THE EGS5 CODE SYSTEM[1]

Hideo Hirayama and Yoshihito Namito
Radiation Science Center
Advanced Research Laboratory
High Energy Accelerator Research Organization (KEK)
1-1 Oho Tsukuba-shi Ibaraki-ken 305-0801 JAPAN

Alex F. Bielajew and Scott J. Wilderman
Department of Nuclear Engineering and Radiological Sciences
The University of Michigan
2355 Bonisteel Boulevard
Ann Arbor, MI 48109, USA

Walter R. Nelson
Department Associate in the Radiation Physics Group (retired)
Radiation Protection Department
Stanford Linear Accelerator Center
Stanford University
2575 Sand Hill Road, Menlo Park, CA 94025, USA

Date of this version: January 14, 2009

Stanford University Notices for
SLAC Manual SLAC-R-730
and its included software
known as the EGS5 Code System

i

Contents

E CONTENTS OF THE EGS5 DISTRIBUTION   **404**

List of Figures

List of Tables

xiii

PREFACE

In the nineteen years since EGS4 was released, it has been used in a wide variety of applications, particularly in medical physics, radiation measurement studies, and industrial development. Every new user and every new application bring new challenges for Monte Carlo code designers, and code refinements and bug fixes eventually result in a code that becomes difficult to maintain. Several of the code modifications represented significant advances in electron and photon transport physics, and required a more substantial invocation than code patching. Moreover, the arcane MORTRAN3[48] computer language of EGS4, was highest on the complaint list of the users of EGS4. The size of the EGS4 user base is difficult to measure, as there never existed a formal user registration process. However, some idea of the numbers may be gleaned from the number of EGS4 manuals that were produced and distributed at SLAC: almost three thousand. Consequently, the EGS5 project was undertaken. It was decided to employ the FORTRAN 77 compiler, yet include as much as possible, the structural beauty and power of MORTRAN3.

We wish to acknowledge Patrick Lui (Technology Transfer Office at SLAC) for administrating the business end of this project and securing sufficient funds to enable its completion. We wish to acknowledge Dr. Ray Cowan (MIT/SLAC BaBar) for converting the SLAC-265 document, written in TeX/Psizzl to LaTeX. SLAC 265 served as the foundation upon which this document is built. Finally, we wish to acknowledge Prof. David W. O. Rogers, co-author of EGS4, for his many years of EGS4 development and Monte Carlo leadership. We also wish to thank Dr. James Liu (SLAC Radiation Physics Group) for helping to create and run the suite of benchmarking codes for EGS5. Dr. Francesc Salvat's assistance in developing the sampling methods for the multiple elastic scattering distributions is most gratefully acknowledged.

This report consists of four chapters and several appendices. Chapter 1 is an introduction to EGS5 and to this report in general. We suggest that you read it. Chapter 2 is a major update of similar chapters in the old EGS4 report[126] (SLAC-265) and the old EGS3 report[61] (SLAC-210), in which all the details of the old physics (*i.e.*, models which were carried over from EGS4) and the new physics are gathered together. The descriptions of the new physics are extensive, and not for the faint of heart! Detailed knowledge of the contents of Chapter 2 is not essential in order to use EGS, but sophisticated users should be aware of its contents. In particular, details of the restrictions on the range of applicability of EGS are dispersed throughout the chapter. First-time users of EGS should skip Chapter 2 and come back to it later if necessary.

With the release of the EGS4 version, a deliberate attempt was made to present example problems in order to help the user "get started," and we follow that spirit in this report. A series of elementary tutorial user codes are presented in Chapter 3, with more sophisticated sample user codes described in Chapter 4. Novice EGS users will find it helpful to read through the initial sections of the EGS5 User Manual (provided in Appendix B of this report), proceeding then to work through the tutorials in Chapter 3.

