energy loss, change in the electromagnetic field and change in particle direction. In the case of a static $\vec{B}$ field ($\vec{E}$ = 0, $\vec{B}$ = $\vec{B}_0$) only the restriction of small changes in the particle direction is relevant. According to Eq.(4.11.12), $|\Delta\vec{u}_{em}|$ should only be allowed to take a value $\delta \ll 1$ from where the restricted step size follows as

$$s = \delta \cdot \frac{E_o \gamma_0 \beta^2}{q \left| \vec{v}_0 \times \vec{B}_0 \right|} = \delta \cdot r_g \tag{4.11.14}$$

the quantity r_g is the well known gyroradius of the particle's trajectory. A value of δ = 0.02 was used to reproduce the analytical trajectory of electrons in vacuum in reference [97]. For more details on the implementation of this algorithm the reader is encouraged to read the very detailed chapter on this topic by Bielajew[97]. For instructions on how to enable transport under electromagnetic fields please refer to section 3.14.1.

3 EGSnrc Reference Manual

3.1 Introduction

This section is based on Appendix 2 of SLAC-265[11] but substantially updated and changed to represent the EGSnrc system rather than EGS4. There have been minor modifications to reflect the EGSnrcMP environment but these are described more fully in PIRS-877[14].

3.1.1 Use of Mortran3

Starting with EGS2, the EGS Code System has been written in an extended Fortran language known as Mortran[98]. Section 8 (page 274) presents a brief overview of the elements of Mortran3 which are needed for users of EGS.

Mortran is a very powerful pre-processor which was ahead of its time back in the 70's and 80's. Today many of its features are available in other languages. Nonetheless we have continued to use Mortran3 because there are so many user codes available in Mortran3 that it makes no sense to abandon it. It is also a very structured language which allows for easy in-line documentation.

Although there might be some resistance by users of EGS to learn another language, we would like to point out two facts:

- The Mortran language (excluding macros) is trivial to learn by those who program in Fortran.
- EGS can be set-up and run by writing entirely in Fortran or some other language should the user so desire.

We would encourage EGS users not to do the latter, however, for this would truly defeat the real purpose for using Mortran—namely, the macro facility.

3.2 General Description of Implementation

The EGS code itself consists of two User-Callable subroutines, `HATCH` and `SHOWER`, which in turn call the other subroutines in the EGS code, some of which call three User- written subroutines, `HOWFAR`, `HOWNEAR` and `AUSGAB`. This is best illustrated with the aid of Fig. 18

To use EGS the user must write a "User Code." This consists of a MAIN program and the subroutines `HOWFAR`, `HOWNEAR` and `AUSGAB`, the latter three determining the geometry and output (scoring), respectively. Additional auxiliary subprograms might be included in the User Code to facilitate matters. The user can communicate with EGS by means of various `COMMON` variables. Usually MAIN will perform any initialisation needed for the geometry routines, `HOWFAR` and `HOWNEAR`, and sets the values of certain EGS `COMMON` variables which specify such things as names of the media to be used, the desired cutoff energies, and the distance unit to be used (e.g., inches, centimeters, radiation lengths, etc.). MAIN then calls

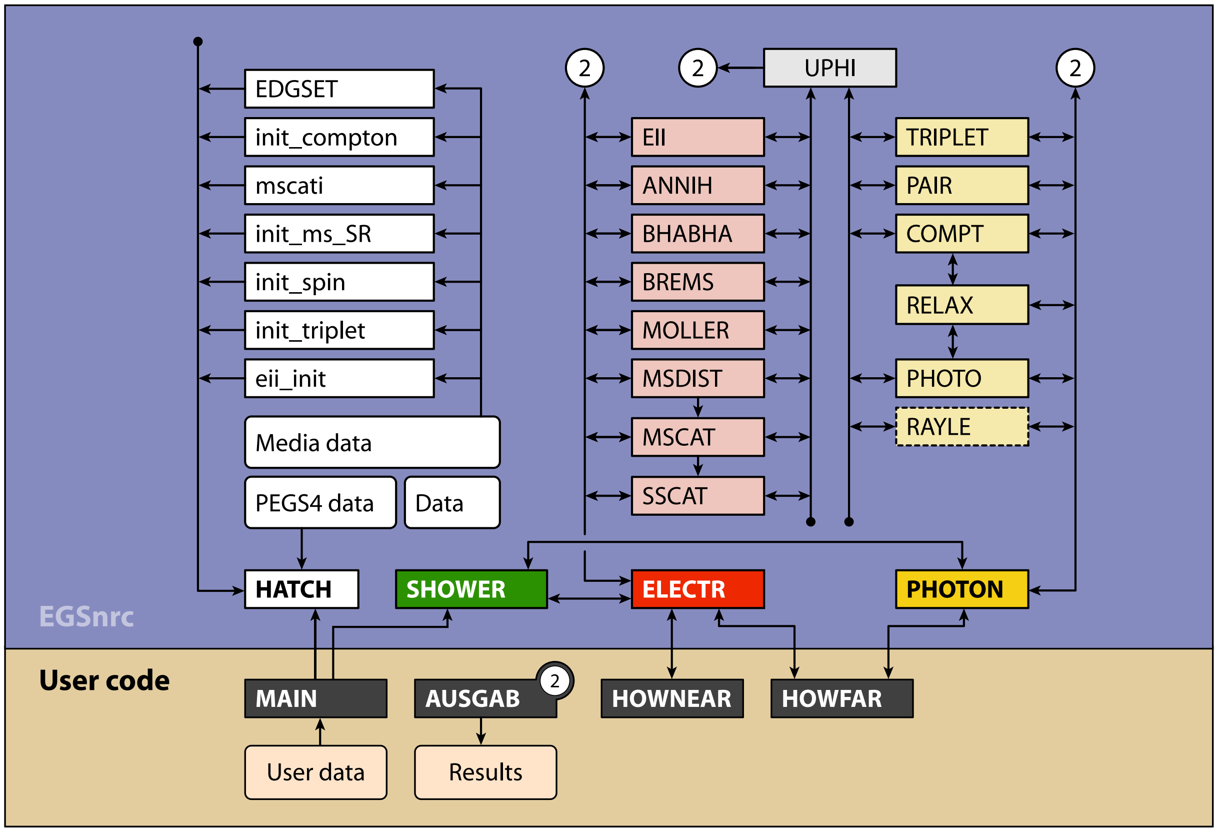


Figure 18: The structure of the EGSnrc code system when used with a user-code.

the `HATCH` subroutine which "hatches EGS" by doing necessary once-only initialisation and by reading material data for the media from a data set that had been previously created by PEGS. This initialisation completed, MAIN may then call `SHOWER` when desired. Each call to `SHOWER` results in the generation of one history (often referred to as a "case"). The arguments to `SHOWER` specify the parameters of the incident particle initiating the cascade.

In addition, macro definitions can be included in MAIN in order to control or over-ride various functions in EGS as well as in the User-Written codes.

In EGSnrc there are many new options compared to EGS4. The system defaults to a set of options which will do the most complete and accurate simulation that EGSnrc is capable of. In some cases this will imply overkill and a reduction in efficiency with no gain in accuracy (e.g. including atomic relaxation or bound Compton scattering for high energy photon calculations). The user has the ability to switch things on or off by setting various flags. So, eg, one could decide to run a calculation which is nearly equivalent to using the EGS4/PRESTA electron transport algorithm (see section 3.4.2.i). Similarly one can choose to model Klein Nishina Compton scattering instead of bound compton scattering by setting a flag.

One other class of new features in EGSnrc is the implementation within the code itself of several variance reduction techniques (range rejection and bremsstrahlung splitting being the main two) since by doing so, a much more efficient implementation is allowed. The user can, of course, completely ignore these features if so desired.

In summary, the user communicates with EGS by means of:

Subroutines

- HATCH — to establish media data
- SHOWER — to initiate the cascade
- HOWFAR& HOWNEAR — to specify the geometry
- AUSGAB — to score and output the results and to control variance reduction

COMMON blocks by changing values of variables

Macro definitions — re-definition of pre-defined features.

To reiterate, we shall refer to the MAIN/HOWFAR/AUSGAB combination (plus auxiliary subprograms and macros) as the User Code. The following sections discuss these things in greater detail.

3.3 The COMMON Blocks

Listed here are the `COMMON` blocks relevant to the user (and relevant variables contained in them) with a brief description of their functions. Their usage will be discussed in more detail in subsequent sections. The easiest way to declare any of the `COMMON` blocks is with the `COMIN` macro. For example, `COMIN/STACK,BOUNDS/;` will automatically expand to the correct `COMMON/STACK/;` and `COMMON/BOUNDS/;` forms.

Note that EGSnrc is, by default, coded completely using IMPLICIT NONE. This means that all parameters in `COMMON` are all explicitly typed. See section 3.4.1.ii for more info.

Table 3: Table describing the EGSnrc `COMMON`s which are accessible to the User.

Common Block	Variable	Description
BOUNDS	ECUT	Array of regions' charged particle cutoff energies(total) in MeV.
	PCUT	Array of regions' photon cutoff energies in MeV.
	VACDST	Distance to transport in vacuum (default=1.E8).
EGS-VARIANCE-REDUCTION		
	e_max_rr	real array ($MXREG) of maximum total energies at which to do range rejection if i_do_rr is set
	i_do_rr	integer array ($MXREG) of flags for range rejection in each region. 0⇒ not done (default); 1⇒ is done.
	i_play_RR	flag specifying if Russian Roulette played on a global basis
	i_survived_RR	an integer flag set every time Russian Roulette is played. If all the particles survive, it is set to 0 (which is the default if not played at all). It is set to n if n particles were eliminated via Russian Roulette on this interaction. It is 0 if a bound compton event is rejected.
	prob_RR	probability of survival if playing Russian Roulette
	n_RR_warning	an internal counter to mark how often Russian Roulette is asked for with prob_RR $\leq$ 0.0. A warning is printed for the first $MAX-RR-WARNING times (default 50).
	nbr_split	For nbr_split >1, nbr_split brems photons are sampled every time there is a brem interaction. The weight is reduced by 1/nbr_split. Default value is 1.0. If set to zero, no brem is generated and the electron loses no energy.

(cont ...)

EGSnrc COMMONs which are accessible to the User -continued

Common Block	Variable	Description
EPCONT	EDEP	Energy deposited in MeV (Double Precision).
	TSTEP	Distance to next interaction (cm)
	TUSTEP	Total (curved) step length requested before check with geometry.
	USTEP	straight step length calculated from TUSTEP.
	TVSTEP	Actual total (curved) step length to be transported.
	VSTEP	actual straight step length after truncation by geometry.
	IDISC	User discard request flag (to be set in `HOWFAR`). IDISC > 0 means user requests immediate discard, IDISC < 0 means user requests discard after completion of transport, and IDISC=0 (default) means no user discard requested. IDISC=99 or -99 means generate annihilation photons when positron is discarded.
	IROLD	Index of previous region.
	IRNEW	Index of new region.
	RHOF	Value of density correction (default=1) (i.e. ratio of real density to that of dataset.
	EOLD	Charged particle (total) energy at beginning of step in MeV.
	ENEW	Charged particle (total) energy at end of step in MeV.
	IAUSFL	Array(35) of flags for turning on various calls to `AUSGAB`. See table 6
	EKE	Electron kinetic energy in MeV.
	ELKE	Natural logarithm of EKE (this is not available for a step in vacuum).
	GLE	Natural logarithm of photon energy.
	E_RANGE	For electron `IARG`=0 steps, this is the range of the electron in the current units (see section 3.11.1).
	x[y][z]_final	position at end of step
	u[v][w]_final	direction at end of step (only used for electrons)

Note: the variable BETA2 is no longer available.

(cont ...)

EGSnrc COMMONs which are accessible to the User -contnued

Common Block	Variable	Description
ET-Control		Electron Transport Control.
	SMAXIR	array($MXREG) defining upper limit on step size in each region (in whatever units defined by DUNIT).(default=1.E10).
	ESTEPE	global energy loss constraint.(default=0.25).
	XIMAX	max. first GS moment per step (roughly half the average MS angle squared.(default 0.5).
	SKINDEPTH _FOR_BCA	distance from a boundary (in elastic MFP) at which to switch to one of the boundary crossing algorithms(BCAs).(default 3). If set 0 by the user initially and BCA_ALGORITHM = 1, then the code assigns a value consistent with BLCMIN in PRESTA-I, otherwise it is 3.0.
	TRANSPORT _ALGORITHM	integer flag telling which transport algorithm to use. 0$\Rightarrow$ PRESTA-II; 1$\Rightarrow$ PRESTA-I.(default 0).
	BCA _ALGORITHM	integer flag telling which BCA to use. 0$\Rightarrow$ use exact(single scattering) algorithm within SKINDEPTH_FOR_BCA of a boundary 1$\Rightarrow$ use multiple scattering but with no lateral deflections within SKINDEPTH_FOR_BCA of a boundary. Default is 0.
	SPIN _EFFECTS	logical variable, `.true.`$\Rightarrow$ use single & multiple scattering theories which include relativistic spin effects; `.false.` $\Rightarrow$ use single and multiple scattering theories based on Rutherford scattering. (default `.true.`)

EGSnrc COMMONs which are accessible to the User (cont)

Common Block	Variable	Description
MEDIA	MEDIA	array(24,$MXMED) of media names.
	NMED	Number of media being used (default=1).
	IRAYLM	Array($MXMED) of flags for turning on (=1) coherent (Rayleigh) scattering in various media. Set in `HATCH` based on values of IRAYLR.
	RLC	Array($MXMED) containing radiation lengths of the media in cm.
	RLDU	Array($MXMED) containing radiation lengths of the media in distance units established by DUNIT.
	RHO	Array($MXMED) containing density of the media in g/cm**3.
	MGE	Array($MXMED) number of photon mapped energy intervals for a given medium.
	MEKE	Array($MXMED) nuber of electron mapped energy intervals for a given medium.
	comp_xsections	character*16 variable holding the name of the file containing user-supplied Compton cross section data. Full name of the file is `$HEN_HOUSE/data/comp_xsections_compton.data`. Only used if `IBCMP`=2 (bound Compton, no doppler effect).
MISC*	MED	Array($MXREG) containing medium index for each region.
	DUNIT	The distance unit to be used. DUNIT=1 (default) establishes all distances in cm; whereas, DUNIT=2.54 establishes all distances in inches.
	KMPI	Fortran unit number (default=12) from which to read material data.
	KMPO	Fortran unit number (default=8) on which to "echo" material data (e.g., printed output, "dummy" output, etc.).
	RHOR	Array($MXREG) containing the density for each region (g/cm**3). If this is different than the default density for the medium for that region, the cross sections and stopping powers (with the exception of the density effect) are scaled appropriately.
	IRAYLR	Array($MXREG) of flags for turning on (=1) coherent (Rayleigh) scattering in various regions (default=1⇒ on).

* NOSCAT is no longer available since there is scattering on all steps.

(cont ...)

EGSnrc COMMONs which are accessible to the User (cont).

Common Block	Variable	Description
STACK		Note: This COMMON block contains the information about the particles currently in the shower. All of the following variables are arrays($MXSTACK) except `NP`, `NPold` and `LATCHI`.
	E	Total energy in MeV (Double Precision).
	X,Y,Z	Position of particle in units established by DUNIT.
	U,V,W	Direction cosines of particle (not necessarily normalized if table lookups used for sines—see section 3.4.1).
	DNEAR	A lower bound of distance from (X,Y,Z) to nearest surface of current region.
	WT	Statistical weight of current particle(default=1.0). To be used in conjunction with variance reduction techniques as determined by user.
	IQ	Integer charge of particle (+1,0,-1).
	IR	Index of particle's current region.
	NP	The stack pointer (i.e., the particle currently being pointed to). Also, the number of particles on the stack.
	NPold	Value of `NP` prior to an interaction (to test how many particles created see section 3.7.2).
	LATCH	An integer variable for use to track histories.
	LATCHI	Initial value of LATCH(1) when shower called.
THRESH	RMT2	Twice the electron rest mass energy in MeV.
	RMSQ	Electron rest mass energy squared in MeV-squared.
	AP	Array($MXMED) containing PEGS lower photon cutoff energy for each medium in MeV.
	UP	Array($MXMED) containing PEGS upper photon cutoff energy for each medium in MeV.
	AE	Array($MXMED) containing PEGS lower charged particle cutoff energy for each medium in MeV.
	UE	Array($MXMED) containing PEGS upper charged particle cutoff energy for each medium in MeV.
	TE	Same as AE except kinetic energy rather than total energy.
	THMOLL	Array($MXMED) containing the Moller thresh- hold energy (THMOLL=AE+TE) for each medium in MeV.

(cont ...)

EGSnrc COMMONs which are accessible to the User (cont).

Common Block	Variable	Description
UPHIOT	THETA	Collision scattering angle (polar).
	SINTHE	Sine of THETA.
	COSTHE	Cosine of THETA.
	SINPHI	Sine of PHI (the azimuthal scattering angle of the collision).
	COSPHI	Cosine of PHI.
	PI	Pi.
	TWOPI	two Pi.
	PI5D2	2.5 $\times$ Pi.
USEFUL	MEDIUM	Index of current medium. If vacuum, then MEDIUM=0.
	MEDOLD	Index of previous medium.
	RM	Electron rest mass energy in MeV.(see also THRESH)
	PRM	"Precision" electron rest mass energy in MeV (Double Precision).
	PRMT2	Twice PRM (Double Precision).
	PZERO	precise 0.0 (Double Precision).
USER		Null by default but available in ELECTR, PHOTON and HATCH to allow users to pass data into the transport routines (eg geometry data, variance reduction data etc).
CH-Steps	`count_pII_steps`	keeps a count of number of electron steps taken using the PRESTA-II multiple scattering model (real*8).
	`count_all_steps`	counts all electron steps (real*8).
	`is_ch_step`	a logical variable set to true if the current electron step is a condensed history (*i.e.* using PRESTA-I or PRESTA-II multiple scattering theory) step.

As well as the `COMIN`s described above, there are several `COMIN`s which are normally just internal to EGSnrc, which have one or two variables which the user may require access to in order to control various options within EGSnrc. In an ideal world these would all be gathered into a single `COMIN` but that would make compatibility with EGS4 user codes even more difficult to achieve. We have adopted this compromise solution which changes as little as possible of what has grown up historically with EGS4. So, for example the parameters `IEDGFL`, `IPHTER`, `IBRDST` and `IPRDST` have not been moved and the new parameters introduced with EGSnrc have similarly been placed in those `COMIN`s associated with the physics being controlled. Table 3.3 summarizes these `COMIN`s which are needed to control further the transport parameters. Note that the user code can ignore these `COMIN`s completely if you are satisfied with the default settings for photon and electron transport in EGSnrc.

Table 4: EGSnrc `COMMON`s which are optionally accessible to the user. Not all elements in each `COMIN` are described since many of them are not to be accessed by the user.

Common Block	Variable	Description
EDGE	IEDGFL	integer array ($MXREG) specifying whether relaxation of the atom is modelled after photo-effect or bound compton events (at present). When on, fluorescent photons, Auger electrons and Coster-Kronig electrons above threshold are modelled explicitly. When not on, the photoelectron acquires the entire energy of the incident photon (contrary to what happened in EGS4). (1⇒ yes (default), 0⇒ no).
	IPHTER	integer array ($MXREG) specifying whether to sample the angular distribution of photo-electrons in each region (1⇒ yes (default), 0⇒ no) .
COMPTON-DATA	IBCMP	integer array ($MXREG) specifying whether to include binding and Doppler broadening effects in Compton scattering events (1⇒ yes (default), 0⇒ no).
	radc_flag	integer flag for radiative Compton corrections. 0⇒ off (default), 1⇒ on.
BREMPR	IBRDST	flag determining how angle of a brem photon is selected. 0⇒ sample leading term of the angular dist'n; 1⇒ sample the ang. distn. of Koch and Motz. Default=1. For <0, the angle is the same as that of the photon so user can use a call to `AUSGAB` to set the angle.
	IPRDST	flag determining how the electron/positron angles are selected after pair production. 0⇒ use the fixed angle approximation of EGS4. 1⇒ sample the leading term in the angular distribution (fast and good enough). 2⇒ sample the complete angular distribution. Default is 1.
	ibr_nist	integer flag determining which differential photon cross section to sample when brem occurs. 0⇒ use Bethe-Heitler as done in EGS4. This is the default. 1⇒ use NIST data base from ICRU Report 37.
	itriplet	set to 1 to explicitly simulate triplet events (default is 0).
	pair_nrc	flag determining which pair production cross-sections are used. 0⇒ use Bethe-Heitler cross-sections (default). 1⇒ use NRC cross-sections

(cont ...)

EGSnrc internal COMMONs optionally accessible to the User (cont).

Common Block	Variable	Description
EII-DATA	`eii_flag`	flag determining which, if any, electron impact ionization theory is used. 0⇒ turn electron impact ionization off (the default). 1⇒ use Kawrakow's EII theory[57]. 2⇒ use the theory of Casnati[59]. 3⇒ use the theory of Kolbenstvedt[58]. 4⇒ use the theory of Gryzinski[60].
RAYLEIGH _INPUTS	`iray_ff_media`	array (with dimension `$MXMED`) containing the names of the media (must appear in the PEGS4 data) for which custom Rayleigh form factors are to be provided. Otherwise, the medium uses the default atomic form factor.
	`iray_ff_file`	array (with dimension `$MXMED`) containing full names (including directory path) of files specifying custom form factors. `iray_ff_file(i)` corresponds to `iray_ff_media(i)`.
EGS-IO	`file_extensions`	character array (with dimension `$mx_units`; default=20) containing extensions of files specified in the code's `.io` file. Maximum extension length is `$max_extension_length` (default=10 characters).
	`file_units`	integer array (with dimension `$mx_units`) where `file_units(i)` specifies Fortran unit number associated with the file having extension `file_extensions(i)`.
	`user_code`	the name of the user code (max. length = 64 chars).
	`input_file`	the name of the input file. Includes directory path but no extension (max. length=256 chars).
	`output_file`	same as above but for the output file.
	`pegs_file`	the name of the pegs data file. Includes directory path and extension (max. length=256 chars).
	`hen_house`	the user's `$HEN_HOUSE` directory (max. length=128 chars).

(cont ...)

EGSnrc internal COMMONs optionally accessible to the User (cont).

Common Block	Variable	Description
EGS-IO (cont)	`egs_home`	the user's `$EGS_HOME` directory (max. length=128 chars).
	`work_dir`	the temporary working directory created in the user's `$EGS_HOME/user_code` during a run (max. length=128 chars).
	`host_name`	the name of the machine being run on (max. length=64 chars).
	`n_parallel`	if >0, the total number of parallel jobs.
	`i_parallel`	if >0, the number of the current parallel job.
	`first_parallel`	the first parallel job number (default=1).
	`n_max_parallel`	max. number of parallel jobs executing simultaneously (updated during run).
	`n_chunk`	no. of histories per calculation chunk during a parallel run. (currently not used).
	`n_files`	no. of files specified in the `user_code.io` file (max=`$mx_units`).
	`i_input`	unit no. for standard input, or the `.egsinp` file (default=5). Note this is used in BEAMnrc to prevent unit collisions when used as a shared library source.
	`i_log`	unit no. for standard output, or the `.egslog` file (default=6). Used in BEAMnrc to prevent unit collisions when used as a shared library source.
	`i_incoh`	unit numbers for data files containing Compton (default=78),
	`i_nist_data`	bremsstrahlung (default=76), multiple scattering (default=11),
	`i_mscat`	photon cross-section (default=79), and photon relaxation
	`i_photo_cs`	data (default=77), respectively. Variable unit numbers
	`i_photo_relax`	are used to allow BEAMnrc shared library sources to access these files as well.
	`xsec_out`	=0 → do not output file `user_code.xsections` containing photon cross-section data (default). =1 → output file containing photon cross-section data.
	`is_batch`	logical variable set to `.true.` if this is a batch job.

3.4 The Sequence of Operations

The sequence of operations needed for the correct operation of EGS is shown below.

Step 0. `call egs_init` for file initialization (see PIRS-877 for details[14]).

Step 1. User Over Ride Of EGS Macros and Defaults (3.4.1)

Step 2. Pre-`HATCH` Call Initialisation (3.4.2)

Step 3. `HATCH` Call (3.4.3)

Step 4. Initialisation For `HOWFAR` & `HOWNEAR` (3.4.4)

Step 5. Initialisation For `AUSGAB` (3.4.5)

Step 5b. Initialisation For Variance Reduction (3.4.6)

Step 6. Determination Of Incident Particle Parameters (3.4.7)

Step 7. SHOWER Call (3.4.8)

Step 8. Output Of Results (3.4.9)

Step 9. `call egs_finish` as the last executable statement. See ref [14] for details. Properly closes files and places them back on the user-code's directory.

The following are restrictions on the order of these operations:

1. Step 1 must precede use of any EGS macros by the user.
2. Step 0 should be the first executable statement and thus is usually after Step 1 and possibly in Step 2.
3. Step 2 must precede Step 3.
4. Steps 3 through 6 must precede Step 7.
5. Step 7 may be repeated as often as desired, depending on whether information on single showers or many showers is desired (e.g., for shower fluctuation or conversion efficiency calculations).
6. At least one Step 7 must precede the first Step 8.

Details for the above steps are given in the following sub-sections.

It is strongly advised that the user echo **ALL** input parameters into the output listing file to ensure that the listing has a complete record of the run. From extensive experience we have found that this is essential and very valuable.

3.4.1 User Over Ride Of EGS Macros and Defaults (Step 1)

EGS macros which the user might want to over-ride include the following:

3.4.1.i `$CALL-HOWNEAR(#)`

For compatibility with EGS4/PRESTA user-codes, the use of `SUBROUTINE HOWNEAR` has been left as a macro call in EGSnrc. There is no default definition of the macro but the following is suggested:

```
REPLACE {$CALL-HOWNEAR(#);} WITH {CALL HOWNEAR({P1},X(NP),Y(NP),Z(NP),IRL);}
```

The user may choose to define an equivalent macro. The parameter that must be returned by the macro is the shortest distance to any boundary from the current position. See section 3.6(page 137) which specifies the macro or subroutine completely. There is also some discussion in section 5.6 (page 208).

3.4.1.ii `$IMPLICIT-NONE, $REAL, $INTEGER`

The EGSnrc system is now coded with `$IMPLICIT-NONE;` (which defaults to `IMPLICIT NONE;`) in all subroutines. This means that any time the user is passing a variable into the EGSnrc system by means of adding to a `COMIN` definition, one must explicitly specify the type of that variable. To turn this feature off one adds the following statement to the user code:

```
REPLACE {$IMPLICIT NONE;} WITH {;}
```

It is strongly recommended that user codes adopt the use of `$IMPLICIT NONE;` since it catches many coding errors and prevents accidental collision of variables.

In addition to using `$IMPLICIT NONE;`, the EGSnrc system has used the macros `$REAL` and `$INTEGER` everywhere to define real and integer variables as well as using generic intrinsic functions such as MAX and MIN. By default `$REAL` and `$INTEGER` are defined as `REAL*8` and `INTEGER*4`. However, to make the entire code run in single precision, one can add the macro:

```
REPLACE {$REAL} WITH {;REAL*4}
```

However, this requires that all type declarations in the user code also use the macros `$REAL` and `$INTEGER` everywhere. With 64 bit machines, one might as well use the `REAL*8` default.

3.4.1.iii Array Dimensions

`$MXMED` Maximum number of media (default=10).

`$MXREG` Maximum number of regions (default=2000).

`$MXSTACK` Maximum number of particles on the `STACK` at once (default = 40).

For example, to extend the number of media to 25, include the following statement in the User Code.

```
REPLACE {$MXMED} with {25}
```

Note that there are often array dimensions defined by the user for scoring arrays and these should be defined at this step as well.

3.4.1.iv Random Number Initialisation

By default the RANLUX random number generator requires no initialisation. However, to use a luxury level different from the default of 1, or a different initial seed, then you must initialize it using:

```
$INITIALIZE RNG USING luxury_level AND iseed;
```

The `luxury levels` are from 0 to 4, but the value 0 is known to cause problems with EGSnrc calculations. The value of `iseed` is from 1 to 1073741824 (2^{30}).

If you have selected the RANMAR random number generator (via the `.configuration` file, then it MUST be initialized before it is first used. This can be accomplished by including the statement:

```
$RNG-INITIALIZATIION;
```

which initializes RANMAR using whatever the current values of `IXX` and `JXX` are and uses default values if they have not been set (they are passed in `COMIN/RANDOM/;`. Alternatively, one can use:

```
$INITIALIZE RNG USING IXX AND JXX;
```

which accomplishes the same thing. The values are restricted to: $0 < \texttt{IXX} \leq 31328$ and $0 < \texttt{JXX} \leq 30081$ and 0 values are set to defaults.

The random number generator may be initialized at any step prior to the call to `SHOWER` in step 7, or prior to the first use in the user code.

For a complete discussion of these and other issues about the random number generators, see section 3.9 below (page 143).

3.4.1.v `$SET-RHOF`

Section 3.4.2(page 126) explains the use of `RHOF`. On each step, EGSnrc calls a macro `$SET-RHOF` which evaluates the ratio of the density at that point to the density given in the PEGS4 data file for the material in that region. Setting `RHOR` allows you to scale the density throughout a region to some new density. If you have a problem in which the density is varying within the region, this can be handled by replacing the default macro:

```
REPLACE {$SET-RHOF;} WITH {RHOF=RHOR(IRL)/RHO(MEDIUM);}
```

by whatever code you want to return the local density ratio. If, on the other hand you do not use density scaling at all in your code, you should replace the default with:

```
REPLACE {$SET-RHOF;} WITH {RHOF = 1.0;}
```

since this saves a division on every step.

3.4.1.vi Sines and Cosines

To increase calculational speed, sines and cosines were not always determined by function (e.g., `SINTHE= SIN(THETA))` in the default EGS4. Instead, the sine was looked-up in a sine-table and the cosine was determined from the sine. However, this was found to lead to very small errors for angles very close to 0 degrees[73]. This can be overcome but with modern computers the speed of the sine and cosine evaluations is as fast as the table lookup method so we have reverted back to direct function evaluations. For slightly older machines, the table lookup feature saved as much as 40% of the CPU time. If you happen to be using one of these machines, it is worthwhile to use the table lookup method unless small angles are critical to you. To re-implement it, define the following two macros in STEP 1.

```
REPLACE {$EVALUATE#USING SIN(#);} WITH {{P1}=SIN1(L{P2})*{P2}+SIN0(L{P2});}}
REPLACE {$SET INTERVAL#,SINC;} WITH {L{P1} = SINC1*{P1} +SINC0;}}}
```

The reader is referred to the Mortran3 User's Guide as an aid in understanding the macros (see section 8).

It should be pointed out that due to the precision involved in the table look-up method, the direction cosines can become slightly unnormalized. Depending on the problem at hand, this can lead to incorrect results—such as when two direction cosines are simultaneously involved in an angular sort of particles. The problem can generally be remedied by renormalizing the direction cosines prior to using them.

3.4.1.vii Charged Particle Transport

The pattern `$CHARGED-TRANSPORT` has been included in subroutine ELECTR in order to allow transport of the charged particles by other means than used in this version. For example,

```
REPLACE {$CHARGED-TRANSPORT;} WITH {CALL MYTRAN;}
```

could be included in Step 1 of the User Code, and an appropriate subroutine MYTRAN would need to be provided by the user. For a detailed discussion of one such implementation, see ref[97, 99]

3.4.2 Pre-HATCH Call Initialisation (Step 2)

This step consists of setting EGS `COMMON` variables that are used by `HATCH` in its initialisation operations. All of these variables are initialized to some reasonable value in the BLOCK DATA subprogram. Therefore, if different values are desired they should be set with executable code (as opposed to another BLOCK DATA). Concurrently, the various `COMMON` blocks (i.e., BOUNDS, MEDIA, MISC) will have to be included in the declaration section of the MAIN program of the User Code.

If the user code reads an input (`.egsinp`) file for defining the simulation geometry, sources, etc, then `COMMON` variables controlling Monte Carlo transport are accessible to the user through the subroutine `get_transport_parameter`. This can be called by a user code prior to calling `HATCH` provided that the user code includes the files `$HEN_HOUSE/src/get_inputs.mortran` (contains the actual `get_transport_parameter` subroutine) and `$HEN_HOUSE/src/transportp.macros` (contains necessary macros) in the `SOURCES` list defined in either `Makefile` or `user_code.make`. Note that `$HEN_HOUSE/src/transportp.macros` must occur before `$HEN_HOUSE/src/get_inputs.mortran` and `$HEN_HOUSE/src/egsnrc.mortran` in the list of `SOURCES`.

The call to `get_transport_parameter` is:

```
call subroutine get_transport_parameter(6)
```

where the parameter "6" instructs the subroutine to read the transport parameters from the `.egsinp` file and to echo the parameters to the screen (or `.egslog` file if running in batch).

Within the `.egsinp` file, the transport parameter settings must appear between the delimiters:

```
:Start MC Transport Parameter:

:Stop MC Transport Parameter:
```

The general format of inputs for `get_transport_parameter` is a text line followed by "=" and then the value(s) that the user wants to assign to the particular parameter. Note that these inputs are case insensitive.

For some more information about `get_transport_parameters` along with a printout of the input description appearing at the top of the code, see the PIRS-702 User Code Manual[100].

The `COMMON` block variables which must be initialized before calling `HATCH`, along with the method to set them if using the `get_transport_parameter` subroutine (where applicable) are:

NMED This must be initialized to the number of media to be used in the shower generation (default=1).

MEDIA This array contains the names of the media required and is dimensioned `MEDIA(24,$MXMED)`, where `$MXMED` is an EGS macro that is currently defined to be 10 (default), and whose value is the maximum number of media for which array space has been allocated (see section 3.4.1 above). The media names are stored in MEDIA in alphameric field specification A1 to ensure portability. Each medium name is 24 characters long. For the convenience of users compiling with EGS' macros, there is a macro to generate A1 strings. For example,

`$S'STRING'` expands to `'S','T','R','I','N','G'`.

One way of implementing this in the User Code is demonstrated in the next example, which is for three media: lead, steel, and air at NTP. A temporary array is declared and initialized in MAIN by:

```
CHARACTER*4 TEMP(24,3)/$S'PB', 22*' ', $S'STEEL', 19*' ', $S'AIR AT
                              NTP',14*' '/;
```

and at Step 2 one puts

```
NMED=3;     "number of media used"
DO J=1,NMED [DO I=1,24 [MEDIA(I,J)=TEMP(I,J);]]
```

MED This array, which is dimensioned `MED($MXREG)`, contains the medium indices for each region (default values are 1 for all `$MXREG`). A medium index of zero means a region is filled with a vacuum. For instance, if we consider the three media example above along with vacuum to define four regions, in Step 2 of the User Code we might have:

```
MED(1)=3;  "first region is air at NTP"
MED(2)=1;  "second region is lead"
MED(3)=0;  "third region is vacuum"
MED(4)=2;  "fourth region is steel"
```

ECUT and PCUT These arrays contain the cutoff energies (in MeV) for charged particles and photons, respectively, for each region. They are dimensioned `ECUT($MXREG)` and `PCUT($MXREG)` and are given temporary (default) values of 0.0 in `BLOCK DATA`. At the time that data for each medium are generated in the preprocessing code (PEGS), two parameters (`AE` and `AP`) are set to the lowest energies at which it will be desired to transport electrons and photons. When the EGS subroutine `HATCH` is called, these `AE` and `AP` values are read in and `HATCH` upgrades the values of `ECUT` and `PCUT` such that the maximum of the current `(ECUT,AE)` (and `(PCUT,AP)`) is chosen. Therefore, by assigning values of `ECUT` and `PCUT` prior to the `HATCH` call, the user can raise (but not lower) the cutoff energies in this manner. For instance, consider the four region example from above. The statement

```
DO I=1,3 [ECUT(I)=10.0; PCUT(I)=100.0;]
```

when put in Step 2 of the User Code results in charged particle histories being terminated at 10.0 MeV (total energy) and photon histories being terminated at 100.0 MeV in the first three regions only. In the fourth region the respective cutoffs are set by `AE` and `AP` as established by PEGS. Of course COMMON/BOUNDS/ will have to be declared in the routine that calls `HATCH` in order to pass ECUT and PCUT to `HATCH`. Combined with `COMMON/MEDIA/` and `COMMON/MISC/`, the macro declaration might look like

```
COMIN/BOUNDS,MEDIA,MISC/;
```

If using the `get_transport_parameter` subroutine, then global values of `ECUT` and `PCUT` can be set using:

```
Global ECUT= ECUT
Global PCUT= PCUT
```

DUNIT This parameter determines the unit of distance to be used in the shower simulation (the default is cm if `DUNIT=1.0`). On input to `HATCH`, this parameter will be interpreted as follows:

DUNIT > 0 means that `DUNIT` is the length of the distance unit expressed in centimeters. For example, setting `DUNIT`=2.54 would mean that the distance unit would be one inch.

DUNIT < 0 means that the absolute value of `DUNIT` will be interpreted as a medium index. The distance unit used will then be the radiation length for this medium, and on exit from `HATCH`, `DUNIT` will be equal to the radiation length of that medium in centimeters. The obvious use of this feature is for the case of only one medium with `DUNIT`=-1. Then the shower is expressed entirely in radiation lengths of the first medium.

The distance unit used by PEGS is the radiation length. After `HATCH` interprets `DUNIT`, it scales all distance- type data from PEGS in the proper way, so that all subsequent operations in EGS will be correctly performed with all distances in units of `DUNIT` (default value: 1.0 cm).

IRAYLR The elements of this integer array (dimensioned `IRAYLR($MXREG)` and passed in `COMMON/MISC/`) are to be set to 1 prior to calling `HATCH` if coherent (Rayleigh) scattering is to be done in a particular region. The default values are 1. See section 2.2.4(page 52). Execution is only terminated if set to 1, user wants to use photon data from PEGS4 file, and Rayleigh data are not included. See section 2.2.4.i(page 52).

If using `get_transport_parameter` routine, then `IRAYLR` can be set to 0 in all regions using the input `Rayleigh scattering= Off` or 1 in all regions using `Rayleigh scattering= On`. If the user intends to supply custom form factor files, then `Rayleigh scattering= custom` must be used and `IRAYLR` will be set to 1.

iray_ff_media A character array (`character*24`) with dimension `$MXMED` (the maximum number of media) defined in `COMMON/RAYLEIGH_SAMPLING`. Prior to calling `HATCH`, this array must be filled with the names of the media (must match names in PEGS4 data) for which the user wishes to supply custom molecular form factor data for Rayleigh scattering. If empty, then default form factors are used. See section 2.2.4.ii (page 53).

With the `get_transport_parameter` routine, `iray_ff_media` can be set using `ff media names=` followed by the list of media.

iray_ff_file A character array (`character*128`) with dimension `$MXMED` defined in `COMMON/RAYLEIGH_SAMPLING`. This holds the full filenames (including directory paths) for the files containing custom Rayleigh form factor data for the media defined in `iray_ff_media` (see above). Each medium specified must have an associated file. Thus, `iray_ff_file(i)` contains the form factor data for `iray_ff_media(i)`. Example files containing form factor data are in `$HEN_HOUSE/data/molecular_form_factors`. See section 2.2.4.ii (page 53).

With the `get_transport_parameter` routine, `iray_ff_file` can be set using `ff file names=` followed by the list of full file names.

RHOR For each medium to be input, there is a default density, `RHO(MED)`. The user may assign an arbitrary density in each geometry region by initializing the array `RHOR($MXREG)`. EGSnrc then appropriately scales all cross sections in each region. This is done by calculating the value of `RHOF = RHOR(IRL)/RHO(MED)` on every step in the calculation, using the macro `$SET-RHOF`(section 3.4.1.v,page 122). The array `RHOR` is initially zero. If the user does not initialize `RHOR`, then in `HATCH` it is set to `RHO(MED)` using the material assigned to each region. In this case `RHOF` is always unity. Note that this scaling is not perfect because the density effect in the electron stopping powers is not scaled, so if you are doing very precisie work, you may need to define a variety of media with different densities. Remember also to use the proper density when calculating the mass of each region for dose calculations.

IBCMP The elements of this integer array (dimensioned `IBCMP($MXREG)` and passed in `COMMON/COMPTON-DATA)` are to be set to 0 prior to calling `HATCH` if Klein-Nishina is to be modelled in a particular region, as opposed to using the default bound Compton formalism. Binding effects can be important for some simulations with photons below 1 MeV but above that is rarely important and only takes extra time. There is an option to model bound Compton scattering without Doppler broadening (`IBCMP`=2). This option must be used used if the user is supplying their own Compton cross section data (see below). There is also an option similar to `IBCMP=1`, but the actual total bound Compton cross section is used and there are no rejections at run time (`IBCMP`=3). The default value is 3 (i.e. uses bound Compton scattering, with no rejection). See section 2.2.2 (page 35) for more details.

If using `get_transport_parameter`, then `IBCMP` can be set to 0 using the line `Bound Compton scattering= Off`, 1 using `Bound Compton scattering= On`, 2 using `Bound Compton scattering= Simple` and 3 using `Bound Compton scattering= norej`.

comp_xsections A character (`character*16`) variable defined in `COMMON/MEDIA/` which must be given the name of the alternative Compton cross section data prior to calling `HATCH` if the user does not want to use the default Compton cross sections. This option can only be used if `IBCMP`=4 (bound Compton without doppler broadening–see above). The cross section data must be contained in file `$HEN_HOUSE/data/comp_xsections_compton.data`. If the user does not supply a name for `comp_xsections`, then the default Compton data used with `IBCMP`=3 is in `$HEN_HOUSE/data/compton_sigma.data`. See section 2.2.2.iv (page 46).

If using `get_transport_parameter`, then `comp_xsections` can be set using `Compton cross sections=` followed by the user definition of `comp_xsections`.

radc_flag Integer flag in `COMMON/COMPTON-DATA` which is set to 1 prior to calling `HATCH` if radiative Compton corrections are to be modeled. The default value is 0 (*i.e.* no radiative corrections). If this is set to 1, then the code must be compiled including `$(EGS_SOURCEDIR)rad_compton1.mortran` just before `$(EGS_SOURCEDIR)get_inputs.mortran` in the `SOURCES` variable defined in the code `MAKEFILE`. See section 2.2.2.iii (page 45).

If using the `get_transport_parameter` subroutine, `radc_flag` can be set to 0 using `Radiative Compton corrections= Off` and 1 using `Radiative Compton corrections= On`.

IEDGFL The elements of this integer array (dimensioned `IEDGFL($MXREG)` and passed in `COMMON/EDGE)` are to be set to 0 prior to calling `HATCH` if one does not want atomic relaxation to be modelled in a given region. The relaxation considers the creation of K, L, M and N shell fluorescent photons, Auger electrons and Coster-Kronig electrons. Relaxation is currently modelled after photo-electric and bound compton events although it may get extended to other processes in time. The default is to have relaxations simulated (*i.e.* `IEDGFL`=1). When relaxation is not simulated and there is a photo-electric event, the full photon energy is transfered to the photo-electron. This differs from EGS4 where the binding energy was subtracted and deposited on the spot. Either option for the energy of the photon-electron is an approximation, and it is our experience that transferring all the energy to the photo-electron is more accurate than dumping the full binding energy on the spot. If this energy transport is important at all, relaxation should be turned on and then it is modelled correctly. See section 2.3 (page 55).

With the `get_transport_parameter` subroutine, `IEDGFL` can be set to 0 in all regions using `Atomic relaxations= Off` and 1 in all regions using `Atomic relaxations= On`.

photon_xsections A character variable (`character*16`) defined in `COMMON/MEDIA` which must be defined prior to calling `HATCH` if the user wants to use photon cross section data other than the default XCOM data. Current possible settings of `photon_xsections` are: `epdl` (the Evaluated Photon Data Library[101]), `xcom` (XCOM from Berger & Hubbell[37], default), `si` (Storm & Israel), and `pegs4` (to use PEGS4 photon data). The user can supply their own cross section data. For a user-supplied data set, `photon_xsections`, the cross section data for photoelectric, pair, triplet and Rayleigh interactions must exist in the files: `photon_xsections_photo.data`, `photon_xsections_pair.data`, `photon_xsections_triplet.data` and `photon_xsections_rayleigh.data` in directory `$HEN_HOUSE/data`. See section 2.2.5 (page 54).

The `photon_xsections` variable can be set if using `get_transport_parameter` using the line `Photon cross sections=` followed by the user-defined value (*i.e.* `xcom`, `epdl`, `si`, `pegs4`, or a custom value).

xsec_out An integer flag defined in `COMMON/EGS_IO` which must be set to 1 prior to calling `HATCH` if the user wants to output a summary of the photon cross section data. The default value is 0. Cross section data is written to the file `$EGS_HOME/user_code/inputfile.xsections`. Previously, the default was to print this file and there was no input for turning it off.

If using `get_transport_parameter`, then `xsec_out` can be set to 1 using `Photon cross-sections output= On` and 0 using `Photon cross-sections output= Off`.

SPIN_EFFECTS A logical variable passed in `COMMON/ET-Control` which must be set false prior to calling `HATCH` if one wants to exclude relativistic spin effects in multiple and single scattering of electrons. The default is `.true.` for highest accuracy. For results closer to EGS4 `SPIN_EFFECTS` should be set to `.false.`, then only Rutherford scattering will be used as the basis of multiple and single scattering. See section 4.8.18 (page 95).

If using the `get_transport_parameter` subroutine, `SPIN_EFFECTS` can be set to `.true.` using the input line `Spin effects= On` and `.false.` using `Spin effects= Off`.

ibr_nist is an integer flag passed in `COMIN/BREMPR/` which must be set prior to calling `HATCH`. The value determines how the brem photon's energy is to be sampled.
0⇒ the Bethe-Heitler cross sections used in EGS4 are sampled. This is the default;
1⇒ sample from the NIST bremsstrahlung cross section data base[48, 49]. 2⇒ same as `ibr_nist`=1 but including electron-electron bremsstrahlung contributions.
See section 4.2.5 (page 68).

If using the `get_transport_parameter` subroutine, then `ibr_nist` can be set to 0 using the input `Brems cross sections= BH`, 1 using `Brems cross sections= NIST` or 2 using `Brems cross sections= NRC`.

eii_flag is an integer flag defined in `COMMON/EII_DATA` which must be set to 1 prior to calling `HATCH` if electron impact ionization is to be included in the simulation. It is set to 0 by default.

`eii_flag` is set based on the input variable `eii_xfile` described below.

eii_xfile A character variable (`character*16`) defined in `COMMON/MEDIA` which must be defined prior to calling `HATCH` if electron impact ionization is to be included in the simulation. Current possible settings of `eii_xfile` are:

`Off`	⇒ do not include electron impact ionization (default);
`On`	⇒ derive cross sections using the theory of Kawrakow[57];
`ik`	⇒ derive cross sections using the theory of Kawrakow([57]);
`casnati`	⇒ derive cross sections using the empirical formula of Casnati[59, 102];
`kolbenstvedt`	⇒ derive cross sections using the theory of Kolbenstvedt[60];
`gryzinski`	⇒ derive cross sections using the theory of Gryziński[58, 103, 104];
`penelope`	⇒ derive cross sections using the theory of Bote and Salvat[61].

Note: For backwards compatibility with older input files, if `eii_xfile` is set to On, `eii_ik.data` will be used.

The user can supply their own EII cross section data. For a user-supplied data set, `eii_xfile`, the EII cross section data must exist in the file: `eii_'eii_xfile'.data` in directory `$HEN_HOUSE/data`.

The `eii_xfile` variable can be set if using `get_transport_parameter` with the line `Electron Impact Ionization =` followed by the user-defined value (*i.e.* `Off`, `On`, `ik`, `casnati`, `kolbenstvedt`, `gryzinski`, `penelope` or a custom value).

IBRDST is an integer flag passed in `COMIN/BREMPR` that specifies what type of angular sampling is done when a bremsstrahlung photon is created.
0⇒ sample from the leading term in the Koch and Motz angular distribution;
1⇒ sample from the Koch and Motz angular distribution[18]. This is the default.
Note that EGS4 used a fixed angle approximation and this clearly causes problems for radiotherapy accelerator calculations[105]. See section 4.2.8 (page 71).

If using `get_transport_parameter`, then `IBRDST` can be set using `Brems angular sampling= Simple` (`IBRDST`=0) or `KM` (`IBRDST`=1).

IPRDST is an integer flag passed in `COMIN/BREMPR` that specifies what type of angular sampling is done when a pair production event occurs.
0⇒ use a fixed angle wrt the photon's direction of m/E_γ which was the default in EGS4;
1⇒ use the leading term in the angular distribution[27]. This is the default.
2⇒ use the angular distribution of Motz, Olsen and Koch[19] as implemented by Bielajew[27]. See section 2.1.18 (page 34).

If using `get_transport_parameter`, then the input `Pair angular sampling= Off` corresponds to setting `IPRDST` to 0, `Pair angular sampling= Simple` corresponds to `IPRDST`=1, and `Pair angular sampling= KM` corresponds to `IPRDST`=2.

IPHTER The elements of this array (dimensioned `IPHTER($MXREG)` and passed in `COMMON/EDGE)` are to be set to 0 if one does not want the photo-electron's angular distribution to be modelled[41]. The default is to model this angular distribution. This rarely, if ever, has an effect on a simulation because the low-energy electrons experience so much multiple scattering anyway. This array need not be reset until before the call to `SHOWER`. See section 2.2.3.iii (page 51).

If using `get_transport_parameter`, `IPHTER` can be set to 0 using `Photoelectron angular sampling= Off` and 1 using `Photoelectron angular sampling= On`.

BCA_ALGORITHM is an integer flag passed in `COMMON/ET-Control` which must be set prior to the call to `HATCH` to specify which boundary crossing algorithm (BCA) to use.
0⇒ use the exact (single scattering) algorithm within a distance of `SKINDEPTH_FOR_BCA` of any boundary. This is the default.
1⇒ approach a boundary using multiple scattering but within a distance of `SKINDEPTH_FOR_BCA` of any boundary, turn off the lateral deflections. See section 2.4.10 (page 100).

If using the `get_transport_parameter` subroutine, `BCA_ALGORITHM` is set to 0 using the input line `Boundary crossing algorithm= Exact` and 1 using `Boundary crossing algorithm= PRESTA-I`.

SKINDEPTH_FOR_BCA is a real variable passed in `COMMON/ET-Control` which must only be set prior to the call to `HATCH` to specify the distance from a boundary (in elastic mean free paths) at which to switch to the boundary crossing algorithm. The value is initialized to 3.0. If set to 0.0 by the user when the exact BCA is being used, it is reset to 3.0 since the value 0.0 means charged particles never get over a boundary. If the PRESTA-I BCA is being used and `SKINDEPTH_FOR_BCA` is set to zero (actually anything less than 10^{-4}), the code uses a value corresponding to the value of `BLCMIN` used by PRESTA-I. If `SKINDEPTH_FOR_BCA` is set to a large value and `TRANSPORT_ALGORITHM` is 0 so that the PRESTA-II algorithm is being used, then a complete single scattering calculation is done. In contrast, if `SKINDEPTH_FOR_BCA` is set to a large value and `TRANSPORT_ALGORITHM` is 1, this turns off lateral deflections everywhere and the algorithm becomes very close to EGS4 (except that now the Lewis pathlength correction is used). See section 2.4.10 (page 100).

With the `get_transport_parameter` subroutine, `SKINDEPTH_FOR_BCA` can be set using the input line `Skin depth for BCA=` followed by the value of `SKINDEPTH_FOR_BCA` requested.

TRANSPORT_ALGORITHM is an integer flag passed in `COMMON/ET-Control` which must be set prior to the call to `HATCH` to to specify which electron transport algorithm to use.
0⇒ use the PRESTA-II algorithm. This is the default.
1⇒ PRESTA-I.
See section 2.4.9 (page 97).

With the `get_transport_parameter` subroutine, `TRANSPORT_ALGORITHM` can be set to 0 using `Electron-step algorithm= PRESTA-II` or 1 using `Electron-step algorithm= PRESTA-I`.

ESTEPE is a real variable passed in `COMMON/ET-Control` which must be set prior to the call to `HATCH` to specify the global maximum fractional energy loss in an electron step due to continuous energy loss. The default value, which is also the maximum allowed value, is 0.25 and should not be changed unless the PRESTA-I `TRANSPORT_ALGORITHM` is being used.

If using `get_transport_parameter`, then `ESTEPE` can be set using the input line `ESTEPE=` followed by the desired value.

ESTEPR The elements of this `$REAL` array (dimensioned `ESTEPR($MXREG)` and passed in `COMMON/ET-Control` are the same as `ESTEPE` except apply just to the local region. The global value of `ESTEPE` overrides the values of `ESTEPR`. The default is 1.0 which means it has no effect.

SMAXIR The elements of this `$REAL` array (dimensioned `SMAXIR($MXREG)` and passed in `COMMON/ET-Control` determine the maximum electron step in this region in whatever units are defined via `DUNIT` (default is cm). The maximum value is very large. This parameter is only needed using the PRESTA-I transport algorithm.

If using the `get_transport_parameter` subroutine, then a global value of `SMAXIR` (*i.e.* applied to all regions) can be set using the input line `Global SMAX=` followed by the global value of `SMAXIR`.

XIMAX This REAL variable is passed in `COMMON/ET-Control` is the maximum first GS moment per step (roughly half the average multiple scattering angle squared, default 0.5). See section 2.4.9 (page 97). This should not be changed.

If using `get_transport_parameter`, then `XIMAX` can be set using `XIMAX=` followed by the input value to assign `XIMAX`.

3.4.2.i Emulating EGS4's implementation of the Condensed History technique

Appropriate selection of the above parameters allows the default behaviour of EGS4's implementation of the Condensed History technique to be emulated. To achieve this set `TRANSPORT_ALGORITHM` to 1, `SKINDEPTH_FOR_BCA` to 1E10, `BCA_ALGORITHM` to 1, `IPRDST` to 0 `IBRDST` to 0 and `SPIN_EFFECTS` to `.false.`. There are other options and improvements which are different from EGS4. These should also be set to match EGS4 (e.g. modelling bound Compton scattering should be off, *i.e.*, the `IBCMP` array should be set 0, atomic relaxation should be off, *i.e.*, the `IEDGFL` array should be set 0, and the differential bremsstrahlung cross sections used should be Bethe-Heitler, *i.e.*set `ibr_nist` = 0). Leave other values at their default and use `ESTEPE` if desired. This emulates EGS4 except for the more accurate pathlength corrections used in EGSnrc, the problem with fictitious cross sections is handled properly (only important for low values of AE), energy loss is calculated more accurately and an exact multiple scattering theory is used.

3.4.2.ii Emulating EGS4/PRESTA's implementation of the Condensed History technique

To emulate EGS4/PRESTA's implementation of the Condensed History technique, set

TRANSPORT_ALGORITHM to 1, SKINDEPTH_FOR_BCA to 0.0, BCA_ALGORITHM to 1, IPRDST to 0 and IBRDST to 0. Leave other values as above and use ESTEPE if desired. This will emulate EGS4/PRESTA reasonably well with the same restrictions as noted above regarding emulation of EGS4.

3.4.3 HATCH Call (Step 3)

This step is very simple—HATCH has no arguments, so all one has to do is:

```
CALL HATCH;
```

The following is a typical output message when DUNIT has not been changed (and Rayleigh data is included in the file):

```
RAYLEIGH DATA AVAILABLE FOR MEDIUM  1 BUT OPTION NOT REQUESTED.
EGSnrc SUCCESSFULLY 'HATCHED' FOR ONE MEDIUM.
```

However, if the user has set DUNIT=2.54 prior to calling HATCH, the message will look like the following (two media, no Rayleigh data):

```
DUNIT REQUESTED&USED ARE:   2.54000E+00   2.54000E+00(CM.)
EGS SUCCESSFULLY 'HATCHED' FOR     2 MEDIA.
```

Failure to "hatch", on the other hand, will result in the message:

```
END OF FILE ON UNIT 12
PROGRAM STOPPED IN HATCH BECAUSE THE
FOLLOWING NAMES WERE NOT RECOGNIZED:
                                           (list of names)
```

followed by a STOP in HATCH. [Note: one cannot ask for the same medium twice].

3.4.4 Initialisation For HOWFAR and HOWNEAR (Step 4)

As stated previously, HOWFAR and HOWNEAR are the routines that specify the geometry of the regions. Although initialisation for items that are used in HOWFAR and HOWNEAR can be done at any step prior to calling SHOWER (Step 7), Step 4 allows a space in MAIN to consider if such initialisation need be performed. For example, if regions are defined by semi-infinite planes, data defining each plane (e.g., coordinates and unit normal vectors) can be established here. The data may be referred to in HOWFAR and HOWNEAR or by user-written subprograms called by HOWFAR or HOWNEAR. It may be that some of the dimensions of the regions are determined at run-time, or the geometry may be so complex that it is desirable to use executable code to generate tables for use by HOWFAR or HOWNEAR. In such cases, initialisation will probably consist of filling up some user-written COMMON blocks.

3.4.5 Initialisation For AUSGAB (Step 5)

This step is similar to Step 4 above in that it provides a specified location in the MAIN code where quantities used in `AUSGAB` can be initialized. For example, suppose that we wished to create an array, `ESUM`, to keep track of the total energy deposited in each of the regions. We could declare

```
COMMON/TOTALS/ESUM($MXREG);
```

in both the MAIN code and in `AUSGAB`, and we could add

```
DO I=1,$MXREG [ESUM(I)=0.0;]
```

to the MAIN code (at Step 5). Then the statement

```
ESUM(IR(NP))=ESUM(IR(NP)) + EDEP;
```

in `AUSGAB` could keep a running total of the energy deposited in each region under consideration.

Note that `EDEP` is a double precision variable, even when the rest of the code is run in single precision (see section 3.4.1.ii). This is established as such via a macro. Therefore, one might wish to establish `ESUM` as double precision as well (in both MAIN and `AUSGAB`). We have experienced situations whereby energy balancing could not be attained due to round-off error difficulties. This was particularly evident for large shower-history problems involving the addition of small energy values to large numbers in various regions. As a result of this experience, certain key energy variables in the EGS code have been defined as double precision. Users may take advantage of this at their discretion.

3.4.6 Initialisation For Variance Reduction (Step 5b)

Variance reduction is often associated with various calls to `AUSGAB` so this step is really part of step 5, but has its own section to emphasize the new variance reduction options within EGSnrc. These options are all controlled by parameters in `COMIN/EGS-VARIANCE-REDUCTION/` (see table 3) and are described in detail in section 3.11 (page 152). The user may wish to introduce their own variance reduction techniques as well.

3.4.7 Determination Of Incident Particle Parameters (Step 6)

This step is really self-explanatory—particularly when looked at in conjunction with Step 7 below. A specific example of such coding might be useful and is given as follows:

```
IQI=-1;                     "incident particle is an electron"
EI=1000.0;                  "total energy (MeV)"
XI=0.0; YI=0.0; ZI=0.0;     "particle coordinates"
UI=0.0; VI=0.0; WI=1.0;     "direction cosines"
IRI=2;                      "region number 2 is the incident region"
WTI=1.0;                    "weight factor in importance sampling"
IXX=12;JXX=3001;            "random number generator seeds"
```

```
NCASES=10;                    "number of histories to run"
LATCHI=0;                     "variable for tracking history or marking"
                              "a history""
```

3.4.8 SHOWER Call (Step 7)

The calling sequence for SHOWER is:

```
CALL SHOWER(IQI,EI,XI,YI,ZI,UI,VI,WI,IRI,WTI);
```

The types of the arguments are given by their starting letter in accordance with standard Fortran convention. These arguments specify the charge, total energy, position, direction, region index, and statistical weight of the incident particle, and are used to fill the corresponding stack variables (see COMMON/STACK/ in Section 3.3). The one exception is the parameter LATCHI which is passed in COMMON STACK directly and for historical reasons is not symmetric with the other parameters. Section 3.4.7 above might be of some aid in understanding the parameter list. The subroutine may be called repeatedly by means of statements like

```
DO I=1,NCASES [CALL SHOWER(IQI,EI,XI,....,etc.);]    .
```

The statistical weight, WTI, is generally taken as unity unless variance reduction techniques are employed by the user. It should noted, however, that if IQI is assigned the value of 2, subroutine SHOWER recognizes this as a pi-zero meson decay event, and two photons are added to the stack with energies and direction cosines appropriately obtained by sampling.

3.4.9 Output-Of-Results (Step 8)

This step has been added for completeness and is self-explanatory.

3.5 Specifications for HOWFAR

On entry to the geometry subprogram, HOWFAR, EGS has determined that it would like to transport the top particle on the stack by a straight line distance USTEP. All of the parameters of the particle are available to the user via COMMON/STACK/ as described earlier. The user controls the transport by setting the following variables:

```
USTEP, IDISC, IRNEW, and DNEAR(NP).
```

Except for the last variable (which is in COMMON/STACK/) , these are available to the user via COMMON/EPCONT/. The ways in which these may be changed, and the way EGS will interpret these changes, will now be discussed in detail.

1. If the user decides that the current particle should be discarded, then IDISC must be set non-zero (the usual convention is to set IDISC=1).
    - A positive value for IDISC will cause the particle to be discarded immediately. The special value IDISC = 99 is used to mark a positron for immediate discard

but **WITH** positron annihilation occurring as if at rest. For `IDISC` = 1 a positron is discarded and no annihilation photons are created.

- A negative value for `IDISC` will cause EGS to discard the particle when it completes the transport. Again, `IDISC` = -99 marks that positron annihilation should take place.

EGS initializes `IDISC` to zero, and if left zero no user requested discard will take place. For example, the easiest way to define an infinite, homogeneous medium is with the `HOWFAR` routine:

```
SUBROUTINE HOWFAR;
RETURN;
END;
```

In this case, particle transport will continue to take place until energy cutoffs are reached. However, a common procedure is to set `IDISC`=1 whenever the particle reaches a discard region.

2. If immediate discard has not been requested, then the user should check to see whether transport by distance `USTEP` will cause a region boundary to be crossed. The presence of the region index for the current particle, `IR(NP)`, should make this task much easier than if only the position of the particle were known. If no boundary will be crossed, then `USTEP` and `IRNEW` may be left as they are. If a boundary will be crossed, then `USTEP` should be set to the distance to the boundary from the current position along the current direction, and `IRNEW` should be set to the region index of the region on the other side of the boundary. For sophisticated geometries, this is the most complex part of the User Code.

3. The setting of `DNEAR(NP)` by the user is optional and is not required for EGSnrc since it is always calculated on each electron step using the routine `HOWNEAR`.

   Boundary checking in `HOWFAR` takes time and should be avoided whenever possible. If EGSnrc had no way of knowing how far it was from a boundary, then it would have to ask the user how far to go every time it wanted to transport a particle. This would not be too serious for photons since they usually go relatively far on each transport. However, the transport of a charged particle from one interaction to the next requires the path-length to be split up into smaller lengths in order to simulate properly the multiple scattering process. If, relative to the small steps being taken, the particle is a fairly good distance from the nearest boundary, then checking for boundary crossings on each transport is a waste of time. In order to avoid this inefficiency, each particle has stored on the stack a variable called DNEAR, which is used by EGS to store a lower bound to the distance from the particle's current position to the nearest region boundary. This variable is used by EGS in the following ways:

   - `DNEAR` for the incident particle is initialized to zero by `EGSnrc`.
   - Whenever a particle is actually moved (by a straight line distance VSTEP) the path length transported is deducted from the `DNEAR` for the particle.

- Whenever a particle interacts, the DNEAR values for the product particles are set from the DNEAR value of the parent particle.
- When EGSnrc has decided it would like to transport the current particle by a distance USTEP (which will be the distance to the next interaction), subroutine HOWFAR will be called to get the user's permission to go that far only if USTEP is larger than DNEAR.

For electron steps the user automatically takes advantage of these efficiency features. It is unlikely that this feature would help during photon transport, but if it would, the user can use HOWNEAR from within HOWFAR for photons only and set DNEAR(NP). If the medium for a region is vacuum, the user need not bother computing DNEAR, as EGS will always transport to the next boundary in only one step in this case.

3.5.1 An example of HOWFAR

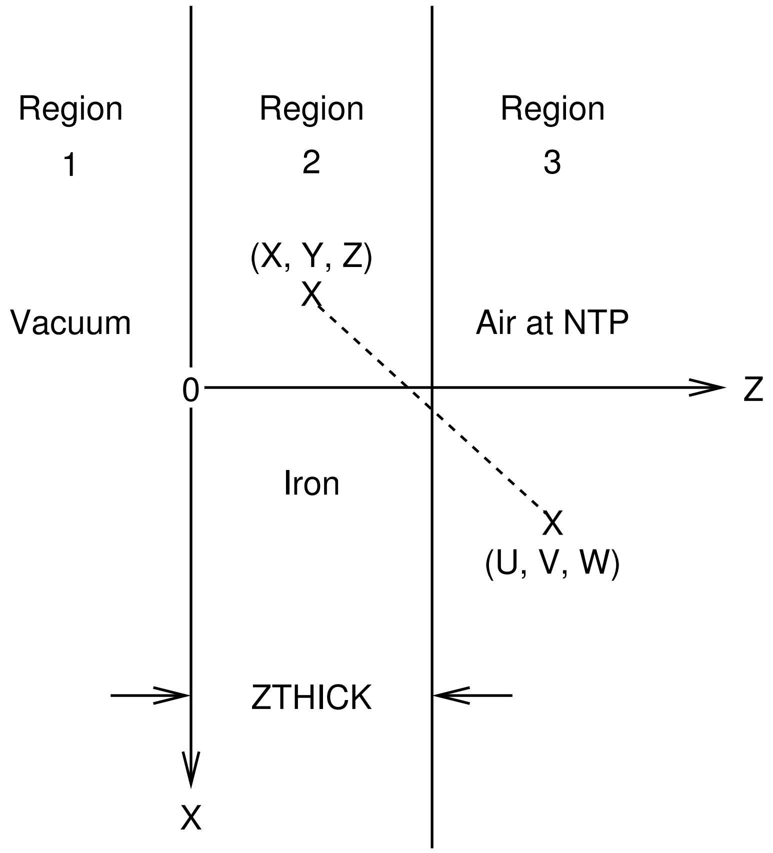


Figure 19: A 3 region geometry example for HOWFAR. The Y axis is into the paper.

Consider, as an example of how to write a HOWFAR subroutine, the three region geometry in fig 19. A particle is shown in Region 2 with coordinates (X,Y,Z) and direction cosines (U,V,W). We will assume that the slab of thickness ZTHICK is semi-infinite (x and y-directions), and that particles are immediately discarded whenever they go into Region 1 or Region 3. The following HOWFAR code is then applicable:

```
SUBROUTINE HOWFAR;
COMIN/EPCONT,STACK/;      "common blocks needed in calculations"
COMMON/GEOM/ZTHICK;       "slab thickness defined in main"
IF(IR(NP) ~= 2) [IDISC=1; RETURN;]
IF(W(NP) =  0.0) [RETURN; "particle going parallel to planes"]
"check forward plane first since shower heading that way"
"                                       most of the time"
IF(W(NP) > 0.0) [DELTAZ=(ZTHICK-Z(NP))/W(NP); IRNEXT=3;]
"otherwise, particle must be heading in backwards direction"
ELSE [DELTAZ=-Z(NP)/W(NP); IRNEXT=1;]
"now check with USTEP and reset things if necessary"
IF(DELTAZ <= USTEP) [USTEP=DELTAZ; IRNEW=IRNEXT;]
RETURN;  END;
```

A number of geometry subprograms and their macro equivalents are distributed with the EGS Code System in order to make it easier to write `HOWFAR`. For example, `SUBROUTINE PLAN2P`, or its equivalent macro `$PLAN2P`, could have been used in place of several lines above and the program would have been easier to read.

For an advanced discussion of coding `HOWFAR` routines, see Alex Bielajew's PIRS-341 report "HOWFAR and HOWNEAR: Geometry Modeling for Monte Carlo Particle Transport"[106].

3.6 Specifications for HOWNEAR

As mentioned in section 3.4.1.i(page 121), for compatibility with previous EGS4/PRESTA user codes, EGSnrc formally requires a macro definition which defaults to:

`REPLACE {$CALL-HOWNEAR(#);} WITH {CALL HOWNEAR({P1},X(NP),Y(NP),Z(NP),IRL);}`

where the variable `#` is `tperp`, the closest distance to any boundary. If you are starting from scratch it is easiest to code `HOWNEAR` as a subroutine.

`SUBROUTINE HOWNEAR` is concerned with the geometry but its job is somewhat simpler to define than for `SUBROUTINE HOWFAR` since it need only return one value, namely `tperp`, the distance to the closest boundary in **any** direction. Unlike `HOWFAR`, `HOWNEAR` passes the transport parameters parameters to the subroutine as:

```
            SUBROUTINE HOWNEAR(tperp, x,y,z, irl);
```

where `x,y,z` are the current positions of the particle and `irl` is its current region. The rest of the information about the geometry is passed in whatever `COMMONs` contain the necessary information. In NRCC user codes and the tutorial codes this is `COMMON/GEOM/;` but it can be called anything the user wants.

One simplification is that the routine does not have to handle regions outside the geometry (as `SUBROUTINE HOWFAR` must in order to have them discarded).

In complex geometries, the mathematics of `HOWNEAR` can become difficult and sometimes almost impossible! If it is easier for the user to compute some lower bound to the nearest distance, this could be used in `HOWNEAR`. In the worst case, one can return `tperp` as 0.0 in which case the code goes into single scattering mode if using the exact boundary crossing algorithm. In fact, this is an easy general way to turn on single scattering throughout the entire geometry.

The following is an example of a HOWNEAR routine for the geometry given above in section 3.5 concerning HOWFAR. See also section 5.6 (page 208).

```
SUBROUTINE HOWNEAR(tperp,X,Y.Z,IRL);
COMMON/GEOM/ZTHICK; "slab thickness defined in main"
tperp = min(Z,ZTHICK-Z);
RETURN;  END;
```

3.7 Specifications for AUSGAB

The subroutine AUSGAB is called by EGS with the statement:

```
CALL AUSGAB(IARG);
```

The argument IARG indicates the situation under which AUSGAB is being called. IARG can take on 35 values starting from zero (i.e., IARG=0 through IARG=34), although only the first five are called by default in EGSnrc. The remaining 30 IARG values must be "switched-on" by means of the array IAUSFL. The value for IARG and the corresponding situations are given in Table 5.

On occasion one wants to terminate a particle histroy from withing AUSGAB. To do this cleanly and efficiently set `E(NP) = 0`. One can also set the weight to zero if you do not want any EDEP energy to be scored.

The IARG values in table 5 are the ones generally required in the majority of situations in which EGS is used to simulate electromagnetic cascade shower development. In particular, IARG=0 is useful whenever track lengths are being calculated or when charged particle ionization loss is needed. Also, as a check on energy conservation, EDEP can be summed in AUSGAB for all IARG values less than 5. The extended IARG range allows the user to extract additional information without making changes to the EGS coding. To do this we have created the integer flag array, IAUSFL(J), for J=1 through 35. It takes on values of 1 or 0 depending on whether AUSGAB is called or not, respectively. For J=1 through 5, which corresponds to IARG=0 through 4, IAUSFL(J)=1 (default). In other words, AUSGAB is always called for the situations listed in Table 5. For the remaining values of J, corresponding to IARG=5 through 34, IAUSFL(J)=0 (default). The value for IARG and the corresponding situations for this upper set of IARG values are shown in Table 6.

As an example of how to write an AUSGAB subprogram, consider the previous three region geometry (Fig. 19). Suppose that we wish to score (i.e., output on the line printer) only photons that go from Region 2 into Region 3. The AUSGAB subprogram that will accomplish this is given below. In this example we print out the stack variables plus IARG.

```
SUBROUTINE AUSGAB(IARG);
COMIN/STACK/;
"only output information for photons that are discarded"
"(by the user) in region 3"
```

Table 5: Values of IARG which are on by default and for which energy is deposited.

IARG	Situation
0	Particle is going to be transported by distance TVSTEP.
1	Particle is going to be discarded because its energy is below the cutoff ECUT (for charged particles) or PCUT (for photons)—but its energy is larger than the corresponding PEGS cutoff AE or AP, respectively.
2	Particle is going to be discarded because its energy is below both ECUT and AE (or PCUT and AP).
3	Particle is going to be discarded because the user requested it (in HOWFAR usually or by range rejection).
4	The difference between the energy of the incident particle and all of the final products is being deposited locally. This energy is due to sub-threshold relaxation events.

```
    IF(IARG = 3 & IQ(NP) = 0 & IR(NP) =  3) [
        OUTPUT E(NP),X(NP),Y(NP),Z(NP),U(NP),V(NP),W(NP),
          IQ(NP),IR(NP),IARG;  (7G15.7,3I5);]
    RETURN;  END;
```

The tutorial programs described in section 4(page 159) give examples of various types of AUSGAB routines.

3.7.1 Checking for STACK overflow

Unlike EGS4, EGSnrc prevents the user from placing too many particles on the STACK. This is implemented via a macro called $CHECK-STACK(#,#); which will terminate the execution if the stack pointer exceeds $MXSTACK. This check takes some time (not much) but to optimize the speed of a calculation, one might want to redefine it as a null macro once one is moving to production runs.

```
REPLACE {$CHECK-STACK(#,#);} WITH {;}
```

Even if the above macro is nulled out, EGSnrc continues to verify that there is adequate space on the stack whenever radiative splitting is being used.

3.7.2 Status of the STACK at various AUSGAB calls

In EGS4, the general rule was that after an interaction, the lowest energy particle was always on the top of the STACK. This general rule has been relaxed in EGSnrc, partially because of

Table 6: Values of `IARG` which are off by default.

`IARG`	`IAUSFL`	Situation
5	6	Particle has been transported by distance TVSTEP.
6	7	A bremsstrahlung interaction is to occur and a call to BREMS is about to be made in ELECTR.
7	8	Returned to ELECTR after a call to BREMS was made.
8	9	A Moller interaction is to occur and a call to MOLLER is about to be made in ELECTR.
9	10	Returned to ELECTR after a call to MOLLER was made.
10	11	A Bhabha interaction is to occur and a call to BHABHA is about to be made in ELECTR.
11	12	Returned to ELECTR after a call to BHABHA was made.
12	13	An in-flight annihilation of the positron is to occur and a call to ANNIH is about to be made in ELECTR.
13	14	Returned to ELECTR after a call to ANNIH was made.
14	15	A positron has annihilated at rest.
15	16	A pair production interaction is to occur and a call to PAIR is about to be made in PHOTON.
16	17	Returned to PHOTON after a call to PAIR was made.
17	18	A Compton interaction is to occur and a call to COMPT is about to be made in PHOTON.
18	19	Returned to PHOTON after a call to COMPT was made.
19	20	A photoelectric interaction is to occur and a call to PHOTO is about to be made in PHOTON.
20	21	Returned to PHOTON after a call to PHOTO was made (assuming `NP` is non-zero).
21	22	Subroutine UPHI was just entered. Not entered in all cases now since the sampling is done more efficiently directly in some subroutines.
22	23	Subroutine UPHI was just exited.
23	24	A coherent (Rayleigh) interaction is about to occur.
24	25	A coherent (Rayleigh) interaction has just occurred.
25	26	A fluorescent photon has just been created in RELAX.
26	27	A Coster-Kronig electron has just been created in RELAX.
27	28	An Auger electron has just been created in RELAX.
28	29	A positron is about to annihilate at rest.

Table 7: Values of `IARG` which are off by default (continued).

`IARG`	`IAUSFL`	Situation
29	30	A photonuclear event is about to occur and a call to PHOTONUC is about to be made in PHOTON.
30	31	Returned to PHOTON after a call to PHOTONUC was made.
31	32	Electron impact ionization is about to occur and a call to eii_sample is about to be made in MOLLER.
32	33	Returned to MOLLER after a call to eii_sample was made.
33	34	A fluorescent photon from relaxations was just discarded below PCUT and the energy stored in edep_local.
34	35	An electron from relaxations was just discarded below ECUT and the energy stored in edep_local.

the new physics, and partially because memory is not as expensive as it once was.

With the addition of relaxation events in EGSnrc, the possibilities about what is on the stack after various events has become more complex than in EGS4. For example, after a Compton scattering event in `EGS4` (`IARG`=18), one could count on the `STACK` having one more particle on it compared to just before the call (`IARG`=17). In EGSnrc, when bound compton is being simulated, this is no longer the case. Firstly, because of the rejection techniques used to determine if the event actually took place (see section 2.2.2,page 35) it is possible that only the original photon is on the stack. At the opposite extreme, the scattered photon and electron may be on the `STACK` along with 2 or 3 relaxation particles (fluorescent x-rays, Auger electrons, Coster-Kronig electrons). Another complication occurs if Russian Roulette is being used since in this case there can be events in which there is nothing left on the `STACK` after the event (e.g. after a pair event where the particles are discarded). To help sort out these situations, EGSnrc has the variable `NPold` in `COMIN STACK` which points to the position on the stack of the initiating particle (the photon prior to Compton scattering, Rayleigh scattering, pair production, photo-electric event or the electron prior to Moller scattering or a bremsstrahlung event, etc).

Another change with EGSnrc is that the ordering of the resultant particles is not rigorously set as lowest to highest energy. This saves computing time and with the reduced cost of memory, the issue of the size of the `STACK` is not so critical, at least not for low energy simulations (<100 MeV). However, for high-energy simulations this may cause problems. To overcome these the user code could define the macros `$PARTICLE-SELECTION-MOLLER;` (where `MOLLER` can be any interaction) to sort the stack by energy after whatever interactions needed it. These macros are by default null macros, but they are placed immediately after the call to each subroutine which samples a type of event.

The ordering on the `STACK` is summarized in the following. Note that the introduction of electron impact ionization and relaxation after other than the photoelectric events has led to significant changes.

Photoelectric `NPold` points at the photoelectron unless the initial photon energy is <1 keV in which case there is an error message and an electron with the photon's energy is created. Another exception occurs if the photon's energy is less than the N-shell binding (which can only occur for Z≥96), in which case `EDEP` is set to the photon's energy and a photon of energy 0.0 is left at `NPold`. For normal events, the particles from `NPold+1` to `NP` are due to relaxation events. If internal Russian Roulette is being played (see section 3.11.3), it is possible for `NP` to be `NPOLD - 1` after the event where there is no fluorescent photon and all the resulting electrons are killed by the Russian Roulette.

Compton scattering For Klein-Nishina modelling, the scattered photon and electron are in `NPold` and `NPold+1 = NP` respectively (i.e. the energy is not ordered). When bound compton scattering is being modelled, there are several possibilities. At one extreme, since a rejection technique is used, the scattering may not occur. In this case, `NPold = NP`. If the scattering occurs, then the scattered photon and compton electron are in `NPold` and `NPold+1`, as in the Klein-Nishina case. If there are any relaxation particles (i.e. `NP > NPold+1`), they are found in `NPold+2` to `NP`. There is a slight complication if the internal Russian Roulette is being used (see section 3.11.3) since, if all the electrons disappear because of Russian Roulette, then NPold=NP. To distinguish these two cases, the flag `i_survived_RR` can be used. It has a value 0 for the case of an unbound interaction being rejected and has a value > 0 if the interaction occurred and all secondaries were eliminated by Russian Roulette.

Pair Production For pair production the electron and positron are in `NPold` and `NPold+1 = NP` with the lower energy particle on the top of the `STACK` (at `NP`). If internal Russian Roulette kills the electron - positron pair, then, `NP = NPold-1` unless this would leave `NP = 0`, in which a zero energy photon is placed on the `STACK` with `NP = 1`.

Rayleigh scattering In this case, the photon is still at `NPold`.

Brem production In this case, the resulting electron is always at `NPold` and the photon is on top of the `STACK` at `NPold+1`, i.e. the lowest energy particle is not necessarily on the top of the STACK. When bremsstrahlung splitting is being used. The photons are between `NPold+1` and `NP`. Note that they are not ordered by energy.

Moller scatter There are 3 possible situations after a call to the Moller routine. If a 'standard' Moller scattering has occurred, the resulting, lower energy electron is in `NPold+1 = NP` and the primary is at `NPold`. It is possible that the incident electron's energy is below the threshold for creating a secondary and the return is with `NP`=`NPold`. If this call to Moller led to an electron impact ionization, the primary electron will be at `NPold` and there may be relaxation events from `NPold+1` to `NP` although in the extreme case of all energy deposition below threshold `NP`=`NPold` and there is only the primary electron.

Bhabha scatter The resulting positron and electron are at `NPold` and `NPold+1`, ordered by their energies.

Annihilation The two resulting photons are at `NPold` and `NPold+1` unless bremsstrahlung splitting is being used in which case the photons go from `NPold` to `NP`. The particle energies are not ordered.

Relaxation There can be separate calls to AUSGAB after the individual relaxation events (fluorescent photon, Coster-Kronig or Auger electrons) but note that after the complete relaxation there may also be ASUGAB calls post Moller, photoelectric and Compton and one should avoid double counting.

3.8 Terminating particle histories

The standard method to terminate a history is by setting IDISC to a non-zero value in HOWFAR (see section 3.5, page 134). Another method is to set the weight of a particle to 0.0, usually in AUSGAB under some conditions (e.g. when doing Russian Roulette). This technique is used in EGS4 by checking the weight of a particle as it enters HOWFAR and setting IDISC non-zero if the weight is zero. This is somewhat wasteful since it means that various parameters are calculated for this particle, despite the fact that it is going to be discarded. In particular, we found that if we were also using the standard photon forcing macro we got into an infinite loop. This could have been corrected by re-coding the macro to handle weight 0.0 particles differently, but it was decided that we could save more time by adding a test at the start of the new electron or new photon loops whereby a particle is discarded immediately via the `USER-PHOTON-DISCARD` or `USER-ELECTRON-DISCARD` if the weight is 0.0. This generates a call to AUSGAB with `IARG = 3`. Positrons discarded this way do not create annihilation photons. Note that `ELKE` is not available with this call since it is assumed the particle is being thrown away. One should still have HOWFAR set `IDISC = 1` when the weight is 0.0, especially if the weight is set to zero for a particle which is not new.

3.9 Random number generators

EGSnrc is supplied with two random number generators, RANLUX and RANMAR. RANMAR is the generator used with EGS4 in the unix distributions[107, 108] and although it is known to fail certain theoretical tests, we have no experience of it causing problems. The RANLUX generator[109, 110], which is treated as the default generator with EGSnrc, is a similar sort of generator which comes with a variety of "luxury levels", from 0 to 4 and a period of greater than 10^{165}. According to James, RANMAR has a quality somewhere between luxury level 1 and 2 of RANLUX and we have found that it gives incorrect answers in some practical EGSnrc calculations with luxury level 0. However, with luxury level 1 or higher we have seen no problems. We have utilized RANLUX as the default rng because it allows explicit testing with higher quality sequences if there are ever any doubts.

Both random number generators offer several important features. Firstly, they are completely portable, producing the same sequences on different machines, although RANMAR occasionally gets slightly out of sequence and sometimes optimizers on a given machine will cause the sequences to differ. We have not seen this behaviour with RANLUX. An even more important feature is that either generator can be initialized and guaranteed to produce a random number sequence which is independent from other sequences. This is very useful for doing runs in parallel on multiple machines.

The default generator is defined in `$HEN_HOUSE/specs/all_common.spec` by the statement RANDOM = $(EGS_SOURCEDIR)ranlux. This can be changed to ranmar or, for individual user codes, it can be changed by adding the statement RANDOM = $(EGS_SOURCEDIR)ranmar to the user_code.make file, somewhere before the SOURCES = statement (if it exists). See Report PIRS-877 for further information about make files in the EGSnrcMP environment[14].

These files provide the following macros which the user is free to use in their user code:

```
;COMIN/RANDOM/;
$RANDOMSET#;
$DEFAULT-LL   (1 by default)
$RNG-INITIALISATION;    (which is only needed optionally)
$INITIALIZE RNG USING # AND #; (a more useful version of the above)
$STORE RNG STATE ON UNIT #;
$PUT RNG STATE ON UNIT #;
$RETRIEVE RNG STATE FROM UNIT #;
$SHOW-RNG-STATE(#);
$PRINT-RNG-STATE(#,#);
$RNG-INPUTS(#,#,#,#);  (uses GET_INPUTS routine)
```

To generate a random number, say RNUMBER, include:

```
$RANDOMSET RNUMBER;
```

Wherever the user needs to use $RANDOMSET#;, they must ensure COMIN/RANDOM/ is present. If the user is happy with luxury level 1 and the same sequence for each calculation, the RANLUX generator is self-initializing. However, to use other luxury levels or other sequences, the user code should include a statement of the type:

```
$INITIALIZE RNG USING luxury_level AND iseed;
```

where luxury_level is an integer between 0 and 4 and iseed is any positive integer (and if left 0, a default of 314159265 is used).

For a standard production run of one of the NRC user codes (CAVRZnrc for ^{60}Co photons incident on a thimble chamber with splitting of 130) we get the timing results shown in table 8, although these values are revised from the pre-2003 printings and depend on the details of any simulation. Given the iproblems encountered using luxury level 0, luxury level 1 has been adopted as the default with EGSnrc. However, given a new coding of RANMAR to generate groups of random numbers using a function call, we find that RANMAR is faster (by 5% say overall). The penalty ifor using higher RANLUX luxury levels becomes increasingly more important at higher luxury levels and a user may want to verify that for their simulations, use of the higher level makes no difference. We would appreciate being informed of any cases found where luxury level 1 was not adequate.

When using the RANMAR generator, the initialisation looks like:

```
$INITIALIZE RNG USING IXX AND JXX;
```

where IXX and JXX are two integers seeds with:

0< IXX<=31328 and 0<JXX<=30081

The other macros are for use when saving the state of a random number generator to

Table 8: Calculation times for runs with CAVRZnrc for the same calculations along with estimates of the time taken by the random number generator at different luxury levels. The results for luxury level 0 are different when high precision is obtained.

Luxury level	total CPU time	time taken by RANLUX	
	s	s	%
0	194	24	12%
1	210	39	18%
2	240	68	28%
3	315	142	45%
4	415	242	58%

disk and possibly restarting a run or other book keeping tasks.

There are two files available for use in codes doing correlated sampling. These files are:

```
ranlux.correlations    or      ranmar.corrrelations
```

These files define the macros:

```
$STORE-RNG(#);
$RESET-RNG(#);
```

which store and reset an arbitrary number of random number states (≤`$MXRNGDIM` which is 5 by default).

Note: because of how the RANLUX generator is implemented, it is essential that any redefined `COMIN/RANDOM` must include an integer variable called `rng_seed`. This variable is initialized to 999999 in `egs_set_default` which replaces `BLOCK DATA` in the EGSnrcMP environment[14].

3.10 Summary of transport parameter

In many cases transport parameters and cross section options will be controlled via an input file that uses a `option = value` format (*eg.* all RZ user codes, all C++ codes). The following provides a summary of the options available along with possible settings.

```
Global ECUT=     Global (in all regions) electron transport cut
                 off energy (in MeV). If this input is missing,
                 AE(medium) will be used.
                 [ ECUT ]

Global PCUT=     Global (in all regions) photon transport cut
                 off energy (in MeV). If this input is missing,
                 AP(medium) will be used.
                 [ PCUT ]
```

```
Global SMAX=      Global (in all regions) maximum step-size
                  restriction for electron transport (in cm).
                  If missing, no geometrical step-size restrictions
                  will be employed. Note that if you use the default
                  EGSnrc electron-step algorithm, no SMAX-restriction
                  is necessary. Option is useful for transport in low
                  density materials (air) when PRESTA behaviour is
                  turned on (see below)
                  [ SMAXIR ]

ESTEPE=           Maximum fractional energy loss per step.
                  Note that this is a global option only, no
                  region-by-region setting is possible. If missing,
                  the default is 0.25 (25%).
                  [ ESTEPE ]

XImax=            Maximum first elastic scattering moment per step.
                  Default is 0.5, NEVER use value greater than 1 as
                  this is beyond the range of MS data available.
                  [ XIMAX ]

Boundary crossing algorithm= EXACT (default), PRESTA-I
                  There are two selections possible: EXACT means
                  the algorithm will cross boundaries in a single
                  scattering (SS) mode, the distance from a boundary
                  at which the transition to SS mode is made is
                  determined by 'Skin depth for BCA' (see below).
                  The second option is PRESTA-I, if selected boundaries
                  will be crossed a la PRESTA, i.e. with lateral
                  correlations turned off and MS forced at boundaries.
                  Default is EXACT.
                  [ bca_algorithm, exact_bca ]

Skin depth for BCA=
                  Determines the distance from a boundary (in elastic
                  MFP) at which the algorithm will go into single
                  scattering mode (if EXACT boundary crossing) or
                  switch off lateral correlations (if PRESTA-I boundary
                  crossing). Default value is 3 for EXACT or
                  exp(BLCMIN)/BLCMIN for PRESTA-I (see the PRESTA paper
                  for a definition of BLCMIN). Note that if you choose
                  EXACT boundary crossing and set Skin depth for BCA
                  to a very large number (e.g. 1e10), the entire
                  calculation will be in single-scattering mode. If you
                  choose PRESTA-I boundary crossing and make Skin depth
                  for BCA large, you will get default EGS4 behaviour
                  (no PRESTA).
                  [ skindepth_for_bca ]
```

```
Electron-step algorithm= PRESTA-II (default), PRESTA-I (legacy)
                 Determines the algorithm used to take into account
                 lateral and longitudinal correlations in a
                 condensed history step.
                 [ transport_algorithm ]

Spin effects=    Off, On (default)
                 Turns off/on spin effects for electron elastic
                 scattering. Spin On is ABSOLUTELY necessary for
                 good backscattering calculations. Will make a
                 difference even in 'well conditioned' situations
                 (e.g. depth dose curves for RTP energy range
                 electrons).
                 [ spin_effects ]

Brems angular sampling= Simple, KM (default)
                 If Simple, use only the leading term of the Koch-Motz
                 distribution to determine the emission angle of
                 bremsstrahlung photons. If KM, complete
                 modified Koch-Motz 2BS is used (modifications
                 concern proper handling of kinematics at low energies,
                 makes 2BS almost the same as 2BN at low energies).
                 [ IBRDST ]

Brems cross sections= BH (default), NIST, NRC
                 If BH is selected, the Bethe-Heitler bremsstrahlung
                 cross sections (Coulomb corrected above 50 MeV)
                 will be used. If NIST is selected, the NIST brems
                 cross section data base (which is the basis for
                 the ICRU radiative stopping powers) will be employed.
                 Differences are negligible for E > ,say, 10 MeV,
                 but significant in the keV energy range. If NRC is
                 selected, the NRC brems cross-section data base will
                 be used, which is a version of the NIST data base
                 with corrected electron-electron brems contributions
                 (corrections to the NIST data is typically only
                 significant for low values of the atomic number Z
                 and for k/T < 0.005).
                 [ ibr_nist ]

Triplet production= On or Off (default).  Turns on/off simulation
                 of triplet production.  If On, then Borsellino's
                 first Born approximation is used to sample triplet
                 events based on the triplet cross-section data.
                 [ itriplet ]

Bound Compton scattering=  On, Off, Simple or norej (default)
                 If Off, Compton scattering will be treated with
```

```
                  Klein-Nishina, with On Compton scattering is
                  treated in the Impulse approximation.
                  With Simple, the impulse approximation incoherent
                  scattering function will be used (i.e., no Doppler
                  broadenning). With norej the actual total bound
                  Compton cross section is used and there are no
                  rejections at run time.
                  Make sure to turn on for low energy applications,
                  not necessary above, say, 1 MeV.
                  [ IBCMP ]

 Radiative Compton corrections= On or Off (default). If On, then
                  include radiative corrections for Compton scattering.
                  Equations are based on original Brown & Feynman
                  equations (Phys. Rev. 85, p 231--1952).  Requires
                  a change to the user codes Makefile to include
                  $(EGS_SOURCEDIR)rad_compton1.mortran in the
                  SOURCES (just before
                  $(EGS_SOURCEDIR)get_inputs.mortran).
                  [ radc_flag ]

 Electron Impact Ionization= Off (default), On, casnati, kolbenstvedt,
                  gryzinski or penelope.  If set to On or ik, then
                  use Kawrakow's theory to derive EII cross-sections.
                  If set to casnati, then use the cross-sections of
                  Casnati (from file $HEN_HOUSE/data/eii_casnati.data).
                  Similar for kolbenstvedt, gryzinski and penelope.
                  This is only of interest in kV X-ray calculations.
                  Note that the user can supply their own EII
                  cross-section data as well. The requirement is that
                  the file eii_suffix.data exists in the $HEN_HOUSE/data
                  directory, where suffix is the name specified.
                  Entry is case-sensitive except for Off, On or ik.
                  [ eii_flag, eii_xfile ]

 Pair angular sampling= Off, Simple (default), KM.
                  If off, pairs are set in motion at an angle m/E
                  relative to the photon direction (m is electron rest
                  energy, E the photon energy). Simple turns on
                  the leading term of the angular distribution
                  (this is sufficient for most applications),
                  KM (comes from Koch and Motz) turns on using 2BS
                  from the article by Koch and Motz.
                  Default is Simple, make sure you always use
                  Simple or KM
                  [ IPRDST ]

 Pair cross sections= BH (default) or NRC.  If set to BH, then use
```

```
                  Bethe-Heitler pair production cross-sections.  If set
                  to NRC, then use NRC pair production cross-sections
                  (in file $HEN_HOUSE/data/pair_nrc1.data).  Only
                  of interest at low energies, where the NRC cross-
                  sections take into account the asymmetry in the
                  positron-electron energy distribution.
                  [ pair_nrc ]

Photon cross sections= Photon cross-section data.  Current options are
                  si (Storm-Israel), epdl (Evaluated Photon Data
                  Library), xcom (default), pegs4, mcdf-xcom and
                  mcdf-epdl:
                  Allows the use of photon cross-sections other than
                  from the PEGS4 file (unless the pegs4 option is
                  specified).  Options mcdf-xcom and mcdf-epdl use
                  Sabbatucci and Salvat's renormalized photoelectric
                  cross sections with either xcom or epdl for all other
                  cross sections.  These are more accurate but can
                  increase CPU time by up to 6 %.
                  Note that the user can supply their own cross-section
                  data as well. The requirement is that the files
                  photon_xsections_photo.data,
                  photon_xsections_pair.data,
                  photon_xsections_triplet.data, and
                  photon_xsections_rayleigh.data exist in the
                  $HEN_HOUSE/data directory, where photon_xsections
                  is the name specified. This entry is case-sensitive.
                  [ photon_xsections ]

Photon cross-sections output= Off (default) or On.  If On, then
                  a file $EGS_HOME/user_code/inputfile.xsections is
                  output containing photon cross-section data used.
                  [ xsec_out ]

Compton cross sections= Bound Compton cross-section data.  User-
                  supplied bound Compton cross-sections in the file
                  $HEN_HOUSE/data/comp_xsections_compton.data, where
                  comp_xsections is the name supplied for this input.
                  This is only used if Bound Compton scattering= Simple
                  and is not available on a region-by-region basis
                  (see below).  The default file (ie in the absence
                  of any user-supplied data) is compton_sigma.data.
                  [ comp_xsections ]

Rayleigh scattering= Off, On (default), custom
                  If On, turns on coherent (Rayleigh) scattering.
                  Default is On. Should be turned on for low energy
```

```
                        applications. If custom, user must provide media names
                        and form factor files for each desired medium. The
                        rest of the media use the default atomic form factors.
                        A PEGS4 data set is not required anymore.
                        [ IRAYLR ]

       ff media names = A list of media names (must match media found in
                        PEGS4 data file) for which the user is going to
                        provide custom Rayleigh form factor data.
                        [ iray_ff_media($MXMED) ]

       ff file names = A list of names of files containing the Rayleigh
                       form factor data for the media specified by
                       the ff media names = input above.  Full directory
                       paths must be given for all files, and for each medium
                       specified, iray_ff_media(i), there must be a
                       corresponding file name, iray_ff_file(i).  For
                       example files, see the directory
                       $HEN_HOUSE/data/molecular_form_factors.
                       [ iray_ff_file($MXMED) ]

       Photonuclear attenuation= Off (default) or On
                        If On, models the photonuclear effect. Current
                        implementation is crude. Available on a
                        region-by-region basis (see below)
                        [ IPHOTONUCR ]

       Photonuclear cross sections= Total photonuclear cross sections. User-
                        supplied total photonuclear cross-sections in
                        $HEN_HOUSE/data/photonuc_xsections_photonuc.data,
                        where photonuc_xsections is the name supplied for
                        this input (case sensitive). In the absence of
                        any user-supplied data, or if photonuc_xsections
                        is set to 'default', the default file is
                        iaea_photonuc.data.
                        [ photonuc_xsections ]

       Photoelectron angular sampling= Off or On (default)
                        If Off, photo-electrons get the direction of the
                        ‘mother’ photon, with On, Sauter’s formula is
                        used (which is, strictly speaking, valid only for
                        K-shell photo-absorption).
                        If the user has a better approach, replace the macro
                            $SELECT-PHOTOELECTRON-DIRECTION;
                        The only application encountered where this option
                        made a small difference was a big ion chamber
                        (cavity size comparable with electron range)
                        with high-Z walls in a low energy photon beam.
```

```
                    [ IPHTER ]

Atomic relaxations= Off, On, eadl (default), simple
                    On defaults to eadl.
                    When simulating atomic relaxations:
                    - In photo-electric absorption events, the element
                      (if material is mixture) and the shell the photon
                      is interacting with are sampled from the appropriate
                      cross sections
                    - Shell vacancies created in photoelectric,
                      compton and electron impact ionization events
                      are relaxed via emission of fluorescent X-Rays,
                      Auger and Koster-Cronig electrons.
                      The eadl option features a more accurate treatment
                      of relaxation events and uses binding energies
                      consistent with those in of the photon cross sections
                      used in the simulation.  If using mcdf-xcom or
                      mcdf-epdl photon cross sections, you cannot use
                      the simple option and this will automatically get
                      reset to eadl. Make sure to use eadl or simple for
                      low energy applications.
                      [ IEDGFL ]
```

Atomic relaxations, Rayleigh scattering, Photoelectron angular sampling, Bound Compton scattering and photonuclear effect can also be turned On/Off on a region-by-region basis. An example for Atomic relaxations on a region-by-region basis is:

```
Atomic relaxations= On in Regions   or
Atomic relaxations= Off in regions
```

Then define the regions in which you want the feature to be turned on:

```
Bound Compton start region=
Bound Compton stop region=
         or
Rayleigh start region=
Rayleigh stop region=
         or
Relaxations start region=
Relaxations stop region=
         or
PE sampling start region=
PE sampling stop region=
```

each followed by a list of one or more start and stop regions separated by commas. Example:

```
 Atomic relaxations= On in Regions
 Relaxations start region=  1, 40
 Relaxations stop region=  10, 99
```

will first turn off relaxations everywhere and then turn on in regions 1-10 and 40-99. Note that the input is checked against minimum and maximum region numbers and ignored if `start region < 1` or `stop_region > $MXREG` or `start region > stop region`.

`ECUT`, `PCUT` and `SMAX` can also be set on a region-by-region basis. To do so, include in the input file

```
Set XXXX=              f_value1, f_value2, ...
Set XXXX start region= i_value1, i_value2, ...
Set XXXX stop region=  j_value1, j_value2, ...
```

where `XXXX` is `ECUT`, `PCUT` or `SMAX`, `f_value1`, `f_value2`,... are the desired values for `XXXX` and `i_value_i` and `j_value_i` are the start and stop regions.

3.11 Variance Reduction Options

Three forms of variance reduction techniques have been implemented directly into the EGSnrc system in order to allow for more efficient calculations. With EGS4 user codes they had to be implemented via calls to `AUSGAB` or other means, and this led to inefficiencies. In all three cases, if the user does nothing to turn on these options explicitly, then they are NOT used.

3.11.1 Range rejection

Within EGSnrc the range of the electron is available at every step. This is not the true range, but the range determined by:

$$\texttt{E\_RANGE} = \int_{E_{min}}^{E} \frac{dE'}{L(E', AE)}$$

where $L(E', AE)$ is the restricted stopping power for a given value of AE and E_{min} is the lowest energy for which PEGS4 produces a stopping power (this is somewhat less than AE, but not much). The value of `E_RANGE` is an upper limit on the distance an electron can travel in the simulation because discrete events may shorten the pathlength.

Range rejection is implemented by a macro `$RANGE-DISCARD` checks the electron range against the distance to the nearest boundary on every step. The history is terminated whenever the range is shorter than the distance to the boundary and if requested by the user. Since the range and distance are calculated for other purposes, this check is very fast and can save a large amount of time, especially in large regions. Since this is a user controlled discard, it goes via the `USER-ELECTRON-DISCARD`, generating and `IARG = 3` call to `AUSGAB`.

This technique does involve an approximation since the electron could emit a brem photon which could escape the region, even if the electron itself could not. To control the extent of this approximation, the range rejection is done only if the electron's energy is below an energy threshold which can be set for every region, viz `e_max_rr`. By judicial choice of this value, the approximation can be made very accurate while still obtaining very significant gains in efficiency (see ref [88] for a detailed discussion, where the parameter `ESAVE` in that paper is equivalent to `e_max_rr` here).

Range rejection is implemented on a regional basis by setting the flags `i_do_rr(irl)` to 1 for all regions `irl` for which range rejection is required and by assigning values to the array `e_max_rr(irl)`. Note that if `e_max_rr(irl)` is not assigned a value, its default value of 0.0 effectively turns off range rejection, even if `i_do_rr(irl)` value is 1. Both arrays are in `COMIN/EGS-VARIANCE-REDUCTION`. They need be set before the first call to `SHOWER`. Note that `e_max_rr(irl)` refers to the electrons total energy (i.e. includes the rest mass, as does `ECUT`).

The user is also free to implement other, possibly more efficient forms of range rejection. This is done by defining the macro `$USER-RANGE-DISCARD` which is called immediately after the above macro since in general, the macro `$RANGE-DISCARD` executes very quickly and can avoid use of the user's, presumably more time consuming range rejection. An example of `$USER-RANGE-DISCARD` is given in CAVRZnrc where range rejection is done on any particle which cannot reach the region where the dose is required. This can terminate many histories earlier than `$RANGE-DISCARD` because it tests against getting to the region of interest as opposed to just getting out of the local region. However, it requires some approximations and takes longer to compute on each step.

3.11.2 Bremsstrahlung Splitting

Bremsstrahlung splitting is a technique which can provide a factor of 4 or more improvement in efficiency when modelling brem beams generated by medical accelerators[88]. Each time an electron emits a photon, the simulation emits an arbitrary number of brem photons with their weight suitably reduced. The electron's energy is decremented by the energy given off by one of these photons. This preserves accurate energy loss straggling of the electron at the expense of no longer having exact energy conservation for each history, although energy is conserved "on average". The advantage of doing this splitting within the routine BREMS is that various constants for the electron's energy are only calculated once and the sampling is therefore faster. The electron is always found at `NPold` on the `STACK` and the photons are not sorted by energy.

To accomplish bremsstrahlung splitting, the variable `nbr_split` which is in `COMIN/EGS-VARIANCE-REDUCTION` must be set to the number of brem photons wanted at each discrete interaction. See section 2.4.2.iv (page 73).

There are no internal limits on the number of splits that may be used except that the stack size, `$MXSTACK`, may not be exceeded. The user code can override the value of `$MXSTACK` if a larger value is needed.

Note that once set, EGSnrc will split brem at all generations. This is appropriate when Russian Roulette is being played since it means that second generation bremsstrahlung photons will have the same weight as first generation photons. However, it can become very time consuming and counter productive when Russian Roulette is not being played because second and higher order bremsstrahlung photons have much smaller weights. To overcome this problem, the user might want to use calls to `AUSGAB` to reduce the value of `nbr_split` after each first generation bremsstrahlung interaction and then re-initialize it at the start of the next history. This is the technique used in the BEAM code[88, 111].

3.11.3 Russian Roulette

Russian Roulette is a standard variance reduction technique which in `EGS4` had to be done via call to `AUSGAB`. This works but is somewhat slower than need be. For example, say that the particles created in a pair event are to be discarded. In EGS4 the sampling routines must still determine all their parameters and then remove them from the `STACK` whereas in EGSnrc Russian Roulette is played prior to the sampling, and then not done if not needed. In less dramatic cases, the EGSnrc approach avoids extra steps were the EGS4 system had to go through to eliminate the particle.

Russian Roulette, as implemented in EGSnrc, is a user option which is turned on by setting the integer flag `i_play_RR` to 1 and by setting the probability `prob_RR` to an appropriate value. Both these parameters are in `COMIN/EGS-VARIANCE-REDUCTION`.

If Russian Roulette is being used in conjunction with bremsstrahlung splitting, the appropriate value of `prob_RR` is `1./nbr_split`.

The integer variable `i_survived_RR` is also in `COMIN/EGS-VARIANCE-REDUCTION`. It is 0 if Russian Roulette is not played or if all particles survived when Russian Roulette was played on the previous interaction. Otherwise, its value tells how many particles were discarded by Russian Roulette on the previous interaction.

3.12 Complete Users Codes Examples

For several examples of complete EGSnrc user codes, see the TUTOR codes discussed in section 4. The EGSnrc distribution also comes with several NRC user codes which have examples of many features[112] but they have grown up over the years and have been patched so often that they are hard to follow.

3.13 Some Utility codes

3.13.1 SUBROUTINE WATCH

`SUBROUTINE WATCH` is in the file `nrcaux.mortran`. With a few simple statements in a user code, it can print a listing to the screen of what is happening in each history. This is very useful for debugging purposes. We strongly recommend that it be used with all user codes. It also creates output suitable for the EGS_Windows graphic display system[113]. For more information, see `tutor4.mortran` for an example of a simple code with WATCH included (see section 4.4, page 176).

3.13.2 ranlux_test.mortran and ranmar_test.mortran

These little codes can be used to verify that the ranlux and ranmar random number generators are working correctly on your system. Since these generators produce machine independent random numbers, the values should be identical on all machines. They are found on

$HEN_HOUSE/user_codes/ranlux_test/ and $HEN_HOUSE/user_codes/ranmar_test/. Each can be compiled and executed as a user code. The output is self explanatory but basically they both calculate 1 million random numbers and compare their sums to the expected values. See the comments at the top of the source code for further information.

Note that even with identical random number sequences, the results of a full EGSnrc calculation may not be identical on different machines or at different optimization levels. This is because statements like

```
          IF ( A < C/D) ) GOTO n;
```

may branch differently, depending on how many digits are stored in C/D on different machines or at different levels of optimization.

3.13.3 EXAMIN

The user code EXAMIN is distributed with the system. It provides easy access to the underlying data produced by PEGS4. EXAMIN tabulates many cross sections and if you have installed the `xmgr` graphics package it also plots the data (see the file `$HEN_HOUSE/utils/HOW_to_get_xmgrace` for information about `xmgrace`). The code outputs quantities such as the gamma mean free path, the relative contributions from the various components of the photon cross sections, the mean free path to discrete interactions for electrons, etc. It is a useful template for seeing how to access the data base. Note that the data presented are those from PEGS4. Although EGSnrc can model bound Compton scattering, this is done using a rejection technique and this code does not allow access to the bound Compton cross sections, only the Klein Nishina cross sections.

3.14 Latest additions

3.14.1 Charged particle transport under electromagnetic fields

Bielajew's EGS4 macros for transport under electromagnetic (EM) fields were adapted for EGSnrc by Amir Keyvanloo and others from the Cross Cancer Institute in Edmonton. These macros have been included in the current EGSnrc release. To turn on transport under EM fields, users must add the file `emf_macros.mortran` to the `MORTRAN` sources to be compiled. The subroutine `ELECTR` in `egsnrc.mortran` contains macro replacements which are empty by default, but which are activated when including the EM macros.

To include these macros in a MORTRAN user code, append `emf_macros.mortran` to `SOURCES` variable in file `user_code.make` (user_code can be any MORTRAN user code other than a BEAMnrc user code):

```
 ...
SOURCES = \
        $(EGS_SOURCEDIR)egsnrc.macros \
        ...
        $(EGS_SOURCEDIR)emf_macros.mortran \
        $(USER_CODE).mortran \
        ...
        $(EGS_SOURCEDIR)get_inputs.mortran\
        $(EGS_SOURCEDIR)get_media_inputs.mortran\
        ...
        $(EGS_SOURCEDIR)pegs4_routines.mortran \
        $(EGS_SOURCEDIR)egsnrc.mortran
 ...
```

To include these macros in an existing BEAMnrc user code, append `emf_macros.mortran` to `SOURCES` variable in file `source.make`:

```
SOURCES = $(EGS_SOURCEDIR)egsnrc.macros \
          ...
          $(EGS_SOURCEDIR)emf_macros.mortran \
          ...
          $(BEAM_HOME)beamnrc_user_macros.mortran \
          ...
          $(BEAM_CODE)_macros.mortran \
          $(EGS_SOURCEDIR)egs_utilities.mortran \
          $(BEAM_HOME)beam_main.mortran\
          $(BEAM_HOME)beamnrc.mortran \
          ...
          $(EGS_SOURCEDIR)egsnrc.mortran
```

In the case of a BEAMnrc source library, append `emf_macros.mortran` to `LIB_SOURCES` variable in file `source.make`:

```
LIB_SOURCES = $(BEAM_HOME)beam_lib.macros \
          $(EGS_SOURCEDIR)egsnrc.macros \
          ...
          $(EGS_SOURCEDIR)emf_macros.mortran \
          ...
          $(BEAM_HOME)beamnrc_user_macros.mortran \
          ...
          $(BEAM_CODE)_macros.mortran \
          $(BEAM_HOME)beam_lib.mortran\
          $(BEAM_HOME)beamnrc.mortran \
          ...
          $(EGS_SOURCEDIR)egsnrc.mortran
```

For C++ user codes one must append `emf_macros.mortran` to the `EGSPP_USER_MACROS` variable in the `Makefile`:

```
 ...
# Specify the name of the user code.
# The name of the executable is determined from this variable.
#
USER_CODE = cavity
```

```
# The following can be used to add user macros and mortran subroutines.
# The file(s) specified here are added after egsnrc.macros, machine.macros
# and egs_c_interface2.macros but before any files that have
# executable code.
#
#EGSPP_USER_MACROS = cavity.macros
EGSPP_USER_MACROS = cavity.macros  \
                    $(EGS_SOURCEDIR)emf_macros.mortran
 ...
```

Details of the field and the step size restriction are entered by input in the MC transport parameters section:

- For magnetic fields:

  ```
  :start MC transport parameter:
   ...
   Magnetic Field = 0, 1.5, 0 # magnetic field in T
   EM ESTEPE = 0.02
   ...
  :stop MC transport parameter:
  ```

- For electric fields:

  ```
  :start MC transport parameter:
   ...
   Electric Field = 0, 3, 0 # electric field in V/cm
   EM ESTEPE = 0.02
   ...
  :stop MC transport parameter:
  ```

Please note than one can define both, an electric and a magnetic field. The `EM ESTEPE` entry sets the value for the step size restrictions based on energy loss, field change and direction change. For instance, in the static $\vec{B}_0$ field case of section 2.4.11.iii, `EM ESTEPE` corresponds to the δ value of Eq(4.11.14). By default it is set to 0.02.

3.14.2 EADL atomic relaxations

In EGSnrc the relaxation cascade of inner shell vacancies is an independent process that is initiated each time a vacancy is created

- In photo-absorption, bound Compton scattering, and electron impact ionization processes.
- Future: triplet production, electron-electron bremsstrahlung

The default implementation include relaxation events from:

- 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.

An option to remove above limitations by using EADL atomic relaxation data has been available in EGSnrc since 2011. The way to turn on EADL relaxation is by setting macro $EADL_RELAX to .true. in egsnrc.macros or in the user-code. It is currently set to .false. by default.

4 Some Short EGSnrc Tutorial Programs

EGSnrc is a powerful system which has been used in order to produce some very complex Monte Carlo simulations. In spite of this complexity the user's interface with the system is, in principle, very simple. In the following series of tutorial programs we use various aspects of these user interfaces in what we refer to as EGSnrc User Codes. In these User Codes we will introduce some of the basic scoring techniques and, at the same time, will demonstrate the power of the Mortran3 language. Formal documentation in the form of EGSnrc and PEGS4 User Manuals can be found in sections 3 and 6, respectively. An EGSnrc User Guide to Mortran3 can be found in section 8 and an overview of the system considerations is given in section 9. With the introduction of the EGSnrcMP environment, the user has a more flexible interface at the system level, but that is described fully in Report PIRS-877[14].

These tutorials are written on the assumption that the reader is generally familiar with the contents of the EGSnrc Reference Manual (section 3 of this manual, page 107), although a complete understanding is not required. In fact, the purpose of these tutorials is to make these manuals more understandable. Although the programs presented here are very simple in construction, it should become clear that with various extentions to them, generally of a bookkeeping nature, a wide range of important problems can be studied. In the following sections, sometimes only partial listings of User Codes are presented. The complete source for each (and the corresponding Fortran 77 code) is found on the EGSnrc Distribution.

Note that default parameters in EGSnrc may change from time to time, in which case there could be some minor differences when you run the calculations. Also, use of the MP system requires a Step 0: `call egs_init;` and a Step 9: `call egs_finish;` which are in the distributed versions but not updated here.

To run the tutorials, a special PEGS4 data set called `tutor_data.pegs4dat` is on the distribution at `$HEN_HOUSE/pegs4/data`. To execute the tutorials issue the command:

```
ex tutor1 "" tutor_data    or      tutor1 -p tutor_data
```

where the `""` signifies that no input file is required (except for `tutor6, tutor7`). Note that with the EGSnrcMP environment there are two other alternatives described in PIRS-877[14] (*viz.*, the right hand form above or use the `egs_gui`). To compile the tutor codes, copy any one to your own user-code area if it is not already there, eg:

```
cd $EGS_HOME/tutor1
cp $HEN_HOUSE/tutor1/tutor1.mortran .
mf tutor1
```

As alternatives to the `mf` command, when using the EGSnrcMP environment, one may use the `egs_gui` or simply issue the command `make` from `$EGS_HOME/tutor1`(see PIRS-877[14]). The meaning of the commands `mf` and `ex` are discussed in section 9 (page 294).

4.1 tutor1.mortran: 20 MeV e$^-$ through 1 mm of Ta

The geometry of all the tutorials is the same, namely, a semi-infinite slab of material is placed in a vacuum and a pencil beam of photons or electrons is incident normal to the surface. The slab is in the X-Y plane and the particles are incident at the origin travelling along the Z-axis. In the first problem, a beam of 20 MeV electrons is incident on a 1 mm thick plate of tantalum. In order to use EGSnrc to answer the question "What comes out the far side of the plate?", we have created the user code `tutor1.mortran` shown below.

Figure 20: tutor1.mortran without any comments.

```
%L
!INDENT M 4;
!INDENT F 2;
$IMPLICIT-NONE; $INTEGER I,IQIN,IRIN; $REAL XIN,YIN,ZIN,EIN,WTIN,UIN,VIN,WIN;
REPLACE {$MXMED} WITH {1}
REPLACE {$MXREG} WITH {3}
REPLACE {$MXSTACK} WITH {15}
REPLACE {;COMIN/GEOM/;} WITH {;COMMON/GEOM/ZBOUND;$REAL ZBOUND;}
REPLACE {$CALL-HOWNEAR(#);} WITH {;CALL HOWNEAR({P1},X(NP),Y(NP),Z(NP),IRL);}
;COMIN/BOUNDS,GEOM,MEDIA,MISC,THRESH/;
CHARACTER*4 MEDARR(24);
DATA  MEDARR /$S'TA',22*' '/;
CALL egs_init;
DO I=1,24[MEDIA(I,1)=MEDARR(I);]
/MED(1),MED(3)/=0;MED(2)=1;
ECUT(2)=1.5;
PCUT(2)=0.1;
;OUTPUT;('\f  Start tutor1'//' CALL HATCH to get cross-section data'/);
CALL HATCH;
;OUTPUT AE(1)-0.511, AP(1);
(/' knock-on electrons can be created and any electron followed down to'
/T40,F8.3,' MeV kinetic energy'/
'   brem photons can be created and any photon followed down to      ',
/T40,F8.3,' MeV ');
ZBOUND=0.1;
;OUTPUT;(/T19,'Kinetic Energy(MeV)',T40,'charge',T48,
'angle w.r.t. Z axis-degrees');
IQIN=-1; EIN=20.511; /XIN,YIN,ZIN/=0.0;
/UIN,VIN/=0.0;WIN=1.0; IRIN=2; WTIN=1.0;
DO I=1,10[OUTPUT I;(' Start history',I4);
CALL SHOWER(IQIN,EIN,XIN,YIN,ZIN,UIN,VIN,WIN,IRIN,WTIN); ]
CALL egs_finish;
STOP;END;
SUBROUTINE AUSGAB(IARG);
$IMPLICIT-NONE;
$INTEGER IARG; $REAL EKINE, ANGLE; COMIN/STACK/;
IF(IARG = 3)[ ANGLE=ACOS(W(NP))*180./3.14159;
   IF(IQ(NP) = 0)[EKINE=E(NP);] ELSE [EKINE=E(NP)-0.511;]
   OUTPUT EKINE,IQ(NP),ANGLE;(T21,F10.3,T33,I10,T49,F10.1);
]
RETURN;END;
```

cont...

Figure 20: continued

```
SUBROUTINE HOWFAR;
$IMPLICIT-NONE;
$REAL TVAL;
COMIN/STACK,EPCONT,GEOM/;
IF(IR(NP) =  3) [IDISC = 1;RETURN;]
ELSEIF(IR(NP) = 2)[
   IF(W(NP) > 0.0)[
      TVAL= (ZBOUND - Z(NP))/W(NP);
      IF(TVAL > USTEP)[RETURN;] ELSE[USTEP = TVAL; IRNEW=3; RETURN;]
   ]
   ELSEIF(W(NP) < 0.0)[ TVAL = -Z(NP)/W(NP);
      IF(TVAL > USTEP)[RETURN;] ELSE[USTEP = TVAL; IRNEW = 1; RETURN;]
   ]
   ELSEIF(W(NP) = 0.0)[RETURN;]
]
ELSEIF(IR(NP) = 1)[
   IF(W(NP) >  0.0)[ USTEP = 0.0;IRNEW = 2;RETURN; ] ELSE[  IDISC=1;RETURN; ]
]
END;
SUBROUTINE HOWNEAR(tperp, x, y, z, irl);
$IMPLICIT-NONE;
$REAL tperp, x, y, z;
$INTEGER irl;
;COMIN/GEOM/;
IF(irl = 3) [OUTPUT;('Called HOWNEAR in region 3'); RETURN;]
ELSEIF(irl = 2)[ tperp = min(z,  (ZBOUND - z)); ]
ELSEIF(irl = 1)[OUTPUT;('Called HOWNEAR in region 1'); RETURN;]
END;
```

Needless to say, the above User Code listing is somewhat difficult to read, and therefore confusing, in spite of the fact that it is complete. The following is a heavily commented version of the same code, where the structure and readability of the Mortran3 language clearly demonstrates itself.

Figure 21: Commented version of tutor1.mortran

```
%L
%E
" $Id  tutor1.mortran,v 1.2 2003/11/03 18:51:53 course Exp $ "
!INDENT M 4;    "indent each mortran nesting level by 4"
!INDENT F 2;    "indent each fortran nesting level by 2"

"******************************************************************"
"                                                                  "
"                     *****************                            "
"                     *               *                            "
"                     * tutor1.mortran *                           "
"                     *               *                            "
"                     *****************                            "
"                                                                  "
"  An EGSnrc user code. It lists the particles escaping from the back "
"  of a 1 mm Ta plate when a pencil beam of  20 MeV electrons       "
"  is incident on it normally.                                     "
"                                                                  "
"  For PIRS-701:  A simple example which 'scores' by listing particles"
"      D.W.O.R.   JAN 1985 updated to EGSnrc Jan 2000              "
"                                                                  "
"  The following units are used: unit 6 for (terminal) output      "
"                                unit 12 is PEGS cross-section  file  "
"                                                                  "
"******************************************************************"
;Copyright NRC
$IMPLICIT-NONE;                        "default is IMPLICIT NONE"
$INTEGER I,IQIN,IRIN;                  "$INTEGER defaults to INTEGER*4"
$REAL XIN,YIN,ZIN,EIN,WTIN,UIN,VIN,WIN;  "$REAL defaults to REAL*4"
"------------------------------------------------------------------"
"STEP 1:  USER-OVERRIDE-OF-EGSnrc-MACROS                           "
"------------------------------------------------------------------"
REPLACE {$MXMED} WITH {1}   "only 1 medium in the problem(default 10)"
REPLACE {$MXREG} WITH {3}   "only 3 geometric regions (default 2000)"
REPLACE {$MXSTACK} WITH {15}"less than 15 particles on stack at once"

"define a common to pass information to the geometry routine HOWFAR"
REPLACE {;COMIN/GEOM/;} WITH {;COMMON/GEOM/ZBOUND;$REAL ZBOUND;}
REPLACE {$CALL-HOWNEAR(#);} WITH {
  ;CALL HOWNEAR({P1},X(NP),Y(NP),Z(NP),IRL);
}
```

cont...

Figure 21: tutor1.mortran - continued

```
;COMIN/BOUNDS,GEOM,MEDIA,MISC,THRESH/;"Note  ; before COMIN"
"       The above expands into a series of COMMON statements"
"       BOUNDS contains ECUT and PCUT"
"       GEOM passes info to our HOWFAR routine"
"       MEDIA contains the array MEDIA"
"       MISC contains MED"
"       THRESH contains AE and AP"
"---------------------------------------------------------------------"
"STEP 2 PRE-HATCH-CALL-INITIALIZATION                                 "
"---------------------------------------------------------------------"
CHARACTER*4 MEDARR(24);
DATA  MEDARR /$S'TA',22*' '/; "place medium name in an array"
"                            $S is a MORTRAN macro to expand strings"

" Step 0:  Initialize the EGSnrc system. "
call egs_init;

DO I=1,24[MEDIA(I,1)=MEDARR(I);]"this is to avoid a DATA STATEMENT for"
"                                    a variable in COMMON"
"NMED and DUNIT default to 1, i.e. one medium and we work in cm"

/MED(1),MED(3)/=0;MED(2)=1;"vacuum in regions 1 and 3, Ta in region 2"

ECUT(2)=1.5;"   terminate electron histories at 1.5 MeV in the plate"
PCUT(2)=0.1;"   terminate   photon histories at 0.1 MeV in the plate"
"               only needed for region 2 since no transport elsewhere"
"               ECUT is total energy = 0.989   MeV kinetic energy"

"---------------------------------------------------------------------"
"STEP 3   HATCH-CALL                                                  "
"---------------------------------------------------------------------"


;OUTPUT;('\f  Start tutor1'//' CALL HATCH to get cross-section data'/);
CALL HATCH;"    pick up cross section data for TA"
"               data file must be assigned to unit 12"

;OUTPUT AE(1)-0.511, AP(1);
(/' knock-on electrons can be created and any electron followed down to'
  /T40,F8.3,' MeV kinetic energy'/
' brem photons can be created and any photon followed down to      ',
  /T40,F8.3,' MeV ');
"Compton events can create electrons and photons below these cutoffs"
                                                                cont...
```

Figure 21: tutor1.mortran - continued

```
"---------------------------------------------------------------------"
"STEP 4  INITIALIZATION-FOR-HOWFAR and HOWNEAR                        "
"---------------------------------------------------------------------"
ZBOUND=0.1;"     plate is 1 mm thick"

"---------------------------------------------------------------------"
"STEP 5  INITIALIZATION-FOR-AUSGAB                                    "
"---------------------------------------------------------------------"
"Print header for output - which is all AUSGAB does in this case"
;OUTPUT;(/T19,'Kinetic Energy(MeV)',T40,'charge',T48,
'angle w.r.t. Z axis-degrees');

"---------------------------------------------------------------------"
"STEP 6   DETERMINATION-OF-INICIDENT-PARTICLE-PARAMETERS              "
"---------------------------------------------------------------------"
"Define initial variables for 20 MeV beam of electrons incident"
"perpendicular to the slab"

IQIN=-1;"               incident charge - electrons"
EIN=20.511;"            20 MeV kinetic energy"
/XIN,YIN,ZIN/=0.0;"     incident at origin"
/UIN,VIN/=0.0;WIN=1.0;" moving along Z axis"
IRIN=2;"                starts in region 2, could be 1"
WTIN=1.0;"              weight = 1 since no variance reduction used"
"---------------------------------------------------------------------"
"STEP 7   SHOWER-CALL                                                 "
"---------------------------------------------------------------------"
"initiate the shower 10 times"
DO I=1,10[
   OUTPUT I;(' Start history',I4);
   CALL SHOWER(IQIN,EIN,XIN,YIN,ZIN,UIN,VIN,WIN,IRIN,WTIN);
]
"-----------------------------------------------------------------"
"STEP 8   OUTPUT-OF-RESULTS                                       "
"-----------------------------------------------------------------"
"note output is at the end of each history in subroutine ausgab"

"-----------------------------------------------------------------"
"STEP 9   finish run                                              "
"-----------------------------------------------------------------"
call egs_finish;

STOP;END;
                                                     cont...
```

Figure 21: tutor1.mortran - continued

```
%E  "tutor1.mortran"
"******************************************************************"
"                                                                  "
SUBROUTINE AUSGAB(IARG);
"                                                                  "
"  In general, AUSGAB is a routine which is called under a series    "
"  of well defined conditions specified by the value of IARG (see the"
"  EGSnrc manual for the list). This is a particularily simple AUSGAB"
"  Whenever this routine is called with IARG=3 , a particle has      "
"  been discarded by the user in HOWFAR                              "
"  We get AUSGAB to print the required information at that point     "
"                                                                  "
"******************************************************************"
$IMPLICIT-NONE;
$INTEGER IARG;
$REAL EKINE, ANGLE;
COMIN/STACK/;

IF(IARG = 3)[
ANGLE=ACOS(W(NP))*180./3.14159;"angle w.r.t. Z axis in degrees"

IF(IQ(NP) = 0)[EKINE=E(NP);]
ELSE          [EKINE=E(NP)-0.511;]"get kinetic energy"
OUTPUT EKINE,IQ(NP),ANGLE;(T21,F10.3,T33,I10,T49,F10.1);]
RETURN;END;"END OF AUSGAB"

%E  "tutor1.mortran"
"*******************************************************************"
SUBROUTINE HOWFAR;
"                                                                   "
" The following is a general specification of HOWFAR                "
"   Given a particle at (X,Y,Z) in region IR and going in direction  "
"   (U,V,W), this routine answers the question, can the particle go  "
"   a distance USTEP without crossing a boundary?                    "
"           If yes, it merely returns                                "
"           If no, it sets USTEP=distance to boundary in the current "
"           direction and sets IRNEW to the region number   on the   "
"           far side of the boundary (this can be messy in general!) "
"                                                                   "
"   The user can terminate a history by setting IDISC>0. Here we     "
"   terminate all histories which enter region 3 or are going        "
"   backwards in region 1                                            "
```

cont...

Figure 21: tutor1.mortran - continued

```
"                   |               |                       "
"   REGION 1        |   REGION 2    |       REGION 3        "
"                   |               |                       "
"   e- =========>   |               | e- or photon ====>    "
"                   |               |                       "
"   vacuum          |     Ta        |       vacuum          "
"                                                           "
"***********************************************************"
$IMPLICIT-NONE;
$REAL TVAL;
COMIN/STACK,EPCONT,GEOM/;
"       COMMON STACK contains X,Y,Z,U,V,W,IR and NP(stack pointer)"
"       COMMON EPCONT contains IRNEW, USTEP and IDISC"
"       COMMON GEOM contains ZBOUND"
IF(IR(NP) = 3) [
   IDISC = 1;RETURN;"terminate this history: it is past the plate"
]
ELSEIF(IR(NP) = 2)["We are in the Ta plate - check the geometry"
   IF(W(NP) > 0.0)[
      "going forward - consider first since  most frequent"
      TVAL= (ZBOUND - Z(NP))/W(NP);"TVAL is dist to boundary"
      "                          in this direction"
      IF(TVAL > USTEP)[RETURN;"can take currently requested step"]
      ELSE[USTEP = TVAL; IRNEW=3; RETURN;]
   ]"END OF W(NP)>0 CASE"

   ELSEIF(W(NP) < 0.0)["going back towards origin"
      TVAL = -Z(NP)/W(NP); "distance to plane at origin"
      IF(TVAL > USTEP)[RETURN;"can take currently requested step"]
      ELSE[USTEP = TVAL; IRNEW = 1; RETURN;]
   ]"END W(NP)<0 CASE"
   ELSEIF(W(NP) = 0.0)["cannot hit boundary"RETURN;]
]"end of region 2 case"

ELSEIF(IR(NP) = 1)["in region with source"
   IF(W(NP) >  0.0)["this must be a source particle on z=0 boundary"
      USTEP = 0.0;IRNEW = 2;RETURN;
   ]
   ELSE[ "it must be a reflected particle-discard it"
      IDISC=1;RETURN;
   ]
]"end region 1 case"
END;"END OF SUBROUTINE HOWFAR"
                                                     cont...
```

Figure 21: tutor1.mortran - continued

```
%E  "tutor1.mortran"
"*******************************************************************"
"                                                                   "
SUBROUTINE HOWNEAR(tperp, x, y, z, irl);
"                                                                   "
" The following is a general specification of HOWNEAR               "
"   Given a particle at (x,y,z) in region irl, HOWNEAR answers the  "
"   question, What is the distance tperp to the closest boundary?   "
"                                                                   "
"  In general this can be a complex subroutine.                     "
"                                                                   "
"*******************************************************************"
$IMPLICIT-NONE;
$REAL tperp, x, y, z;
$INTEGER irl;

;COMIN/GEOM/;      "       COMMON GEOM contains ZBOUND"

IF(irl = 3) [OUTPUT;('Called HOWNEAR in region 3'); RETURN;]

ELSEIF(irl = 2)["We are in the Ta plate - check the geometry"
    tperp = min(z,  (ZBOUND - z));
]

ELSEIF(irl = 1)[OUTPUT;('Called HOWNEAR in region 1'); RETURN;]

END;"end of subroutine HOWNEAR"

"===========================end of tutor1.mortran================="
```

This User Code produces the following output on unit 6 (the terminal by default).

Figure 22: Output of `tutor1.mortran` using the EGSnrcMP environment[14] on Linux.

```
33_irs45_group1>tutor1 -p tutor_data
================================================================================
EGSnrc version 4 for i686-pc-linux-gnu         Mon Nov 10 11:06:22 2003
================================================================================
configuration............................................g77
user code................................................tutor1
pegs file................................................tutor_data on
HEN_HOUSE
using host...............................................irs45
output file(s)...........................................test
================================================================================
  Start tutor1

 CALL HATCH to get cross-section data
 RAYLEIGH DATA AVAILABLE FOR MEDIUM  1 BUT OPTION NOT REQUESTED.
  Reading screened Rutherford MS data ...............  done

Reading spin data base from /usr/people/course/HEN_HOUSE/data/spinms.data
EGSnrc spin data, version 2.0
Data generated on a machine with 1234 endianess
The endianess of this CPU is 1234
Ranges:      1.00   100.00  0.30054  1.00000
  medium    1 ....................  done
  Medium  1 sige =   1.7946409  1.78708172

  Initializing tmxs for estepe =   0.25 and ximax =   0.5
 Output from subroutine EDGSET:
 ==============================
  Atomic relaxations requested!
 Reading photo-absorption data ..... Done
 Reading relaxation data ....  Done
 Reading photo cross section data ....  Done
 Bound Compton scattering requested, reading data ...... Done
 Initializing Bound Compton scattering ......
 Medium   1 has  21 shells:
  1 861  1  0.02740 0.141E+01    67.413
                 |
                 |
 20 880  7  0.05479 0.944E+02     0.008
 21 881  7  0.01370 0.221E+03     0.008
...... Done.
                                                     cont...
```

Figure 22: continued

```
 EGSnrc SUCCESSFULLY 'HATCHED' FOR ONE MEDIUM.
 knock-on electrons can be created and any electron followed down to
                                         0.189 MeV kinetic energy
 bremsstrahlung photons can be created and any photon followed down to
                                         0.010 MeV
                  Kinetic Energy(MeV)  charge  angle w.r.t. Z axis-degrees
 Start history   1
                         0.962           0           6.1
                        17.185          -1          24.0
 Start history   2
                         0.268           0          21.9
                         0.159           0          26.7
                         0.935           0          24.1
                         5.167           0          21.7
                        11.702          -1          26.0
 Start history   3
                         1.110           0           7.9
                        17.168          -1          18.3
 Start history   4
                         4.332           0           9.9
                        13.522          -1          42.6
 Start history   5
                         0.335           0          18.6
                        17.751          -1          17.7
 Start history   6
                         3.536           0           6.0
                         2.302           0           6.0
                        12.417          -1           9.1
 Start history   7
                         0.229           0           6.9
                         0.179           0          14.3
                        17.369          -1          12.0
 Start history   8
                         0.577           0          28.3
                        17.505          -1          37.2
 Start history   9
                         0.266           0          11.7
                        17.087          -1          20.0
 Start history  10
                        14.479          -1          46.8
========================================================================
Finished simulation
  Elapsed time:                  0.3 s (  0.000 h)
  CPU time:                      0.1 s (  0.000 h)
  Ratio:                   2.493
End of run                                     Mon Nov 10 11:06:22 2003
========================================================================
```

By keeping track of many of these histories, we could answer a lot of questions about what comes out the far side of the plate, but it should be recognised that these are all bookkeeping extensions to the problem—the physics itself is already accomplished with EGSnrc and the relatively small amount of User Code listed above. The scoring routine for this problem is the simplest possible; namely, it outputs on the terminal some of the parameters of the various particles leaving the plate.

In addition, this User Code includes examples of the following items that are discussed further in the EGSnrc User Manual (section 3).

- Defining simple macro replacements for templates (e.g. the character string `$MXMED` is replaced by 1 everywhere in the EGSnrc system).
- The use of `COMIN` statements (which is an EGSnrc macro to allow easy insertion of `COMMONS`).
- The use of `$REAL, $IMPLICIT-NONE, $INTEGER`.
- The technique required in order to define the array `MEDIA`
- The use of `OUTPUT` statements (which are an easy way to output things to Fortran Unit 6).
- The definition of calling parameters for the SHOWER routine.
- A very simple AUSGAB routine.
- The replacement of `$CALL-HOWNEAR(#)` with the recommended subroutine call.
- Simple HOWFAR and HOWNEAR subroutines.
- Relational and logical operators such as `>= (.GE.), & (.AND.), ~= (.NE.)` and `| (.OR.)`. The only warning is not to mix modes since this will generate errors (e.g., don't use `IF(A=B.AND.C=D))`.

4.2 tutor2.mortran: energy transmitted, reflected, deposited

In this example we use the same geometry as above, but we want to score the fraction of the incident energy that is reflected from, transmitted through, and deposited in the plate using the default parameter settings. The coding is essentially the same as in `tutor1.mortran` except that `COMMON/SCORE/` and a new array `ESCORE` are defined at Step 1. The latter is initialised to zero (Step 2) and subsequently printed out on the line printer (Step 8). The AUSGAB routine is considerably different as shown below.

AUSGAB is still very simple since all we need to do is to keep track of the energy deposited in the three regions. The variable `EDEP` (available through `COMMON/EPCONT/`) contains the energy deposited during a particular step for a variety of different `IARG`-situations, as described in the comments in fig 23 and further elaborated upon in section 3.7(page 138). In this example, but not always, we can sum `EDEP` for any value of `IARG` up to 4. Figure 24 shows the output from `tutor2.mortran`.

Figure 23: AUSGAB subroutine from tutor2.mortran

```
%E  "tutor2.mortran"
"******************************************************************"
"                                                                  "
SUBROUTINE AUSGAB(IARG);
"                                                                  "
" In this AUSGAB routine for tutor2, we score the energy deposited  "
"  in the various regions. This amounts to the total energy         "
"  reflected,deposited and transmitted by the slab.                 "
"                                                                  "
"  For IARG=0, an electron or photon step is about to occur and we  "
"  score the energy deposited, if any. Note that only electrons     "
"  deposit energy during a step, and due to our geometry, electrons "
"  only take steps in region 2 - however there is no need to check. "
"  For IARG=1,2 and 4, particles have been discarded for falling    "
"  below various energy cutoffs and all their energy is deposited   "
"  locally (in fact EDEP = particles kinetic energy).               "
"  For IARG=3, we are discarding the particle since it is in        "
"  region 1 or 3, so score its energy.                              "
"                                                                  "
"******************************************************************"
IMPLICIT NONE;
INTEGER IARG,IRL;

COMIN/EPCONT,SCORE,STACK/; "we use EDEP from EPCONT,IR(NP) from STACK"
"                         ESCORE is passed in user defined COMIN SCORE"
IF(IARG <= 4) [
   IRL = IR(NP);" pick up current region number"
   ESCORE(IRL)=ESCORE(IRL)+EDEP;
]
RETURN;END;"end of AUSGAB"
```

Figure 24: Output from `tutor2.mortran` excluding some input data related outputs that are the same as in `tutor1`.

```
79_irs45_group1>ex tutor2 "" tutor_data
==========================================================================
EGSnrc version 4 for i686-pc-linux-gnu          Tue Nov 11 11:46:59 2003
==========================================================================
configuration...........................................g77
user code...............................................tutor2
pegs file...............................................tutor_data on
HEN_HOUSE
using host..............................................irs45
output file(s)..........................................test
==========================================================================
 Start tutor2
 CALL HATCH to get cross-section data
                   |
                   |
 20 880  7  0.05479 0.944E+02     0.008
 21 881  7  0.01370 0.221E+03     0.008
...... Done.
 EGSnrc SUCCESSFULLY 'HATCHED' FOR ONE MEDIUM.
 knock-on electrons can be created and any electron followed down to
                                          0.189 MeV kinetic energy
 bremsstrahlung photons can be created and any photon followed down to
                                          0.010 MeV

 For   10000 incident particles
 Fraction of energy reflected from plate=            1.230%
 Fraction of energy deposited in plate=             12.956%
 Fraction of energy transmitted through plate=      85.814%
                                                -----------
 Total fraction of energy accounted for=           100.000%

========================================================================
Finished simulation
  Elapsed time:            1.0 s (  0.000 h)
  CPU time:                0.9 s (  0.000 h)
  Ratio:                 1.011
End of run                                    Tue Nov 11 11:46:59 2003
========================================================================
```

4.3 tutor3.mortran: NaI response function

The geometry in this example is similar to the previous two but the problem is very different. Here we investigate the energy response function for a 2.54 cm thick slab of NaI when a 5 MeV beam of photons is incident on it. In this case the final scoring and binning is done at the end of each history (*i.e.*, after all the descendants from each initial photon have been tracked completely). Figure 25 shows the changes required (at Steps 7 and 8) and the new AUSGAB routine. Figure 26 shows the output from this code. For a detailed discussion of the use of EGS to calculate response functions see Rogers (1984)[114]. The user code DOSRZnrc calculates response functions in any cylindrical geometry[112].

Figure 25: Portions of `tutor3.mortran`

```
"----------------------------------------------------------------------"
"STEP 7   SHOWER-CALL                                                  "
"----------------------------------------------------------------------"
NCASE=5000;  "INITIATE THE SHOWER NCASE TIMES"

DO I=1,NCASE [
   EHIST = 0.0; "zero energy deposited in this history"
   CALL SHOWER(IQIN,EIN,XIN,YIN,ZIN,UIN,VIN,WIN,IRIN,WTIN);
   "increment bin corresponding to  energy deposited in this history "
   IBIN= MIN(IFIX(EHIST/BWIDTH + 0.999), $EBIN);
   IF(IBIN ~= 0) [EBIN(IBIN)=EBIN(IBIN) + 1;]
]
"----------------------------------------------------------------------"
"STEP 8   OUTPUT-OF-RESULTS                                            "
"----------------------------------------------------------------------"
"Pick up maximum bin for normalization                                 "
BINMAX=0.0; DO J = 1,$EBIN [BINMAX = MAX(BINMAX,EBIN(J));]
OUTPUT EIN,ZBOUND;
('0Response function'/' For a',F8.2,' MeV pencil beam of',
' photons on a',F7.2,' cm thick slab of NaI'/
T6,'Energy  Counts/incident photon');

DO I=1,48 [LINE(I) = ' ';]  "blank entire output array"
DO I=1,$EBIN [
   ICOL=IFIX(EBIN(I)/BINMAX*48.0+0.999);
   IF(ICOL = 0) ICOL=1;
   LINE(ICOL)='*';  "load output array at desired location"
   OUTPUT BWIDTH*I,EBIN(I)/FLOAT(NCASE),LINE;
   (F10.2,F10.4,48A1);  LINE(ICOL)=' ';"reblank"
]
STOP;END;"end of tutor3 main routine"
                                                  cont...
```

Figure 25: Portions of `tutor3.mortran` - continued

```
%E  "tutor3.mortran"
"*********************************************************************"
"                                                                     "
SUBROUTINE AUSGAB(IARG);
"                                                                     "
" In this AUSGAB routine for TUTOR3, we score the energy deposited    "
" in the detector region, region 2                                    "
"                                                                     "
"  For IARG=0, an electron or photon step is about to occur and we    "
"  score the energy deposited, if any. Note that only electrons       "
"  deposit energy during a step, and due to our geometry, electrons   "
"  only take steps in region 2 - however there is no need to check    "
"  this here                                                          "
"  For IARG=1,2 and 4,particles have been discarded for falling below "
"  various energy cutoffs and all their energy is deposited locally   "
"  (in fact EDEP = particles kinetic energy). This only happens in    "
"  region 2.  For IARG=3, we are discarding the particle since it is  "
"   in region 1 or 3, so we do not score its energy                   "
"                                                                     "
"  EHIST keeps track of the total energy deposited during each        "
"  history. In the main routine it is zeroed at the start of each     "
"  history and binned at the end of each history.                     "
"                                                                     "
"*********************************************************************"

COMIN/EPCONT,SCORE/; "we use EDEP from EPCONT and EHIST from SCORE    "

IF(IARG <= 2 | IARG = 4) [EHIST=EHIST + EDEP;]

RETURN;END;"end of ausgab"
```

Figure 26: Portions of output from `tutor3.mortran`.

```
Start tutor3
           .
           .
           .
           .
knock-on electrons can be created and any electron followed down to
                                         0.189 MeV kinetic energy
brem photons can be created and any photon followed down to
                                         0.010 MeV
Response function
For a    5.00 MeV pencil beam of photons on a   2.54 cm thick slab of NaI
    Energy  Counts/incident photon
     0.20    0.0054       *
     0.40    0.0052      *
     0.60    0.0038     *
     0.80    0.0072         *
     1.00    0.0056       *
     1.20    0.0040     *
     1.40    0.0056       *
     1.60    0.0054       *
     1.80    0.0058       *
     2.00    0.0056       *
     2.20    0.0052      *
     2.40    0.0052      *
     2.60    0.0060       *
     2.80    0.0086           *
     3.00    0.0058       *
     3.20    0.0090           *
     3.40    0.0108              *
     3.60    0.0098            *
     3.80    0.0110              *
     4.00    0.0256                                 *
     4.20    0.0196                         *
     4.40    0.0142                  *
     4.60    0.0364                                               *
     4.80    0.0232                              *
     5.00    0.0358                                               *
```

4.4 tutor4.mortran: use of SUBROUTINE WATCH

The `tutor4.mortran` user code is functionally identical to `tutor2.mortran`, *i.e.* it scores the total amount of energy reflected, deposited and transmitted when a 20 MeV beam of electrons is incident on a 1 mm slab of Ta. However it has the added ability to use the auxiliary `SUBROUTINE WATCH` which is part of the file `$HEN_HOUSE/nrcaux.mortran`. As set up, `tutor4.mortran` outputs to the terminal, detailed information about each particle interaction which occurs during the simulation.

Implementing the use of `WATCH` consists of two mandatory calls and one optional call, which has been included in `tutor4.mortran`. These extra calls are shown in fig 27. Figure 28 shows the header of `SUBROUTINE WATCH` which is part of `nrcaux.mortran`. This listing describes how to use `SUBROUTINE WATCH`.

Figure 29 shows portions of the output from `tutor4`. Note that the full output requires 132 columns and this figure has been edited slightly so the font size is still more or less visible! It is well worth studying this output carefully and being sure that you understand what is happening in the shower.

In `tutor4` the value of `IWATCH` has been set to 1 and hence the output lists every time an interaction occurs or a particle is discarded for some reason. If one changes the value of `IWATCH` to 2 the output includes information on every single electron step and hence explodes dramatically in quantity. Finally, if one has installed the code `EGS_Windows` for doing a 3-D graphics display of EGS simulations, then `IWATCH = 4` will write a files (to unit 13) which is the input for this code. This site also presents a series of examples of the use of `EGS_Windows`. It should be noted that starting with Version 4 of the `EGS_Windows` system, it works on any X-windows based Unix/Linux system[113]. Figure 30 presents an example of the environment file needed with `tutor4` if it is to write a data file for display by the `EGS_Windows` system. In this case the file is named `tutor4.egsgph`.

All users are strongly advised to build `SUBROUTINE WATCH` into any user code that they write since it has proven absolutely invaluable for debugging programs. When things don't work, being able to track a few histories inevitably helps to sort out the problem.

Although not apparent in this example, WATCH also displays when bremsstrahlung photons are split, when relaxation particles are created (fluorescent photons, Auger electrons, Coster-Kronig electrons) or when particles are discarded by internal Russian Roulette.

One major warning: when using WATCH you must only use it for a few histories at a time due to the large quantity of output!

Figure 27: Portions of `tutor4.mortran` showing the calls to `SUBROUTINE WATCH`.

```
"---------------------------------------------------------------------"
"STEP 5  INITIALIZATION-FOR-AUSGAB                                    "
"---------------------------------------------------------------------"
DO I=1,3[ESCORE(I)=0.0;]"zero scoring array before starting"

IWATCH=1;     "This determines the type and amount of output"
              "=1 => print info about each interaction"
              "=2 => print info about same + each electron step"
              "=4 => create a file to be displayed by EGS_Windows"
              " Note that these files can be huge"
              "IWATCH 1 and 2 outputs to unit 6, 4 to unit 13"

CALL WATCH(-99,IWATCH);  "Initializes calls to AUSGAB for WATCH"
                            |
                            |
"---------------------------------------------------------------------"
"STEP 7   SHOWER-CALL                                                 "
"---------------------------------------------------------------------"
NCASE=10;  "INITIATE THE SHOWER NCASE TIMES"

DO I=1,NCASE [
    IF((IWATCH ~= 0) & (IWATCH ~= 4))[
       OUTPUT 1,EI,IQIN,IRIN,XIN,YIN,ZIN,UIN,VIN,WIN,LATCHI,WTIN;
              (/' INITIAL SHOWER VALUES',T36,':',
              I5,F9.3,2I4,3F8.3,3F7.3,I10,1PE10.3);
    ]
   CALL SHOWER(IQIN,EIN,XIN,YIN,ZIN,UIN,VIN,WIN,IRIN,WTIN);
   CALL WATCH(-1,IWATCH);  "print a message that this history is over"
]
                            |
                            |
SUBROUTINE AUSGAB(IARG);
IMPLICIT NONE;
INTEGER IARG,IRL;
COMIN/EPCONT,SCORE,STACK/; "we use EDEP from EPCONT,IR(NP) from STACK"
"                         ESCORE is passed in user defined COMIN SCORE"

IF (IWATCH > 0 ) CALL WATCH(IARG,IWATCH); "handles printouts of data"
                                          "IWATCH is passed in SCORE"
IF(IARG <= 4) [
   IRL = IR(NP);" pick up current region number"
   ESCORE(IRL)=ESCORE(IRL)+EDEP;
]
RETURN;END;"end of ausgab"
```

Figure 28: Header of SUBROUTINE WATCH.

```
%E   "start of nrcaux.mortran (SID 1.29 last edited 00/04/24)"
"================================================================================"
"                                                                              "
"                         WATCH                                                "
"                                                                              "
SUBROUTINE WATCH(IARG,IWATCH);
"                                                                              "
"      A general purpose auxiliary routine for use with the EGSnrc system      "
"                                                                              "
"      It prints out information about the particle transport                  "
"                                                                              "
"        For IWATCH = 1 it prints information about each discrete interaction  "
"        For IWATCH = 2 or 3 it prints information about each step as well     "
"        For IWATCH = 4 it prints graphing data for use with EGS_Windows       "
"                                                                              "
"     Routine is used via two mandatory and 1 optional call from the user's    "
"           code                                                               "
"                                                                              "
"    1)The routine must be initialized by a call with IARG=-99 before the first"
"           call to SHOWER. It should be after all inputs are in place.        "
"    2)The routine must be called near the beginning of the AUSGAB subroutine  "
"           IF (IWATCH > 0 ) CALL WATCH(IARG,IWATCH);                          "
"    3)The routine may be called at the end of each history with IARG = - 1 so "
"           a message will get printed stated history is complete              "
"                                                                              "
"     Since WATCH cannot output values related to the initial values in a      "
"     shower call, it is useful to also put something like the following       "
"     immediately prior to the CALL SHOWER stmt                                "
"            IF((IWATCH ~= 0) & (IWATCH ~= 4))[                                "
"               OUTPUT 1,EIN,IQI,IRI,XI,YI,ZI,UI,VI,WI,LATCHI,WTI;             "
"                (/' INITIAL SHOWER VALUES',T36,':',                           "
"                I5,F9.3,2I4,3F8.3,3F7.3,I10,1PE10.3);                         "
"            ]                                                                 "
"    Note EIN is the kinetic energy of the incident particle.                  "
"                                                                              "
"    The routine uses up to 132 columns for output.                            "
"                                                                              "
"******************************************************************************"
```

Figure 29: Portions of output from `tutor4.mortran` (slightly edited for space).

```
Start tutor4
CALL HATCH to get cross-section data
RAYLEIGH DATA AVAILABLE FOR MEDIUM  1 BUT OPTION NOT REQUESTED.
 Reading screened Rutherford MS data ..............  done
 Initializing spin data for medium    1 ....................  done
 Medium  1 sige =   1.79464114  1.78708148
 Initializing tmxs for estepe =   0.25 and ximax =   0.5
Output from subroutine EDGSET:
==============================
 Atomic relaxations requested!
Reading photo-absorption data ..... Done
Reading relaxation data ....  Done
Reading photo cross section data ....  Done
Bound Compton scattering requested, reading data ...... Done
Initializing Bound Compton scattering ......
Medium   1 has  21 shells:
 1 861  1  0.02740 0.141E+01    67.413
 2 862  2  0.02740 0.503E+01    11.680
                  |
21 881  7  0.01370 0.221E+03     0.008
...... Done.
EGSnrc SUCCESSFULLY 'HATCHED' FOR ONE MEDIUM.
knock-on electrons can be created and any electron followed down to
                                         0.189 MeV kinetic energy
brem photons can be created and any photon followed down to
                                         0.010 MeV

                              NP ENERGY  Q REGION X      Y      Z     U      V      W  LATCH  WEIGHT

INITIAL SHOWER VALUES         :1 20.000  -1 2   0.000  0.000  0.000  0.000  0.000  1.000  0 1.00E+00
bremsstrahlung  about to occur:1 19.656  -1 2   0.001  0.002  0.020 -0.046  0.089  0.995  0 1.00E+00
        Resulting electron    :1 18.695  -1 2   0.001  0.002  0.020 -0.046  0.089  0.995  0 1.00E+00
        Resulting photon      :2  0.962   0 2   0.001  0.002  0.020 -0.052  0.092  0.994  0 1.00E+00
Discard -user request         :2  0.962   0 3  -0.003  0.009  0.100 -0.052  0.092  0.994  0 1.00E+00
        Now on top of stack   :1 18.695  -1 2   0.001  0.002  0.020 -0.046  0.089  0.995  0 1.00E+00
bremsstrahlung  about to occur:1 18.196  -1 2   0.003  0.007  0.049  0.329  0.264  0.907  0 1.00E+00
        Resulting electron    :1 18.120  -1 2   0.003  0.007  0.049  0.329  0.264  0.907  0 1.00E+00
        Resulting photon      :2  0.076   0 2   0.003  0.007  0.049  0.330  0.268  0.905  0 1.00E+00
Discard  AE,AP<E<ECUT         :2  0.076   0 2   0.003  0.007  0.049  0.330  0.268  0.905  0 1.00E+00
        Now on top of stack   :1 18.120  -1 2   0.003  0.007  0.049  0.329  0.264  0.907  0 1.00E+00
Discard -user request         :1 17.185  -1 3   0.009  0.026  0.100 -0.073  0.401  0.913  0 1.00E+00
END OF HISTORY       1   ***************************************


INITIAL SHOWER VALUES         :1 20.000  -1 2   0.000  0.000  0.000  0.000  0.000  1.000  0 1.00E+00
bremsstrahlung  about to occur:1 19.016  -1 2  -0.007 -0.001  0.058 -0.158 -0.188  0.969  0 1.00E+00
        Resulting electron    :1 18.747  -1 2  -0.007 -0.001  0.058 -0.158 -0.188  0.969  0 1.00E+00
        Resulting photon      :2  0.268   0 2  -0.007 -0.001  0.058 -0.012 -0.347  0.938  0 1.00E+00
Discard -user request         :2  0.268   0 3  -0.007 -0.017  0.100 -0.012 -0.347  0.938  0 1.00E+00
        Now on top of stack   :1 18.747  -1 2  -0.007 -0.001  0.058 -0.158 -0.188  0.969  0 1.00E+00
bremsstrahlung  about to occur:1 18.577  -1 2  -0.009 -0.004  0.067 -0.268 -0.271  0.925  0 1.00E+00
        Resulting electron    :1 18.418  -1 2  -0.009 -0.004  0.067 -0.268 -0.271  0.925  0 1.00E+00
        Resulting photon      :2  0.159   0 2  -0.009 -0.004  0.067 -0.293 -0.298  0.909  0 1.00E+00
Discard -user request         :2  0.159   0 3  -0.020 -0.015  0.100 -0.293 -0.298  0.909  0 1.00E+00
        Now on top of stack   :1 18.418  -1 2  -0.009 -0.004  0.067 -0.268 -0.271  0.925  0 1.00E+00
bremsstrahlung  about to occur:1 18.256  -1 2  -0.011 -0.007  0.076 -0.177 -0.310  0.934  0 1.00E+00
        Resulting electron    :1 17.321  -1 2  -0.011 -0.007  0.076 -0.177 -0.310  0.934  0 1.00E+00
        Resulting photon      :2  0.935   0 2  -0.011 -0.007  0.076 -0.175 -0.335  0.926  0 1.00E+00
Discard -user request         :2  0.935   0 3  -0.016 -0.016  0.100 -0.175 -0.335  0.926  0 1.00E+00
        Now on top of stack   :1 17.321  -1 2  -0.011 -0.007  0.076 -0.177 -0.310  0.934  0 1.00E+00
bremsstrahlung  about to occur:1 17.231  -1 2  -0.012 -0.009  0.081 -0.278 -0.252  0.927  0 1.00E+00
        Resulting electron    :1 12.065  -1 2  -0.012 -0.009  0.081 -0.278 -0.252  0.927  0 1.00E+00
        Resulting photon      :2  5.167   0 2  -0.012 -0.009  0.081 -0.261 -0.216  0.941  0 1.00E+00
Discard -user request         :2  5.167   0 3  -0.017 -0.013  0.100 -0.261 -0.216  0.941  0 1.00E+00
        Now on top of stack   :1 12.065  -1 2  -0.012 -0.009  0.081 -0.278 -0.252  0.927  0 1.00E+00
Discard -user request         :1 11.720  -1 3  -0.018 -0.015  0.100 -0.311 -0.280  0.908  0 1.00E+00
END OF HISTORY       2   **********************************
                                                                                           cont...
```

Figure 29: Portions of output from `tutor4.mortran` -continued

```
INITIAL SHOWER VALUES         :1 20.000  -1 2   0.000  0.000  0.000  0.000  0.000  1.000  0 1.00E+00
bremsstrahlung  about to occur:1 19.308  -1 2   0.003  0.001  0.041 -0.039  0.106  0.994  0 1.00E+00
        Resulting electron    :1 19.298  -1 2   0.003  0.001  0.041 -0.039  0.106  0.994  0 1.00E+00
        Resulting photon      :2  0.010   0 2   0.003  0.001  0.041 -0.032  0.112  0.993  0 1.00E+00
Discard  AE,AP<E<ECUT         :2  0.010   0 2   0.003  0.001  0.041 -0.032  0.112  0.993  0 1.00E+00
        Now on top of stack   :1 19.298  -1 2   0.003  0.001  0.041 -0.039  0.106  0.994  0 1.00E+00
bremsstrahlung  about to occur:1 18.709  -1 2   0.008  0.002  0.075  0.090  0.061  0.994  0 1.00E+00
        Resulting electron    :1 17.599  -1 2   0.008  0.002  0.075  0.090  0.061  0.994  0 1.00E+00
        Resulting photon      :2  1.110   0 2   0.008  0.002  0.075  0.137 -0.015  0.990  0 1.00E+00
Discard -user request         :2  1.110   0 3   0.011  0.001  0.100  0.137 -0.015  0.990  0 1.00E+00
        Now on top of stack   :1 17.599  -1 2   0.008  0.002  0.075  0.090  0.061  0.994  0 1.00E+00
Discard -user request         :1 17.168  -1 3   0.010  0.007  0.100  0.026  0.313  0.950  0 1.00E+00
END OF HISTORY       3   ************************************


INITIAL SHOWER VALUES         :1 20.000  -1 2   0.000  0.000  0.000  0.000  0.000  1.000  0 1.00E+00


                              NP ENERGY  Q REGION  X     Y      Z     U      V      W   LATCH WEIGHT

bremsstrahlung  about to occur:1 19.316  -1 2  -0.001  0.001  0.040 -0.101  0.057  0.993  0 1.00E+00
        Resulting electron    :1 10.849  -1 2  -0.001  0.001  0.040 -0.101  0.057  0.993  0 1.00E+00
        Resulting photon      :2  8.467   0 2  -0.001  0.001  0.040 -0.104  0.069  0.992  0 1.00E+00
Discard -user request         :2  8.467   0 3  -0.007  0.005  0.100 -0.104  0.069  0.992  0 1.00E+00
        Now on top of stack   :1 10.849  -1 2  -0.001  0.001  0.040 -0.101  0.057  0.993  0 1.00E+00
Moller   about to occur       :1  9.707  -1 2  -0.034 -0.009  0.098 -0.687 -0.390  0.613  0 1.00E+00
         Resulting electrons  :1  9.027  -1 2  -0.034 -0.009  0.098 -0.741 -0.387  0.548  0 1.00E+00
                              :2  0.680  -1 2  -0.034 -0.009  0.098  0.045 -0.274  0.961  0 1.00E+00
Discard  AE,AP<E<ECUT         :2  0.680  -1 2  -0.034 -0.009  0.098  0.045 -0.274  0.961  0 1.00E+00
        Now on top of stack   :1  9.027  -1 2  -0.034 -0.009  0.098 -0.741 -0.387  0.548  0 1.00E+00
Discard -user request         :1  8.957  -1 3  -0.037 -0.011  0.100 -0.866 -0.269  0.421  0 1.00E+00
END OF HISTORY       4   **************************************


INITIAL SHOWER VALUES         :1 20.000  -1 2   0.000  0.000  0.000  0.000  0.000  1.000  0 1.00E+00
                                                       |
                                                       |
                                                       |
                                                       |


         Resulting electrons  :1  7.569  -1 2   0.026  0.016  0.094  0.117  0.073  0.990  0 1.00E+00
                              :2  0.275  -1 2   0.026  0.016  0.094 -0.726 -0.409  0.553  0 1.00E+00
Discard  AE,AP<E<ECUT         :2  0.275  -1 2   0.026  0.016  0.094 -0.726 -0.409  0.553  0 1.00E+00
        Now on top of stack   :1  7.569  -1 2   0.026  0.016  0.094  0.117  0.073  0.990  0 1.00E+00
Discard -user request         :1  7.464  -1 3   0.027  0.017  0.100  0.118  0.247  0.962  0 1.00E+00
END OF HISTORY      10   ***********************************


Fraction of energy reflected from plate=            0.000%
Fraction of energy deposited in plate=             10.939%
Fraction of energy transmitted through plate=      89.061%
                                                -----------
Total fraction of energy accounted for=           100.000%
```

Figure 30: tutor4.io file needed to create an EGS_Windows data file .egsgph.

```
# $Id tutor4.io,v 1.2 2003/11/11 19:10:52 dave Exp $
#
#  This file determines which files are to be connected to which Fortran
#  I/O unit. Lines starting with # are ignored.
#  The first column is the Fortran I/O unit number, the second a
#  file extension => e.g. unit 1 will be
#  connected to output_file.egslst, etc.
#
#  This .io file is for the NRC user code tutor4.
#  The only additional file is unit 13 for possible output for
#  the EGS_WINDOWS graphics package.
#
#  In the following, unit 13 is connected to test.egsgph  by default.
#  However, by using tutor4 -o file -p tutor_data      to run the code
#  one gets  file.egsgph
#
13 .egsgph
```

4.5 tutor5.mortran: using LATCH and Rayleigh scattering

This tutorial program is an example that includes Rayleigh scattering and which makes use of the variable `LATCH` (contained in `COMMON/STACK/`). `LATCH` can be set for any particle on the "stack" of particles being transported, and it is passed on to all its progeny. This provides a simple procedure for keeping track of the history of a particle. In this case we make use of `LATCH` to keep track of how often photons from an incident 50 keV beam are Compton or Rayleigh scattered while passing through a 0.5 cm slab of water.

The program also demonstrates the use of the `IAUSFL` array of flags (in `COMMON /EPCONT/`). By setting the appropriate flags, the user can cause the EGSnrc system to call the AUSGAB subroutine in any combination of 28 well specified situations (see section 3.7, page 138). By default, EGSnrc calls AUSGAB only for 5 out of the possible 28 situations. Here, by setting `IAUSFL(18)` and `IAUSFL(24)` from 0 (default) to 1 in the main program, we cause EGSnrc to call AUSGAB with `IARG=17` and `IARG=23` (*i.e.*, just before a Compton or a Rayleigh scattering event, respectively). We make use of these calls to set some flags associated with each photon rather than for scoring any variables. The AUSGAB routine also makes use of a few simple local macros in order to make the logic of the coding more transparent. Perhaps the greatest strength of Mortran3 is this ability to create readable and hence more accurate coding. A complete listing of `tutor5.mortran`, except for the HOWFAR/HOWNEAR routines which are similar to the other examples, is given below.

Note that the logic in `tutor5.mortran` for EGSnrc is the same as it was for EGS4. This is only possible because in water there are no fluorescent photons above the 10 keV PCUT value in the problem. If the material were changed to a higher Z material, e.g. lead, one would need to change the logic since the present logic allows fluorescent photons created via relaxation after a Compton interaction to be scored as a Compton scattered photon.

Figure 31: Portions of `tutor5.mortran`

```
%L
%E    "tutor5.mortran (SID 1.1 last edited 00/1/18)"

!INDENT M 4;   "indent MORTRAN listing 4 per nesting level"
!INDENT F 2;   "indent FORTRAN output 2 per nesting level"

"*********************************************************************"
"                                                                     "
"                     *****************                               "
"                     *               *                               "
"                     * tutor5.mortran *                              "
"                     *               *                               "
"                     *****************                               "
"                                                                     "
"                                                                     "
" An EGSnrc user code which scores the number and average energy of   "
" primary, Rayleigh scattered and Compton scattered photons passing   "
" through a  5 mm thick slab of water when a 50 keV pencil beam of    "
" photons is incident normally                                        "
"                                                                     "
" For PIRS-701: Example of including Rayleigh scattering, using macros"
"               to facilitate logic and use of the LATCH feature      "
"      D.W.O.R.   Feb 1985  updated Jan 2000 for EGSnrc               "
"                                                                     "
"*********************************************************************"
$IMPLICIT-NONE;
$INTEGER I,IQIN,IRIN,NCASE;
$REAL XIN,YIN,ZIN,EIN,WTIN,UIN,VIN,WIN,ANORM;

"---------------------------------------------------------------------"
"STEP 1:  USER-OVERRIDE-OF-EGSnrc-MACROS                              "
"---------------------------------------------------------------------"
REPLACE {$MXMED} WITH {1}   "only 1 medium in the problem(default 10)"
REPLACE {$MXREG} WITH {3}   "only 3 geometric regions (default 2000)"
REPLACE {$MXSTACK} WITH {15}"less than 15 particles on stack at once"

"Define a common to pass information to the geometry routine HOWFAR"
REPLACE {;COMIN/GEOM/;} WITH {;COMMON/GEOM/ZBOUND;$REAL ZBOUND;}
"Define a common for scoring in AUSGAB"
REPLACE {;COMIN/SCORE/;} WITH {
   ;COMMON/SCORE/COUNT(3),ENTOT(3); $REAL COUNT,ENTOT;}
"  in COUNT(1),(2),(3) AUSGAB will count the number of transmitted  "
"  primaries, rayleigh scattered or only compton scattered photons  "
"  ENTOT adds up  the total energy of each of these components      "
```

Figure 31: Portions of `tutor5.mortran` - continued.

```
REPLACE {$CALL-HOWNEAR(#);} WITH {
   ;CALL HOWNEAR({P1},X(NP),Y(NP),Z(NP),IRL);}

;COMIN/BOUNDS,EPCONT,GEOM,MEDIA,MISC,SCORE,STACK,THRESH/;
"       the above expands into a series of COMMON statements"

"---------------------------------------------------------------------"
"STEP 2 PRE-HATCH-CALL-INITIALIZATION                                  "
"---------------------------------------------------------------------"
CHARACTER*4 MEDARR(24);
DATA  MEDARR /$S'H2O',21*' '/; "place medium name in an array"

Step0: Initialize the EGSnrc system
call egs_init;

DO I=1,24[MEDIA(I,1)=MEDARR(I);]"this is to avoid a data statement for"
"                                a variable in COMMON"
"NMED and DUNIT default to 1, i.e. one medium and we work in cm"

/MED(1),MED(3)/=0;MED(2)=1; "regions 1 and 3 are vacuum, region 2, H2O"

ECUT(2)=1.5;"     terminate electron histories at 1.5 MeV in the slab"
PCUT(2)=0.010;"   terminate   photon histories at 0.01 MeV in the slab"
IRAYLR(2)=1;"     turn on Rayleigh scattering in the slab "
"NOTE, above three parameters need to be set for all regions in which "
"there is particle transport - just region 2 in this case             "

"---------------------------------------------------------------------"
"STEP 3   HATCH-CALL                                                   "
"---------------------------------------------------------------------"
;OUTPUT;('1  Start tutor5'//' Call HATCH to get cross-section data'/);
CALL HATCH;"    pick up cross section data for H2O"
"               data file must be assigned to unit 12"
;OUTPUT AE(1)-0.511, AP(1);
(/' Knock-on electrons can be created and any electron followed down to'
  /T40,F8.3,' MeV kinetic energy'/
' Brem photons can be created and any photon followed down to      '
  /T40,F8.3,' MeV ');"NOTE, AE values can over-ride ECUT values"

"---------------------------------------------------------------------"
"STEP 4  INITIALIZATION-FOR-HOWFAR and HOWNEAR                         "
"---------------------------------------------------------------------"
ZBOUND=0.5;"     plate is 5 mm thick"
```

Figure 32: Portions of `tutor5.mortran` - continued.

```
"----------------------------------------------------------------------"
"STEP 5  INITIALIZATION-FOR-AUSGAB                                     "
"----------------------------------------------------------------------"
DO I=1,3 [ COUNT(I)=0.0;ENTOT(I)=0.0;]"zero scoring array at start"
"  We want to set flags in AUSGAB every time a Rayleigh scattering   "
"  or compton scattering occurs. Set the flags in IAUSFL(COMIN      "
"  EPCONT) to signal the  EGS system to make the appropriate calls  "
/IAUSFL(18),IAUSFL(24)/=1;
"----------------------------------------------------------------------"
"STEP 6   DETERMINATION-OF-INICIDENT-PARTICLE-PARAMETERS               "
"----------------------------------------------------------------------"
"define initial variables for 20 MeV beam of electrons  incident"
"perpendicular to the slab"
IQIN=0;"                 incident photons"
EIN=0.050;"              50 keV"
/XIN,YIN,ZIN/=0.0;"      incident at origin"
/UIN,VIN/=0.0;WIN=1.0;" moving along z axis"
IRIN=2;"                 starts in region 2, could be 1"
WTIN=1.0;"               weight = 1 since no variance reduction used"
LATCHI=0;"               LATCH set to zero at start of each history"
"----------------------------------------------------------------------"
"STEP 7   SHOWER-CALL                                                  "
"----------------------------------------------------------------------"
NCASE=10000;  "initiate the shower NCASE times"
DO I=1,NCASE [CALL SHOWER(IQIN,EIN,XIN,YIN,ZIN,UIN,VIN,WIN,IRIN,WTIN);]

"----------------------------------------------------------------------"
"STEP 8   OUTPUT-OF-RESULTS                                            "
"----------------------------------------------------------------------"
ANORM = 100./FLOAT(NCASE); "normalize to % of photon number"
DO I=1,3[
   IF(COUNT(I)~=0)[ENTOT(I)=ENTOT(I)/COUNT(I);]"get average energies"
]
OUTPUT EIN*1000.,ZBOUND, PCUT(2), (ANORM*COUNT(I),ENTOT(I),I=1,3);
(//' For',F6.1,' keV photons incident on',F4.1,' cm OF H2O'/
' with PCUT=',F5.3,' MeV'
//' Transmitted primaries=',T40,F8.2,'%  ave energy=',F10.3,' MeV'//
' Fraction rayleigh scattering=',T40,F8.2,'%  ave energy=',F10.3,' MeV'
//' Fraction compton scattering only=',T40,F8.2,'%  ave energy=',F10.3,
' MeV'//);
"------------------------------------------------------------------"
"STEP 9   finish run                                               "
"------------------------------------------------------------------"
call egs_finish;
STOP;END;"end of tutor5 main routine"
```

Figure 32: Portions of `tutor5.mortran` - continued.

```
"******************************************************************"
"                                                                  "
SUBROUTINE AUSGAB(IARG);
"                                                                  "
" In this AUSGAB routine for tutor5 we both set flags whenever there is"
" a scattering event and then count histories when they have come     "
" through the slab , according to what kind of scattering they have   "
" undergone.                                                       "
" The logic is as follows                                          "
" set FLAG1 if a compton event occurs                              "
" set FLAG2 if a Rayleigh event occurs                             "
" The FLAGS are the units and thousands digits in the parameter LATCH  "
"                                                                  "
" When a history is terminated, increment various counters according   "
" to whether no flags are set - i.e. its a primary, FLAG2 is set,      "
" i.e. it has Rayleigh scattered or FLAG1 is set and FLAG2 is not set  "
" i.e. only  compton scattering has occured.                       "
"                                                                  "
" First a few macros are defined to make the logic simpler to read and "
" therefore verify                                                 "
"                                                                  "
"******************************************************************"
$IMPLICIT-NONE;
$INTEGER IARG,JJ;
REPLACE {$SET-FLAG1;} WITH {LATCH(NP)=LATCH(NP)+1;}
REPLACE {$SET-FLAG2;} WITH {LATCH(NP)=LATCH(NP)+1000;}
REPLACE {$FLAG1} WITH {MOD(LATCH(NP),100)}"i.e. units digit of LATCH"
REPLACE {$FLAG2} WITH {MOD(LATCH(NP),10000)-$FLAG1} "thousands digit"

;COMIN/SCORE,STACK/; "we use IR(NP) from STACK"

"   first set flags when scattering events occur - IAUSFL was set "
"   in step 5 of main to ensure AUSGAB was called at these points "

IF(IARG=17)    [ "a compton scatter is about to occur"  $SET-FLAG1;]

ELSEIF(IARG=23)[ "a rayleigh scatter is about to occur" $SET-FLAG2;]
```

Figure 32: Portions of `tutor5.mortran` - continued.

```
" if a history has terminated because leaving the slab, score it"
ELSEIF(IARG = 3 ) [ "particle has left slab"
   IF(IR(NP)=3 | IR(NP) = 1) [
      "it is transmitted or reflected"
      JJ=0;
      IF(LATCH(NP) = 0)  [ "no scattering - a primary"       JJ=1;]
      ELSEIF($FLAG2 ~= 0)[ "at least one rayleigh scatter"   JJ=2;]
      ELSEIF($FLAG1 ~= 0)[">=1 compton scatter, no Rayleigh" JJ=3;]
      ELSE ["debug";OUTPUT JJ,LATCH(NP); (' JJ,LATCH(NP)=',2I10);]

      IF(JJ ~= 0) [
         COUNT(JJ)=COUNT(JJ) + 1.;
         ENTOT(JJ) = ENTOT(JJ) + E(NP);
      ]
   ]"end region 3 or 1 block"
]"end IARG 3 block"
RETURN;END;"end of AUSGAB"
```

Figure 33: Portions of output from `tutor5.mortran`

```
Start tutor5
                     |
                     |
                     |
                     |
EGSnrc SUCCESSFULLY 'HATCHED' FOR ONE MEDIUM.
Knock-on electrons can be created and any electron followed down to
                                         0.189 MeV kinetic energy
Brem photons can be created and any photon followed down to
                                         0.010 MeV
For  50.0 keV photons incident on 0.5 cm OF H2O
with PCUT=0.010 MeV
Transmitted primaries=                   89.35%  ave energy=    0.050 MeV
Fraction rayleigh scattering=             0.84%  ave energy=    0.048 MeV
Fraction compton scattering only=         8.40%  ave energy=    0.046 MeV
```

4.6 tutor6.mortran: modifying the transport options

As seen from the previous tutorial programs, EGSnrc can be run in the default mode without making any selections regarding transport algorithms or about which features to include. In these cases it it uses default algorithms which apply in the vast majority of cases. However, there are some options which are wasteful to model in many simulations, but critical for others. For example, taking into account binding effects in Compton scattering can be important for low-energy calculations but has no effect for most cases above 1 MeV. The same is true for Rayleigh (coherent) scattering of photons. In addition, there have been many changes made to the electron transport algorithm in EGS4/PRESTA and EGSnrc. To our knowledge the default simulations with EGSnrc provide accurate electron transport but to allow us and others to understand the differences between EGSnrc and EGS4/PRESTA, we have left hooks in the EGSnrc code which allow various old transport options to still be used (see section 3.4.2.i,page 131). So, for example, EGSnrc can be run with the PRESTA-I electron transport algorithm turned on instead of the new PRESTA-II transport algorithm. Also one can chose to vary the boundary crossing algorithm used in EGSnrc and mimic that used in EGS4/PRESTA. This can be useful for investigations about the impact of the changes on a given user code. More importantly, it allows some of the time consuming features of the simulation in the default EGSnrc to be "turned off" if it turns out that they do not affect the users problem.

The `tutor6.mortran` code does a simple calculation, very similar to those in the previous tutorial codes, but asks the user for a series of options to use. In fact, this little code gives an example of how to change every EGSnrc transport option that we can think of. The purpose is to show how easy it is. Basically you must assign a value to some flag or variable and ensure that the appropriate `COMIN` is defined in the subroutine or main routine where the inputs are read.

The variables which are set are summarised in Table 9 and figure 34 shows one part of code which inputs two of these variables (please see the full version of the code on the distribution at `$HEN_HOUSE/tutor6/tutor6.mortran`).

`tutor6.mortran` also exhibits two other interesting features. It initialises two variables, viz `IREJECT` and `ESAVE`, which allow EGSnrc's internal range rejection to be utilised by setting the arrays `i_do_rr` and e_max_rr. By setting `IREJECT` to 1, EGSnrc terminates the history of any electron below the energy ESAVE if that particle is unable to reach the closest geometric boundary. See section 3.11.1 (page 152) for more information. `tutor6.mortran` also does a statistical analysis of its results using a new procedure suggested by Francesc Salvat[115] which doesn't use the common "batch technique" but scores the statistics on an event by event basis in an efficient manner. This procedure is shown in figure 35.

Note that if `tutor6.mortran` is run with bremsstrahlung splitting turned on it will be seen that the total energy is not exactly 1.0. This is because of the lack of exact energy conservation as discussed in section 3.11.2 (page 153). As expected, as the number of histories increases, the energy conservation gets closer to unity.

Table 9: The transport control variables set in `tutor6.mortran` and the `COMIN`s they are contained in. These represent all user controllable transport variables. See the source code of `tutor6.mortran` and sections 3.3 and 3.4.2 for more detailed descriptions of the meaning of each.

Variable	`COMIN`
ECUT,PCUT	BOUNDS
IBRDST	BREMPR
IPRDST	BREMPR
ibr_nist	BREMPR
IBCMP	COMPTON-DATA
IEDGFL	EDGE
IPHTER	EDGE
ESTEPE	ET-CONTROL
XIMAX	ET-CONTROL
SKINDEPTH_FOR_BCA	ET-CONTROL
TRANSPORT_ALGORITHM	ET-CONTROL
BCA_ALGORITHM	ET-CONTROL
EXACT_BCA	ET-CONTROL
SPIN_EFFECTS	ET-CONTROL
IRAYLR	MISC
luxury_level,iseed	none

Table 10: The intrinsic variance reduction parameters set in `tutor6.mortran`. They are included in EGSnrc via `COMIN/EGS-VARIANCE-REDUCTION`.

Variable
nbr_split
i_play_RR
prob_RR
i_do_rr
e_max_rr

Figure 34: Examples from `tutor6.mortran` of the reading in of three of the many variables read in, specifically `IEDGFL, IPHTER and IBCMP`.

```
         from subroutine inputs with COMIN COMPTON-DATA and EDGE included

"Relaxations switch, must be done before HATCH so that the necessary"
"additional data can be read in in HATCH if the user requsted relaxations"
LOOP [
  OUTPUT; (/' Atomic relaxations on (1) or off (0)? ',$);
  INPUT iedgfl(1); (I5);
] UNTIL ( iedgfl(1) = 1 | iedgfl(1) = 0 );
"Now set iedgfl for all regions to the value input by the user"
DO i=2,$MXREG [ iedgfl(i) = iedgfl(1); ]

"Photo-electron angular distribution switch. It does not need to be "
"before HATCH, we do it here because this is the most logical place"
LOOP [
  OUTPUT; (/' Photo-electron angular distribution on (1) or off (0)? ',$);
  INPUT iphter(1); (I5);
] UNTIL ( iphter(1) = 1 | iphter(1) = 0 );
"Now set iphter for all regions to the value input by the user"
DO i=2,$MXREG [ iphter(i) = iphter(1); ]

"Bound Compton scattering switch"
"Must be done before HATCH in order to get the data for bound Compton"
LOOP [
  OUTPUT; (/' Binding effects for Compton scattering on (1) or off (0)?
',$);
  INPUT ibcmp(1); (I5);
] UNTIL ( ibcmp(1) = 1 | ibcmp(1) = 0 );
"Now set ibcmp for all regions to the value input by the user"
DO i=2,$MXREG [ ibcmp(i) = ibcmp(1); ]
```

Figure 35: Portions of tutor6.mortran related to statistical analysis without batching.

```
"STEP 8   OUTPUT-OF-RESULTS                                            "
"----------------------------------------------------------------------"
total = 0;
anorm = 1/(ein+float(iqin)*rm); "for e+ add 2*rm to k.e.
DO i=1,nzb+1 [
    "first put non-scored energy portions into sc_array and sc_array2"
    aux = sc_tmp(i); aux2 = aux*aux;
    sc_array(i) = sc_array(i) + aux;
    sc_array2(i) = sc_array2(i) + aux2;
    aux = sc_array(i)/ncase;    aux2 = sc_array2(i)/ncase;
    aux2 = (aux2 - aux*aux)/(ncase-1);
    IF( aux2 > 0 ) aux2 = sqrt(aux2);
    aux = aux*anorm;         aux2 = aux2*anorm;
    sc_array(i) = aux;       sc_array2(i) = aux2;     total = total + aux;
]
OUTPUT sc_array(1),sc_array2(1);
   ('   Reflected energy fraction: ',f10.6,' +/- ',f10.6);
                     etc
"**************************************************************************"
subroutine ausgab(iarg);
" For tutor6.mortran.  Provides an example of doing statistics       "
" without batches using technique suggested by Francesc Salvat.      "
" Copyright National Research Council of Canada 2000                 "
"**************************************************************************"
$IMPLICIT-NONE;            $INTEGER iarg,irl;          real*8   aux;
;COMIN/SCORE,   "to get the scoring arrays and iscore"
       EPCONT,  "to get EDEP"
       STACK/;  "to get the region number"
IF( iarg < 5 ) [ "energy is being deposited"
    irl = ir(np);
    IF( icase = sc_last(irl) )["still the same shower that deposited energy"
                                 "last time in this region"
        sc_tmp(irl) = sc_tmp(irl) + edep*wt(np);
    ]
    ELSE [ "we have the next shower depositing energy into region irl"
           " => put sc_tmp into  the scoring arrays and set sc_last"
        aux = sc_tmp(irl);
        sc_array(irl) = sc_array(irl) + aux;
        sc_array2(irl) = sc_array2(irl) + aux*aux;
        sc_tmp(irl) = edep*wt(np); sc_last(irl) = icase;
    ]
]
return; end;         "end of subroutine ausgab for tutor6.mortran"
```

Figure 36: `5mev_e_1mm_Ta.egsinp`, an input file to run `tutor6`. Note that the text on the right are not a necessary part of the file but are effectively comments.

```
5mev_e_1mm_Ta.egsinp
TA
1.0                    ,ECUT
0.01                   ,PCUT
0                      ,IRAYLR
1                      ,IEDGFL
0                      ,IPHTER
1                      ,IBCMP
1                      ,IPRDST
1                      ,IBRDST
1                      ,ibr_nist
1                      ,ISPIN
0                      ,ESTEPE
0                      ,XIMAX
0                      ,transport_algorithm
0                      ,bca_algorithm
3                      ,skindepth_for_bca
1,1                    ,luxury_level,iseed
0.1                    ,ZBOUND
1                      ,IREJCT
5.0                    ,ESAVE
1                      ,nbr_split
0                      ,i_play_RR Russian Roulette
-1                     ,IQIN
5.0                    ,EIN
0.                     ,ANGLE
1000                   ,NCASE
```

`tutor6.mortran` is designed to be run interactively, prompting for responses from the terminal. This can become tedious and to shorten the process the scripts used to run EGSnrc allow the user to specify an input file which is read as if it were input from the terminal. Figure 36 shows an example of such an input file to be used with `tutor6.mortran`. The only trouble is that the prompts become rather garbled since the code was not written to handle this case. Figure 37 shows the terminal session when `tutor6.mortran` is run interactively using the same inputs as in the sample input file.

Figure 37: Output from tutor6.mortran when run interactively using the same parameters as in the example file.

```
Starting non-debug interactive run with no input file

Input name of medium   1:TA
Input minimum electron transport energy (total, MeV): 1.0
Input minimum photon transport energy (MeV): 0.01
Rayleigh scattering on (1) or off (0)? 0
Atomic relaxations on (1) or off (0)? 1
Photo-electron angular distribution on (1) or off (0)? 0
Binding effects for Compton scattering on (1) or off (0)? 1
Pair angular distribution: the following choices are available:
  0: fixed pair angle (EGS4 default)
  1: leading term of the distribution
  2: Koch and Motz
your choice: 1
Bremsstrahlung angular distribution,
   0: leading term of Koch and Motz distn
   1: Koch and Motz 2BS(modified):
your choice: 1
Bremsstrahlung differential photon cross section to sample,
  0: use Bethe-Heitler distribution as in EGS4
  1: use NIST/ICRU 37 distributions
your choice: 1
Spin effects on (1) or off (0)? 1
Input maximum fractional energy loss per step (estepe): 0.0
using default value:      0.250
Input maximum 1st elastic scattering moment per step: 0.0
using default, ximax =      0.500
Electron-step algorithm: EGSnrc default (0) or PRESTA (1)? 0
Boundary crossing algorithm: exact (0) or PRESTA (1)? 0
Skin-depth for BCA:
  this is the distance from a boundary
  (measured in elastic mean-free-paths) at which the
  simulation switches to single scattering mode
  Best choice for efficiency is 3. If you set this
  parameter to a very large number (e.g. 1e10), you
  can force single scattering simulation in the entire
  geometry (this is very slow)
your choice: 3
Input random number luxury level (0-4) & seed (>0) (0 defaults OK): 1,1
```

continued...

Figure 37: tutor6.mortran output - continued

```
 ***************** RANLUX initialization *****************
 luxury level:  1
 initial seed:           1
***********************************************************

 Input slab thickness: 0.1
 Use (1) or do not use (0) electron range rejection? 1
 Input the maximum energy to apply range rejection: 5.0
 How many bremsstrahlung photons to create per event (0=>just 1): 1
 Russian Roulette all secondary charged particles(yes=1,no=0)?  0
 Input incident charge (-1, 0, +1):  -1
 Input incident kinetic energy (MeV): 5.0
 Input incident angle (degrees): 0.
 Input number of showers to be simulated: 1000

  Start tutor6

 CALL HATCH to get cross-section data
 RAYLEIGH DATA AVAILABLE FOR MEDIUM  1 BUT OPTION NOT REQUESTED.

  Reading screened Rutherford MS data ...............  done
  Initializing spin data for medium    1 .................  done
  Medium  1 sige =   1.79464114  1.78708148
  Initializing tmxs for estepe =   0.25 and ximax =   0.5

 Output from subroutine EDGSET:
 ==============================
  Atomic relaxations requested!
 Reading photo-absorption data ..... Done
 Reading relaxation data ....  Done
 Reading photo cross section data ....  Done

 Bound Compton scattering requested, reading data ...... Done
 Initializing Bound Compton scattering ......
 Medium   1 has  21 shells:
  1 861  1  0.02740 0.141E+01    67.413
  2 862  2  0.02740 0.503E+01    11.680
  3 863  3  0.02740 0.247E+01    11.136
  4 864  4  0.05479 0.271E+01     9.881
  5 865  5  0.02740 0.111E+02     2.705
  6 866  5  0.02740 0.652E+01     2.466

                                                continued...
```

Figure 37: `tutor6.mortran` output - continued

```
  7 867  5  0.05479 0.696E+01     2.191
               |
               |
 20 880  7  0.05479 0.944E+02     0.008
 21 881  7  0.01370 0.221E+03     0.008
...... Done.

  Using NIST brems cross sections!
  Initializing brems data for medium  1...
  Max. new cross sections per energy loss:   1.79542971  1.78786802

 EGS SUCCESSFULLY 'HATCHED' FOR ONE MEDIUM.
 knock-on electrons can be created and any electron followed down to
                                          0.189 MeV kinetic energy
 bremsstrahlung photons can be created and any photon followed down to
                                          0.010 MeV
 Starting shower simulation ...
 Finished    10.0% of cases
              |
 Finished   100.0% of cases

 Finished shower simulation with      1000 cases.

   Reflected energy fraction:   0.139521 +/-   0.007509
   Deposited energy fraction:   0.716428 +/-   0.007973
 Transmitted energy fraction:   0.144051 +/-   0.006239
-------------------------------------------------------------
                       total:   1.000000

=======================================================================
Finished simulation
  Elapsed time:                   0.7 s (  0.000 h)
  CPU time:                       0.7 s (  0.000 h)
  Ratio:                     1.026
End of run                                     Tue Nov 11 14:37:29 2003
=======================================================================
```

4.7 tutor7.mortran: using SUBROUTINE GET_INPUTS

`tutor7.mortran` is like the same as `tutor6.mortran` but it uses `SUBROUTINE GET_INPUTS` to input the necessary inputs. This is a powerful package which allows for a very descriptive input format which makes the creation of input files much simpler. This example is given here to encourage people to learn about `SUBROUTINE GET_INPUTS` which is described in detail in another manual[112].

4.8 Sophisticated User Codes

The EGSnrc distribution includes a variety of NRC written general purpose user codes such as DOSRZnrc, FLURZnrc etc. These are described in a separate manual "NRC User Codes for EGSnrc" [112]. These provide many working examples of how to do more sophisticated operations such as handle cylindrical geometries, properly score fluence, use complex LATCH algorithms, handle statistical analysis etc. The BEAM system is for modelling linear accelerators and doing dose calculations in a CT-based patient phantom[88, 111, 116].

5 Adapting EGS4 User Codes to EGSnrc

5.1 Introduction

Section 1.4 presents a brief summary of the major changes in EGS4 found in EGSnrc. More details are found in section 2 although not necessarily contrasted to EGS4. These changes taken together represent a major change in the code system. Moreover, we strongly advise to use the newer multi-platform EGSnrc version, and to read the related user manual, PIRS-877.

Nonetheless, we have tried to make EGSnrc as much as possible compatible with existing EGS4 user codes. A full compatibility could not be achieved because of the various changes and enhancements in the modelling of the underlying physical processes. In addition, we have decided not to support what we now consider bad coding practices, especially replacement of EGS internal (private) common blocks or subroutine `COMIN`'s. This will increase the portability and insensitivity of user codes to future changes and enhancements of the system, once they are adapted to EGSnrc.

In general, the following main incompatibilities may occur in the process of adaptation of user codes to EGSnrc:

1. Incompatibilities due to redefinitions in the user code of EGS internal common blocks (`COMIN`s).
2. Incompatibilities due to the lack of explicit data typing in any user replacement of macro templates used within EGSnrc since EGSnrc uses IMPLICIT NONE; everywhere.
3. Incompatibilities in the scoring routines due to implicit or explicit assumptions about what particles are in what order on the STACK after a given interaction.
4. The lack of a `HOWNEAR` routine if the user code did not use PRESTA-I, but otherwise HOWNEAR is used in a compatible manner.
5. Incompatibilities (or possibly just lack of efficiency) for user codes which implemented their own versions of range rejection, bremsstrahlung splitting and/or Russian Roulette, especially range rejection.

This section of the manual gives guidelines for the adaptation of existing user codes. The next section briefly summarises changes to EGS4 common blocks and subroutines. Section 5.3 discusses the use of `APPEND` *vs* `REPLACE`. Section 5.4 is devoted to the use of explicit data typing. Section 5.5 deals with the resolution of incompatibilities in the user scoring routine `AUSGAB`. Possible approaches for the coding of `HOWNEAR` and the options available if this task is too complicated are given in Sec. 5.6. Section 5.7 deals with the use of electron range rejection, Sec. 5.8 with the input of transport options. Section 5.9 presents a "cook book" fashion guide for the adaptation procedure. Finally, section 5.10 shows an example for the adaptation of the user code `XYZDOS`.

5.2 Short description of changes

5.2.1 The system

Most of the NRCC changes/additions to EGS4 over the years have been included in the standard EGSnrc files,

`egsnrc.macros`	standard EGSnrc macros and replacements
`egsnrc.mortran`	standard EGSnrc subroutines.

The use of the `nrcc4mac.mortran`, `presta.macros` and `presta.mortran` files is therefore not necessary any more. In addition, `BLOCK DATA`, previously in `egs4blok.mortran`, is included in `egsnrc.mortran`.

5.2.2 EGS4 COMMON blocks

- `COMIN/BREMPR/` was modified to provide data for the new sampling techniques employed in the routines `BREMS` and `PAIR`. In addition, the material composition which is needed for the bremsstrahlung and pair angular selection macros (NRC extensions), is included now by default. `BREMPR` also holds flags that determine the angular selection scheme (`IBRDST` and `IPRDST` and the bremsstrahlung cross section employed, (`ibr_nist`);

- `COMIN/ELECIN/`: Unused variables were removed from the definition of this COMIN. Additional variables that hold information necessary for the modified implementation of the fictitious cross section method, electron range calculations and multiple elastic scattering are included in `COMIN/ELECIN/`;

- `COMIN/EPCONT/`: `BETA2` and `TSCAT` were removed (not needed), `IAUSFL` was expanded to 28 elements in order to define calls to the scoring routine after various relaxation transitions;

- `COMIN/MULTS/` and `COMIN/PATHCM/` which were needed to implement Molière's multiple scattering theory and path-length corrections, are not necessary in the new system and are therefore removed;

- EGSnrc provides two random number generators: `RANMAR`[107, 108] and `RANLUX`[109, 110]. They are included via the configuration file, `ranmar.macros` (macros) and `ranmar.mortran` (initialisation routine) are needed in order to use `RANMAR`, `ranlux.macros` (macros) and `ranlux.mortran` (initialisation and sampling routines) are needed for `RANLUX`. The definition of `COMIN/RANDOMM/` is removed from the main system and included in the `ranmar.macros` or `ranlux.macros` files. An additional consequence is that the random number generator is no longer initialised in `HATCH`, the user must provide the initialisation in their user code (although RANLUX is self-initialising to a default luxury level of 1 and fixed initial random number seed).

- The NRCC additions `LATCH` and `LATCHI` to `COMIN/STACK/` are now included by default. In addition, a variable `NPold`, which points to the location of the particle initiating a given discrete interaction, i.e. the top of the stack before the last interaction, is included in `COMIN/STACK/`.

- `COMIN/THRESH/`: removed `ESCD2` (not needed);
- `COMIN/USEFUL/`: removed `IBLOBE` which is not needed with the new implementation of atomic relaxations;

5.2.3 NRC extensions to EGS4

- `COMMON/EDGE/` used for fluorescent X-ray emission is now a standard EGSnrc common block which is included in `$COMIN-PHOTO` and `$COMIN-BLOCK`. It is completely different from the definition found in `nrcc4mac.mortran` as EGSnrc models K,L and M fluorescence and Auger and Coster-Kronig relaxation transitions.
- The common block `/USERXT/IPHTER($MXREG)` which used to turn on photo-electron angular sampling and was defined to be part of `COMIN/USER-MISC/` in `nrcc4mac.mortran`, is now a part of a standard EGSnrc common block. The variable `IPHTER` is included in `COMMON/EDGE/` and should be discarded from definitions and replacements in user codes;
- Electron transport parameters such as the maximum geometrical step-size (`SMAX` or `SMAXIR($MAXRG)`), the maximum energy loss per step (`ESTEPR($MAXRG)`) etc., have been frequently included in `COMIN/USER/` in EGS4 user codes. Most electron transport parameter are now included in `COMIN/ET-Control/`;

5.2.4 New EGSnrc COMMON blocks of interest

- `COMIN/NIST-BREMS/`: contains information necessary to sample the energy of brem from the NIST cross section data base, which is the basis for ICRU radiative stopping powers;
- `COMIN/COMPTON-DATA/`: contains information necessary for the modelling of binding effects and Doppler broadening in incoherent photon scattering events;
- `COMIN/ET-Control/`: combines all variables that determine electron transport settings, such as maximum fractional energy loss per step, maximum first elastic moment per step, maximum geometrical step-size restriction, boundary crossing algorithm, electron-step algorithm, etc;
- `COMIN/MS-Data/`: contains information necessary for the new multiple scattering theory based on the screened Rutherford cross section;
- `COMIN/Spin-Data/`: data for the additional rejection loop in the multiple scattering routine that is necessary to take into account spin effects;
- `EGS-VARIANCE-REDUCTION`: A few variance reduction techniques that have proved to be very useful for applications we are interested in at the National Research Council have been implemented internally. The `COMIN EGS-VARIANCE-REDUCTION` contains flags and variables necessary for the operation of these variance reduction techniques.

5.2.5 Explicit data typing

In all of the EGSnrc routines there is a macro `$IMPLICIT-NONE;` followed by a macro `$DEFINE-LOCAL-VARIABLES-XXXX;`, `XXXX` denoting the subroutine name. The default replacement is `implicit none` to allow for explicit data typing. The `$DEFINE-LOCAL-VARIABLES-XXXX;` macros, to be found in `egsnrc.macros`, define all of the local variables in the corresponding routines. See section 5.4 below about dealing with these features.

5.2.6 Changes in EGS4 subroutines

Details concerning the following changes are contained in section 2 of this manual.

- `ANNIH`: Now implements radiative splitting internally, the time consuming evaluation of a logarithm was taken out of the main sampling loop.
- `BHABHA`: No changes compared to EGS4 version 3.0 (but note there was a sampling bug present in older EGS4 distributions[56])
- `BLOCK DATA`: Numerous changes to provide initialisations for EGSnrc additions.
- `BREMS`: There is a bug in the EGS4 sampling algorithm which shows up only if the electron kinetic energy is not much larger than the bremsstrahlung production threshold `AP`. The sampling algorithm was completely recoded to remove the bug and increase the efficiency. Radiative splitting is implemented internally, the bremsstrahlung angular selection macro (NRC extension) is modified to allow for the use of the leading term of the angular distribution, the EGS4 fixed angle selection scheme is removed. The new `BREMS` routine can also use the NIST bremsstrahlung cross section data base to sample photon energies if the flag `ibr_nist` is set to 1.
- `COMPT`: Completely recoded to take into account binding effects and Doppler broadening in incoherent scattering events.
- `ELECTR`: Completely recoded to include the improvements in the implementation of the condensed history technique some of which have become known as PRESTA-II.
- `HATCH`: Now calls several additional subroutines which read in data and perform initialisations necessary for the EGSnrc enhancements.
- `MOLLER`: No changes to the EGS4 version 3.0 (but note the Møller bug present in older EGS4 distributions[56]).
- `MSCAT`: Completely recoded to implement the new exact multiple scattering theory.
- `PAIR`: Sampling technique modified to make it more efficient at photon energies below 20-30 MeV. Pair production is now skipped if the electron Russian Roulette flag is set and the pair electrons are selected to be "killed" (for efficiency).

- `PHOTO`: Completely recoded. When relaxation is being modelled, in the case of mixtures, `PHOTO` explicitely samples the atomic number of the element involved in the interaction and explicitly selects the shell. This prevents the local absorption of all sub-K-shell photons typical for EGS4.
- `EDGSET`: Completely recoded, reads in the data necessary for the implementation of K,L,M shell atomic relaxations.
- `PHOTON`: Only a minor change to reflect the fact that, due to the internal implementation of variance reduction techniques, after pair events the top particle is not necessarily an electron or a positron.
- `SHOWER`: No changes
- `UPHI`: Modified to implement a more efficient azimuthal angle sampling algorithm. The use of interpolation tables for the evaluation of sines and cosines is disabled by default. Note, however, that the calculation of a sine and a cosine is necessary only after pair events in EGSnrc. It is planned that the next release of EGSnrc will sample the pair angle cosines directly so that the issue of trigonometric evaluations will become entirely irrelevant.

5.2.7 New EGSnrc subroutines

- `fix_brems`: Re-calculates some quantities in the common block `BREMPR` for use with the new `BREMS` sampling algorithm.
- `init_compton`: Reads in data necessary for the modelling of binding effects and Doppler broadening in incoherent scattering events.
- `mscati`: Performs various initialisations such as the calculation of maximum step-size, the calculation of the maximum electron/positron cross section per unit energy loss (necessary for the modified implementation of the fictitious cross section method) and calls the multiple scattering initialisation routines.
- `spin_rejection`: A function that returns the value of the rejection function due to spin effects for the current step-length, material, energy, and scattering angle.
- `sscat`: Samples an elastic scattering angle from the single elastic scattering cross section.
- `init_ms_SR`: Reads-in and initialises data necessary to sample multiple scattering angles from the MS theory based on the screened Rutherford cross section.
- `init_spin`: Reads in and initialises data necessary to sample multiple scattering angles when spin effects are taken into account.
- `set_spline`: An auxiliary subroutine which sets cubic spline interpolation coefficients.
- `spline`: An auxiliary function which performs a cubic spline interpolation. The spline coefficients must have been set with `set_spline`.

- `msdist_pII`: Performs a condensed history step using the improved electron-step algorithm (*i.e.* samples a multiple scattering angle and the final position, given a pathlength and energy loss).
- `msdist_pI`: Performs a condensed history step à la PRESTA.
- `RELAX`: Performs the de-excitation cascade of K,L,M and N shell vacancies.
- `init_nist_brems`: Reads-in and initialises data necessary for the sampling of brems energies from the NIST cross section data base.
- `prepare_alias_table`: Prepares an alias sampling table using linear interpolation between data given in the arrays `xs_array` (abscissas) and `ys_array` (ordinates) which both have the dimension `(0:nsbin)`.
- `alias_sample1`: Employs a modified version of the alias sampling technique to sample a random quantity from an alias table previously initialised with `prepare_alias_table`. Used for sampling bremsstrahlung energies from the NIST cross section data base.
- `gauss_legendre`: Calculate abscissas and weights for a Gauss-Legendre quadrature.
- `vmc_electron`: Implements the VMC electron transport algorithm (much faster but not so general). Not distributed with the system for commercial reasons regarding the clinical implementation of Monte Carlo treatment planning.

5.2.8 Changes since 2005 printing

In addition to the changes from EGS4 to EGSnrc outlined above, there have been further changes to the EGSnrc system since the 2005 printing of this manual. They are summarized in this section.

5.2.8.i Changes to existing COMMON blocks

- `COMIN/COMPTON-DATA`: Added an alias table, `eno_atbin_array`. Allows the `COMPT` subroutine to pick the interacting shell instead of looking it up sequentially each time. This makes simulating Compton interactions more efficient. Also added the flag `radc_flag` for turning on radiative Compton corrections.
- `COMIN/ELECIN`: Added logical variable `sig_ismonotone` which is set to `.true.` if the electron cross section is an increasing function of energy.
- `COMIN/CH_steps`: Added a boolian variable `is_ch_step` which is set to `.true.` if the current electron step is a condensed history (*i.e.* PRESTA-II) step. Useful for some functions in user codes and for counting in general.
- `COMIN/EPCONT`: Added real variables `x_final`, `y_final`, `z_final`, `u_final`, `v_final`, `w_final`, which store the final position and direction of an electron at the end of its step. Of use in codes that do sub-voxel dose scoring.

- `COMIN/MEDIA`: Added character variables, `eii_xfile`, `photon_xsections` and `comp_xsections` which store the names (or prefixes) of the data files for electron impact ionization, photon cross sections and Compton cross sections respectively. Also added real variables, `apx`, `upx` which store photon cross section thresholds.

5.2.8.ii Changes to existing subroutines

- `BREMS`: Added option for `ibr_nist`=2: includes effect of electron-electron bremsstrahlung.
- `COMPT`: Changes to allow user to simulate radiative Compton corrections. Also, some speed improvements.
- `ELECTR`: In single-scattering mode, the sampling of the electron MFP has been changed to eliminate errors when `lambda` > `lambda_max`. Also now calculates energy loss to `AE` instead of 0 in case step is longer than range to `AE`.
- `init_compton`: Modifications for more efficient Compton sampling.
- `msdist_PI` and `msdist_PII`: Fixed a bug which occurred when particles were traveling exactly backwards (`w(np)`=-1), the transport distance and direction was reversed.
- `init_nist_brems`: Modifications to allow reading in of bremsstrahlung cross section data that takes into account electron-electron bremsstrahlung (`ibr_nist`=2).

5.2.8.iii New COMMON blocks of interest

- `COMIN/EII-DATA/`: Contains cross section data necessary for modeling electron impact ionization. Also, contains the variable `eii_flag` which is used to turn electron impact ionization on or off.
- `COMIN/RAYLEIGH_INPUTS/`: Contains arrays of medium names and file names for custom molecular form factor data when modeling Rayleigh scattering.
- `COMIN/RAYLEIGH_SAMPLING/`: Arrays storing data required to simulate Rayleigh events.
- `COMIN/EGS-IO/`: Contains file names, unit numbers for I/O and data files used during an EGSnrc system. Also contains variables used for parallel runs. User codes commonly access these variables.
- `COMIN/RAD_COMPTON/`: Contains cross-section arrays necessary for the modeling of radiative Compton corrections (*i.e.* if `radc_flag`=1) . Note that this common block is defined in a separate file, `$HEN_HOUSE/src/rad_compton1.mortran` which must be included in the `SOURCES` list (defined in `Makefile` or `user_code.make`) of the user code if radiative Compton corrections are to be used.
- `$COMIN-INIT-NIST-BREMS`: a macro to replace list of `COMINS` at the top of subroutine `init_nist_brems`. Allows access to the variable `ibr_nist` through C++ codes.

5.2.8.iv New subroutines

- `ANNIH_AT_REST`: Used to be handled in the `ELECTR` subroutine. Putting it on its own allows positron annihilation at rest to be initiated from within the `AUSGAB` of a user code.

- `egs_init_user_photon`: Subroutine which opens the photon cross section data files, including user-supplied Compton cross sections, if applicable, and fills cross section arrays for each medium. Previously this was done in `HATCH`. Now this subroutine is called by `HATCH`. Implemented at the same time as the feature allowing the user to select different photon cross sections.

- `egsi_get_data`: Subroutine called by `egs_init_user_photon` which actually reads the photon cross section data from the applicable files.

- `radc_init` and `sample_double_compton`: Subroutines contained in the file `$HEN_HOUSE/src/rad_compton1.mortran` used to apply radiative corrections to Compton events. `radc_init` is called by the subroutine `init_compton` at the beginning of a run and reads in the radiative Compton data from `$HEN_HOUSE/data/rad_compton1.data`. `sample_double_compton` is called by the subroutine `COMPT` to simulate double Compton events if radiative corrections are turned on (`radc_flag`=1).

- `egs_scale_photon_xsection`: Subroutine to allow scaling of photon cross sections by a given factor for a given medium. Not sure if this is currently used.

- `eii_init`: Subroutine to open data file used to calculate cross sections for electron impact ionization. Called by subroutine `HATCH`.

- `egs_init_rayleigh`, `egs_rayleigh_sigma`, `egs_rayleigh_sampling` and `prepare_rayleigh_data`: Subroutines/functions for the new Rayleigh implementation. `egs_init_rayleigh` is called from `egs_init_user_photon` and is used to calculate Rayleigh cross sections from either user-supplied form factor data or from the default atomic form factors. `egs_rayleigh_sigma` is a real function, called by `egs_init_rayleigh` used to calculate the Rayleigh cross section in barns. `egs_rayleigh_sampling` is called by subroutine `PHOTON` to simulate the Rayleigh interaction. `prepare_rayleigh_data` is a subroutine called by `egs_init_rayleigh` to put the Rayleigh data in a form suitable for run-time sampling.

5.2.8.v `$egs_info`, `$egs_fatal` and `$egs_warning`

In addition, the current release of EGSnrc replaces outputs to STDOUT previously coded with `write(6,*)` statements with calls to new `$egs_info` (for information or echoing of input), `$egs_fatal` (for fatal error messages) and `$egs_warning` (for non-fatal warning messages). This has made it easier to assign a unit number other than 6 to the `.egslog` file, which is necessary in the case of, say, a BEAMnrc shared library source.

5.3 `REPLACE` *vs* `APPEND`

It seems to have been a common practise that in user codes, standard EGS4 comin's and common's are replaced by user definitions. A typical example is the inclusion of `COMIN/GEOM/` in `$COMIN-PHOTON` in order to implement photon interaction forcing (this is accomplished by the appropriate replacement of `$SELECT-PHOTON-MFP`). The replacement

```
REPLACE {$COMIN-PHOTON;} WITH
{;COMIN/DEBUG,BOUNDS,GEOM,MEDIA,MISC,EPCONT,PHOTIN,STACK,THRESH,
    UPHIOT,USEFUL,USER,RANDOM/;}
```

(*e.g.* in `CAVRZ`) has the obvious disadvantage that if the default definition of `$COMIN-PHOTON` is changed in a new EGS release (*e.g.* in order to include new common blocks necessary to model photo-nuclear reactions), the user code will not work properly with the updated system (because the new common blocks are missing from `$COMIN-PHOTON`). If we compare the above replacement with the default definition of `$COMIN-PHOTON` in `egs4mac.mortran`, we see that the only difference is the inclusion of `COMIN/GEOM/`! This task can be accomplished by the use of `APPEND`,

```
APPEND {;COMIN/GEOM/;} TO {$COMIN-PHOTON;} .
```

This technique has the advantage of making the user code independent of changes/updates in the EGS system. We therefore strongly recommend the use of `APPEND` instead of `REPLACE`, not only for the adaptation process but also for the writing of new user codes. In fact, the misuse of the mortran `REPLACE` capability was one of the main source for coding efforts necessary to adapt NRC user codes to the EGSnrc system.

Note that when you `APPEND` something to another macro, the other macro MUST BE DEFINED FIRST. In the above example, the `APPEND` statement must occur after the definition of `$COMIN-PHOTON`.

5.4 Use of explicit data typing

The FORTRAN "advantage" of pre-defined data types for variables frequently leads to programming errors that are sometimes difficult to find (such as typos). The EGS4 system is well benchmarked and therefore one can be quite confident that there are no such bugs in the EGS4 routines. However, the redefinition of standard EGS macros such as `$SELECT-PHOTON-MFP, $SELECT-ELECTRON-MFP, $USER-RANGE-DISCARD,$CALL-HOWNEAR, $PARTICLE-SELECTION-XXXX` (`XXXX` stands for a subroutine name) introduces the possibility that EGS internal variables are used and eventually changed in these macros. The easiest way to avoid conflict of variables is to always use separate routines to perform the tasks of the above macros, *e.g.*

```
REPLACE {$SELECT-PHOTON-MFP;} WITH
{dpmfp = select_photon_mfp(gle,x(np),y(np),z(np),u(np),v(np),w(np),ir(np));
}
```

(this will also make unnecessary the inclusion of `COMIN/GEOM/` into `$COMIN-PHOTON`). In terms of efficiency, use of routine calls instead of macros is not crucial as modern compilers will inline the routine if it is simple enough (and if the routine is not inlined, then the overhead due to a routine call is most likely to be negligible compared to the time necessary to execute the routine).

Another way of preventing conflict of variables is to make use of the explicit data typing introduced into the EGSnrc system (see section 5.2.5). In order to do so, all macro variables must be defined in the user code using the `APPEND` statement, *e.g.*

```
APPEND {;
$REAL real_var1,real_var2,...;
$INTEGER int_var1,int_var2,...; } TO
{$DEFINE-LOCAL-VARIABLES-XXXX;}
```

Here `$REAL` and `$INTEGER` are macros defined in `egsnrc.macros` with the default replacement `real*4` and `integer*4` and `XXXX` stands for the name of the subroutine where the macro is placed (if double precision and logical variables are used as well, they of course have to be defined too). If it happens that the macro uses EGSnrc internal variables that are already defined in `$DEFINE-LOCAL-VARIABLES-XXXX` or one of the common blocks, a good FORTRAN compiler will complain about multiple definition of variables[15]

Note also that the use of explicit data typing together with the `$REAL` macro allows for the easy switch to double precision, if double precision turns out to be important for your application (*e.g.* high energy applications are good candidates for use of double precision). If you want your application to produce the same history sequence on different machines, double precision is a good idea, too).

We strongly recommend the use of explicit data typing in EGSnrc user codes. However, if you are confident that your macro replacements and main code do not contain collisions with EGSnrc internal variables and you are too lazy to declare all variables used, the simplest approach is to include the statement

```
REPLACE {$IMPLICIT-NONE;} WITH {;}
```

at the beginning of your user code.

5.5 The scoring routine

It is impossible to predict all possible applications of a coupled electron photon Monte Carlo simulation package and therefore it is impossible to give guidelines for the adaptation of all user codes available. We believe, however, that the only problems that may arise in `AUSGAB` are problems related to assumptions about the outcome of a particular interaction made in the coding of the scoring routine.

As an example, consider a call to `AUSGAB` after Compton events in order to flag scattered photons (so that, for instance, dose due to scattered photons can be scored separately). A typical implementation is the following piece of code in `AUSGAB`:

[15] Note, however, that the warning/error messages can not always be immediately attributed to the multiple definition of a variable. The GNU g77 compiler, for instance, takes out a variable, declared as a member of a common block and then re-declared, out of the common block, and then prints warnings that the common block has different lengths at different places of the program!

```
IF(IARG = 18) [ "a Compton event just occurred"
  IF(IQ(NP) = -1) [ LATCH(NP-1) = 1; "photon is at NP-1"]
  ELSE [ LATCH(NP) = 1; "photon is at NP" ]
]
```

If binding effects are turned on in EGSnrc, the above may lead to unpredictable results because the outcome of a Compton event may be any one of the following:

1. The original photon, if the interaction was rejected due to binding effects
2. A scattered photon and a Compton electron, if the interaction took place and no vacancy with a binding energy above the specified transport threshold energies (ECUT and PCUT) was created
3. A scattered photon, a Compton electron and one or more relaxation particles.

If, for instance, the interaction was rejected and the photon was the only particle on the stack (NP=1), the above code will override a portion of the memory leading to an unpredictable outcome of the simulation. Problems of that type can be resolved by including a loop over all particles from NPold (the stack pointer before the last interaction) to NP (the current stack pointer). In the current EGSnrc version problems may arise only after Compton scattering and photo-electric absorption. However, to guarantee compatibility of your user code to future EGSnrc releases, it is a good idea to *always* use a loop over all particles created and not assume a certain outcome of interactions. The implementation of electron impact ionization, which will affect the routines MOLLER and BHABHA, is scheduled for the next EGSnrc release, if we one day decide to explicitely simulate triplet production and large energy transfer electron-electron bremsstrahlung, the outcome of PAIR and BREMS will also become unpredictable.

The above piece of code is adapted to EGSnrc using

```
IF( iarg = 18 ) [ "A Compton event has occurred"
  IF( NP > NPold ) [ "i.e.  the interaction really took place"
    DO ip = NPold, NP [ IF( iq(ip) = 0 ) latch(ip) = 1; ] ] ]
```

Note that this coding defines fluorescent photons as scattered photons and only works if Russian Roulette is not being played (since if it were, all electrons might be gone and NP = NPold even when the interaction took place). To explicitly flag only the Compton scattered photons one could use:

```
IF( iarg = 18 ) [ "A Compton event has occurred"
  IF( NP > NPold ) [ "i.e.  the interaction really took place"
    IF( iq(NPold) = 0 ) latch(NPold) = 1; ] ]
```

which makes use of the fact that the scattered photon is always at NPold but also makes the assumption that Russian Roulette is not being played on the charged particles (see section 3.7.2, page 139 for more details – the variable i_survived_RR allows these cases to be distinguished).

5.6 HOWNEAR

In order to be able to use the improved condensed history implementation, EGSnrc requires the knowledge of the perpendicular distance to the closest boundary at the beginning of each electron step, as did PRESTA-I. The user is asked for this distance via the macro

```
$CALL-HOWNEAR(#);
```

The default replacement of `$CALL-HOWNEAR(#)` in `egsnrc.macros` prints a message to the standard output that a user definition of this macro is required and aborts the execution of the code. If your user code is not using the PRESTA algorithm, the recommended replacement is

```
REPLACE {$CALL-HOWNEAR(#);} WITH {
  call hownear({P1},x(np),y(np),z(np),ir(np)); }
```

together with the code for the subroutine `hownear`. If your user code is using the PRESTA algorithm, there must be a definition of `$CALL-HOWNEAR` somewhere in your user code. We recommend modifying the definition to a subroutine call, but this is not a necessary requirement. If you decide not to modify an existing `$CALL-HOWNEAR` definition, you need to append declarations of all variables that this macro uses to `$DEFINE-LOCAL-VARIABLES-ELECTR`, or replace `IMPLICIT-NONE` with null at the beginning of your user code (see section 5.4).

If your geometry is too complex and you are not able to code a `hownear` routine (even some second order surfaces require the solution of a sixth order equation in order to determine the minimum perpendicular distance), you should include the following replacement in your user code:

```
REPLACE {$CALL-HOWNEAR(#);} WITH { {P1} = 0; }
```

EGSnrc will then always assume that the particle is very close to a boundary. The accuracy and speed of the simulation will crucially depend on the setting of the `bca_algorithm` variable which is in `COMIN/ET-Control/`:

- `bca_algorithm = 0`: The entire simulation will be performed in single elastic scattering mode. This is very accurate but extremely slow and probably not feasible for most applications.

- `bca_algorithm = 1`: The entire simulation will be performed without lateral deflections in the individual electron steps taken into account. This is similar to the original EGS4 condensed history implementation although the path-length correction is more accurate. We recommend to set the parameter `ximax`, which is also in `COMIN/ET-Control/`, to not more than 0.05 for accurate results. Reducing `ximax` is better than reducing `ESTEPE` as it automatically takes into account differences in the strength of elastic scattering (and so, you don't need to use `ESTEPE` of 3-4% for low-Z materials but `ESTEPE` only 0.3% for high-Z materials).

5.7 Use of electron range rejection

Many user codes employ some form of electron range rejection technique (which terminates electrons if their range is less than the perpendicular distance to the closest boundary, or to a region of interest). To implement this technique, the `$USER-RANGE-DISCARD` macro is replaced with some code that estimates the CSDA range of electrons and then compares it with the minimum perpendicular distance. In EGSnrc the range is used (and therefore calculated) internally to calculate energy loss to sub-threshold processes. It is ABSOLUTELY ESSENTIAL for the correct operation of EGSnrc that the variable `range`, which holds the current electron CSDA range, is NOT MODIFIED by a user estimate of the range in the macro `$USER-RANGE-DISCARD`! You should check your user code for a replacement of the macro `$USER-RANGE-DISCARD` and rename the variable used for the electron range if it is called `range`! (such a collision with the EGSnrc internal `range` variable would be automatically detected with the use of implicit data typing, see section 5.4). If you fail to do so, the most likely result will be an abort of execution a short time after the begin of the simulation and a message in the output file from the `mscat` routine which says that the maximum step-size in `mscat` was exceeded.

Note that EGSnrc already implements electron range rejection on a region-by-region basis internally. Therefore, a user range rejection is not necessary, unless it is more sophisticated and rejects electrons that can not reach certain volume elements even if capable of escaping the current region.

5.8 Input of electron transport parameters and cross section options

If you don't make any changes to the electron transport and cross section settings, EGSnrc will run in its most accurate mode possible. If you want to modify some of the transport parameter settings (default values are set in the `block data`), you may

1. Hard-code the values you want to use for the simulations. In order to do so, you need to include the appropriate `COMIN`'s into the routine that sets the desired values, *i.e.*

   ```
   subroutine my_routine;
   ;COMIN/COMPTON-DATA/;
   ...
   DO j=1,$MXREG [ ibcmp(j) = 0; ]
   return; end;
   ```

   will turn off bound Compton scattering. The tutorial code, `tutor6.mortran` contains examples of setting all the parameters we could think of for a simple calculation (section 4.6, page 188).

2. Read desired transport settings from an input file. The easiest way to do so is to include the statement

   ```
   call get_transport_parameter(ounit);
   ```

in your user code anywhere before the call to `HATCH`. Here, `ounit` is an integer variable that is the unit number of the file to which the `get_transport_parameter` routine will print a summary of the transport parameter settings (if `ounit` is less or equal to zero, no output will be produced). If you decide to use `get_transport_parameter`, you need to include the files `transportp.macros` and `get_inputs.mortran` via the configuration file. The syntax used by `get_transport_parameter` is described extensively in the user-code manual[112] and in the file `get_inputs.mortran`, the tutorial code `tutor7.mortran` gives an example of the usage of `get_transport_parameter`. The advantage of this approach is twofold:

i. Future changes of the system will be readily available to your user code without changes.

ii. You don't need to specify *all* transport parameters in the input file, just the ones that you want to change. Missing transport parameter input will cause the system to use default values for the corresponding variables.

5.9 Adapting user codes: cook book instructions

After the discussions of the previous sections, we are ready to give the steps that must be followed to adapt existing EGS4 user codes to EGSnrc in a "cook book fashion". If your user code is basically a minor modification of an NRC user code such as DOSRZ, it may be more efficient to adapt the corresponding EGSnrc user code. Note that this section assumes that you have the EGSnrc system running so, e.g. you have already redefined `$HEN_HOUSE` to point to the EGSnrc system and you have sourced `Cshrc_additions_for_egsnrc` which means you will pick up the new scripts using the old commands (so, e.g. the command `mf` now picks up `egs_compile` instead of `egs4_compile`) and the naming convention for files is slightly different (e.g., `.egsinp` instead of `.egs4inp`, etc). After ensuring the system is ready (see section 9, page 294), do the following steps with your EGS4 user code.

We also strongly advise to use the newer multi-platform EGSnrc version, and to read the related user manual, PIRS-877.

Step 1: Check whether your code defines the `$CALL-HOWNEAR` macro. If it is defined, decide whether it is simple enough to change the definition to a subroutine call. If yes, do so (see section 5.6). If not, append `COMIN/GEOM/` (or whatever COMIN you need for your geometry routines) to `$COMIN-ELECTR` (see section 5.3). If `$CALL-HOWNEAR` is not defined in your user code, try to code a subroutine that calculates the minimum perpendicular distance to the closest boundary for your geometry and replace `$CALL-HOWNEAR` with a call to this subroutine. If your geometry is too complex, make `$CALL-HOWNEAR` return a `tperp` value of zero (see section 5.6). For a complete discussion of HOWNEAR routines, see section 3.6, page 137).

Step 2: If your user code redefines the `$RANDOMSET` macro and/or the `COMIN/RANDOM`, consider whether you want to leave it redefined or use the RANLUX or RANMAR generators. Adjust any I/O related to the RNG's seeds. If the user code uses the default RNG, make any adjustments to the fact that the default is now RANLUX. Check whether your code initialises the random number generator. If not, provide initialisation by putting *e.g.* the macro `$RNG-INITIALIZATION` somewhere in your user code before the user code uses a random number or at least before the first call to `SHOWER`. Note that you don't need an initialisation if you use the `RANLUX` generator (which is the default), but it is also not wrong to use it. Note that if you redefine `COMIN/RANDOM`, it must include integer variable `rng_seed`. See sections 3.4.1.iv (page 122) and 3.9 (page 143) for further details.

Step 3: Check your user code for replacements of `COMIN/XXXX/` where `XXXX` stands for an EGS4 or EGSnrc COMIN block name other than `USER, USER-MISC`, etc. If you find such replacements, compare them with the corresponding `COMIN` definition in `egsnrc.macros`. If the definitions match, remove the replacement from your user code. If not, put user variables into a new common block that will later become a part of `COMIN/USER/` (see step 6) and remove this `COMIN` definition from your user code. A typical example of a user re-defined `COMIN` is `COMIN/BREMPR/` with the inclusion of parameters that control bremsstrahlung splitting. Such variables should be in a user common block and passed to the EGSnrc routines via `COMIN/USER/`(see section 3).

Step 4: Check your user code for replacements of `$COMIN-XXXX`, `XXXX` being a EGS4/EGSnrc subroutine name. If you find such replacements, compare them to the corresponding definition in `egsnrc.macros`. If they match, remove the replacement statement from your user code. If not, change the replacement statement to `APPEND` to include necessary comin's with `$COMIN-XXXX` (see section 5.3). Don't forget that some EGS4 comin's have been eliminated (see section 5.2).

Step 5: If you are using the NRC extensions from `nrcc4mac.mortran` (check your configuration file), `COMIN/USER/` is replaced there by

```
COMIN/USER-STEP-CONTROLS,USER-VARIANCE-REDUCTION,USER-MISC/
```

In the present release, most of the NRC extensions are included in `egsnrc.macros` and so the use of `nrcc4mac.mortran` is not necessary. However, we found that it is better to return to the original EGS4 definition of `COMIN/USER/` namely a null (;) and let the user define their own `COMIN/USER/`. Check therefore your user code for definitions of `USER-STEP-CONTROLS, USER-VARIANCE-REDUCTION` and `USER-MISC`. Combine all of these definitions to a `COMIN/USER/` definition. Include also common blocks or variables needed to account for previous replacements of EGS4 / EGSnrc internal comin's (see step 4 above).

Step 6: Check whether one or more of the common blocks included in `COMIN/USER/` are not already defined in `COMIN/EDGE/` (*e.g.* `COMMON/USERXT/`) or in `COMIN/ET-Control/` (e.g. `SMAXIR`) and discard them if so. Check also for appearances of `ESTEP, ESTEPE`, etc. If you intend to use the `get_transport_parameter` routine (see step 13 and section 5.8), remove all of them. If not, make sure that `COMIN/ET-Control/` is available.

Step 7: Check for definitions of `$SET-BREMS-ANGLE, $SET-PAIR-ANGLE`, `$RAYLEIGH-CORRECTION` and `$SELECT-PHOTOELECTRON-DIRECTION` in your user code. Discard them unless you are using angle selection schemes for bremsstrahlung, pair production or photo-electron production or a treatment of Rayleigh scattering which differ from what are now the internal EGSnrc sampling routines.

Step 8: Check for a definition of `$USER-RANGE-DISCARD`. If you find one, make sure that it does not use the variable `range`. If your `$USER-RANGE-DISCARD` macro performs only electron range rejection on a region-by-region basis, you may consider using the internal EGSnrc range rejection capability and deleting the definition of `$USER-RANGE-DISCARD` from your user code.

Step 9: Decide whether you want to use implicit data types. If no (*e.g.* in order to save work), insert in your user code the statement `REPLACE {$IMPLICIT-NONE;} WITH {;}` If yes, perform the following steps:

- Include the `$IMPLICIT-NONE;` macro at the beginning of the main routine, `HOWFAR, HOWNEAR, AUSGAB` and all other user-written routines.
- Declare all local variables used in these routines.
- Declare all variables in `COMIN/USER/`;

- Check for definitions of the following in your user code. If one or more of these macros is redefined, declare all variables used by these macros using `APPEND` (see section 5.3) with the appropriate `$DEFINE-LOCAL-VARIABLES-XXXX;` (see section 5.4).

`$SELECT-ELECTRON-MFP`	used in `ELECTR`
`$USER-RANGE-DISCARD`	
`$DE-FLUCTUATION`	
`$PARTICLE-SELECTION-MOLLER`	
`$PARTICLE-SELECTION-BHABHA`	
`$PARTICLE-SELECTION-ANNIH`	
`$PARTICLE-SELECTION-BREMS`	
`$SELECT-PHOTON-MFP`	used in `PHOTON`
`$RAYLEIGH-CORRECTION`	
`$RAYLEIGH-SCATTERING`	
`$PARTICLE-SELECTION-PAIR`	
`$PARTICLE-SELECTION-COMPT`	
`$PARTICLE-SELECTION-PHOTO`	
`$SET-BREMS-ANGLE`	used in `BREMS`
`$SET-PAIR-ANGLE`	used in `PAIR`
`$SELECT-PHOTOELECTRON-DIRECTION`	used in `PHOTO`

Step 10: Try to compile your user code. If you get error messages check carefully steps 1-10 to confirm that you have performed all changes described above.

Step 11: Check your scoring routine for assumptions about the outcome of interactions (see section 5.5 above and section 3.7.2 on page 139). If you find any, modify accordingly.

Step 12: Decide if you want to be able to modify transport parameter settings. If yes, include a call to `get_transport_parameter` somewhere before the call to `HATCH`, code your own input routine or hard code the parameters you wish.

Step 13: Check your environment file. In addition to the unit numbers already reserved in EGS4 for data files (*e.g.* unit 12 is connected to the PEGS4 data file), EGSnrc uses various other units to read additional data required:

- `unit 11`: Multiple scattering data (file `msnew.data`)
- `unit 76`: NIST bremsstrahlung cross section data (file `nist_brems.data`)
- `unit 77`: Relaxation data (file `photo_relax.data`)
- `unit 78`: Bound Compton data (file `incoh.data`)
- `unit 79`: Photo-absorption cross section data (file `photo_cs.data`)

The above links are made directly in `egs_run`. If your user code uses any of the above units (via link statements in the environment file), modify accordingly.

Step 14: If your user code uses the PRESTA algorithm, the input file has most likely a PRESTA associated input (5 numbers on a line). If the PRESTA associated input is not the last line in your input file, DELETE IT before running the code.

Step 15: Run test calculations. If you encounter statistically significant differences to your previous results in a way which differs from what has been discussed in the literature or the documentation, please send a message with a detailed description of your results to one of the authors since we would like to make a collection of situations where the use of the new electron transport physics and various cross section options is necessary.

5.10 Example: adapting XYZDOSnrc

The user code `XYZDOS` has been used for timing studies of the EGS4 system[117]. It is a relatively simple code that provides a good example for the adaptation procedure.

Step 1

There is no definition of the `$CALL-HOWNEAR` macro in `xyzdos.mortran` so include the recommended replacement

```
REPLACE {$CALL-HOWNEAR(#);} WITH {call
hownear({P1},x(np),y(np),z(np),ir(np)); }
```

near the beginning of `xyzdos.mortran`. The geometry information is contained in the comin `GEOM` which must be made available to the `hownear` routine. The code for `hownear`, which is added at the end of `xyzdos.mortran`, looks therefore as follows (using explicit data typing):

```
subroutine hownear(tperp,x,y,z,ir);

$IMPLICIT-NONE;

COMIN/GEOM/;
"Input/Output variables"
$REAL tperp,x,y,z;
$INTEGER ir;
"Local variables"
$INTEGER irx,iry,irz;

$DECODEIR(ir,irx,iry,irz); "xyzdos defined macro to decode region number"
tperp = 1e10;
tperp = min(tperp,xbound(irx+1)-x,x-xbound(irx));
tperp = min(tperp,ybound(iry+1)-y,y-ybound(iry));
tperp = min(tperp,zbound(irz+1)-z,z-zbound(irz));
return;
end;
```

Step 2

The macro `$RNG-INITIALIZATION;` is added to the main routine after the input is complete. XYZDOS reads-in a variable `INSEED` which is used to initialise the old EGS4 random number generator (which had just one seed called `IXX`). XYZDOSnrc, the EGSnrc version will not modify the initial seed(s) and therefore all code associated with `INSEED` and `IXX` are removed. The default RNG for the system, viz RANLUX will be used with `luxury_level = 1`.

Step 3

There are no replacements of EGSnrc comin's.

Step 4
There are no replacements of `$COMIN-XXXX` macros.

Step 5
`XYZDOS` uses the `nrcc4mac.mortran` file but there are no definitions of `USER-STEP-CONTROLS`, `USER-VARIANCE-REDUCTION` and `USER-MISC`.

Step 6
The only relevant definition and use encountered is `ESTEPE`. It is declared as `REAL ESTEPE($MXMED)` and used to input maximum allowed fractional energy losses for each medium. It is also passed to the `FIXTMX` routine which initialises step-lengths for the `ESTEPE` selected. In EGSnrc this task is performed internally by the `mscati` routine. Since the `get_transport_parameter` routine will be used to parse transport parameter settings, all appearances of `ESTEPE` are removed.

Step 7
There are no definitions of these macros.

Step 8
Range rejection is not used in `XYZDOS`.

Step 9
Explicit data typing is being used. Therefore

- The statement `$IMPLICIT-NONE;` is included just before the line

  ```
  ;COMIN/BOUNDS,GEOM,MEDIA,MISC,RANDOM,SCORE,STACK,THRESH/;
  ```

  in the main program and also in `HOWFAR` and `AUSGAB`.
- All variables used by the main program, in the common blocks, and by `HOWFAR` and `AUSGAB` are declared explicitly. Variable declaration for `HOWNEAR` was already performed in step 2.
- No declarations are needed for `COMIN/USER/`
- None of the macros from step 9 of the instructions is used in `XYZDOS`

Step 10
When trying to compile, there were a couple of error messages. All of them are because of failure to declare some of the variables used. Once declared the code compiles without problems.

Step 11
The scoring routine does not need adaptation. However, it performs a check for stack overflow. In EGSnrc this check is performed internally and therefore the associated code is removed.

Step 12
To be able to modify transport parameter settings, a call to the `get_transport_parameter` routine is included just before the call to `HATCH`.

Up to this point the whole procedure took about 1.5 hours (including writing down what was done).

Step 13
The environment file does not use any of the fortran units reserved for EGSnrc data input.

Step 14
`XYZDOS` does not use the PRESTA algorithm. one can therefore just take an example input file (*e.g.* `benche.egs4inp` which is the input file for the EGS4 timing benchmark) and run `XYZDOS`. However, let us modify the transport parameter settings from their default values to the ones that correspond as closely as possible to default EGS4. To do so, copy `benche.egs4inp` to `test.egsinp` (the default input file extension for EGSnrc is `.egsinp` instead of `.egs4inp`) and include the `MC Transport Parameter` section from the `test_tutor7.egsinp` file found in the `tutor7` directory on the `$HEN_HOUSE`.

At this point it was noted that the transport cut-off energies (`ECUT,PCUT`) are input in `get_transport_parameter` and also at the beginning of the main `XYZDOS` routine. This duplication renders one of the inputs unnecessary and so one should remove the code associated with `ECUT,PCUT` input in the main `XYZDOS` routine. Next modify the input file to make transport parameter settings in EGSnrc match EGS4 as closely as possible (see section 3.4.2.i. page 131). The modified `test.egsinp` file follows:

```
BENCHE: 20 MeV e- on 19cm**3 WATER Phantom - dual waters,rho=1.001
2
H2O
H2O2
-3,-3,-3
0.0,
1.0,9
0.25,4
1.0,9
0.0
1.0,9
0.25,4
1.0,9
0.0
1.0,6
0.25,4
1.0,12
1,22,1,22,1,22,1,1.001
1,22,1,22,2,2,2,1.001
1,22,1,22,4,4,2,1.001
1,22,1,22,6,6,2,1.001
1,22,1,22,8,8,2,1.001
1,22,1,22,10,10,2,1.001
1,22,1,22,12,12,2,1.001
1,22,1,22,14,14,2,1.001
1,22,1,22,16,16,2,1.001
1,22,1,22,18,18,2,1.001
```

```
1,22,1,22,20,20,2,1.001
1,22,1,22,22,22,2,1.001
0,0,0,0,0,0,0,0.0
10,13,10,13,1,20,1
1,20,1,20,1,1,0
1,20,1,20,8,8,0
0,0,0,0,0,0,0
4.5,14.5
4.5,14.5
0.0,90.0,90.0
20,-1,100000,0,11.89,0

##############################
:Start MC Transport Parameter:

 Global ECUT=                     1.060     #as in the benchmark
 Global PCUT=                     0.010     #as in the benchmark
 Global SMAX=                     1e10      #no geom step-size restriction
 Bound Compton scattering=        Off       #use Klein-Nishina as in EGS4
 Rayleigh scattering=             Off       #Rayleigh scattering is off
 Atomic relaxations=              Off       #No relaxations as in EGS4
 Photoelectron angular sampling=  Off       #No PE angular distribution
 Brems cross sections=            BH        #use Bethe-Heitler(like EGS4)
 Brems angular sampling=          Simple    #EGS4 fixed angle not available,
                                            #this is closest we can get
 Pair angular sampling=           Off       #Fixed angle for pairs(as EGS4)
 ESTEPE=                          0.04      #original benche value
 XIMAX=                           0.5       #roughly corresponds to Bethe's
                                            #restriction for Molière theory
 Skin depth for BCA=              1e10      #together with inexact boundary
                                            #crossing => CH a la EGS4
 Boundary crossing algorithm=     PRESTA-I  #i.e.  don't use exact BCA
 Electron-step algorithm=         PRESTA-I  #this does not matter with
                                            #inexact BCA & large skin-depth
 Spin effects=                    Off       #no have option to use Molière
                                            #MS but at least can turn off
                                            #spin to get closer to Moliere

:Stop MC Transport Parameter:
##############################
```

Step 15

Running the adapted `XYZDOS` code with this input file on NRC computers (600 MHz PIII with SuSE Linux and egcs-2.91.66 compiler, -O2 optimization) results in a CPU time of 93 seconds. This is quite disappointing compared to the 69 seconds for the benchmark using standard EGS4 (in view of the fact that very similar transport parameter settings were used). However, different random number generators are used in the two codes and the EGSnrc version needs to calculate the perpendicular distance to the closest boundary at the beginning of each electron step, information which is not used in the simulation mode mimicking EGS4. To check the influence of these two differences

- Re-implement the simple multiplicative random number generator used in the original XYZDOS version. This is most easily done by including

  ```
  REPLACE {;COMIN/RANDOM/;} WITH {
    ;COMMON/RANDOMM/IXX;
    $INTEGER IXX;
  }
  REPLACE {$RANDOMSET#;} WITH {
    IXX=IXX*663608941;
    {P1}=0.5 + IXX*0.23283064E-09;
  }
  REPLACE {$RNG-INITIALIZATION;} WITH {IXX=987654321;}
  ```

  in the xyzdos.mortran file (this overrides the definitions from ranlux.mortran). Running the code with this random number generator reduces the CPU time to 85 seconds.

- Remove the calculation of the minimum perpendicular distance by replacing the $CALL-HOWNEAR macro with

  ```
  REPLACE {$CALL-HOWNEAR(#);} WITH { {P1} = 0; }
  ```

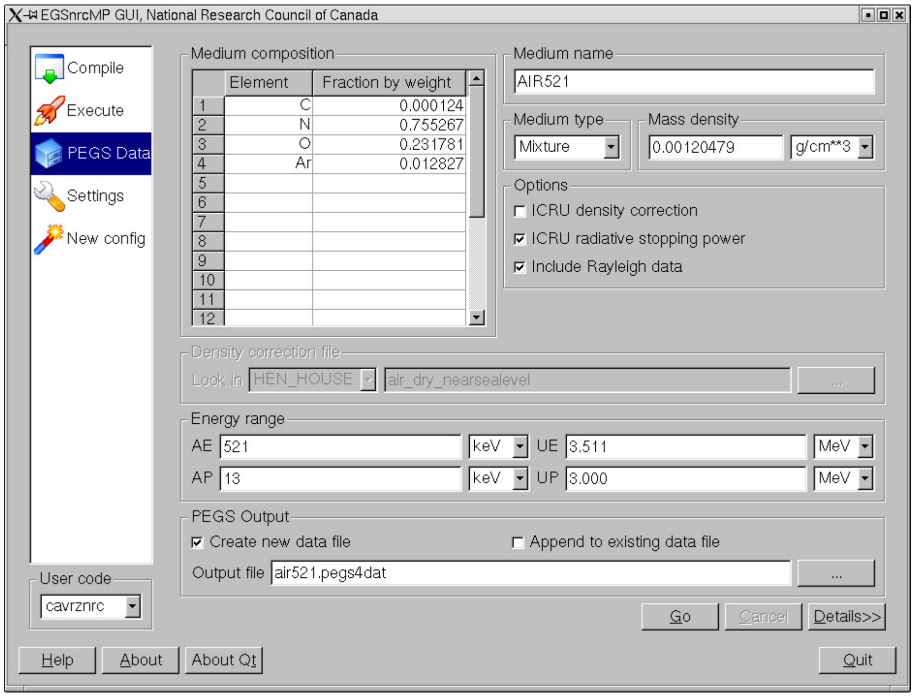
  After that modification the CPU time becomes 79 seconds.

The remaining 20% CPU time difference must be due to the various modifications of the sampling and transport techniques. However, all modified sampling routines run faster, or at least as fast, compared to the original EGS4 implementation, if tested alone. The difference must be therefore due to the more accurate evaluation of energy dependent quantities in EGSnrc (such as CSDA energy loss, multiple scattering related quantities, etc.). The very purpose of implementing more accurate techniques for energy dependent quantities is to allow for longer electron steps and so, the use of the 4% maximum fractional energy loss per step in EGSnrc is wasteful. On the other side, using an EGS4-type algorithm for path-length corrections and boundary crossing is not particularly accurate and step-sizes should not be allowed to exceed certain limits. This is best accomplished by reducing the maximum allowed first elastic scattering moment per step (ximax), i.e. ESTEPE is set to 0.25 and ximax to 0.1 in the input file. With this parameter selection the EGSnrc version of XYZDOS needs only 48 seconds for the benchmark, *i.e.* 1.5 times less that the original EGS4 version.

6 PEGS4 User Manual

The PEGS4 system has been modified very little for use with EGSnrc although EGSnrc requires considerably more data than provided by PEGS4 and this is read in directly via the HATCH routine (see fig 18 on page 108). Most of this section is basically a reprint of the PEGS4 User Manual from SLAC-265. There have been a few additions to PEGS4 since 1985 and these are summarized in the next section.

In addition to what is described here, the EGSnrcMP environment includes a user-friendly GUI for using PEGS4. See Report PIRS-877[14] for a detailed description.



Medium composition

	Element	Fraction by weight
1	C	0.000124
2	N	0.755267
3	O	0.231781
4	Ar	0.012827
5		
6		
7		
8		
9		
10		
11		
12		

Medium name: AIR521

Medium type: Mixture

Mass density: 0.00120479 g/cm**3

Options
- [ ] ICRU density correction
- [x] ICRU radiative stopping power
- [x] Include Rayleigh data

Density correction file: Look in HEN_HOUSE air_dry_nearsealevel

Energy range
AE 521 keV UE 3.511 MeV
AP 13 keV UP 3.000 MeV

PEGS Output
- [x] Create new data file
- [ ] Append to existing data file

Output file air521.pegs4dat

User code: cavrznrc

Figure 38: Screen shot of the `egs_gui` set to run PEGS4 to create an air data set. Filling in this form is much easier than creating an input file and access to the density effect corrections is easy. The GUI does not allow any value except `IUNRST = 0`. For details, see PIRS-877[14].

6.1 Some new documentation

6.1.1 Photon data for EGSnrc

Originally, EGS main field of application was high energy physics. For this reason, no attention was paid to the accuracy of the photon cross sections near atomic absorption edges[118] and PEGS4 was designed to generate a grid with a maximum number of photon energy points, `$MXGE`, in the entire energy range requested by the user. `$MXGE` is a macro which sets the array dimensions for the energy-dependent photon data. Although `$MXGE` can be changed in the PEGS4 code, PEGS4's algorithm for deciding how many bins to use might still pick many fewer bins than `$MXGE`. The decision is based on the overall accuracy to reproduce the cross sections in the entire energy interval.

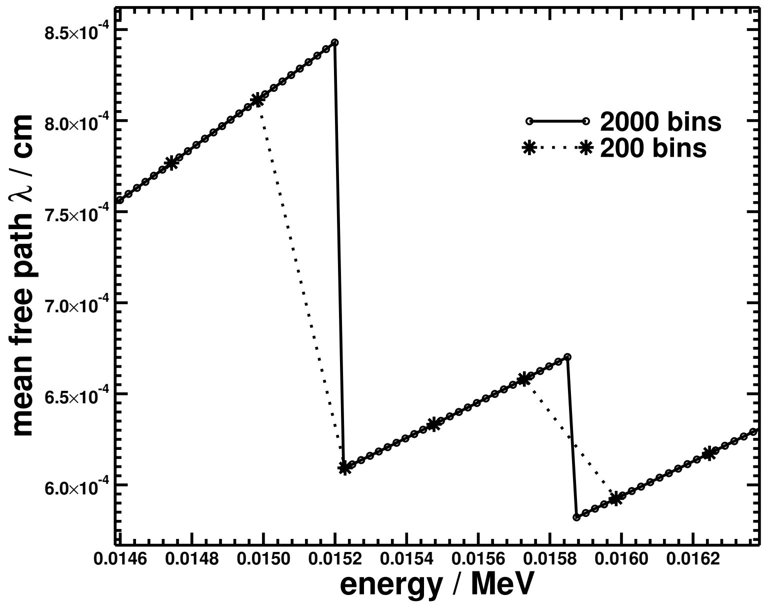


Figure 39: Mean free path λ for lead in the vicinity of the L1 and L2 atomic absorption edges for two different energy grids. Data sets generated by EGSnrc from the XCOM cross section compilation for an energy range 10 keV to 250 keV.

Typical PEGS4 data sets used in radiotherapy calculations are created for an energy range between 10 keV and 20 MeV. If these data sets are used in the kilovoltage energy range, inaccuracies in the cross section interpolation procedure near absorption edges can potentially lead to errors in the calculations. Fig. 39 shows two cross section data sets for the same energy range (10 keV - 250 keV) when using different energy grids (200 and 2000 bins). A coarser energy grid will cause larger interpolation errors near the absorption edge.

EGSnrc default behavior has been set to ignore PEGS4 photon data and recreate the photon cross sections using a logarithmic energy grid with `$MXGE` energy points. However, EGSnrc still needs the PEGS4 data file for material information, photon threshold energies (`AP, UP`), and a fraction of the electron data. By default data is read from the XCOM[119] photon cross section compilation (`xcom_pair.data, xcom_photo.data, xcom_rayleigh.data and xcom_triplet.data`). As alternatives, EGSnrc offers cross section data files for the Storm and Israel[21](`si_*.data`) and EPDL97[39] (`epdl_*.data`) photon cross section compilations. New photon cross section compilations can be added following the same format as above, *i.e.*, `prefix_*.data`. The choice of the photon cross section compilation is done by input as shown in the example below:

```
:start MC transport parameter:
    .                              # This entry is case sensitive
    Photon cross sections = xcom   # xcom (default), epdl, si or any_prefix
    .
:stop MC transport parameter:
```

The prefix used is case sensitive and must be such that the required files are available in the `$HEN_HOUSE/data` directory.

Since the 2011 release (V4-r2-3-2) EGSnrc allows the use of photon cross section data in the PEGS4 data file. This could come in handy to compare with older calculations for validation purposes. To accomplish this, one must set the photon cross section key to `pegs4` (case insensitive) as shown in the following input file snippet:

```
:start MC transport parameter:
    .
    Photon cross sections = pegs4 # case insensitive
    .
:stop MC transport parameter:
```

6.1.2 Some additional outputs- unrestricted cross sections

The original version of PEGS4 contained a few undocumented options which many people have made use of, so they are now documented here. Basically there is an additional parameter, `IUNRST` which allows various stopping powers to be calculated, rather than just the restricted stopping powers normally calculated when `IUNRST = 0`. The `IUNRST` input is made as part of the namelist input for INP (along with NE, AE etc). Basically `IUNRST` gives access to a variety of different stopping powers and allows for simulations which model various types of CSDA calculations.

IUNRST = 0, restricted stopping powers: This is the default case which is needed for normal simulations. The stopping powers output by PEGS4 are the restricted collision and radiative stopping powers.

IUNRST = 1, unrestricted collision stopping power: This is useful for calculating unrestricted stopping powers for comparison to frequently published values.

IUNRST = 2, CSDA data set: This produces a data set which can do one form of CSDA calculation. The stopping power produced is the unrestricted total (collision + radiative) stopping power and the distances to discrete electron interactions is infinite (i.e. they never occur). A simulation done with this data set is a form of CSDA calculation, with all of the bremsstrahlung energy deposited locally.

IUNRST = 3, CSDA calculation with brem interaction: The stopping powers are the sum of the unrestricted collision stopping power plus the restricted stopping power and the distance to discrete interactions takes into account only bremsstrahlung events.

IUNRST = 4, CSDA calculation with delta-ray interactions: The stopping powers are the sum of the restricted collision stopping powers and the unrestricted collision stopping powers and the distance to discrete interactions takes into account only the creation of knock-on electrons or delta-rays.

IUNRST = 5, unrestricted radiative stopping power: This complements `IUNRST = 1` and allows comparison to published radiative stopping powers.

IUNRST = 6, restricted radiative stopping power: Allows for direct calculation of the restricted radiative stopping powers.

IUNRST = 7, restricted collision stopping powers: Allows for direct calculation of the restricted collision stopping power.

Note that for low values of AP (e.g. 0.010 keV) the restricted radiative collision stopping power is very close to zero and hence the stopping powers with `IUNRST = 0` are close to those for `IUNRST = 7`.

The code EXAMIN (see section 3.13.3, page 155) will take the data sets produced by PEGS4 and print tables of cross section data and/or plot these same data in user friendly units.

6.1.3 Use of ICRU Report 37 Collision Stopping Powers

For very precise dosimetry work it is often advantageous to make use of collision stopping powers recommended by the ICRU in their Report 37[50]. This option was added to PEGS4 in 1989[70]. Basically PEGS4 allows the user to read in a file which contains an arbitrary density effect data set (δ values). The EGSnrc distribution supplies the data sets required for a large number of materials as calculated by Berger and Seltzer[68] for ICRU Report 37 (see `$HEN_HOUSE/pegs4/density_corrections`).

To implement this option, one adds `EPSTFL=1` to the `INP` namelist input for the `ELEM`, `COMP` or `MIXT` option and executes PEGS4 as:

```
pegs4.exe -i inputfile [-o ofile] [-a] [-d density] [-x crosssection] [-e HEN_HOUSE]

inputfile.pegs4inp   the input file
output defaults to $HEN_HOUSE/pegs4/data/inputfile.pegs4dat
                or, if ofile is given, to $HEN_HOUSE/pegs4/data/ofile.pegs4dat
[-a]            => append results to output file
[-d density]    => use density.density   for density effect
[-x crosssection] => use $HEN_HOUSE/pegs4/crosssection instead of
                                $HEN_HOUSE/pegs4/pgs4pepr.dat
[-e HEN_HOUSE]  => use this absolute location as the HEN_HOUSE
```

where `input1.pegs4inp` contains the standard PEGS4 input file (with `EPSTFL=1`) and `input2.density` is the file with the density effect data needed.

PEGS4 verifies that the data in `input2.density` corresponds to the same material as described in `input1.pegs4inp`, and in particular demands that the densities match. If you want to create a data set using the density effect data for graphite based on a density of 2.26 g/cm^3 but the real bulk data is 1.70 g/cm^3, you must edit the density effect file and artificially change the density to match the bulk density you are after, otherwise PEGS4 will stop.

Note that the density effect file also provides the ICRU 37 value of the I value for the material and PEGS4 uses this rather than its internally calculated value.

Data sets created with `EPSTFL=1` and `IUNRST = 1` should match the collision stopping powers in ICRU Report 37 exactly.

6.1.4 Use of ICRU Report 37 Radiative Stopping Powers

In 1989 an option was added to PEGS4[53] which ensured that the unrestricted radiative stopping powers calculated by PEGS4 were identical to those calculated by Berger and Seltzer[68] and included in ICRU Report 37[50]. This was done by scaling the cross sections used in PEGS4 to ensure that these stopping powers matched. This made a significant difference to the bremsstrahlung cross sections at low energies. It is strongly recommended that this option always be used and has been made the default value in PEGS4. To restore the original PEGS4 values, enter the input `IAPRIM=0` to the `INP` namelist input for the `ELEM,` `COMP` or `MIXT` option.

Note that this change does not produce the same photon differential cross sections as calculated by Seltzer and Berger[48], but this option has been added to EGSnrc (see section 3.4.2) and turned on by setting `ibr_nist = 1`.

6.1.5 A Bug in PEGS4

Although the PEGS4 manual reports that one may produce data sets for many materials in one run, this is not in fact possible. One must run the code separately for each material desired and then concatenate these files into one master file.

6.2 Original PEGS4 User Manual

SLAC265 - APPENDIX 3

PEGS4 User Manual

By

Walter R. Nelson
Stanford Linear Accelerator Center
Stanford University
Stanford, CA 94305, U.S.A.

Hideo Hirayama
National Laboratory for High Energy Physics (KEK)
Oho-machi, Tsukuba-gun, Ibaraki, Japan

David W. O. Rogers
National Research Council of Canada
Ottawa K1A 0R6, Canada

31 December 1985

[This PEGS4 User Manual is based directly on Appendix 3 of
SLAC-265, The EGS4 Code System]

```
A3.  PEGS4 USER MANUAL
-----------------
A3.1 Introduction
-----------------

     The PEGS code (Preprocessor for EGS) is a stand alone
utility program written in Mortran **.  PEGS' purpose is to
generate material data for the EGS code, and to provide other
services for the user who is studying or simulating electro-
magnetic interactions.  The active operations of PEGS are
functionals; that is, they are operations whose arguments are
functions (the functions related to physics interactions).
Included among these operations are:

     -  Fitting of functions by means of piecewise linear fits.

     -  Production of print plots of selected functions.

     -  Evaluation of functions at selected points.

     -  Comparision of functions with sampled spectra.


Associated with these active functionals are other operations;
namely,


     -  Selection of material to which the functions refer.

     -  Selection of energy cutoffs for fits.

     -  Punching of fit data.


[Note: Those interested in preparing data sets for EGS4 can go
       directly to Section A3.3]

---------------
```

** A. J. Cook, "Mortran3 User's Guide"[98]. Also, see ‘‘An EGS Users Guide to Mortran3’’ (section 8).

A3.1-1

```
-----------------------------------
A3.2 Structural Organization of PEGS
-----------------------------------
```

The PEGS code contains over 4200 Mortran source lines which are the source for a MAIN program, BLOCK DATA subprogram, 12 subroutines, and 83 functions. Despite the large number of subprograms, PEGS has a simple structure. Fig. A3.2.1 shows a flowchart of the MAIN program of PEGS. After the once-only initializations an option loop is entered. Each time through the option loop, an option is read (option names are four characters and are read as 4A1), numeric control parameters are read (using NAMELIST/INP/), and then the option name is looked up in the option table. If not found, the job is aborted. If found, the appropriate code is executed and return is passed to the beginning of the option loop. Normal exit from the loop is by selection of the STOP option, or detection of an End of File condition on the control input file. The details for the use of the options are contained in Section A3.3.

Fig. A3.2.2 shows the subprogram relationships of PEGS. Boxed items are subprograms, and labels for option names (i.e., :CALL:) are used to show which subprograms correspond to which options. It can be seen that the physical routines are accessed directly for the PWLF option. For utility options (TEST, PLTN, PLTI, HPLT, and CALL) the physical routines are referenced using the function FI---the so-called "function multiplexer". Function FI has five arguments. The first argument (I) tells which physical function to invoke, and the other four arguments (X1, X2, X3, X4) are used as needed as arguments for the called function. FI then returns the value returned by the called function.

This method of implementing options that are functionals was selected to avoid the necessity of having a separate call to the associated utility routines for each physical function on which it might be desired to operate. It was also desired to be able to refer to the particular function symbolically, both at compile time and at run-time, and to know the number of arguments to each function. In order to have these conveniences and also allow easy insertion or deletion of functions to the list of functions accessible to FI, a Mortran macro ($FUNCTIONS) was written which takes a list of names of functions (each of which is immediately preceded by the number of arguments it has) and generates other macros containing the desired information. In particular,
A3.2-1

```
                         +------+
                         | PEGS |
                         +------+
                           |  |       +-------------+
               initialize  |  +------>| OPTION LOOP |<-----+
            +--------------+          +-------------+      A
            |                                |             |
            |                                |             |
            V                                V             |
+------------------------+          +-----------------+    |
|   Compute Physical &   |          |   Read Option   |    |
| Mathematical Constants |          |    Name (4A1)   |    |
+------------------------+          |        &        |    |
            |                       |   Read Control  |    |
            |                       |    Parameters   |    |
            V                       | (NAMELIST/INP/) |    |
+------------------------+          +-----------------+    |
| Read Pair Production & |                   |             |
|  Photo Cross Sections  |                   |             |
|    from File PHPRDAT   |                   |             |
+------------------------+                   |             |
                                             V             |
               +------+     Yes      +--------------+      |
               | Stop |<------------ | End of File? |      |
               +------+              +--------------+      |
                                             |             |
                                             | No          |
                                             |             |
                                             V             |
     +-----------------------+       +---------------+     |
     | Illegal Option---Stop |<----- | Select Option |     |
     +-----------------------+       +---------------+     |
                                             |             |
                                             |             |
                                             V             |
  +<---------------------------------------- +             |
  |                                                        |
  V                                                        |
  *                                                        *
 * *                                                      * *
* 1 *                                                    * 2 *
 * *                                                      * *
  *                                                        *
```

Fig. A3.2.1 Flowchart of the MAIN Program of PEGS

(continued on next page)

A3.2-2

```
   *                                                          *
  * *                                                        * *
 * 1 *                                                      * 2 *
  * *                                                        * *
   *                                                          *
   |          +-----------------------+                       ^
   +-->:ELEM: | Set Up Element Medium | -------------------> |
   |          +-----------------------+                       |
   |          +-----------------------+                       |
   +-->:MIXT: | Set Up Mixture Medium | -------------------> |
   |          +-----------------------+                       |
   |          +------------------------+                      |
   +-->:COMP: | Set Up Compound Medium | ------------------> |
   |          +------------------------+                      |
   |          +------------------------+                      |
   +-->:ENER: | Set Energy Cutoffs and | ------------------> |
   |          | Compute Thresholds     |                      |
   |          +------------------------+                      |
   |          +---------------------+                         |
   +-->:PLTN: | Plot Named Function | ---------------------> |
   |          +---------------------+                         |
   |          +-----------------------+                       |
   +-->:PLTI: | Plot Indexed Function | -------------------> |
   |          +-----------------------+                       |
   |          +-----------------------+                       |
   +-->:HPLT: | Histogram Theoretical | -------------------> |
   |          |  vs Sampled Spectrum  |                       |
   |          +-----------------------+                       |
   |          +-------------------------+                     |
   +-->:CALL: | Evaluate Named Function | -----------------> |
   |          +-------------------------+                     |
   |          +-----------------------------+                 |
   +-->:TEST: | Plot Functions To Be Fitted | -------------> |
   |          +-----------------------------+                 |
   |          +----------------------+                        |
   +-->:PWLF: | Piecewise Linear Fit | --------------------> |
   |          +----------------------+                        |
   |          +------------------------+                      |
   +-->:DECK: | Punch Deck of Material | ------------------> +
   |          | Dependent Data         |
   |          +------------------------+
   |          +------+
   +-->:STOP: | Stop |
              +------+
      Fig. A3.2.1  Flowchart of the MAIN Program of PEGS
                     (continued from previous page)
 A3.2-3
```

```
    +------------+         +------+
    | BLOCK DATA |         | MAIN |
    +------------+         +------+
                               |
    + --- + -- + ----- + ----- + ----- + ----- + ---- + ---- +
    |     |            |       |       |       |      |      |
    |     |         :PWLF:  :DECK:  :TEST:  :HPLT: :CALL: :ENER:
    |     |            |       |    :PLTN:     |      |
    |     |            |     +---+  :PLTI:  +-----+   |
    |     |            |     |LAY|     |    |HPLT1|   |
    |     |            |     +---+     |    +-----+   |
    |     |            |            +----+     |      |
 +------+ |        +---+----+       |PLOT|     |      |
 |PMDCON| |        |        |       +----+     |      |
 +------+ |     +-----+  +-----+       |       |      |
          |     |EBIND|  |PWLF1|       +-----> | <--- +
          |     +-----+  +-----+               |
       :ELEM:               |                  |
       :MIXT:            +----+                |
       :COMP:            |QFIT|                *
          |              +----+               * *
  + ----- + ----- +         |                * 3 *
  |       |       |         |                 * *
+---+ +------+ +------+     |                  *
|MIX| |SPINIT| |DIFFER|     |
+---+ +------+ +------+     |
                       + -- + -- + --------------- +
                       |         |                 |
                    +-----+   +-----+           +-----+
                    |EFUNS|   |GFUNS|           |RFUNS|
                    +-----+   +-----+           +-----+
                       |         |                 |
   +-----+----+-----+--+---+---+ |   +------+   +-----+
   |     |    |     |      |   | +---|PHOTTE|   |AINTP|
+------+ | +------+ | +------+ | |   +------+   +-----+
|SPTOTP| | |ANIHTM| | |BHABTM| | |   +------+
+------+ | +------+ | +------+ | +---|COMPTM|
         |          |          | |   +------+
+------+ | +------+ | +------+ | |   +------+
|SPTOTE|-+ |AMOLTM|-+ |BREMTM|-+ +---|PAIRTU|
+------+   +------+   +------+   |   +------+
                                 |   +------+
                                 +---|COHETM|
                                     +------+
```

Fig. A3.2.2 Subprogram Relationships of PEGS

(continued on next page)

A3.2-4

```
                            *
                           * *
                          * 3 *
                           * *
                            *
                            |
                            |
                            |
                            |
             + --------------------- +
             |           FI          |
             | "Function Multiplexer" |
             + --------------------- +
                            |
                            |
                            |
      + ------------------------------------ +
      |                                      |
      |   ALIN     APRIM    COMPFM   PAIRTE  |
      |   ALINI    BHABDM   COMPRM   PAIRTM  |
      |            BHABFM   COMPTM   PAIRTR  |
      |   ADFMOL   BHABRM   CRATIO   PAIRTU  |
      |   ADIMOL   BHABTM   EBIND    PAIRTZ  |
      |   ADDMOL   BREMDR   EBR1     PBR1    |
      |   ALOG     BREMFR   EDEDX    PBR2    |
      |   EXP      BREMDZ   ESIG     PDEDX   |
      |   AREC     BRMSDZ   FCOULC   PHOTTZ  |
      |   ALKE     BREMFZ   GBR1     PHOTTE  |
      |   ALKEI    BRMSFZ   GBR2     PSIG    |
      |   AMOLDM   BREMRR   GMFP     SPIONE  |
      |   AMOLFM   BREMRM   PAIRDR   SPIONP  |
      |   AMOLRM   BREMRZ   PAIRFR   SPTOTE  |
      |   AMOLTM   BREMTM   PAIRDZ   SPTOTP  |
      |   ANIHDM   BREMTR   PAIRFZ   TMXB    |
      |   ANIHFM   BRMSRM   PAIRRM   TMXS    |
      |   ANIHRM   BRMSRZ   PAIRRR   TMXDE2  |
      |   ANIHTM   BRMSTM   PAIRRZ   XSIF    |
      |            COHETM                    |
      |            COHETZ                    |
      |            COMPDM                    |
      + ------------------------------------ +
```

Fig. A3.2.2 Subprogram Relationships of PEGS
(continued from previous page)

A3.2-5 .

the following macros are defined:

$NFUNS - Gives the number of functions.

$FLIST$DATA(FNAME) - Generates a data statement initializing the array FNAME(6,$NFUNS) so that FNAME(i,j) has the ith character of the name of the jth function.

$FLIST$NARGS - Gives a list of the number of arguments for each function, which is used to initialize the run-time array NFARG($NFUNS).

$FLIST$FNUMS - Gives a list of numbers from 1 to $NFUNS, which is used to generate the computed GO TO in FI.

$FLIST$FCALLS - Generates the function calls in FI with the proper number of arguments for each function taken from the list X1, X2, X3, X4.

$FN(function name) - Gives the function index of the specified function.

$NA(function name) - Gives the number of arguments for the specified function.

It should also be noted that there are relationships between the functions shown in Fig. A3.2.2 that are not indicated there. We show the most complicated of these in Figs. A3.2.3a,b (Bremsstrahlung Related Functions) and in Figs. A3.2.4a,b (Pair Production Related Functions). One reason for the complexity of these is that the higher level forms of the cross sections must be obtained by numerical integration of the more differential forms.

A3.2-6

```
                        +-------+
                        |BREMTM|
                        +-------+
                            |
                            V
                        +-------+
                        |BREMRM|
                        +-------+
                            |
                            V
 +-------+              +-------+  initialize  +-------+
 |  QD  |<------------- |BREMRZ|-------------> |BREMDZ|
 +-------+              +-------+    BREMFZ    +-------+
    |                                              |
    V                                              |
 +-------+                                         |
 |DCADRE|                                          |
 +-------+                                         |
    |                                              |
    V                                              V
 +-------+              +-------+              +-------+
 |BREMFZ|-------------> |BRMSFZ|<------------- |BRMSDZ|
 +-------+              +-------+              +-------+
                            A                    A |
                            |                    | |   +-------+
            +---------------+                    | +-->|APRIM |
            |                                    | |   +-------+
        +-------+       +-------+  initialize    | |
        |DCADRE|   +----|BRMSRZ|---------------> + |   +-------+
        +-------+  |    +-------+    BRMSFZ        +-->| XSIF |
            A      |        A                      |   +-------+
            |      |        |                      |
        +-------+  |    +-------+                  |   +-------+
        |  QD  |<--+    |BRMSRM|                   +-->|FCOULC|
        +-------+       +-------+                      +-------+
                            |
   +-------+            +-------+            +-------+
   |SPTOTE|-----------> |BRMSTM|<----------- |SPTOTP|
   +-------+            +-------+            +-------+
      |                                          |
      V                                          V
   +-------+            +-------+            +-------+
   |SPIONE|-----------> |SPIONB|<----------- |SPIONP|
   +-------+            +-------+            +-------+
```

Fig. A3.2.3a Bremsstrahlung Related Functions---Most Accurate Form
   (Used to Produce the Total Cross Sections and Stopping Power).
 A3.2-7

```
                            +-------+
                            |BREMTR|
                            +-------+
                                |
                                |
                                V
                            +-------+
                            |BREMRR|
                            +-------+
                                |
                                |
                     initialize V
                    + <---------+--------> +
                    |  BREMFR              |
                    |                      |
                    V                      V
                 +-------+              +-------+
                 |BREMDR|               |  QD  |
                 +-------+              +-------+
                    |                      |
                    |                      |
                    |                      V
                    |                   +-------+
                    |                   |DCADRE|
                    |                   +-------+
                    |                      |
                    |                      |
                    |                      |
                    |       +-------+      |
                    + ----> |BREMFR| <---- +
                            +-------+
```

Fig. A3.2.3b Bremsstrahlung Related Functions---With Run-
Time Approximations (For Comparison
with Sampled Spectra).

A3.2-8

```
                        +------+
                        |PAIRTU|
                        +------+
                          |  |
   + <------------------- +  + -------------------> +
   |                                                |
   V                                                V
+------+                                         +------+
|PAIRTM|                                         |PAIRTE|
+------+                                         +------+
   |                                                |
   V                                                V
+------+                                         +------+
|PAIRRM|                                         |PAIRTZ|
+------+                                         +------+
   |                                                |
   V                                                V
+------+   initialize                            +------+
|PAIRRZ|-------------> +                         |AINTP |
+------+     PAIRFZ    |                         +------+
   |                   |
   |                   |
   |                   |
   |                   |
   V                   V
+------+            +------+            +------+
|  QD  |            |PAIRDZ|-----+----> | XSIF |
+------+            +------+     |      +------+
  |                    |         |
  |                    |         |      +------+
  |                    |         +----> |FCOULC|
  V                    V                +------+
+------+            +------+
|DCADRE|----------> |PAIRFZ|
+------+            +------+
```

Fig. A3.2.4a Pair Production Related Functions---Most
Accurate Form (Used to Produce the Total
Cross Sections and Stopping Power).

```
                   +------+
                   |PAIRTR|
                   +------+
                       |
                       |
                       V
                   +------+
                   |PAIRRR|
                   +------+
                       |
                       |
             initialize V
            + <-------- + -------> +
            |  PAIRFR              |
            |                      |
            V                      V
         +------+              +------+
         |PAIRDR|              |  QD  |
         +------+              +------+
            |                      |
            |                      |
            |                      V
            |                  +------+
            |                  |DCADRE|
            |                  +------+
            |                      |
            |                      |
            |                      |
            |      +------+        |
            + ---> |PAIRFR| <---- +
                   +------+
```

Fig. A3.2.4b Pair Production Related Functions---With
              Run-Time Approximations (For Comparison
              with Sampled Spectra).

A3.2-10

Table A3.2.1 lists the SUBROUTINES used in PEGS. A brief description of their use and page references for a fuller discussion is given.

Table A3.2.2 lists the FUNCTIONS used in PEGS along with their mathematical symbols, definitions, and locations in this report for a fuller discussion. The names of most of the functions have been chosen in a rather mnemonic way. The first three or four letters suggest the process being considered. The last letter designates the form of the cross section (Z for element, M for mixture, and R for "run-time" mixture). The next to last letter describes either the particular form of the cross section (such as D for differential, T for total or R for range-integrated), or it indicates that only the secondary energy is to vary, with other data being passed through a common. The letter F is used in such cases and the data in common is initialized using the corresponding function that has a next to last letter of D. If the function word begins with an I through N (i.e., the FORTRAN integer convention) the word is prefixed with the letter A. A few examples are given below:

AMOLDM is the differential Moller cross section for a mixture of elements.

BREMDR is the differential bremsstrahlung cross section for a "run-time" mixture of elements.

BREMRM is the bremsstrahlung cross section, integrated over some energy range, for a mixture of elements.

BRMSTM is the soft bremsstrahlung total cross section for a mixture of elements.

PAIRRR is the pair production cross section, integrated over some energy range, for a "run-time" mixture of elements.

PAIRTZ is the total cross section for pair production for an element.

This method of naming is not strictly adhered to, however. For example, SPIONE is the ionization stopping power for an electron, PBR1 and PBR2 are positron branching ratios, and GMFP is the gamma-ray mean free path.
A3.2-11

Table A3.2.1
SUBROUTINES Used In PEGS

NAME	DESCRIPTION	PAGES
DIFFER	Determines the various parameters needed for bremsstrahlung and pair production energy sampling.	A3.2-4, A3.3-11
EFUNS	Subprogram to compute electron functions to be fit in a way that avoids repetition.	A3.2-4
GFUNS	Subprogram to compute photon functions to be fit in a way that avoids repetition.	A3.2-4
HPLT1	Creates line printer plot comparisons of EGS-sampled data (UCTESTSR User Code) and theoretical functions of PEGS.	A3.2-4, A3.3-20,21
LAY	Subprogram to produce a deck of material dependent data (for sub- sequent use by EGS).	A3.2-4, A3.3-17
MIX	Computes Z-dependent paramaters that reside in COMMON/MOLVAR/.	A3.2-4, A3.3-11
MOLIER	Computes material independent multiple scattering data (EGS2 only!).	A3.2-4
PLOT	Subprogram to plot a given function (referenced by number).	A3.2-4
PMDCON	Determines the physical, mathematical, & derived constants in a mnemonic way.	A3.2-4
PWLF1	Subprogram to piecewise linearly fit up to 10 functions simultaneously on an interval (XL,XU).	A3.2-4,19, A3.3-14
RFUNS	Subprogram to compute Rayleigh scattering functions to be fit in a way that avoids repetition.	A3.2-4
SPINIT	Initializes stopping power functions for a particular medium.	A3.2-4, A3.3-11

A3.2-12

Table A3.2.2

FUNCTIONS Used In PEGS

NAME	DESCRIPTION	PAGES
AFFACT	Atomic form factor (squared) for an element or mixture of elements.	
AINTP	Linear or log interpolation function.	A3.2-4,9
ALKE	Log of kinetic energy (ALOG(E-RM)), used as a cumulative distribution function for fits and plots.	A3.2-5, A3.3-20
ALKEI	Inverse of ALKE (=EXP(X)+RM).	A3.2-5
ALIN	Linear cumulative distribution function for plots (ALIN(X)=X).	A3.2-5, A3.3-20
ALINI	Inverse of ALIN (=same as ALIN). Used as inverse cumulative distribution function in plots.	A3.2-5
ADFMOL	Approximate cumulative distribution function for Moller and Bhabha cross sections (ADFMOL(E)=-1/(E-RM)).	A3.2-5, A3.3-20
ADIMOL	Inverse of ADFMOL.	A3.2-5
ADDMOL	Derivative of ADFMOL.	A3.2-5
AMOLDM	Moller differential cross section for a mixture of elements.	A3.2-5,11,14

(continued on next page)

A3.2-13

Table A3.2.2
(continued)

FUNCTIONS Used In PEGS

NAME	DESCRIPTION	PAGES
AMOLFM	"One argument" form of AMOLDM.	A3.2-5
AMOLRM	Moller cross section, integrated over some energy range, for a mixture of elements.	A3.2-5
AMOLTM	Moller total cross section for a mixture of elements.	A3.2-4,5
ANIHDM	Annihilation differential cross section for a mixture of elements.	A3.2-5
ANIHFM	"One argument" form of ANIHDM.	A3.2-5
ANIHRM	Annihilation cross section, integrated over some energy range, for a mixture of elements.	A3.2-5
ANIHTM	Annihilation total cross section for a mixture of elements.	A3.2-4,5
APRIM	Empirical correction factor in bremsstrahlung cross section.	A3.2-5,7
AREC	Reciprocal function (=derivative of ALOG(X)). Used as probability density function in log plots (AREC(X)=1/X).	A3.2-5
BHABDM	Bhabha differential cross section for a mixture of elements.	A3.2-5
BHABFM	"One argument" form of BHABDM.	A3.2-5

(continued on next page)

Table A3.2.2
(continued)

FUNCTIONS Used In PEGS

NAME	DESCRIPTION	PAGES
BHABRM	Bhabha cross section, integrated over some energy range, for a mixture of elements.	A3.2-5
BHABTM	Bhabha total cross section for a mixture of elements.	A3.2-4,5
BREMDR	Bremsstrahlung differential cross section for a "run-time" mixture of elements.	A3.2,5,8,11
BREMFR	"One argument" form of BREMDR.	A3.2-5,8
BREMDZ	Bremsstrahlung differential cross section for an element.	A3.2-5,7
BREMFZ	"One argument" form of BREMDZ.	A3.2-5,7
BREMRM	Bremsstrahlung cross section, integrated over some energy range, for a mixture of elements.	A3.2-5,7,11
BREMRR	Bremsstrahlung cross section, integrated over some energy range, for a "run-time" mixture of elements.	A3.2-5,8
BREMRZ	Bremsstrahlung cross section, integrated over some energy range, for an element.	A3.2-5,7
BREMTM	Bremsstrahlung total cross section for a mixture of elements.	A3.2-4,5,7
BREMTR	Bremsstrahlung total cross section for a "run-time" mixture of elements.	A3.2-5,8

(continued on next page)

A3.2-15

Table A3.2.2
(continued)
FUNCTIONS Used In PEGS

NAME	DESCRIPTION	PAGES
BRMSDZ	Soft bremsstrahlung differential cross section for an element.	A3.2-5,7
BRMSFZ	"One argument" form of BRMSDZ.	A3.2-5,7
BRMSRM	Soft bremsstrahlung cross section, integrated over some energy range, for a mixture of elements.	A3.2-5,7
BRMSRZ	Soft brems cross section integrated over an energy range,for an element.	A3.2-5,7, A3.3-19,20
BRMSTM	Soft bremsstrahlung total cross section for a mixture of elements.	A3.2-5,7,11
COHERTM	Coherent (Rayleigh) total cross section for a mixture of elements.	A3.2-4,5
COHETZ	Coherent (Rayleigh) total cross section for an element.	A3.2-5
COMPDM	Compton differential cross section for a mixture of elements.	A3.2-5
COMPFM	"One argument" form for COMPDM.	A3.2-5
COMPRM	Compton cross section, integrated over an energy range, for a mixture of elements.	A3.2-5
COMPTM	Compton total cross section for a mixture of elements.	A3.2-4,5
CRATIO	Coherent (Rayleigh) cross section ratio	A3.2-5
DCADRE	Quadrature routine to integrate f(x) from a->b by cautious Romberg extrapolation.	A3.2-7-10,19
EBIND	Function to get an average photo-electric binding energy.	A3.2-4,5

(continued on next page)

A3.2-16

Table A3.2.2
(continued)

FUNCTIONS Used In PEGS

NAME	DESCRIPTION	PAGES
EBR1	Function to determine the electron(-) branching ratio (Brem/Total).	A3.2-5
EDEDX	Evaluates SPTOTE with cutoff energies of AE and AP.	A3.2-5
ESIG	Determines the total electron(-) interaction cross section (probability per radiation length).	A3.2-5
FCOULC	Coulomb correction term in pair production and bremsstrahlung cross sections.	A3.2-5,7,9
FI	Function multiplexer.	A3.2-1,5,6
GBR1	Function to determine the gamma-ray branching ratio (Pair/Total).	A3.2-5
GBR2	Function to determine the gamma-ray branching ratio ((Pair+Compton)/Total).	A3.2-5
GMFP	Function to determine the gamma-ray mean free path.	A3.2-5,11, A3.3-18,19
IFUNT	Given PEGS function name, it looks it up name in table and returns the function index. Used by options that specify functions by name.	
PAIRDR	Pair production differential cross section for a "run-time" mixture of elements.	A3.2-5,10,18
PAIRDZ	Pair production differential cross section for an element.	A3.2-5,9,18

(continued on next page)

A3.2-17

Table A3.2.2
(continued)
FUNCTIONS Used In PEGS

NAME	DESCRIPTION	PAGES
PAIRFR	"One argument" form of PAIRDR.	A3.2-5,10
PAIRFZ	"One argument" form of PAIRDZ.	A3.2-5,10
PAIRRM	Pair production cross section, integrated over some energy range, for a mixture of elements.	A3.2-5,9
PAIRRR	Pair production cross section, integrated over some energy range, for a "run-time" mixture of elements.	A3.2-5,10,11
PAIRRZ	Pair production cross section, integrated over energy range, for element.	A3.2-5,9
PAIRTE	"Empirical" total pair production production cross section for a mixture (=SUM(PZ(I)*PAIRTZ(Z(I))).	A3.2-5,9
PAIRTM	Pair production total cross section for a mixture of elements, obtained by numerical integration of differential cross section.	A3.2-5,9
PAIRTR	Pair production total cross section for a "run-time" mixture of elements.	A3.2-5,10
PAIRTU	Pair production total cross section actually "used". Same as PAIRTE for primary energy less than 50 MeV; otherwise, same as PAIRTM.	A3.2-4,5,9
PAIRTZ	Computes contribution to empirical pair production total cross section for an element assuming one atom per molecule. It is obtained by log-linear interpolation of Israel-Storm data.	A3.2-5,9,11
PBR1	Function to determine the positron branching ratio (Brem/Total).	A3.2-5,11

(continued on next page)

Table A3.2.2
(continued)
FUNCTIONS Used In PEGS

NAME	DESCRIPTION	PAGES
PBR2	Function to determine the positron branching ratio ((Brem+Bhabha)/Total).	A3.2-5,11
PDEDX	Evaluates SPTOTP with cutoff energies of AE and AP.	A3.2-5
PHOTTE	Determines the proper mix of PHOTTZ's for a mixture.	A3.2-4,5
PHOTTZ	Determines the interpolated total photoelectric cross section from tabulated data.	A3.2-5
PSIG	Determines the total positron interaction cross section (probability per radiation length).	A3.2-5
QD	Driver function for DCADRE, the numerical integration routine.	A3.2-7-10
QFIT	Utility logical function for the piecewise linear fit subroutine, PWLF1. It returns .TRUE. if a given partition gives a good fit.	A3.2-4, A3.3-14,15
SPIONB	Does the work for SPIONE and SPIONP. One argument tells whether to compute stopping power for electron or positron.	A3.2-7
SPIONE	Calculates the stopping power due to ionization for electrons(-).	A3.2-5,7
SPIONP	Calculates the stopping power due to ionization for positrons.	A3.2-5,7
SPTOTE	Calculates the total stopping power (ionization plus soft bremsstrahlung) for electrons(-) for specified cutoffs.	A3.2-4,5,7

(continued on next page)

A3.2-19

Table A3.2.2
(continued)

FUNCTIONS Used In PEGS

NAME	DESCRIPTION	PAGES
SPTOTP	Calculates the total stopping power (ionization plus soft bremsstrahlung) for positrons for specified cutoffs.	A3.2-4,5,7,19
TMXB	Determines the maximum total step length consistent with Bethe's criterion.	A3.2-5
TMXS	Determines the minimum of TMXB and 10 radiation lengths.	A3.2-5
TMXDE2	Included for possible future modification purposes (=TMXB/(E**2*BETA**4)). It might be easier to fit this quantity than to fit TMXB and then apply the the denominator in EGS at run-time.	A3.2-5
XSIF	Function to account for bremsstrahlung and pair production in the field of the atomic electrons.	A3.2-5,7,9
ZTBL	Given the atomic symbol for an element, it returns the atomic number.	

A3.3 PEGS Options and Input Specifications

A3.3.1 Interrelations Between Options

Fig. A3.3.1 illustrates the logical relationship between options of PEGS. For example, in order to be able to use the PLTN option, one of the material specification options (ELEM, MIXT, COMP) must have already been processed. The PWLF option requires that both the ENER option and one of the material specification options precede it. To use the DECK option, it is sufficient to have validly invoked the PWLF option. The STOP and MIMS options are seen to be independent of the others.

In the following sections, for each option we will give its function, parameters which control it, the format of cards needed to invoke it, and an explanation of the routines (if any) that are used to implement it. The cards for a given option are named with the first part of their name being the option name, and the last part the card number. For example, MIXT2 is the name of the second card needed for the MIXT option. The information is summarized in Table A3.3.1. It should be noted that IBM and CDC require different formats for NAMELIST data. Also, the single card referred to as being read by NAMELIST may in fact be several cards, provided that the proper convention for continuing NAMELIST cards is followed. Once the first card (indicating the option) has been read in, however, the second card (i.e., NAMELIST/INP/) must follow (see examples at the end of Section A3.3.2). We will use the IBM form of NAMELIST in our examples.

A3.3.2 The ELEM, MIXT, COMP Options

The purpose of the ELEMent, MIXTure, and COMPound options is to specify the material used by the PEGS functions. The parameters needed to specify a material are its density (RHO), the number of different kinds of atoms (NE), and, for each different kind of atom, its atomic number (Z(I)), its atomic weight (WA(I)), and its proportion either by number (PZ(I)) for a compound or by weight (RHOZ(I)) for a mixture. PEGS has tables for the atomic symbol (ASYMT(1:100)) and the atomic weight (WATBL(1:100)) for elements I=1 through I=100, so the type of atom is specified by giving its atomic symbol (ASYM(I)). PEGS also has a table of the densities of the elements (RHOTBL(1:100)). Rayleigh (coherent) scattering data will be appended to the normal output data if the IRAYL flag is set to 1 in NAMELIST/INP/.

```
  +------+    +------+    +------+          +------+
  |:ELEM:|    |:MIXT:|    |:COMP:|          |:ENER:|
  +------+    +------+    +------+          +------+
      |          |           |                  |
      |          |           |                  |
      |          |           |                  |
      |          V           |                  |
      |                      |                  |
      + ------> OR <-------- +                  |
                                                |
                 |                              V
                 |
                 +---------------------------> AND ----+
                 |                                     |
                 |                              |      |
                 V                              |      |
   + -------+----+----+------- +                |      |
   |        |         |        |                |      |
   |        |         |        |                |      |
   V        V         V        V                |      |
+------+ +------+ +------+ +------+ +------+    |      |
|:PLTN:| |:PLTI:| |:HPLT:| |:CALL:| |:TEST:|<---+      |
+------+ +------+ +------+ +------+ +------+           |
                                                       |
                                                       V
                                                    +------+
                                                    |:PWLF:|
+------+                                            +------+
|:MIMS:|                                               |
+------+                                               |
                                                       V
+------+                                            +------+
|:STOP:|                                            |:DECK:|
+------+                                            +------+
```

Fig. A3.3.1 Logical Relationship Between the Options of PEGS

A3.3-2 APPENDIX 3

Table A3.3.1
PEGS Control Cards

CARD	FORMAT	VARIABLES READ	COMMENTS
ELEM1	(4A1)	OPT(1:4)	'ELEM'. Means "select material that is an element."
ELEM2	NAMELIST/INP/	RHO	Optional. If given, this over-rides the PEGS default density (g/cm**3) for the element.
		WA(1)	Optional. Atomic weight of element. If given, this over-rides the PEGS default.
		IRAYL	Optional. Set to unity to included Rayleigh output.
		IUNRST	Optional. Set to unity for unrestricted collision stopping power.
		ISSB	Optional. Set to unity to use own density effect parameters (see text below).
		GASP	Optional. See MIXT or COMP
ELEM3	(24A1, 6X,24A1)	MEDIUM(1:24)	Identifier assigned to data set to be produced.
		IDSTRN(1:24)	Optional. Identifer of medium name under which desired Sternheimer-Seltzer-Berger coeffcients are given in PEGS. If not specified, identifier in MEDIUM(1:24) is used.
ELEM4	(24(A2,1X))	ASYM(1)	Atomic symbol for element.
COMP1	(4A1)	OPT(1:4)	'COMP'. Means "select material that is a compound."
COMP2	NAMELIST/INP/	NE	Number of elements in compound.
		RHO	Density (g/cm**3) of compound (at NTP for gases).

A3.3-3 (continued on next page)

Table A3.3.1
(continued)
PEGS Control Cards

CARD	FORMAT	VARIABLES READ	COMMENTS
		(PZ(I),I=1,NE)	Relative numbers of atoms in compound.
		GASP	Optional. Defines state of compound: zero (default) for solid or liquid, otherwise value gives gas pressure (atm).
		(WA(I),I=1,NE)	Optional. May be used to over-ride default atomic weights (e.g., to allow for special isotopes).
		IRAYL	Same as for ELEM2.
		IUNRST	Same as for ELEM2.
		ISSB	Same as for ELEM2.
COMP3	(24A1, 6X,24A1)	MEDIUM,IDSTRN	Same as for ELEM3.
COMP4	(24(A2, 1X))	(ASYM(I),I=1,NE)	Atomic symbols for the atoms in the compound. Duplicates are allowed if several iso-topes of the same element are present, or may be required for diatomic molecules (e.g. nitrogen gas).
MIXT1	(4A1)	OPT(1:4)	'MIXT'. Means "select mat-erial that is a mixture."
MIXT2	NAMELIST/INP/	NE	Number of elements in mixture.
		RHO	Density (g/cm**3) of mixture (at NTP for gases).
		(RHOZ(I),I=1,NE)	Relative amount of atom in mixture (by weight).

(continued on next page)

A3.3-4

Table A3.3.1
(continued)
PEGS Control Cards

CARD	FORMAT	VARIABLES READ	COMMENTS
		GASP	Optional. Defines state of mixture: zero (default) for solid or liquid, otherwise value gives gas pressure (atm).
		(WA(I),I=1,NE)	Optional. May be used to over-ride default atomic weights.
		IRAYL	Optional. Set to unity to included Rayleigh output.
		IUNRST	Same as for ELEM2.
		ISSB	Same as for ELEM2.
MIXT3	(24A1, 6X,24A1)	MEDIUM,IDSTRN	Same as for ELEM3.
MIXT4	(24(A2, 1X))	(ASYM(I),I=1,NE)	Same as for COMP4.
ENER1	(4A1)	OPT(1:4)	'ENER' => "select energy limits."
ENER2	NAMELIST/INP/	AE	Lower cutoff energy (total) for charged particle transport (MeV).
		UE	Upper limit energy (total) for charged particle transport (MeV).
		AP	Lower cutoff energy for photon transport (MeV).
		UP	Upper limit energy for photon transport (MeV).

Note: If the user supplies negative values for the energy limits above, the absolute values given will be interpreted as in units of the electron rest mass energy. Thus, AE=-1 is equivalent to AE=0.511 MeV.

(continued on next page)

A3.3-5

Table A3.3.1
(continued)

PEGS Control Cards

CARD	FORMAT	VARIABLES READ	COMMENTS
PWLF1	(4A1)	OPT(1:4)	'PWLF'. Means "select piecewise linear fit."
PWLF2	NAMELIST/INP/	Note:	The following PWLF parmeters (see Section A3.3.4) are optional and may be over-ridden by the user. The default values (in BLOCK DATA) are indicated below.
		EPE/0.01/	Electron EP parameter.
		EPG/0.01/	Gamma EP parameter.
		ZTHRE(1:8)/8*0./	Electron ZTHR parameter.
		ZTHRG(1:3)/0.,.1,0./	Gamma ZTHR parameter.
		ZEPE(1:8)/8*0./	Electron ZEP parameter.
		ZEPG(1:3)/0.,.01,0./	Gamma ZEP parameter.
		NIPE/20/	Electron NIP parameter.
		NIPG/20/	Gamma NIP parameter.
		NALE/$MXEKE/	Electron NIMX parameter.
		NALG/$MXGE/	Gamma NIMX parameter.
DECK1	(4A1)	OPT(1:4)	'DECK'. Means "write fit data and other useful parameters."
DECK2	NAMELIST/INP/		No parameters.

(continued on next page)

Table A3.3.1
(continued)

PEGS Control Cards

CARD	FORMAT	VARIABLES READ	COMMENTS
MIMS1	(4A1)	OPT(1:4)	'MIMS'. Means "Calculate Material Independent Multiple Scattering Data" (for EGS2 only).
MIMS2	NAMELIST/INP/		MIMS is controlled by macro settings and by data in BLOCK DATA that is not accessible to the NAMELIST.
TEST1	(4A1)	OPT(1:4)	'TEST'. Means "Plot the fitted functions."
TEST2	NAMELIST/INP/	NPTS	Optional. Number of points to plot per function (Default=50).
CALL1	(4A1)	OPT(1:4)	'CALL'. Means "Call the designated function and print value."
CALL2	NAMELIST/INP/	XP(1:4)	Values for up to four arguments of the function.
CALL3	(6A1)	NAME(1:6)	Name of function to be evaluated.
PLTI1	(4A1)	OPT(1:4)	'PLTI'. Means "Plot function given its index and the index of the distribution function."
PLTI2	NAMELIST/INP/	IFUN	The index of the function to be plotted.
		XP(1:4)	Values for the static arguments (parameters).

(continued on next page)

Table A3.3.1
(continued)

PEGS Control Cards

CARD	FORMAT	VARIABLES READ	COMMENTS
		IV	Variable telling which argument is to be varied (e.g., IV=2 means plot function vs. its second argument).
		VLO	Lower limit for argument being varied.
		VHI	Upper limit for argument being varied.
		NPTS	Number of points to plot.
		IDF	Index of distribution function used to select independent variable.
PLTN1	(4A1)	OPT(1:4)	'PLTN'. Means "Plot the named function."
PLTN2	NAMELIST/INP/	XP(1:4),IV, VLO,VHI,NPTS, IDF,MP	Same as PLTI2.
PLTN3	(2(6A1))	NAME(1:6)	Name (6 characters) of function to be plotted.
		IDFNAM(1:6)	Name of distribution function to be used.

(continued on next page)

Table A3.3.1
(continued)
PEGS Control Cards

CARD	FORMAT	VARIABLES READ	COMMENTS
HPLT1	(4A1)	OPT(1:4)	'HPLT'. Means "Plot histogram to compare the sampled spectrum with the range-integrated & the differential theoretical values."
HPLT2	NAMELIST/INP/	EI	Test particle total energy (MeV).
		ISUB	Variable telling which function is being tested: 1=PAIR 2=COMPT 3=BREMS 4=MOLLER 5=BHABHA 6=ANNIH
HPLT3	(' TEST DATA FOR ROUTINE=',12A1,',#SAMPLES=', I10,',NBINS=',I5)		
		NAMESB(1:12)	Name of subroutine tested.
		NTIMES	Number of samples.
		NBINS	Number of histogram bins.
HPLT4	(' IQI=',I2,',RNLO,RNHI=',2F12.8,',IRNFLG=',I2)		
		IQI	Charge of test particle.
		RNLO,RNHI	Lower and upper limits to random number preceding call to test function.
		IRNFLG	Non-zero means to "apply above limits to preceding random number to test for correlation." Zero value means "don't do this."
HPLT5...etc.	(9I8)	NH(1:NBINS)	Sampled data (from User Code UCTESTSR).

A3.3-9

(text continued from page A3.3-1)

The ELEMent option is used if the material in question has only one type of atom. In this case PEGS knows that NE=1, has the density in a table, sets PZ(1)=1, and deduces Z(1) and WA(1) from ASYM(1). Thus the atomic symbol (ASYM(1)) is the only information that the user need supply. Before each option, RHO and the WA(I) are saved and then cleared so that it can be determined whether these have been set by the user. If so, they over-ride the table values in PEGS. This allows the different atoms to be non-standard isotopes and/or allows the overall density to be adjusted to the experimental state. For options other than ELEM, MIXT, or COMP, RHO and WA(I) are restored after reading the NAMELIST.

The COMPound option is used when there is more than one different kind of atom and it is desired to give the proportions by relative number of atoms (PZ(I)). The only required data is NE, ASYM(I), and PZ(I) (for I=1,NE). Optionally, any of the WA(I) can be over-ridden.

The MIXTure option is similar to the COMPound option except that the relative atomic proportions are given by weight (RHOZ(I)) rather than by number.

When the PZ(I) values have been specified, PEGS obtains the RHOZ(I) using RHOZ(I)=PZ(I)*WA(I); otherwise, the PZ(I) are obtained from PZ(I)=RHOZ(I)/WA(I). The absolute normalization of the PZ(I) and RHOZ(I) values is not important because of the way the quantities are used. For example, the macroscopic cross sections contain factors like

```
PZ(I)/SUM(PZ(I)*WA(I))
```

where the denominator is the "molecular weight".

In addition to physically specifying the material being used, a name for it must be supplied (MEDIUM(1:24)) for identification purposes. This name is included in the output deck when the DECK option is selected. The name can be different for any two data sets that are created, even though the same material has been used. For example, one might produce PEGS output using a particular material but different energy limits (or fit tolerances, density effect parameters, etc.), with separate identification names for each (e.g., FE1, FE2, etc.).

A3.3-10

The quantity IDSTRN(1:24) is used to identify the Sternheimer-Seltzer-Berger density effect parameters that are tabulated in BLOCK DATA (see Table 2.13.2 of SLAC-265 for complete list of identifer names). If IDSTRN(1) is blank, then IDSTRN is given the same value as MEDIUM. If this name is not identifiable with any of those in BLOCK DATA, the Sternheimer density effect scheme is replaced with a general formula by Sternheimer and Peierls. Although not recommended, setting IUNRST to unity causes the unrestricted collision stopping power to be used (instead of the sum of the restricted and radiative stopping powers). An option is available to allow users to supply their own values for the various parameters used to calculate the density effect correction (see Section 2.13 of SLAC-265 for a discussion). To initiate this option, all six parameters (AFACT, SK, X0, X1, CBAR, and IEV)) must be read in the NAMELIST input in ELEM, MIXT, or COMP, and ISSB must be set to non-zero as a flag. Note that if one only wants to override IEV, one must still input all six parameters (see Table 2.13.2 of SLAC-265).

After reading the input data for these options, subroutine MIX is called in order to compute the Z-related parameters that reside in COMMON/MOLVAR/, subroutine SPINIT is called to initialize the stopping power routines for this material, and subroutine DIFFER is called to compute run-time parameters for the pair production and bremsstrahlung sampling routines. The reader might find the comments in subroutine MIX useful.

The following are examples of sets of data cards that can be used with the ELEM, MIXT, and COMP options:

```
(Note: The NAMELIST data (i.e., &INP...&END) starts in column 2).
-----------------------------------------------------------------
  A. Material - Element is Iron with defaults taken.
-----------------------------------------------------------------
                               Column
Card    1234567891123456789212345678931234567894123456789..etc.

ELEM1   ELEM
ELEM2    &INP &END
ELEM3   IRON                         FE
ELEM4   FE
```

A3.3-11

```
 -----------------------------------------------------------------
   B. Material - He-3 with density & atomic weight over-ridden by user.
 -----------------------------------------------------------------
                                Column
 Card    1234567891123456789212345678931234567894123456789..etc.
 ELEM1   ELEM
 ELEM2    &INP RHO=1.E-2,WA(1)=3 &END
 ELEM3   HELIUM-3                    HE
 ELEM4   HE
 -----------------------------------------------------------------

   C. Material - Compound is sodium iodide with
                 IDSTRN(1:24) defaulting to MEDIUM(1:24).
 -----------------------------------------------------------------
                                Column
 Card    1234567891123456789212345678931234567894123456789..etc.
 COMP1   COMP
 COMP2    &INP NE=2,RHO=3.667,PZ(1)=1,PZ(2)=1 &END
 COMP3   NAI
 COMP4   NA I
 -----------------------------------------------------------------

   D. Material - Compound is polystyrene scintillator (e.g., PILOT-B
               or NE-102A) with data taken from: "Particle Properties
               Data Booklet, April 1982" (Physics Letters 111B, April
               1982).  Sternheimer-Peierls default.
 -----------------------------------------------------------------
                                Column
 Card    1234567891123456789212345678931234567894123456789..etc.
 COMP1   COMP
 COMP2    &INP NE=2,RHO=1.032,PZ(1)=1,PZ(2)=1.1 &END
 COMP3   POLYSTYRENE SCINTILLATOR
 COMP4   C  H
 -----------------------------------------------------------------

   E. Material - Mixture is lead glass, consisting of five
                 specified elements (and 1 per cent of the trace
                 elements unspecified).  Sternheimer-Peierls default.
 -----------------------------------------------------------------
                                Column
 Card    1234567891123456789212345678931234567894123456789..etc.
 MIXT1   MIXT
 MIXT2    &INP NE=5,RHO=3.61,RHOZ=41.8,21.0,29.0,5.0,2.2 &END
 MIXT3   LEAD GLASS
 MIXT4   PB SI O  K  NA

 A3.3-12                                           APPENDIX 3
```

```
-----------------------------------------------------------------
  F. Material - Mixture is U-235, U-238, and carbon (not a real
                material).  Sternheimer-Peierls default.
-----------------------------------------------------------------
                              Column
Card    123456789112345678921234567893123456789412345678..etc.

MIXT1   MIXT
MIXT2    &INP NE=3,RHO=16,WA=235,238,RHOZ=50,30,10 &END
MIXT3   JUNK
MIXT4   U  U  C


-----------------------
A3.3.3  The ENER Option
-----------------------
```

The ENERgy option is used to define the electron and photon energy intervals over which it is desired to transport particles, and hence, over which fits to total cross sections and branching ratios must be made. The electron energy interval is (AE,UE) and the photon interval is (AP,UP). If any of these is entered negative, it is multiplied by -RM=-0.511 MeV; that is, the absolute magnitude is assumed to be the energy in units of the electron rest mass energy. The quantities TE=AE-RM, TET2=2*TE, and TEM=TE/RM, as well as the bremsstrahlung and Moller thresholds (RM+AP and AE+TM, respectively), are then computed and printed out.

The following are examples of sets of data cards that can be used with the ENER option:

(Note: The NAMELIST data (i.e., &INP...&END) starts in column 2).

```
-----------------------------------------------------------------
  A. Electron and photon cutoff energies are 1.5 MeV and
     10 keV, respectively.  The upper energy limit for both
     is set at 100 GeV.   (Note: All energies are in MeV and
     are total energies).
-----------------------------------------------------------------
                              Column
Card    123456789112345678921234567893123456789412345678..etc.
ENER1   ENER
ENER2    &INP AE=1.5,UE=100000.,AP=0.01,UP=100000. &END
```

A3.3-13

```
-----------------------------------------------------------------
  B. Same as above, except AE=3*RM.
-----------------------------------------------------------------
                               Column
Card    12345678911234567892123456789312345678941234567 8..etc.
ENER1   ENER
ENER2    &INP AE=-3,UE=100000.,AP=0.01,UP=100000. &END
```

```
-----------------------
A3.3.4  The PWLF Option
-----------------------
```

The PieceWise Linear Fit option performs a simultaneous piecewise linear (vs. ln(E-RM)) fit of eight electron functions over the energy interval (AE,UE) and a simultaneous piecewise linear (vs. ln E) fit of three or four photon functions over the energy interval (AP,UP). Each simultaneous fit over several functions is accomplished by a single call to subroutine PWLF1---once for the electrons and once for the photons.

By simultaneous fit we mean that the same energy subintervals are used for all of the functions of a set. Alternately, we could describe it as fitting a vector function. The PWLF1 subroutine is an executive routine that calls the function QFIT. Function QFIT, which does most of the work, tries to perform a fit to the vector function by doing a linear fit with a given number of subintervals. It returns the value .TRUE. if the fit satisfies all tolerances and .FALSE. otherwise. Subroutine PWLF1 starts out doubling the number of subintervals until a successful fit is found. Additional calls to QFIT are then made to determine the minimum number of subintervals needed to give a good fit. Sometimes, because of discontinuities in the functions being fitted, a fit satisfying the specified tolerances cannot be obtained within the constraints of the number of subintervals allowed by the array sizes of EGS. When this happens, PEGS prints out the warning message (for example):

```
  NUMBER OF ALLOCATED INTERVALS(=  150) WAS INSUFFICIENT
  TO GET MAXIMUM RELATIVE ERROR LESS THAN       0.01
```

Even in this case a fit is produced which is sufficient most of the time.

A3.3-14

Let NFUN be the number of components to the vector function F(IFUN,E(J)) (where IFUN=1,NFUN), and let E(J) be a sequence of points (J=1,NI) covering the interval being fitted. The number of points (NI) is about ten times the number of fit intervals (NINT) in order that the fit will be well tested in the interiors of the intervals. If FEXACT(IFUN,J) and FFIT(IFUN,J) are the exact and fitted values of the IFUN-th component at E(J), then the logical function QFIT may be given as follows:

```
  LOGICAL FUNCTION QFIT(NINT);
  COMMON.......etc.
  QFIT=.TRUE.;
  REM=0.0;  "RELATIVE ERROR MAXIMUM"
  NI=10*NINT;
  DO J=1,NI [
   DO IFUN=1,NFUN [
    AER=ABS(FEXACT(IFUN,J)-FFIT(IFUN,J));
    AF=ABS(FEXACT(IFUN,J));
    IF(AF.GE.ZTHR(IFUN))[IF(AF.NE.0.0) REM=AMAX1(REM,AER/AF);]
    ELSE [IF(AER.GT.ZEP(IFUN)) QFIT=.FALSE.;]
   ]
  ]
  QFIT=QFIT.AND.REM.LE.EP;
  RETURN;  END;
```

Thus we see that EP is the largest allowed relative error for those points where the absolute computed value is above ZTHR(IFUN), and ZEP(IFUN) is the largest allowed absolute error for those points where the absolute computed value is less than ZTHR(IFUN).

Other features of the QFIT routine include provisions for aligning a subinterval boundary at a specified point in the overall interval (in case the fitted function has a discontinuous slope such as at the pair production or Moller thresholds), and computation of fit parameters in bins flanking the main interval to guard against truncation errors in subinterval index computations.

The net result of the fit is to obtain cofficients AX, BX, AF(IFUN,J), and BF(IFUN,J) such that

```
  FVALUE(E)=AF(IFUN,INTERV)*XFUN(E) + BF(IFUN,INTERV)
```

is the value of the IFUN-th function, and where

```
  INTERV=INT(AX*XFUN(E) + BX).
```

XFUN is called the distribution function and is ln(E-RM) for electrons and ln(E) for photons.

The coding of EGS and its original $EVALUATE macros are designed to allow a "mapped PWLF" in which we have AX, BX, AF(IFUN,J), BF(IFUN,J), and M(I), such that when

```
  I=INT(AX*XFUN(E)+BX)
```

and

```
  J=M(I),
```

then

```
  FVALUE(E)=AF(IFUN,J)*XFUN(E) + BF(IFUN,J)
```

for the IFUN-th function. This kind of fit has the advantage that it could get a better fit with a smaller amount of stored data. However, this fitting scheme has never been implemented in PEGS. With the present scheme more data than necessary is used in describing the functions at the higher energies where they vary quite smoothly.

The following is an example of the data cards that can be used with the PWLF option:

```
                               Column
Card    123456789112345678921234567893123456789412345678..etc.

PWLF1   PWLF
PWLF2    &INP &END
```

```
-----------------------
A3.3.5  The DECK Option
-----------------------
```

The DECK option, with the aid of subroutine LAY, prints and punches the data needed to specify the current material, the energy intervals specified, various computed molecular parameters (e.g., the radiation length), the run-time parameters for pair production and bremsstrahlung, and the fit data produced by the PWLF option. In other words, DECK prints and punches anything that might be of use to EGS in simulating showers, or to the user in analysis routines. The macros ECHOREAD and ECHOWRITE have been written to give nicely captioned print-outs of data (read or written) and to eliminate the need for creating separate write statements to echo the values.

Subroutines LAY (in PEGS) and HATCH (in EGS) are a matched pair in that HATCH reads what LAY writes (PEGS "lays" and EGS "hatches"). Thus, if the users would like to get more information at EGS run-time, they need only modify LAY and HATCH accordingly.

DECK should be invoked when either ELEM, MIXT, or COMP and ENER and PWLF have been run for the current material and before any of these have been executed for the next material (see Fig. 3.3.1).

The following is an example of the data cards that can be used with the DECK option:

```
                              Column
Card    12345678911234567892123456789312345678941234567 8..etc.
DECK1   DECK
DECK2    &INP &END
```

```
-------------------------------------------
A3.3.6  The MIMS Option and the EGSCMS Code
-------------------------------------------
```

The MIMS option produces a data set for EGS (Version 2 only) that contains Material Independent Multiple Scattering data. A stand alone program, EGSCMS (EGS Continuous Multiple Scattering), performs an analogous function for EGS3/EGS4, except that the data is held in a BLOCK DATA subprogram in the EGS code itself rather than in an external data set.

```
     In the sense that these data sets are produced already,
the MIMS option and EGSCMS code are now dispensible.  However,
as documentation for the origin of the present data, they are
valuable and they will be maintained with the EGS Code System.
(Note: EGSCMS was originally written in Mortran2 and will
remain that way since it is not expected to be used again).

-----------------------
A3.3.7  The TEST Option
-----------------------

     The TEST option is used as an easy way to obtain plots
of all the functions (not the fits) that the PWLF option fits.
These plots are valuable in getting a feel for the magnitudes
and variations of the functions to be fitted.


     The following is an example of the data cards that can be
used with the TEST option:

                               Column
Card    12345678911234567892123456789312345678941234567 8..etc.

TEST1   TEST
TEST2    &INP NPTS=50 &END


-----------------------
A3.3.8  The CALL Option
-----------------------

     The CALL option is used whenever one desires to have
PEGS evaluate a particular function and print out the results.


     The following is an example of the data cards that can be
used with the CALL option in order to test for discontinuities
in GMFP (Gamma Mean Free Path) near 50 MeV.  (Note: In this
example we have included the (necessary) ELEM option cards
for Lead).




A3.3-18
```

```
                              Column
Card    123456789112345678921234567893123456789412345678..etc.

ELEM1   ELEM
ELEM2    &INP &END
ELEM3   PB
ELEM4   PB
CALL1   CALL
CALL2    &INP XP(1)=49.99 &END
CALL3   GMFP
CALL1   CALL
CALL2    &INP XP(1)=50.01 &END
CALL3   GMFP
```

The resulting output from PEGS is:

```
OPT=CALL
FUNCTION CALL:     1.95522     = GMFP   OF     49.9900
OPT=CALL
FUNCTION CALL:     1.97485     = GMFP   OF     50.0100
```

(Note: This calculation was done on the IBM-3081 computer).

A3.3.9 The PLTI and PLTN Options

The PLTI and PLTN options may be used to obtain printer---and with some work, possibly graphic---plots of any of the functions in the PEGS function table. The PLTI option is rather primitive in that the functions involved must be specified by number, so we shall instead concentrate on the PLTN option in which the functions are specified by name.

Consider the function BRMSRZ(Z,E,K1,K2) which is the soft bremsstrahlung cross section (for an electron of total energy energy E and element Z) integrated over the photon energy range (K1,K2). Suppose we would like to see a plot of BRMSRZ(2,E,0.0,1.5) for values of E from 5 to 100 MeV. Also assume we want the data points evenly spaced in ln(E). Then (see Table 3.3.1) the function name is ’BRMSRZ’, the distribution function name is IDNAM=’ALOG’, the static arguments are

```
XP(1)=2., XP(3)=0.0, XP(4)=1.5, the independent variable is
the second argument (i.e., IV=2), and its limits are VLO=5.0
and VHI=100.0.  If we want 100 points on the plot we let
NPTS=100.

The cards necessary to accomplish this plot are:

                              Column
Card    12345678911234567892123456789312345678941234567 8..etc.
PLTN1   PLTN
PLTN2    &INP XP(1)=2.,XP(3)=0.0,XP(4)=1.5,IV=2,VLO=5.,
         VHI=100.,NPTS=100 &END
PLTN3   BRMSRZALOG



Distribution functions that are available are indicated below:

-----------------------------------------------------------------
    IDFNAM                   Purpose
-----------------------------------------------------------------
    'ALIN'    Linear plot.
    'ALOG'    Natural log plot.
    'ALKE'    Natural log of electron kinetic energy plot.
   'ADFMOL'   Approximation to Moller and Bhabha distributions
              (i.e., 1/K.E. distribution).


------------------------
A3.3.10  The HPLT Option
------------------------

     The Histogram PLoT option is designed to be used in
conjunction with UCTESTSR (User Code to TEST Sampling
Routine), which is provided on the EGS4 distribution tape .
(Note: UCTESTSR was simply referred to as TESTSR in the
the EGS3 User Manual (see Section 2.6 of SLAC-210)).

     The basic idea is that a probability density function,
PDF(X) (see Section 2.2 of SLAC-265), is to be sampled
by EGS (note: PDF(X) will have other static arguments
which we ignore for this discussion).  Let CDF(X) be the
cumulative distribution function associated with PDF(X).



A3.3-20
```

If PDF(X) drops sharply with increasing X, we will not get many samples in the bins with large X unless we make the bins themselves larger in such regions. We accomplish this by finding another p.d.f.and c.d.f., PDG(X) and CDG(X), respectively, such that PDG(X) approximates PDF(X). If we want N bins, we then pick the X(I) such that

```
  CDG(X(I+1))-CDG(X(I))=(CDG(X(N+1))-CDG(X(I)))/N.
```

For all I, this implies that

```
  X(I)=CDGI((CDG(X(N+1))-CDG(X(1)))*I/(N+1) + CDG(X(1)))
```

where CDGI is the inverse function of CDG. Thus, if PDG(X) is a reasonable approximation, the histogram bins at large X should have the same order of magnitude of counts as those at lower X. The function CDG(X) is called the "distribution function" in the context of the HPLT option. CDG(X), CDGI(X), and PDG(X) are used by the UCTESTSR and HPLT1 programs. If we let SPDF(X) and SCDF(X) be the "sampled" data, and PDF(X) and CDF(X) be the theoretical data, then the routine HPLT1 can be summarized by the pseudo-code:

```
  DO I=1,N [
  PLOT((SCDF(X(I+1))-SCDF(X(I)))/(CDG(X(I+1))-CDG(X(I))));
  PLOT((CDF(X(I+1))-CDF(X(I)))/(CDG(X(I+1))-CDG(X(I))));

  DO....X(I) at 10 points in the interval (X(I),X(I+1)) [
    PLOT(d(SCDF)/d(CDG)=PDF(X)/PDG(X));]
  ]
  RETURN;  END;
```

Thus the theoretical and sampled distributions can be compared and problems with the sampling routine (or the random number generator, for example) can be detected.

All of the control cards for the HPLT option are punched directly by UCTESTSR. The reader should refer to the listing for UCTESTSR and comments therein for a better understanding of the HPLT option (see also Chapter 6 of SLAC-210).

A3.3-21

```
-----------------------
A3.4 Concluding Remarks
-----------------------

     In the previous sections we have seen the various uses
for PEGS.  We summarize by giving the option sequences most
generally used.

-----------------------------------------------------------------
  A. Minimal material data set creation (for use by EGS).
-----------------------------------------------------------------
         1.  ELEM (or MIXT, or COMP)
         2.  ENER
         3.  PWLF
         4.  DECK

-----------------------------------------------------------------
  B. Same as A. with default plots of all the functions
     that the PWLF option fits.
-----------------------------------------------------------------
         1.  ELEM (or MIXT, or COMP)
         2.  ENER
         3.  TEST
         4.  PWLF
         5.  DECK

-----------------------------------------------------------------
  C. Comparison of theoretical and sampled distributions
     by means of the HPLT option.
-----------------------------------------------------------------
         1.  ELEM (or MIXT, or COMP)
             Note: Data cards should agree with those used
                   with the UCTESTSR run.
         2.  HPLT
             Note: Output data from UCTESTSR run.

-----------------------------------------------------------------
  D. Selective plotting of various functions.
-----------------------------------------------------------------
         1.  ELEM (or MIXT, or COMP) - for material 1
         2.  PLTN  -  for function 1
         3.  PLTN  -  for function 2,....etc.
         4.  ELEM (or MIXT, or COMP) - for material 2,....etc.
         5.  PLTN  -  for function 1
         6.  PLTN  -  for function 2,....etc.
```

7 Pegsless Mode

As of 2013, EGSnrc user codes can be run in pegsless mode. User codes run in this mode do not require the use of interaction cross sections calculated a-priori using the PEGS4 code (*i.e.* no `.pegs4dat` file). Photon cross sections have been generated on-the-fly since 2006. Thus, migration to a fully pegsless implementation is a logical next step.

7.1 User Code Inputs for Pegsless Mode

To run a user code in pegsless mode, the user must include parameters for calculating cross-sections, including specifications for all media used in the simulation, in their `.egsinp` file between the delimiters, `:start media definition:` and `:stop media definition:`. This is best illustrated with an example input:

```
:start media definition:

AE=0.521
UE=50.511
AP=0.01
UP=50.

material data file=/home/username/HEN_HOUSE/pegs4/data/material.dat

:start H2O521ICRU:
  elements = H, O
  number of atoms = 2,1
  rho = 1.0
  bremsstrahlung correction = KM
:stop H2O521ICRU:

:start AIR521ICRU:
  elements = C,N,O,AR
  mass fractions = 1.24000E-04, 7.55200E-01, 2.31800E-01, 1.28300E-02
  rho = 1.2048E-03
  bremsstrahlung correction = NRC
  gas pressure = 1.0
:stop AIR521ICRU:

:start PMMA521ICRU:
  bremsstrahlung correction = NRC
  density correction file = /home/uname/EGSnrc/HEN_HOUSE/pegs4/density_corrections/
compounds/polymethylmethacrylate__lucite___perspex___plexiglas_.density
:stop PMMA521ICRU:

:stop media definition:
```

where:

`AE,UE,AP,UP` are the kinetic energy limits for calculating photon (`AP,UP`) and electron (`AE,UE`) cross sections in MeV. These energy limits are also mentioned in the context of the EGSnrc system in Section 3.3. If `AE` is not specified it defaults to the highest value of `ECUT` (electron transport cutoff energy–see Section 3.4.2) specified in the simulation. If `AP` is unspecified, then it defaults to the highest value of `PCUT` (photon transport cutoff energy–see Section 3.4.2) specified. `UE` and `UP` default to 50.511 MeV and 50.0 MeV, respectively.

`material data file` is the name (including full directory path) of a file containing specifications for the media used in the simulation. Provided with the EGSnrc distribution is the material data file `$HEN_HOUSE/pegs4/data/material.dat` which contains specifications necessary to reproduce all of the cross-section data in `521icru.pegs4dat` and `700icru.pegs4dat` (provided that the appropriate values of `AE, UE, AP and UP` are specified–see above). Note that the format for media specifications in the material data file is similar to that used for specifying media directly in the `.egsinp` file, described immediately below. In the case of the material data file, however, there is some redundancy in the specification to allow the user to see the composition and density of the media.

the `:start MEDNAME:` and `:stop MEDNAME:` delimiters are used to specify medium, `MEDNAME`, directly in the `.egsinp` file. This method of specifying a medium is used if `MEDNAME` is not included in the material data file or if you wish to override some or all of the specifications for `MEDNAME` in the material data file. If no material data file is input, then all media in the simulation must be specified in this way. The inputs between these delimiters are described below. Variables in square brackets are the analogous PEGS4 variables described in the PEGS4 Manual (Section 6) above.

`elements` specifies the elements comprising the medium. Elements are specified using chemical symbols separated by commas. Case is unimportant.

`number of atoms` [PZ] or `mass fractions` [RHOZ]. For each of the `elements`, specify either the number of atoms in a molecule of the medium (*i.e.* stoichiometric coefficients), if the medium is a compound, or the mass fractions of the elements in the medium, if the medium is a mixture. Values are separated by commas and are input in the same order as their corresponding elements. In the example above, the composition of `H2O521ICRU` is defined using the number of atoms of each element, while that of `AIR521ICRU` is defined using the mass fraction of each element. Note that this input is omitted if the medium is an element.

`rho` specifies the bulk density of the medium in g/cm^3.

`bremsstrahlung correction` [IAPRIM] specifies the correction to apply to calculated bremsstrahlung cross-sections. Options are:

- `KM` [IAPRIM=0]: Apply Koch and Motz[47] empirical corrections.
- `NRC` [IAPRIM=1]: (the default) Apply NRC corrections based on NIST/ICRU[53]. These corrections are read from the file `$HEN_HOUSE/pegs4/aprime.data`.
- `None` [IAPRIM=2]: No corrections applied.

`density correction file` [EPSTFL] is the name of a file containing density effects which, when applied to calculated collision stopping powers, results in agreement with collision stopping powers published in ICRU37[50]. In general, density correction files are specified including their full directory path and `.density` file extension. However, the file can be specified by its prefix only if it exists in, in order of search priority:

1. `$EGS_HOME/pegs4/density_corrections`
2. `$EGS_HOME/pegs4/density_corrections/elements`
3. `$EGS_HOME/pegs4/density_corrections/compounds`
4. `$EGS_HOME/pegs4/density`
5. `$HEN_HOUSE/pegs4/density_corrections/elements`
6. `$HEN_HOUSE/pegs4/density_corrections/compounds`

Note that the density correction files for many elements, compounds and mixtures are supplied with the distribution. Density correction files have a header portion from which the composition and bulk density of the medium are read. These values override any user inputs for `elements=`, `number of atoms=` or `mass fractions=`, and `rho=`. Thus, as in the case of `PMMA521ICRU` in the example above, it is possible to specify the composition of a medium simply by specifying a density correction file.

`gas pressure` [GASP] is the pressure of the medium in atm if the medium is a gas. This input is only relevant (and necessary for a gas) if a density correction file is not used, in which case `gas pressure` is used to modify the calculated density effect parameters. `gas pressure` defaults to 0 (*i.e.* the medium is not a gas).

Inputs specifying media are case insensitive with the exception of the medium name (*e.g.* `H2O521ICRU` is different than `h2o521ICRU`).

For more details on the inputs for specifying a medium, please refer to the PEGS4 Manual above (Section 6).

7.2 Running User Codes in Pegsless Mode

EGSnrc user codes that read an input file can be run interactively in pegless mode using the command line input:

```
user_code -i inputfile
```

where `inputfile` is the name of the input file (with no `.egsinp` extension).

Pegsless batch runs use the command line syntax:

```
exb user_code inputfile pegsless [short|medium|long] [batch=batch_system] [p=N]
```

This is identical to the syntax for a batch run with pegs data but with the word "`pegsless`" in place of the name of the pegs data file.

When running in pegsless mode, EGSnrc outputs a file, `inputfile.mederr`, which, for each medium used in the simulation, indicates where each specifying parameter has been read (*i.e.* from a material data file or directly from the `.egsinp` file). The file also includes warnings when `AE`, `UE`, `AP` or `UP` have not been specified and have been set to their default values and when a material data file has not been specified. For parallel runs in pegsless mode (parameter `p` > 1 in the batch command syntax above), the `.mederr` file is only output by the first job.

The actual values of the media specifications (including defaults) used to calculate cross-sections are output in the listing file, `inputfile.egslst`, and to the screen for interactive runs or in the log file, `inputfile.egslog`, for batch runs.

7.3 Implementation of Pegsless Code

To run an EGSnrc user code in pegsless mode, the following files, all located in `$HEN_HOUSE/src`, must be included at compile time:

`pegs4_macros.mortran`:

- Contains PEGS4 common block variables used to calculate electron cross sections by subroutines in `pegs4_routines.mortran` (see below). In many cases these are modified versions of the common blocks used by `pegs4.mortran` (the original PEGS4 code).
- Contains the macro `$GET_PEGSLESS_XSECTIONS`. This is coding that is inserted at compile time into the subroutine `HATCH` in `egsnrc.mortran` and is executed when the code is run in pegsless mode. First, this block of coding calls the subroutine `get_media_inputs` (see below) to read the pegsless inputs. The coding then loops over all media in the simulation and, for each medium:
  1. copies media parameters stored in EGSnrc common block variables by `get_media_inputs` into the corresponding PEGS4 common block variables used by the cross section calculation subroutines
  2. calls the necessary subroutines in `pegs4_routines.mortran` for calculating electron cross sections
  3. copies cross section data back out from the PEGS4 common block variables into the corresponding EGSnrc common block variables for use in the simulation.

  Finally, `show_media_parameters` is called (see below) to output the media specifications read in to standard output (the `.egslog` file in the case of a batch run).
- Contains the macro `$DECLARE-PEGS4-COMMON-BLOCKS`, called at the beginning of `HATCH` to declare the PEGS4 common block variables that `HATCH` needs to have access to, so that they can be copied from/to their EGSnrc common block analogs. This macro also declares some local and external types used by `HATCH` in pegsless mode.
- Contains the macro `$INIT-PEGS4-VARIABLES` used to define some constants in the PEGS4 common blocks.

- Must be included after `egsnrc.macros` and before the user source code.

`pegs4_routines.mortran`: This contains the subroutines necessary to calculate electron cross sections. In most cases, these subroutines have been copied directly from `pegs4.mortran` with some modifications to variable names, suppressed output, and the handling of density correction files. Subroutines that appear directly in the `$GET_PEGSLESS_XSECTIONS` coding block (see above) are:

- `MIX`: This subroutine calculates multiple scattering parameters used to calculate electron cross sections.
- `SPINIT`: This subroutine determines the density correction to apply to electron stopping powers. It takes as an argument the name of the density correction file for the medium (if specified). It then determines the appropriate density corrections as outlined in Subsection 7.1 above.
- `PWLF1`: This subroutine performs a piece-wise linear fit of functions describing bremsstrahlung, Moeller, Bhaba and positron annihilation cross sections over the electron kinetic energy interval defined by [`AE, UE`]. The subroutine determines the number of sub-intervals required for the fits and returns a discrete point for each interaction type for each sub-interval.

`get_media_inputs.mortran`:

- Contains the `get_media_inputs(ounit)` subroutine, which interprets the pegsless inputs between the delimiters, `:start media definition:` and `:stop media definition:` in the `.egsinp` file. The subroutine is also responsible for assigning default values for input parameters that are out of range, omitted, or left blank. Input values are then passed to the appropriate common block variables for use in calculating/modifying electron cross sections for the media in the simulation. The parameter, `ounit`, specifies the unit number to which media specifications, once read, will be echoed. Usually this is unit number 6 (standard output). If `ounit`$\leq$0, then the media specs are not echoed. The subroutine also has an entry point, `show_media_parameters(ounit)`, which can be called from the user code to echo the media specifications to any unit at any point after they have been read in. Finally, this subroutine opens up and writes to the `.mederr` file, containing information about the source of media specifications used (*i.e.* the material data file or the `.egsinp` file).
- Contains the subroutine `GET_INPUT_PLUS`, which is a modified version of `GET_INPUT` (in the file `get_inputs.mortran`). Modifications include the ability to read a unit other than standard input (the `.egsinp` file) and the ability to have different starting and ending delimiters within which to search for inputs. `GET_INPUT_PLUS` is used by `get_media_inputs` to read and interpret the material data. file.

Apart from ease of bookkeeping, the main reason for separating the PEGS4 common block variables used to calculate cross sections from their EGSnrc common block analogs is to allow calculation of cross sections with the same precision (`REAL*4`) as `pegs4.mortran`. This conveniently allows comparisons between results in pegsless mode and those using

PEGS4 data files (created with identical media specifications and over the same energy range). Note, however, that there will still be differences within uncertainty between the two calculations. This is because the reading of cross section data from fixed-format PEGS4 data files necessarily results in a loss of precision compared to passing values internally from the PEGS4 common block variables to the EGSnrc variables.

To change electron cross section calculations to double precision, go into `$HEN_HOUSE/src/pegs4_macros.mortran` and change the macro, `$REAL4` from `real*4` to `real*8` and recompile your user code.

7.4 Changes to EGSnrc Source Codes

Implementation of pegsless mode necessitated the following changes to EGSnrc source coding:

In `$HEN_HOUSE/src/egsnrc.macros`:

- Definition of empty macros `$GET-PEGSLESS-XSECTIONS`, `$INIT-PEGS4-VARIABLES` and `$DECLARE-PEGS4-COMMON-BLOCKS` which are used in place of their versions defined in `$HEN_HOUSE/src/pegs4_macros.mortran` when user codes are compiled without pegsless implementation.
- Definition of a new logical variable, `is_pegsless`, in the `EGS-IO` common block. This variable is set to `.true.` for pegsless runs (see below).

In `$HEN_HOUSE/src/egsnrc.mortran`:

- Introduction of `$INIT-PEGS4-VARIABLES` and `$DECLARE-PEGS4-COMMON-BLOCKS` macros at the beginning of subroutine `HATCH`.
- Within `HATCH`, the introduction of a new conditional statement where the `.pegsdat` file is read if `is_pegsless=.false.` and the `$GET-PEGSLESS-XSECTIONS` macro is executed if `is_pegsless=.true.`

In `$HEN_HOUSE/src/egs_utilities.mortran`

- Subroutine `egs_check_arguments` sets `is_pegsless=.true.` if the "`-p pegsfilename`" argument is not supplied on the command line when running a user code.
- Subroutine `egs_init1` now only opens the PEGS4 data file if `is_pegsless=.false.`

8 EGS User Guide to MORTRAN3

SLAC265 - APPENDIX 4

EGS User Guide to Mortran3

By

Walter R. Nelson
Stanford Linear Accelerator Center
Stanford University
Stanford, CA 94305, U.S.A.

David W. O. Rogers
National Research Council of Canada
Ottawa K1A 0R6, Canada

31 December 1985

[This EGS User Guide to Mortran3 was originally Appendix 4 of SLAC-265. This guide was excerpted from SLAC Computation Research Group Report CGTM No. 165 (June 1975): "A User's Guide to Mortran2" by A. James Cook and L. J. Shustek and modified by the authors to represent Mortran3.]

With the EGSnrcMP environment, there have been a few changes to the MORTRAN3 processor. In source code, C-style comments (viz /* comment */) are now allowed, and any line starting with a # is output directly without any processing (this is to allow C-preprocessor statements to be used.

PREFACE

EGS is written in a structured language called Mortran3 which has been developed at SLAC by A. James Cook and L. J. Shustek. The Mortran3 precompiler is written in 1966 ANSI standard FORTRAN IV and 1977 ANSI standard FORTRAN 77, and both are included on the EGS4 distribution tape. This guide, which is also maintained on the EGS4 distribution tape, is meant to be a brief introduction to Mortran3 to allow EGS4 users to use it. For a complete description, the user is referred to the "Mortran3 User's Guide" by A. J. Cook, SLAC Computation Research Technical Memorandum CGTM-209 (1983).

Although it is possible to write EGS User Codes entirely in FORTRAN, we strongly urge you to use Mortran because of how easily one can write readable (and therefore more likely correct) code, because of its flexibility, and because of its ability to improve execution time by allowing for in-line code generation.

8.1 Introduction

The term Mortran, like FORTRAN, has several meanings, depending upon the context in which the term is used. Mortran has come to mean:

- A Structured Language. - A Translator for that language. - A Macro-processor.

The structured language is implemented as a set of macros which are used by the macro processor to translate the language into FORTRAN. The resulting FORTRAN program is then run like any other FORTRAN program. User-defined macros are easily added to the standard (language-defining) set of macros so that the language is "open-ended" in the sense that extensions to the language may be made at any time by the user. Extensions have ranged from very simple ones like matrix multiplication, to complex ones like those which define new data types.

Users need not concern themselves with the method of implementation or the macro facility in order to take advantage of the structured language which is provided by the standard set of macros. The features of this language include:

```
     - Free-field (column and card boundaries may be
     ignored).  - Alphanumeric labels of arbitrary length.
     - Comments inserted anywhere in the text.  - Nested
     block structure.  - Conditional statements which
     may be nested (IF, IF-ELSE,
       and ELSEIF).
     - Loops (repetitively executed blocks of statements)
     which
       test for termination at the beginning or end or
       both or neither (WHILE, UNTIL, FOR-BY-TO, LOOP
       and DO).
     - EXIT (jump out of) any loop.  - NEXT (go to NEXT
     iteration) of any loop.  - Multiple assignment
     statements.  - Conditional (alternate) compilation.
     - Program listing features include:
         - Automatic printing of the nesting level.
         - Automatic indentation (optional) according
         to nesting
           level.
     - Abbreviations for simple I/O statements.  -
     Interspersion of FORTRAN text with Mortran text.
```

The user may elect to override the standard set of macros and write a set which defines another, perhaps "problem oriented", language.

The Mortran3 processor is a FORTRAN program of approximately 2000 statements. The 1966 ANSI standard has been observed throughout, so that transportability of the processor is assured.

8.2 Coding Rules

Mortran programs may be written without regard to column or card boundaries. Statements may begin anywhere on the input line (card image), and may end anywhere on the same line or on a succeeding line. The end of a statement is determined by a semicolon (;). This feature permits "free-field" or "free-form" programming. Normally, only the first 72 columns of the input line are intepreted as program text, but this can be changed (see Section A4.6).

Character strings comprising Hollerith fields are enclosed in apostrophes (as in 'THIS IS HOLLERITH DATA'). If an embedded apostrophe is desired as a character within a quoted string, use a pair of apostrophes to represent each such embedded apostrophe (as in 'DON''T').

Comments in Mortran are enclosed in quotation marks (as in "COMMENT") and may be inserted anywhere in the program (except in character strings or macros).

In Mortran, an alphanumeric label is a character sequence of arbitrary length enclosed in colons (as in :TOMATOES:). The characters which comprise the sequence may be any combination of letters and digits. An alphanumeric label may be used anywhere a FORTRAN statement label is allowed.

Multiple blanks (a sequence of two or more blanks) in a Mortran program are equivalent to a single blank except in quoted strings, where all blanks are preserved, and in macros (discussed in Section A4.5).

Summary of coding rules:

- Terminate statements with a semicolon (;).

- Enclose comments in quotation marks (").

- Enclose labels in colons (:).

- Enclose character strings in apostrophes (’).

- Blanks may be inserted freely except in labels, character strings and user-defined macros.

It should be pointed out that any extensions provided by a particular FORTRAN compiler may be used, provided that they do not conflict with Mortran’s coding conventions. However, if transportability of the programs being written in Mortran is a consideration, the ANSI FORTRAN standard should be adhered to. The standard set of macros which define the language described in this Guide do not generate non-ANSI FORTRAN.

8.3 Structure

8.3.1 Statements

FORTRAN may be regarded as a subset of Mortran, since (with the exception of multiple blanks in a Hollerith string) any valid FORTRAN statement becomes a valid Mortran statement when

- it is terminated by a semicolon, and
- continuation marks (if any) are deleted.

8.3.2 Blocks

A Mortran block is a sequence of Mortran statements enclosed in the special characters [and], which we will call "brackets" (or "square brackets"). The left bracket may be read "begin" and the right bracket may be read "end". Let

```
                   S1; S2; S3;...Sk;...Sn;             (A4.3-1)
```

be a sequence of statements. The sequence becomes a block when it is enclosed in brackets

```
                 [ S1; S2; S3;...Sk;...Sn; ]  .         (A4.3-2)
```

(Note: the ellipses (...) are meta-symbols indicating arbitrary repetition. The brackets are not meta-symbols; they are delimiters in the Mortran language).

Blocks may be nested. That is, any of the statements in a block may be replaced by a block. For example, in sequence (A4.3-2) we could replace Sk by a block and write

```
     [ S1; S2; S3;...[ T1; T2; T3;...Tm; ]...Sn; ]      (A4.3-3)
```

The block containing the sequence T1;...Tm; is completely contained, or nested, within the block containing the sequence S1;...Sn;. We will frequently write an ellipsis enclosed in brackets [...] to denote a block.

Example of a block: [X=Y; CALL SUB(A); B=1;]   .

8.3.3 Conditional Statements

The simplest form of a conditional statement in Mortran is written

```
                   IF e [...]                           (A4.3-4)
```

where e is an arbitrary logical expression, and the ellipsis enclosed in brackets denotes a block as described above. If e is true then the statements in the block are executed. If e is false, control is transferred to the first statement following the block. For example:

```
                   IF A.LT.B [ C=D; E=F; ] G=H;
```

If A is less than B then the statements C=D and E=F are executed after which G=H is executed. If A is not less than B control is transferred directly to the statement G=H.

Next in complexity is the IF-ELSE statement, which is written

```
                   IF e [...] ELSE [...]                (A4.3-5)
```

If e is true then the statements in the first block are executed and control is transferred to the statement following the second block. If e is false then the statements in the second block are executed and control is transferred to the statement following the second block. For example, consider

```
                      IF A.LT.B [C=D; E=F;]
                           ELSE   [G=H; I=J;]
                      K=L;
```

If A is less than B the statements C=D and E=F are executed after which control is transferred to the statement K=L. If A is not less than B the statements G=H and I=J are executed after which control is transferred to the statement K=L.

Consider

```
                         IF A.EQ.B [ X=Y;]
```

Here the block to be executed, whenever A is equal to B, consists of the single statement X=Y;. An alternate form acceptable in Mortran is the standard FORTRAN logical IF:

```
                          IF (A.EQ.B) X=Y;
```

IF-ELSE statements may be nested to any depth. Even so, the IF-ELSE is not really adequate (in terms of clarity) for some problems that arise. For example, consider the following "case analysis" problem. Suppose that we have four logical expressions

p, q, r, and s,

and five blocks of statements

A, B, C, D, and E.

Now suppose that p, q, r, and s are to be tested sequentially. When the first TRUE expression is found we want to execute the statements in the corresponding block (p corresponds to A, q to B, etc.) and then transfer control to the statement following block E. If none of them is true we want to execute block E. Using nested IF-ELSE statements we could write

```
                   IF p [A]
                   ELSE [ IF q [B]
                          ELSE [ IF r [C]
                                 ELSE [ IF s [D]
                                        ELSE [E]
                                      ]
                               ]
                        ]
```

While this does what we want, it is awkward because each ELSE increases the level of nesting. Mortran offers the ELSEIF statement as an alternative:

```
                        IF     p [A]
                        ELSEIF q [B]
                        ELSEIF r [C]                    (A4.3-6)
                        ELSEIF s [D]
                        ELSE     [E]
```

Using ELSEIF instead of ELSE allows all the tests to be written at the same nest level.

In summary, an IF statement may be optionally followed by any number of ELSEIF clauses which in turn may be optionally followed by a single ELSE clause.

8.3.4 Iteration

A Mortran loop is a block which is preceded by, and optionally followed by, a "control phrase". One such phrase is " WHILE e ". One of the loops we may write with this phrase is

```
                        WHILE e  [...]                  (A4.3-7)
```

The logical expression e is tested first. If e is true the block is executed and then control is returned to test e again. When e becomes false, control is transferred to the first statement following the block.

Another control phrase is " LOOP ". If we wanted to test at the end of the loop instead of the beginning, we could write

```
                        LOOP [...] WHILE e ;            (A4.3-8)
```

In (A4.3-8) the block is executed first. Then, if the logical expression e is true, the block is executed again. When e becomes false control is transferred to the statement following the loop (that is, the statement following the " WHILE e ;").

The logical converse of the WHILE loop is the UNTIL loop.

```
                    UNTIL e  [...]              (A4.3-9)
```

The logical expression e is tested first. If e is false the block is executed and then control is returned to test e again. When the logical expression becomes true, control is transferred to the first statement following the block. Similarly, the logical converse of (A4.3-8) may be written by replacing the WHILE in (A4.3-8) by UNTIL.

Tests for loop termination may be made at both ends of a loop. For example, if e and f are logical expressions

```
                       WHILE e [...] UNTIL f ;
                       WHILE e [...] WHILE f ;

                       UNTIL e [...] WHILE f ;
                       UNTIL e [...] UNTIL f ;  ,
```

all test at both the beginning and the end. The above list is by no means exhaustive, but we must develop other "control phrases" in order to complete the discussion.

The iteration control phrases discussed above do not involve "control variables"; that is, variables whose values are automatically changed for each execution of the loop.
The following loop involves a control variable:

```
                 FOR  v = e  TO  f  BY  g  [...]        (A4.3-10)
```

where v is the control variable and e, f, and g are arbitrary arithmetic expressions. The control variable v must be of type REAL or INTEGER, and may be an array element (subscripted variable). The value of any of the arithmetic expressions may be positve or negative. Moreover, the magnitudes as well as the signs of f and g may change during the execution of the loop.

The control variable v is set to the value of e and the test for loop termination (see below) is made. If the test is passed then the block is executed, after which v is incremented by the value of g and control is returned to the test. Note that the block is never executed if the test fails the first time.

The "test" for the termination of a FOR-loop refers to the logical expression

```
                     g * (v-f) .GT. 0                   (A4.3-11)
```

```
If the value of (A4.3-11) is true, then the test is said to have
failed and control is transferred to the statement following the
loop.  Multiplication by g in (A4.3-11) assures that loops in
which the increment is (or becomes) negative will terminate
properly.

     The "FOR-loop" (A4.3-10) has two alternate forms
              FOR v = e  BY  g TO f  [...]             (A4.3-12)
and           FOR v = e  TO  f       [...]             (A4.3-13)

In (A4.3-13) no increment is given, so it is assumed to be one.

     The iteration control phrase "DO I=J,K,N" also involves
a control variable.  In this case I,J,K, and N must all be of
type INTEGER and may not be array elements or expressions (these
are the standard FORTRAN rules for DO-loops).  The following
generates a standard FORTRAN DO-loop:

                   DO I=J,K,N [...]                    (A4.3-14)

     There is one exception to the rule that loops must be
preceded by control phrases; namely, the compact DO-loop notation

                   [I=J,K,N;...]                       (A4.3-15)

which generates a standard FORTRAN DO-loop.  This form permits
compact notation for nests like

          [I=1,N1; [J=1,N2; [K=1,N3; A(I,J,K)=exp; ]]].

(The use of the compact DO-loop notation is controversial;
some people feel that it obscures the loop control.  If desired,
it can be be removed from the language by deleting a single macro
from the standard set).

     The Mortran FOR- and DO-loops apply only to blocks of
statements, not to I/O lists.  The usual FORTRAN implied DO
should be used within READ or WRITE statements.

     There remains one more type of loop to be discussed.  This
loop is sometimes referred to as the "forever loop".  One writes
the forever loop in Mortran by preceding a block with the phrase
"LOOP":
                    LOOP [...] REPEAT                  (A4.3-16)
or simply           LOOP [...]                         (A4.3-17)
```

The block is executed and control is transferred back to the beginning of the loop. The optional phrase "REPEAT" in (A4.3-16) is sometimes useful as a visual aid in locating the ends of deeply nested loops.

A reasonable question might be: "How do you get out of a forever loop? Or, for that matter, any of the loops?". One rather obvious way is to write

```
                          GO TO :CHICAGO: ;
```

where the label :CHICAGO: is on some statement (or block) outside the loop. If a convenient label doesn't already exist, creating one for the sole purpose of jumping out of the loop can be annoying and distracting. For the case in which the jump is to the statement following the loop, the GO TO may be replaced by the Mortran "EXIT;" statement, which is written

```
                         EXIT;                    (A4.3-18)
```

or, with a conditional statement

```
                     IF (e) EXIT;                 (A4.3-19)

or                   IF  e  [...EXIT;]            (A4.3-20)
```

In any Mortran loop, the occurrence of the statement "EXIT;" causes a transfer of control to the first statement following the loop in which it occurs.

A companion to the "EXIT;" statement is the "NEXT;" statement, which is written

```
                         NEXT;                    (A4.3-21)
```

or, with a conditional statement

```
                     IF (e) NEXT;                 (A4.3-22)

or                   IF  e  [...NEXT;]            (A4.3-23)
```

The occurrence of a "NEXT;" statement (which is short-hand for "go to the next iteration of this loop") in any Mortran loop causes a transfer of control to the beginning of the loop in which it occurs, incrementing the control variable (if any) before making the test for continuation in the loop. In loops which test at both ends of the loop, only the test at the beginning of the loop is made; tests at the end of the loop are made only when the end of the loop is reached. The tests of control variables in FOR- and DO-loops are considered to be at the beginning of the loop.

Any Mortran loop may be optionally preceded by a label. We will call loops which are preceded by labels "labeled loops". Any EXIT or NEXT statement may be optionally followed by a label. Any labeled loop may contain one or more statements of the form

```
                 EXIT :label: ;                        (A4.3-24)
```

which transfers control to the first statement following the labeled loop. For example EXIT :ALPHA:; would transfer control to the statement following the loop which had been labeled :ALPHA:. This transfer of control takes place regardless of nesting, and thus provides a "multi-level" EXIT capability. The statement

```
                 NEXT :label: ;                        (A4.3-25)
```

transfers control in the manner described above for the NEXT; statement.

Suppose we have a nest of loops which search some arrays. The outer loop has been labeled :SEARCH:, and two of the inner loops have been labeled :COLUMN: and :ROW:. Now we may write

```
  NEXT :ROW: ;    or    NEXT :COLUMN: ;   or    EXIT :SEARCH: ;
```

when the transfer involves more than one level of nesting, or

```
                  NEXT;     or       EXIT;
```

when only one nest level is involved. Of course, the form

```
                      EXIT :label:;
```

may also be used to exit a single level if desired.

We can now summarize Mortran loops in the following chart:

```
         | WHILE a          |
         | UNTIL b          |    | NEXT;        |
         | LOOP             |    | EXIT;        |      | WHILE c; |
|:label:|  FOR v=x BY y TO z [...| EXIT :label:;| ...] | UNTIL d; |
         | FOR v=x TO y BY z|    | NEXT :label:;|      | REPEAT   |
         | FOR v=x TO y     |
         | DO i=j,k,n       |

         where  a, b, c, and d are arbitrary logical expressions,
                x, y, and z are arbitrary arithmetic expressions,
                v is a subscripted or non-subscripted variable of
                   type INTEGER or REAL,
                j, k, and n are non-subscripted INTEGER variables
                   or INTEGER constants, and
                i is a non-subscripted INTEGER variable.

         | |  indicates "choose one",
         [ ]  indicates "optional", and
         ...  indicates a (possibly null) sequence of statements.
```

8.4 Miscellaneous Features

8.4.1 Multiple Assignment

It is sometimes useful to be able to assign the value of some expression or variable to several variables in a single statement. In Mortran one writes

```
               / p,q,r...,z / = e                      (A4.4-1)
```

where p,q,r,...,z are variables and e is an expression. The expression e assigned to each variable in turn. For example,

```
         / I, A(I,K), J / = SQRT(X/2.0);
```

produces the following FORTRAN statements:

```
         I = SQRT(X/2.0)
         A(I,K) = SQRT(X/2.0)
         J = SQRT(X/2.0)
```

8.4.2 I/O Abbreviations

```
     Another instance in which the creation of a label can be
annoying because it is used only once and contains no mnemonic
information is the following:

 READ (5,:NONSENSE:) i/o list;  :NONSENSE: FORMAT(format list);

In Mortran one may write

              INPUT i-o list; (format list);            (A4.4-2)
or            OUTPUT i-o list; (format list);           (A4.4-3)

whenever the input or output is to the standard FORTRAN input or
output units (5 and 6, respectively).  (If your FORTRAN compiler
is already "extended" to allow the keywords INPUT and OUTPUT or
if these keywords are already used in other contexts, the macros
defining these statements may be removed or modified to use other
keywords.  Similarly, the input and output units may be easily
changed to units other than 5 and 6.)
```

8.4.3 Operators

```
     Relational operators (e.g., .LT. etc. in FORTRAN notation) may
be denoted by:

          <, <=, =<, =, ~=, =>, >=, and >.

Logical operators may be denoted by:

          & (and), | (or), and ~ (not).

The usual FORTRAN notation is still valid, but one must NOT mix
modes within a single statement (i.e., (A>B.AND.C>D) will be
translated incorrectly.
```

8.5 User-Defined Macros

8.5.1 String Replacement

```
     Macro definitions are written in the following form:

       REPLACE {pattern} WITH {replacement}             (A4.5-1)
```

Macro definitions are not statements and therefore need not be terminated by semicolons (if you put one in it will be ignored). Macro definitions are "free field" in the sense that you may write more than one definition on one line, or extend one definition to several lines.

The pattern and replacement parts of a macro definition are character strings in the sense described in Section A4.2. Since embedded strings are permitted in macro definitions, the usual rules regarding the doubling of apostrophes apply.

The simplest kind of macro is one which contains neither parameters nor embedded strings. For example, one could write

```
      REPLACE {ARRAYSIZE} WITH {50}                     (A4.5-2)
```

after which all occurrences of the characters ARRAYSIZE in the program text would be replaced by 50. For example,

```
         DIMENSION X(ARRAYSIZE); ... DO J=1,ARRAYSIZE [...]...
```

would produce the same FORTRAN program as if

```
         DIMENSION X(50);...  DO J=1,50 [...]...
```

had been written.

Blanks are generally not significant when searching for occurrences of the pattern in the program text. For example, the macro

```
           REPLACE {SIGMA(1)} WITH {SIGMA1}
```

would match the program text

```
                    SIGMA (1)
```

as well as

```
                    SIGMA(1)
```

In some cases it is desirable to require that one or more blanks be present in the program text in order that a match occur; this can be done by writing a single blank in the pattern part of the macro. For example the macro

```
      REPLACE {DUMP X;} WITH {OUTPUT X; (F10.2);}
```

would match

```
                   DUMP X;
```

but not

```
                   Y = DUMPX;
```

Normally, the text generated by a macro is itself eligible for replacement by other macros, or even by the same macro that generated the text.

8.5.2 Parameters in Macros

The pattern part of a macro definition may contain up to nine formal (or "dummy") parameters, each of which represents a variable length character string. The parameters are denoted by the symbol #. For example,

```
            {EXAMPLE#PATTERN#DEFINITION}              (A4.5-3)
```

contains two formal parameters. The formal parameters are "positional". That is, the first formal parameter is the first # encountered (reading left to right), the second formal parameter is the second # encountered and so on. The corresponding actual parameters are detected and saved during the matching process. For example, in the string

```
      EXAMPLE OF A PATTERN IN A MACRO DEFINITION        (A4.5-4)
```

(assuming (A4.5-3) is the pattern to be matched), the first actual parameter is the string "OF A", and the second actual parameter is the string "IN A MACRO". The parameters are saved in a "holding buffer" until the match is completed.

After a macro is matched, it is "expanded". The expansion process consists of deleting the program text which matched the pattern part of the macro and substituting for it the replacement part of the macro.

The replacement part may contain an arbitrary number of occurrences of formal parameters of the form {Pi} (i=1,2,...,9). During expansion, each formal parameter {Pi} of the replacement part is replaced by the i-th actual parameter. A given formal parameter may appear zero or more times in the replacement part. For example, the pattern part of the macro definition

```
      REPLACE {PLUS #;} WITH {{P1}={P1}+1;}             (A4.5-5)
```

would match the program text

```
                  PLUS A(I,J,K);                          (A4.5-6)
```

During the matching process the actual parameter A(I,J,K) is saved in the holding buffer. Upon completion of the matching process (that is when the semicolon in the program text matches the semicolon in the pattern part), the "expansion" of the macro takes place, during which the actual parameter A(I,J,K) replaces all occurrences of the corresponding formal parameter, producing

```
               A(I,J,K)=A(I,J,K)+1;                        (A4.5-7)
```

Note that the single formal parameter {P1} occurs twice in the replacement part and therefore the single actual parameter A(I,J,K) occurs twice in the resulting string.

The program text which may be substituted for the formal (dummy) parameter is arbitrary except for the following restrictions:

1. It may not be text containing an end-of-statement semicolon. This restriction prevents "run-away" macros from consuming large parts of the program.

2. Parentheses and brackets must be correctly matched (balanced). This facilitates the construction of macros which treat expressions as indivisible units.

3. Quoted character strings are considered to be indivisible units. If the opening apostrophe of a character string is part of the actual parameter, then the entire string must be within the actual parameter.

8.6 Control Cards

More properly called "processor control directives", Mortran3 control cards may appear anywhere within the program and fall into two categories:

1. Column-one-restricted directives
2. Free-form directives

Column-one-restricted directives MUST begin in column one and only one directive per line is recognized. Free-form directives, on the other hand, may appear anywhere on a line and are not limited by number. Some of both types have been found to be useful with the EGS4 Code System and they will be presented next [additional information is provided in A. J. Cook, "Mortran3

```
User's Guide", SLAC Computation Research Group Technical Memor-
andum CGTM-209 (1983)].
```

8.6.1 Column-One-Restricted Directives

```
     Each of these control cards begins with a "%" in column 1,
and should not contain embedded blanks or program text.
Column 1
|
V
%%   Signals end of Mortran input.  This is the ONLY control card
     that is required.  All others are optional.  Unlike Mortran2,
     %% signals the final end of input and must not be used to
     signal end of input from this particular unit.

%F   Switch to FORTRAN mode (initial mode is Mortran).  While
     in FORTRAN mode, cards are read and written without any
     processing. This feature allows the interspersion of FORTRAN
     and Mortran text.  If this feature is used, all FORTRAN
     statement labels should be restricted to four digits
     (or less) in order to avoid possible conflict with Mortran
     generated statement labels, all of which are five digits
     long (in the default mode).

%M   Switch back to Mortran mode (initial mode is Mortran).

%E   Eject (start new page) in Mortran listing.

%L   List (i.e., turn on Mortran listing (initally ON)).

%N   Nolist (i.e., turn off Mortran listing).
%In  Indent n places per nest level in Mortran listing, where
     n=0,1,2,...,99 (initally 0). (Note: leading blanks are
     automatically suppressed when n>0 so that "ragged" programs
     will be "straightened" by Mortran).

%Cn  Set input line width to n, where n=10,11,...,80
     (initially n=72).  Characters in columns n+1 thru 80 will
     appear in the Mortran listing, but will be ignored by the
     processor.

%An  Annotation mode switch.  Controls the generation of Mortran
     source as comments in the generated FORTRAN as follows:

     n=0 Suppress Mortran text in FORTRAN file (initially n=0).
     n=1 Interleave Mortran text as comments in the FORTRAN
         program starting in column 2 and extending through
         column 80.  If column 80 of the Mortran source line is
         not blank, a second comment line is generated containing
         the 80th character.

     n=2 Interleave the Mortran text as comments in the FORTRAN
         program in columns 40 thru 80.  Each Mortran source line
         will appear as two comment lines in the FORTRAN listing,
         each of which contains one half of the Mortran source
         line.

%Qn  Quote switch (initally n=0).  Controls Mortran comments as
     follows:
     n=0 Comments must be fully enclosed in quotation marks (").

     n=1 All comments not closed at the end of each line will be
         closed by Mortran.

%Un  Unit switch.  Causes Mortran to switch to FORTRAN input
     unit n for further input, where n=1,2,3...99.  When an
     end-of-file is read from unit n, input is switched back
     again to the unit from which the %Un was read.  If another
     %Un is read before an end-of-file, the current input unit
     is stacked and the Mortran input unit is again switched.
     This control statement provides a facility similar to that
     implemented in other languages by "COPY" or "INCLUDE"
     statements.   It is particularly convenient for introducing
     standard declarations, common blocks, or macro definitions
     from a predefined external file.
```

8.6.2 Free-Form Directives

```
     Free-form directives begin with an exclamation point
("bang!") and end with a semicolon.  The following are of
particular use with EGS4/PEGS4:

!ANNOTATE;     Interleave Mortran source in FORTRAN output.
               Mortran statements become COMMENTs in FORTRAN
               output. [!NOANNOTATE; means "Turn Off !ANNOTATE"].

!COMMENTS;     Print Mortran comments as FORTRAN comments.
               Mortran comments are output to FORTRAN file with
               'C' in column one.  [!NOCOMMENTS; means "Turn
```

```
              Off !COMMENTS"].

!INDENT Mn;   Set automatic indentation of Mortran listing to
              n columns per nesting level (same as %In above).

!INDENT Fn;   Set automatic indentation of FORTRAN source to
              n columns per nesting level (like !INDENT Mn;).

!INDENT Cn;   Set automatic indentation of FORTRAN comments to
              n columns per nesting level (but the 'C' remains
              in column one).

!LIST;        Turn on Mortran listing.  Same as %L above.
              [!NOLIST; means "Turn Off !LIST;"].

!LABELS n;    Reset FORTRAN statement label generator to n.
```

9 EGSnrc System Considerations

This section is almost completely superceded by Report PIRS-877 which describes the EGSnrcMP environment or system for using the EGSnrc system[14]. This section is left here for old time's sake! A few of the specific commands have been updated to avoid total confusion.

9.1 Introduction

The EGSnrc Code System works in a manner which is very similar to the EGS4 system so that those using the EGS4 system already should find this very familiar. The major changes are to make the user codes subdirectories of `$EGS_HOME` instead of `$HOME/egs4` and to define the files related to specific runs as `.egsinp` instead of `.egs4inp` etc. These changes have been made to make it possible to run both systems in parallel for a while when making the transition, and to clearly identify which system various files refer to. This can be done by merely changing the definition of the environment variable `HEN_HOUSE` to point at either the EGSnrc or the EGS4 area, and sourcing `Cshrc_additions_for_egsnrc` instead of `Cshrc_additions_for_egs4` in your `.cshrc` file. If you are already an `EGS_pert` then the rest of this section will be of little interest. If you are new to EGS and the above makes no sense at all, then this is the section for you!

9.2 Overview

The EGSnrc system, which was originally developed for Unix based systems by Alex Bielajew, is very flexible and powerful. This comes at the expense of being somewhat complicated at first sight. However, the scripts which come with the system make it quite easy to use and flexible once you are familiar with the over-all design.

The EGSnrc system consists of two major directory area. One area, the `HEN_HOUSE`, holds all the standard EGSnrc files and does not get changed. The structure of the `HEN_HOUSE` is shown in fig 40. The `HEN_HOUSE` can be maintained anywhere on the system that all local EGSnrc users can read. One useful approach in a multi-user environment is to have a user called `egsnrc` and make the `HEN_HOUSE` the `$HOME` directory of that user. If you are the only local EGSnrc user, you can make the `HEN_HOUSE` a subdirectory of your `$HOME` area. In either case, when you initiate the `INSTALL_EGS` procedure, you should do so on the directory where you want the `HEN_HOUSE` to reside (see section 9.7 on page 303 for more on installation).

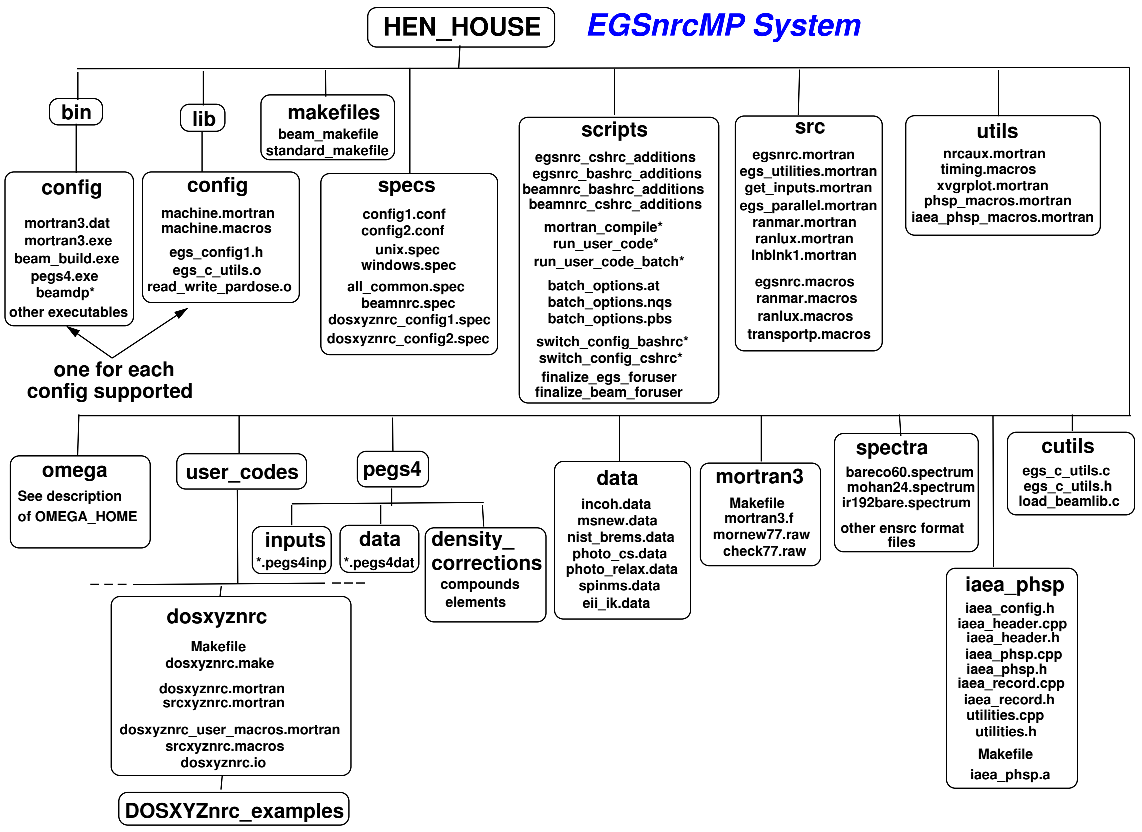


Figure 40: The components of the HEN_HOUSE area for the EGSnrc system. There are also subdirectories related to documentation and NRC user codes which are not shown here and the main HEN_HOUSE box actually has multiple subdirectories which are not shown. See PIRS-877 for details[14].

The second major directory area of the EGSnrc system is on the individual user's area and holds all the user's user codes and run associated files. It is mandatory that you set up a subdirectory which is referred to as `$EGS_HOME` (e.g., frequently `$HOME/egsnrc`) and for each `user_code.mortran` which you use or write, there must be a sub-directory with the same name as the user code, *i.e.* `$EGS_HOME/user_code`. A typical user's area is shown in figure 41.

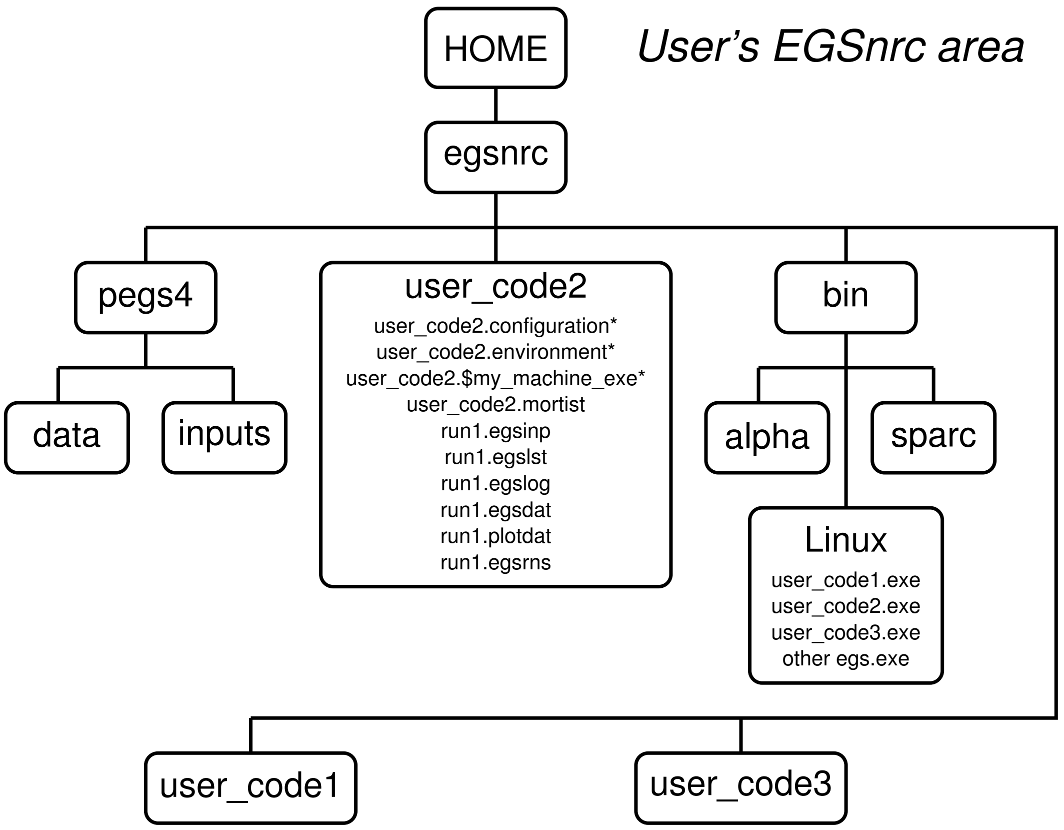


Figure 41: Components of a typical EGS user's area where, for this example, `$EGS_HOME` is `$HOME/egsnrc`. On a given system there can be an arbitrary number of different user's areas.

9.2.1 A complication for multi-architecture systems

The install scripts are clever enough to establish various machine or compiler dependencies and create the files `$HEN_HOUSE/lib/my_config/machine.macros` and `$HEN_HOUSE/lib/my_config/machine.mortran` where `my_config` is the name of your particular configuration (eg, g77, gfortran, Mac).

9.2.2 System aliases and environment variables

The EGSnrc system is based on using `csh` C-shell scripts or `sh`, bash shell scripts and to facilitate this one should run from the C-shell (`csh` or `tcsh`) or the bash shell `sh`. To define the aliases and other variables needed by the system, there are files called `egsnrc_cshrc_additions` and `egsnrc_bashrc_additions` available on `$HEN_HOUSE/scripts`. However, one first needs to set the environment variable `HEN_HOUSE` to point at the location of the `HEN_HOUSE`. This is most conveniently done with a declaration in your `.login` file or `.cshrc` file or `.bashrc` file depending on which shell is your default.

```
      setenv HEN_HOUSE location_of_HEN_HOUSE (e.g. $HOME/HEN_HOUSE)
```

or for `.bashrc`

```
      HEN_HOUSE=/absolute_location_of_your_HEN_HOUSE/ (e.g. $HOME/HEN_HOUSE)
```

Then it is **essential** that in your `.cshrc` or `.bashrc` file you source the `egsnrc_cshrc_additions` or `egsnrc_bashrc_additions` file, i.e. add the statement:

```
     source $HEN_HOUSE/scripts/egsnrc_cshrc_additions}
```

or, for those using `bash`:

```
     source $HEN_HOUSE/scripts/egsnrc_bashrc_additions}
or
     source /absolute_location_of_your_HEN_HOUSE/scripts/egsnrc_bashrc_additions}
```

These additions file set up many aliases and other variables for you. These are most easily seen by executing the commands `alias`, `set` and `setenv` although this will show you all of the systems definitions as well. We will explain the use of many of these definitions in the remainder of this section but the most important of the aliases are:

mor MORTRAN a stand alone code

m MORTRAN a specified users code

mf MORTRAN and Fortran compile and link a specified users code

f Fortran compile and link a specified users code

ex execute an egs user code

exb execute an egs user code in batch mode

The use of these is described below in detail but note that the command `make` is now the easiest way to mortran and fortran user codes as described in PIRS-877[14].

9.3 PEGS4

As described elsewhere in this report, PEGS4 is the program which prepares much of the material dependent cross section data sets required by EGSnrc to do the simulations (see section 6 on page 219). As discussed there, use of the `egs_gui` greatly facilitates the use of the `PEGS4` program.

The user must have the directories `$EGS_HOME/pegs4`, `$EGS_HOME/pegs4/inputs` and `$EGS_HOME/pegs4/data`. A user's PEGS4 input file is on `$EGS_HOME/pegs4/inputs` and has the name `my_data.pegs4inp`. The PEGS4 program outputs the data file for EGSnrc to `$EGS_HOME/pegs4/data/my_data.pegs4dat`. The listing file from the PEGS4 run is found on `$EGS_HOME/pegs4` and is called `my_data.pegslst`. To invoke PEGS4 one enters:

```
pegs4.exe -i inputfile [-o ofile] [-a] [-d density] [-x crosssection] [-e HEN_HOUSE]

inputfile.pegs4inp   the input file
output defaults to $HEN_HOUSE/pegs4/data/inputfile.pegs4dat
                or, if ofile is given, to $HEN_HOUSE/pegs4/data/ofile.pegs4dat
[-a]            => append results to output file
[-d density]    => use density.density   for density effect
[-x crosssection] => use $HEN_HOUSE/pegs4/crosssection instead of
                                $HEN_HOUSE/pegs4/pgs4pepr.dat
[-e HEN_HOUSE]  => use this absolute location as the HEN_HOUSE
```

where `my_data.pegsinp` is the input file described in detail in section 6.2 (page 246) and `density.density` is a file containing the density effect information needed for this particular calculation IF it is needed by `my_data.pegsinp`. The PEGS4 script checks the following directories, in order, for the file `density.density`.

```
$EGS_HOME/pegs4/inputs
$EGS_HOME/pegs4/density_corrections/elements
$EGS_HOME/pegs4/density_corrections/compounds
$EGS_HOME/pegs4/density_corrections
$HEN_HOUSE/pegs4/density_corrections/elements
$HEN_HOUSE/pegs4/density_corrections/compounds
```

9.3.1 Where pegs4 data is kept

Note that PEGS4 can only create one material output file per run and if the simulation you want to run requires data for more than one material, these must be joined into one `file.pegs4dat` file, either by concatenation, by using an editor or by the `-a` option in the command line.

The EGSnrc run scripts look for the data file requested (see section 9.5), first on `$EGS_HOME/pegs4/data` and if not found there, then it looks on `$HEN_HOUSE/pegs4/data`.

9.3.2 examin

The command:
`examin -p file`
(note there is no "e" at the end of `examin`) will allow you to plot and/or list much of the photon and electron cross section data in the file `$EGS_HOME/pegs4/data/file.pegs4dat` or if that is not present, then in the file `$HEN_HOUSE/pegs4/data/file.pegs4dat`. Note that `examin` no longer displays the photon cross section data from the PEGS4 data file but displays the data actually used by EGSnrc. The default dataset used by `examin` is xcom. If you are using something else in your user code, you must change the following line in the code:

```
REPLACE {$XDATA-DEFAULT} WITH {'xcom'}
```

The output is found on `$EGS_HOME/examin`. If requested, during execution `examin` pops open `xmgrace` (a 2-D graphing package), otherwise you have to use the listing files created. See section 3.13.3 (page 155).

9.4 Compilation (MORTRAN and Fortran)

The EGSnrc system is written in the Mortran3 language which is a pre-processor for Fortran. Mortran3 is itself written in Fortran. The Mortran3 processor is part of the EGSnrc distribution and is discussed in detail in section 8, page 274 of this report. In order to create an executable file from the mortran source files, there are two steps needed. The first step, converts the mortran source code into standard Fortran77. We call this `MORTRAN` compiling. The next step turns the Fortran source into an executable, using the standard Fortran compile and link process.

Using the EGSnrcMP system (see PIRS-877[14]), the most direct way to do these steps is using the `make` command. The following are the old way to do it but they still work.

To "compile" the user code `user_code.mortran`, one issues the command:

```
 mf user_code a [opt0|opt1|opt2|opt3|opt4]
```

or

```
m user_code
```

The `mf` command will MORTRAN, Fortran and link the entire user-code called `user_code`, including the EGSnrc system. The Fortran and the execute modules are machine specific (SUN vs SGI vs Linux etc) but this is handled in a transparent manner by the scripts.

The `m` command will MORTRAN the entire user-code called `user_code`, including the EGSnrc system. The output is a Fortran file `user_code_{my_machine}.f` which is specific for the machine being used since the script picks up a set of macros which handles any differences between the machines.

The `f` command will Fortran and link the entire Fortran version of the user-code called `user_code_{my_machine}.f`, which includes the EGSnrc system. The scripts automatically detect the value for `my_machine` and use the appropriate value.

`user_code.mortran` or `user_code_{my_machine}.f` must be present on area `$EGS_HOME/user_code`.

The letter `a` after the user code name is a residual option which no longer exists, but which has been left for compatibility with a wide range of other scripts at NRC. This option was associated with a technique to avoid recompiling unchanged Fortran subroutines from one pass to the next, but this was removed since compilers have become so fast that it was not very useful and occasionally led to problems and always left many extra files around. Note that the standard `make` facility does not work because the date on the Fortran file is always more recent than that on the object module since MORTRAN creates a new version of the Fortran each time.

The option `[opt0|opt1|opt2|opt3|opt4|debug]` allows selection of the optimisation level to be used. For production codes, be sure to use at least `opt2` since these can save up to 50% over `opt0`, but they usually take much longer to compile and link.

9.4.1 The user_code.configuration file (obsolete)

This approach has been completely revised and replaced by the `user_code.make` file with `SOURCES =` formats. See PIRS-877[14]. The same general capabilities have been maintained in the new format.

During the MORTRAN stage, the scripts still create a file, called `.mortjob.mortran` on `$EGS_HOME/user_code`. This file contains all the source code for this particular user_code (macros, user_code, and any other routines the user wants, etc). The files which are concatenated together to form `.mortjob.mortran` are defined by the user.

The `.configuration` file (or new equivalent `.make` file) allows for very flexible options when compiling a given user code. For example, it is possible to change which random number generator is being used throughout the system by merely changing which pair of files are included in the configuration file, either `ranlux.macros` and `ranlux.mortran` or `ranmar.macros` and `ranmar.mortran` (see section 3.9, page 143). However if anything other than the default seeds and luxury levels are desired, there may be some minor changes needed n `user_code.mortran` unless one is very careful (*eg.* DOSRZnrc is designed to use either rng).

9.4.2 Order is Important

The following still applies. When the file `.mortjob.mortran` is created, the order in which the various files are placed in the file is important. This is primarily because in Mortran3 the last definition of any macro is the one that is in force. Thus it is important that the `egsnrc.macros` file be read in early since this defines all of the default values of the macros. As long as it appears before the user code, then any macro redefinitions done in the user code take precedence.

9.4.3 machine.mortran files

Different compilers often have slightly different ways of handling certain things. The most common differences are regarding how files are opened (i.e., the format of the OPEN statement), and what is more difficult, how they handle time. EGSnrc itself has very few of these constructs within it, but the user codes cannot avoid them. To overcome these differences, there are a series of macros defined in machine.mortran files which are found on $HEN_HOUSE/lib/$my_machine/machine.mortran. It is even more complex for Linux machines where the machine.mortran files differ depending on what version of the compiler you are using. The default settings are for a fairly recent g77 compiler but if you are using a slightly older one you may need to edit the egs_compile script to select a different default for the variable Linux_compiler.

The machine.mortran files define a macro $MACHINE which the user codes pick up. It is useful to define this properly in the machine.mortran which is picked up in your local environment.

The above requirements are handled by a machine.mortran which is created for your system by the EGSnrcMP system as described in PIRS-877.

9.5 Execution

The following works mostly still,, but there are new ways described in PIRS-877[14] for the EGSnrcMP system.

Once a user code has been compiled, it is executed by issuing:

```
ex[b] user_code input data [queue|debug]
```

$EGS_HOME/user_code/input.egsinp and
$EGS_HOME/user_code/user_code.{my_machine}_exe must exist. The PEGS4 data is in data.pegs4dat which is located either on the users area or the system area, as discussed in section 9.3.1.

If queue is not specified when the batch option “b” is being used, then the default queue on your system is used.

In interactive mode, the 4-th input can be debug if a debug run is wanted. There must be an execute module created with the debug switch set during compilation for this to work.

Execution of EGSnrc jobs is handled by the script egs_run which is on the HEN_HOUSE.

At the completion of the run, all output and batch output files are found on $EGS_HOME/user_code/ unless the user has specified something different in the environment file (see below).

During execution, the egs_run script creates a separate subdirectory area and only modifies the files there. This is usually a subdirectory of $EGS_HOME/user_code/ but on a Linux system is on /tmp (this is to make the subdirectory local to the machine running the job since on the NRC system, having the disk on another machine would seriously slow down the job). All the files are moved back to `$EGS_HOME/user_code/` after completion of the run. This can be confusing unless you are aware of this behaviour.

9.5.1 user_code.environment file (obsolete)

This functionality is reworked and described in PIRS-877[14]. In particular the `user_code.io` file is relevant although the general ideas below are still applicable.

Almost all Fortran compilers associate a file called `fort.n` with any I/O unit `n` which is not attached explicitly within the code to some file name. Thus if `HATCH` read data from unit 12 without any open statement, then it looks for `fort.12`. The system must associate this file name to the real file name. There are a series of similar links established for files which have common names for all user codes.

The following files are associated with a normal type of run with an NRC user code (although the environment files can add whatever files the user may want and the files below are not mandatory). For this example, the input file name is `input.egsinp`.

input.egsinp: defined by the user.

input.egslst: the output listing file from the run. Erased at start of next run if not renamed.

input.egslog: The log file which echos the inputs and prompts when running in batch. Note, this has most of the **error** messages too, so get in the habit of reading this file.

input.egsdat: A file containing all the information needed to allow a calculation to be restarted again.

input.egsrns: Some codes allow a record of random number seeds at the start of each history to be kept (just most recent). These are kept in this file if requested.

input.egsplot (etc): Plotting files for various routines, the most up to date are for `xmgrace`.

input.egsgeom: file with output description of geometry for `EGS_windows`.

input.egsgph: file with output phase space of each history for `EGS_windows`.

9.5.2 Batch Queues (obsolete)

See PIRS-877 for how this is handled now.

9.6 Parallel Processing

The following is replaced by a more sophisticated approach in the EGSnrcMP system and the standard NRC user codes. See PIRS-877[14].

Monte Carlo calculations with EGSnrc are easily run in parallel and the results combined at the end of the runs because every history is completely independent, as long as the random number sequences are independent. One of the reasons for selecting the random number generators distributed with EGSnrc is that there is a simple procedure for ensuring independent sequences (see section 3.9 on page 143).

The NRC user codes[112] have a parallel processing option built in. The system used requires the NQS batch submission system to be installed on all machines to be run in parallel. The option is implemented via a script, called `pprocess` which makes an arbitrary number of input files from the original file, ensuring different random number sequences for each and then submitting the job to an arbitrary number of machines. At the end of the runs, the main user code is run once more with instructions to combine the results of the previous runs. Although crude, it is highly effective. We routinely run using more than 30 of our 38 CPUs running in parallel. The script `combine_egsnrc` has been written to automatically edit the required input file and execute the needed final run for the analysis of parallel runs with the NRC user codes. There is more discussion in reference [112].

Note that there is a useful script called `clean_after_parallel` which is found on `$HEN_HOUSE/scripts`. Parallel runs create a huge number of files and this script helps clean them up, but only once they have been analysed.

9.7 Distribution / Installation of EGSnrc

This installation script requires that the environment variables `HEN_HOUSE` not be defined when the script is called and not be defined in the user's `.cshrc` or `.bashrc` file. The installation will create the entire `$HEN_HOUSE` structure shown in fig 40 (page 295) and compile a variety of codes for the machine the user is doing the installation from (*viz.* Mortran3, PEGS4, DOSRZnrc). It also partially sets up the user's area `$EGS_HOME`.

Once the `INSTALL_EGS` script has been successfully executed, one must ensure that the environment variables `HEN_HOUSE` and `EGS_HOME` are defined to point at wherever you have placed the EGSnrc system and your EGSnrc user area.

Most importantly, the installation leaves behind a file called `egsnrc_cshrc_additions` or `egsnrc_bashrc_additions` which MUST be sourced from the user's `.cshrc` script or `.bashrc` script depending on which default shell is used. Once the installation is complete, add the following to your `.cshrc` file:
`source /explicit_path_to_your_HEN_HOUSE/egsnrc_cshrc_additions`.
or the following to your `.bashrc` file:
`source /explicit_path_to_your_HEN_HOUSE/egsnrc_bashrc_additions`.

The user should try to run the `tutor` codes to ensure the system is working.

References

[1] I. Kawrakow and A. F. Bielajew. On the representation of electron multiple elastic-scattering distributions for Monte Carlo calculations. *Nuclear Instruments and Methods*, 134B:325 – 336, 1998.

[2] A. F. Bielajew. A hybrid multiple-scattering theory for electron-transport Monte Carlo calculations. *Nucl. Inst. Meth.*, B111:195 – 208, 1996.

[3] I. Kawrakow. Electron transport: multiple and plural scattering. *Nuclear Instruments and Methods*, B108:23 – 34, 1996.

[4] I. Kawrakow and A. F. Bielajew. On the condensed history technique for electron transport. *Nuclear Instruments and Methods*, 142B:253 – 280, 1998.

[5] I. Kawrakow. Accurate condensed history Monte Carlo simulation of electron transport. I. EGSnrc, the new EGS4 version. *Med. Phys.*, 27:485 – 498, 2000.

[6] I. Kawrakow. Accurate condensed history Monte Carlo simulation of electron transport. II. Application to ion chamber response simulations. *Med. Phys.*, 27:499 – 513, 2000.

[7] Y. Namito, H. Hirayama, and S. Ban. Improvements of low-energy photon transport in EGS4. *Radiation Phys. Chem.*, 53:283 – 294, 1998.

[8] Y. Namito, S. Ban, and H. Hirayama. Compton scattering of 20 to 40-keV photons. *Phys. Rev. A*, 51:3036 – 3043, 1995.

[9] Y. Namito, S. Ban, and H. Hirayama. Implementation of Doppler broadening of Compton-scattered photons into the EGS4 code. *Nucl. Inst. Meth.*, A349:489 – 494, 1994.

[10] Y. Namito, S. Ban, and H. Hirayama. Implementation of linearly-polarized photon scattering into the EGS4 code. *Nuclear Instruments and Methods*, A322:277 – 283, 1993.

[11] W. R. Nelson, H. Hirayama, and D. W. O. Rogers. The EGS4 Code System. Report SLAC–265, Stanford Linear Accelerator Center, Stanford, California, 1985.

[12] B. R. B. Walters, J. Treurniet, D. W. O. Rogers, and I. Kawrakow. QA tests and comparisons of the EGSnrc system with EGS4. Technical Report PIRS–703, National Research Council of Canada, Ottawa, Canada, 2000.

[13] R. L. Ford and W. R. Nelson. The EGS code system – Version 3. *Stanford Linear Accelerator Center Report SLAC-210*, 1978.

[14] I. Kawrakow, E. Mainegra-Hing, and D. W. O. Rogers. EGSnrcMP: the multi-platform environment for EGSnrc. Technical Report PIRS–877, National Research Council of Canada, Ottawa, Canada, 2003.

[15] M. J. Berger. Monte Carlo Calculation of the penetration and diffusion of fast charged particles. In B. Alder, S. Fernbach, and M. Rotenberg, editors, *Methods in Comput. Phys.*, volume 1, pages 135 – 215. Academic, New York, 1963.

[16] A. F. Bielajew and D. W. O. Rogers. Electron Step-Size Artefacts and PRESTA. In T. M. Jenkins, W. R. Nelson, A. Rindi, A. E. Nahum, and D. W. O. Rogers, editors, *Monte Carlo Transport of Electrons and Photons*, pages 115 – 137. Plenum Press, New York, 1988.

[17] A. F. Bielajew and D. W. O. Rogers. PRESTA: The Parameter Reduced Electron-Step Transport Algorithm for electron Monte Carlo transport. *Nuclear Instruments and Methods*, B18:165 – 181, 1987.

[18] A. F. Bielajew, R. Mohan, and C. S. Chui. Improved bremsstrahlung photon angular sampling in the EGS4 code system. *National Research Council of Canada Report PIRS-0203*, 1989.

[19] J. W. Motz, H. A. Olsen, and H. W. Koch. Pair production by photons. *Rev. Mod. Phys.*, 41:581 – 639, 1969.

[20] H. Davies, H. A. Bethe, and L. C. Maximon. Theory of bremsstrahlung and pair production. II. Integral cross sections for pair production. *Phys. Rev.*, 93:788, 1954.

[21] E. Storm and H. I. Israel. Photon cross sections from 1 keV to 100 MeV for elements Z=1 to Z=100. *Atomic Data and Nuclear Data Tables*, 7:565 – 681, 1970.

[22] Y. S. Tsai. Pair Production and Bremsstrahlung of Charged Leptons. *Rev. Mod. Phys.*, 46:815, 1974.

[23] J. C. Butcher and H. Messel. Electron number distribution in electron-photon showers in air and aluminum absorbers. *Nucl. Phys.*, 20:15 – 128, 1960.

[24] Ingjald Øverbø, Kjell J. Mork, and A. Olsen. Pair production by photons: Exact calculations for unscreened atomic field. *Phys. Rev. A*, 8:668 – 684, 1973.

[25] V. Votruba. *Bull. Intern. Acad. Tcheque Sci.*, 49:19, 1948.

[26] Kjell J. Mork. Pair production by photons on electrons. *Phys. Rev.*, 160:1065 – 1071, 1967.

[27] A. F. Bielajew. Improved angular sampling for pair production in the EGS4 code system. *NRCC Report: PIRS-0287R*, 1994.

[28] O. Klein and Y. Nishina. Über die Streuung von Strahlung durch freie Elektronen nach der neuen relativistischen Quantendynamik von Dirac. *Z. für Physik*, 52:853–868, 1929.

[29] R. Ribberfors. . *Phys. Rev. B*, 12:2067, 1975.

[30] R. Ribberfors and K. F. Bergen. . *Phys. Rev. A*, 26:3325, 1982.

[31] F. Biggs, L. B. Mendelsohn, and J. B. Mann. . *At. Data and Nucl. Data Tables*, 16:201, 1975.

[32] D. Brusa, G. Stutz, J. A. Riveros, J. M. Fernández-Varea, and F. Salvat. Fast sampling algorithm for the simulation of photon Compton scattering. *Nuclear Instruments and Methods*, A379:167–175, 1996.

[33] F. Salvat, J. M. Fernandez-Varea, J. Baro, and J. Sempau. PENELOPE, an algorithm and computer code for Monte Carlo simulation of electron-photon showers. *University of Barcelona Report*, 1996.

[34] C. M. Lederer and V. S. Shirley (eds.). *Table of Isotopes, 7th ed.* (Wiley, New York), 1978.

[35] M. Abramowitz and I. A. Stegun, editors. *Handbook of mathematical functions with formulae, graphs and mathematical tables.* Number 55 in Applied Mathematics. National Bureau of Standards, Washington, D.C., 1964.

[36] L.M. Brown and R.P. Feynman. Radiative corrections to Compton scattering . *Phys. Rev.*, 85:231–244, 1952.

[37] M. J. Berger and J. H. Hubbell. XCOM: Photon Cross Sections on a Personal Computer. Report NBSIR87–3597, NIST, Gaithersburg, MD20899, 1987.

[38] E. E. Lewis and W. F. Miller. *Computational Methods of Neutron Transport.* John Wiley and Sons, New York, 1984.

[39] D. E. Cullen, M. H. Chen, J. H. Hubbell, S. T. Perkins, E. E. Plechaty, J. A. Rathkopf, and J. H. Scofield. Tables and Graphs of Photon-Interaction Cross Sections from 10 eV to 100 GeV Derived from the LLNL Evaluated Photon Data Library (EPDL), Part A: Z = 1 to 50, Part B: Z = 51 to 100. *Lawrence Livermore National Laboratory Report UCRL-50400, Volume 6, Revision 4 (Livermore, Calif)*, 1989.

[40] F. Sauter. Über den atomaren Photoeffekt in der K-Schale nach der relativistischen Wellenmechanik Diracs. *Ann. Physik*, 11:454 – 488, 1931.

[41] A. F. Bielajew and D. W. O. Rogers. Photoelectron angular distribution in the EGS4 code system. *National Research Council of Canada Report PIRS-0058*, 1986.

[42] J. H. Hubbell and I. Øverbø. Relativistic atomic form factors and photon coherent scattering cross sections. *J. Phys. Chem. Ref. Data*, 9:69, 1979.

[43] D. E. Peplow and K. Verghese. Measured molecular coherent scattering form factors of animal tissues, plastics and human breast tissue. *Phys. Med. Biol.*, 43:2431–52, 1998.

[44] D.E. Cullen, J.H. Hubbell, and L. Kissel. EPDL97: The evaluated photon data library,97 version. *Lawrence Livermore National Laboratory Report No. UCRL-50400*, 1997.

[45] S. T. Perkins, D. E. Cullen, M. H. Chen, J. H. Hubbell, J. A. Rathkopf, and J. H. Scofield. Tables and Graphs of Atomic Subshell and Relaxation Data Derived from the LLNL Evaluated Atomic Data Library (EADL), Z = 1–100. *Lawrence Livermore National Laboratory Report UCRL-50400, Volume 30 (Livermore, Calif)*, 1991.

[46] P. V. Vavilov. . *Soviet Physics JETP*, 5:749, 1957.

[47] H. W. Koch and J. W. Motz. Bremsstrahlung cross-section formulas and related data. *Rev. Mod. Phys.*, 31:920 – 955, 1959.

[48] S. M. Seltzer and M. J. Berger. Bremsstrahlung spectra from electron interactions with screened atomic nuclei and orbital electrons. *Nucl. Inst. Meth. Phys. Res. B* **12**, 12:95 – 134, 1985.

[49] S. M. Seltzer and M. J. Berger. Bremsstrahlung energy spectra from electrons with kinetic energy from 1 kev to 10 gev incident on screened nuclei and and orbital electrons of neutral atoms with z = 1-100. *Atomic Data and Nuclear Data Tables*, 35:345–418, 1986.

[50] ICRU. Stopping powers for electrons and positrons. ICRU Report 37, ICRU, Washington D.C., 1984.

[51] H. K. Tseng and R. H. Pratt. Exact screened calculations of atomic-field bremsstrahlung. *Phys. Rev. A*, 3:100–115, Jan 1971.

[52] Frederic Tessier and Iwan Kawrakow. Calculation of the electron-electron bremsstrahlung cross-section in the field of atomic electrons . *Nucl. Inst. Meth. B*, 266:625 – 634, 2008.

[53] D. W. O. Rogers, S. Duane, A. F. Bielajew, and W. R. Nelson. Use of ICRU-37/NBS radiative stopping powers in the EGS4 system. *National Research Council of Canada report PIRS-0177*, 1989.

[54] C. Møller. Zur Theorie des Durchgangs schneller Elektronen durch Materie. *Ann. Phys.*, 406:531–585, 1932.

[55] H. J. Bhabha. . *Proc. Royal Society London*, A154:195, 1935.

[56] A. F. Bielajew and D. W. O. Rogers. Effects of a Møller cross section error in the EGS4 code. *Med. Phys. (abstract)*, 23:1153, 1996.

[57] I Kawrakow. Electron impact ionization cross sections for egsnrc. *Med. Phys. (abstract)*, 29:1230, 2002.

[58] M. Gryziński. Two-particle collisions. I. General relations for collisions in the laboratory system. *Phys. Rev. A*, 138:305 – 321, 1965.

[59] E. Casnati, A. Tartari, and C. Baraldi. An empirical approach to K-shell ionisation cross section by electrons. *J. Phys. B*, 15:155 – 167, 1982.

[60] H. Kolbenstvedt. Simple theory for K-ionization by relativistic electrons. *J. Appl. Phys.*, 38:4785 – 4787, 1967.

[61] D. Bote and F. Salvat. Calculations of inner-shell ionization by electron impact with the distorted-wave and plane-wave Born approximations. *Physical Review A*, 77(4):42701, 2008.

[62] H. Messel and D. F. Crawford. Electron-photon shower distribution function. *(Pergamon Press, Oxford)*, 1970.

[63] M. J. Berger and S. M. Seltzer. Tables of energy losses and ranges of electrons and positrons. *NASA Report SP-3012 (Washington DC)*, 1964.

[64] H. A. Bethe. Theory of passage of swift corpuscular rays through matter. *Ann. Physik*, 5:325, 1930.

[65] H. A. Bethe. Scattering of electrons. *Z. für Physik*, 76:293, 1932.

[66] F. Bloch. Stopping power of atoms with several electrons. *Z. für Physik*, 81:363, 1933.

[67] R. M. Sternheimer and R. F. Peierls. General expression for the density effect for the ionization loss of charged particles. *Phys. Rev.*, B3:3681 – 3692, 1971.

[68] M. J. Berger and S. M. Seltzer. Stopping power and ranges of electrons and positrons. *NBS Report NBSIR 82-2550-A (second edition)*, 1983.

[69] R. M. Sternheimer, S. M. Seltzer, and M. J. Berger. Density effect for the ionization loss of charged particles in various substances. *Phys. Rev.*, B26:6067, 1982.

[70] S. Duane, A. F. Bielajew, and D. W. O. Rogers. Use of ICRU-37/NBS collision stopping powers in the EGS4 system. *NRCC Report PIRS-0173, Ottawa, March*, 1989.

[71] G. Z. Molière. Theorie der Streuung schneller geladener Teilchen. II. Mehrfach- und Vielfachstreuung. *Z. Naturforsch*, 3a:78 – 97, 1948.

[72] G. Z. Molière. Theorie der Streuung schneller geladener Teilchen. I. Einzelstreuung am abgeschirmten Coulomb-Field. *Z. Naturforsch*, 2a:133 – 145, 1947.

[73] X. A. Li and D. W. O. Rogers. Electron mass scattering powers: Monte Carlo and analytical calculations. *Med. Phys.*, 22:531 – 541, 1995.

[74] N. F. Mott. . *Proc. Royal Society London*, A124:425, 1929.

[75] J. W. Motz, H. A. Olsen, and H. W. Koch. Electron scattering with atomic or nuclear excitation. *Rev. Mod. Phys.*, 36:881 – 928, 1964.

[76] N. F. Mott. . *Proc. Royal Society London*, A135:429, 1932.

[77] M. J. Berger and R. Wang. Multiple-scattering angular deflections and energy-loss straggling. In T. M. Jenkins, W. R. Nelson, A. Rindi, A. E. Nahum, and D. W. O. Rogers, editors, *Monte Carlo Transport of Electrons and Photons*, pages 21 – 56. Plenum Press, New York, 1988.

[78] S. M. Seltzer. An overview of ETRAN Monte Carlo methods. In T. M. Jenkins, W. R. Nelson, A. Rindi, A. E. Nahum, and D. W. O. Rogers, editors, *Monte Carlo Transport of Electrons and Photons*, pages 153 – 182. Plenum Press, New York, 1988.

[79] M. E. Riley. Relativistic Elastic Electron Scattering from Atoms at Energies Greater than 1 keV. Report SLA–74-0107, Sandia Laboratories, 1974.

[80] J. P. Desclaux. A Multiconfiguration Relativistic Dirack-Fock Program. *Comp. Phys. Commun.*, 9:31–45, 1975.

[81] J. A. Halbleib, R. P. Kensek, T. A. Mehlhorn, G. D. Valdez, S. M. Seltzer, and M. J. Berger. ITS Version 3.0: The Integrated TIGER Series of Coupled Electron/Photon Monte Carlo Transport Codes. *Sandia report SAND91-1634*, 1992.

[82] W. T. Scott. The theory of small-angle multiple scattering of fast particles. *Rev. Mod. Phys.*, 35:231 – 313, 1963.

[83] U. Fano. Inelastic Collisions and the Moliere Theory of Multiple Scattering. *Phys. Rev.*, 93:117 – 120, 1954.

[84] I. Kawrakow. Improved modeling of multiple scattering in the Voxel Monte Carlo model. *Med. Phys.*, 24:505 – 517, 1997.

[85] S. A. Goudsmit and J. L. Saunderson. Multiple scattering of electrons. *Phys. Rev.*, 57:24 – 29, 1940.

[86] S. A. Goudsmit and J. L. Saunderson. Multiple scattering of electrons. II. *Phys. Rev.*, 58:36 – 42, 1940.

[87] G. J. Lockwood, L. E. Ruggles, G. H. Miller, and J. A. Halbleib. Calorimetric measurement of electron energy deposition in extended media—Theory versus measurement. *Sandia report SAND79-0414*, 1980.

[88] D. W. O. Rogers, B. A. Faddegon, G. X. Ding, C.-M. Ma, J. Wei, and T. R. Mackie. BEAM: A Monte Carlo code to simulate radiotherapy treatment units. *Med. Phys.*, 22:503 – 524, 1995.

[89] E. W. Larsen. A theoretical derivation of the condensed history algorithm. *Ann. Nucl. Energy*, 19:701 – 714, 1992.

[90] J. M. Fernández-Varea, R. Mayol, J. Baró, and F. Salvat. On the theory and simulation of multiple elastic scattering of electrons. *Nuclear Instruments and Methods*, B73:447 – 473, 1993.

[91] I. Kawrakow. Electron transport: longitudinal and lateral correlation algorithm. *Nuclear Instruments and Methods*, B114:307 – 326, 1996.

[92] H. W. Lewis. Multiple scattering in an infinite medium. *Phys. Rev.*, 78:526 – 529, 1950.

[93] A. F. Bielajew, D. W. O. Rogers, and A. E. Nahum. Monte Carlo simulation of ion chamber response to ^{60}Co – Resolution of anomalies associated with interfaces. *Phys. Med. Biol.*, 30:419 – 428, 1985.

[94] B. J. Foote and V. G. Smyth. The modelling of electron multiple-scattering in EGS4/PRESTA and its effect on ionization-chamber response. *Nucl. Inst. Meth.*, B100:22 – 30, 1995.

[95] D. W. O. Rogers. Low energy electron transport with EGS. *Nucl. Inst. Meth.*, 227:535 – 548, 1984.

[96] C.-M. Ma and A. E. Nahum. A new algorithm for EGS4 low-energy electron transport to account for the change in discrete interaction cross-section with energy. *Nucl. Instr. Meth.*, B72:319 – 330, 1992.

[97] A. F. Bielajew. Electron Transport in $\vec{E}$ and $\vec{B}$ Fields. In T. M. Jenkins, W. R. Nelson, A. Rindi, A. E. Nahum, and D. W. O. Rogers, editors, *Monte Carlo Transport of Electrons and Photons*, pages 421 – 434. Plenum Press, New York, 1988.

[98] A. J. Cook. Mortran3 user's guide. *SLAC Computa. Res. Group Tech. Memo. No. CGTM 209*, 1983.

[99] A. F. Bielajew. The effect of strong longitudinal magnetic fields on dose deposition from electron and photon beams. *Med. Phys.*, 20:1171 – 1179, 1993.

[100] D. W. O. Rogers, I. Kawrakow, J. P. Seuntjens, B. R. B. Walters, and E. Mainegra-Hing. NRC User Codes for EGSnrc. Technical Report PIRS–702(RevB), National Research Council of Canada, Ottawa, Canada, 2003.

[101] D. E. Cullen, S. T. Perkins, and J. A. Rathkopf. The 1989 Livermore Evaluated Photon Data Library (EPDL). *Lawrence Livermore National Laboratory Report UCRL-ID-103424 (Livermore, Calif)*, 1990.

[102] E. Casnati, A. Tartari, and C. Baraldi. An empirical approach to K-shell ionisation cross section by electrons. *J. Phys. B*, 16:505, 1983.

[103] M. Gryziński. Two-particle collisions. II. Coulomb collisions in the laboratory system of coordinates. *Phys. Rev. A*, 138:322 – 335, 1965.

[104] M. Gryziński. Classical theory of atomic collisions. I. Theory of inelastic collisions. *Phys. Rev. A*, 138:336 – 358, 1965.

[105] B. A. Faddegon, C. K. Ross, and D. W. O. Rogers. Angular distribution of bremsstrahlung from 15 MeV electrons incident on thick targets of Be, Al and Pb. *Med. Phys.*, 18:727 – 739, 1991.

[106] A. F. Bielajew. HOWFAR and HOWNEAR: Geometry Modeling for Monte Carlo Particle Transport. *National Research Council of Canada Report PIRS-0341*, 1995.

[107] G. Marsaglia and A. Zaman. A new class of random number generators. *Annals of Applied Probability*, 1:462 – 480, 1991.

[108] G. Marsaglia, A. Zaman, and W. W. Tsang. Toward a universal random number generator. *Statistics and Probability Letters*, 8:35 – 39, 1990.

[109] M. Lüscher. A portable high-quality random number generator for lattice field theory simulations. *Computer Phys. Commun.*, 79:100 – 110, 1994.

[110] F. James. RANLUX: A Fortran implementation of the high-quality pseudorandom number generator of Lüscher. *Computer Phys. Commun.*, 79:111 – 114, 1994.

[111] D. W. O. Rogers, C.-M. Ma, G. X. Ding, B. Walters, D. Sheikh-Bagheri, and G. G. Zhang. BEAM98 Users Manual. *NRC Report PIRS 509(a)revC*, 1998.

[112] D. W. O. Rogers, I. Kawrakow, J. P. Seuntjens, and B. R. B. Walters. NRC User Codes for EGSnrc. Technical Report PIRS–702, National Research Council of Canada, Ottawa, Canada, 2000.

[113] J. A. Treurniet and D. W. O. Rogers. EGS_Windows4.0 User's Manual. *NRC Report PIRS–0669*, 1999.

[114] D. W. O. Rogers. More realistic Monte Carlo calculations of photon detector response functions. *Nucl. Instrum. Meth.*, 199:531 – 548, 1982.

[115] J. Sempau and A. F. Bielajew. Towards the elimination of Monte Carlo statistical fluctuations from dose volume histograms for radiotherapy treatment planning. *Phys. Med. Biol.*, 45:131 – 157, 2000.

[116] C.-M. Ma, D. W. O. Rogers, and B. Walters. DOSXYZnrc Users Manual. *NRC Report PIRS 509b(revF)*, 2001.

[117] A. F. Bielajew and D. W. O. Rogers. A standard timing benchmark for EGS4 Monte Carlo calculations. *Med. Phys.*, 19:303 – 304, 1992.

[118] D. W. O. Rogers and A. F. Bielajew. Monte Carlo techniques of electron and photon transport for radiation dosimetry. In K. R. Kase, B. E. Bjärngard, and F. H. Attix, editors, *The Dosimetry of Ionizing Radiation, Vol III*, pages 427 – 539. Academic Press, 1990.

[119] J. H. Hubbell and S. M. Seltzer. Tables of X-Ray Mass Attenuation Coefficients and Mass Energy-Absorption Coefficients 1 keV to 20 MeV for Elements Z = 1 to 92 and 48 Additional Substances of Dosimetric Interest. Technical Report NISTIR 5632, NIST, Gaithersburg, MD 20899, 1995.

Index