The User Manuals and other materials found in the appendices contain detailed flow charts, variable lists, and subprogram descriptions of EGS5 and PEGS. Included are step-by-step instructions for developing basic EGS5 user codes and for accessing all of the physics options available in EGS5 and PEGS. Once acquainted with the basic structure of EGS5, users should find the appendices the most frequently consulted sections of this report.

H. H.

Y. N.

A. F. B.

S. J. W.

W. R. N.

December 2005

Chapter 1

INTRODUCTION

1.1   Intent of This Report

The EGS (Electron-Gamma Shower) code system 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 hundred GeV (depending on the atomic numbers of the target materials). This report introduces a new, enhanced version called EGS5. In addition to explaining and documenting the various enhancements and changes to the previous version (EGS4[126]), this document includes several introductory and advanced tutorials on the use of EGS5, and also contains the EGS5 User Manual. Our intention has been to make this document wholly self-contained so that the user need not refer to the original EGS4 manual (SLAC-265[126]) in order to use the code. To this end, we have taken the liberty of incorporating into Chapter 2 of this report those portions of Chapter 2 of SLAC-265 which describe physics models of EGS4 retained by EGS5, thereby documenting all the physics contained in EGS5.

On the practical side this report can be used in order to create EGS5 user codes and to run EGS5. Formerly a stand-alone data preparation code, PEGS (Preprocessor for EGS), is now integrated in this version of EGS.

1.2   History of EGS

A great deal of the content of the following historical sections of this report has been plagiarized unashamedly from the original EGS3 document (SLAC-210) authored by Richard Ford and Ralph Nelson[61]. There are several reasons for this aside from laziness. This history predates one of the author's (AFB) involvement with EGS and he found it very difficult to improve upon the words penned by Ford and Nelson in that original document. Moreover, the EGS3 manual is now out-

of-print and this history might have eventually been lost to the ever-burgeoning EGS-community now estimated to be at least 7000 strong. There had been one previous attempt to give a historical perspective of EGS[125]. However, this article was very brief and did not convey the large effort that went into the development of EGS. In this report the historical section on EGS4 as well as the summary of EGS3 to EGS4 conversion and the overview of EGS4 was taken directly from the EGS4 manual[126]. The technical descriptions herein are necessarily brief. The new physics of EGS5 is described in detail in the next Chapter. However, the reader should consult the references cited in this report for more details regarding motivation and implementation.

1.2.1 Before EGS

The Monte Carlo method was originally suggested by Ulam and von Neumann[174], and was first used by Goldberger[62] in order to study nuclear disintegrations produced by high-energy particles. The first application of the Monte Carlo technique to study shower production was done by Wilson[179]. Wilson's approach was a simple graphical-mechanical one that was described as follows:

"The procedure used was a simple graphical and mechanical one. The distance into the lead was broken into intervals of one-fifth of a radiation length (about 1 mm). The electrons or photons were followed through successive intervals and their fate in passing through a given interval was decided by spinning a wheel of chance; the fate being read from one of a family of curves drawn on a cylinder...

A word about the wheel of chance; The cylinder, 4 in. outside diameter by 12 in. long is driven by a high speed motor geared down by a ratio 20 to 1. The motor armature is heavier than the cylinder and determines where the cylinder stops. The motor was observed to stop at random and, in so far as the cylinder is concerned, its randomness is multiplied by the gear ratio..."

from R. R. Wilson, *op. cit.*

Although apparently quite tedious, Wilson's method was still an improvement over the analytic methods of the time—particularly in studying the average behavior and fluctuations about the average[141].

The first use of an electronic digital computer in simulating high-energy cascades by Monte Carlo methods was reported by Butcher and Messel[38, 39], and independently by Varfolomeev and Svetlolobov[176]. These two groups collaborated in a much publicized work[104] that eventually led to an extensive set of tables describing the shower distribution functions[103]—the so-called

"shower book".

For various reasons two completely different codes were written in the early-to-mid-1960's to simulate electromagnetic cascades. The first was written by Zerby and Moran[180, 181, 182] at the Oak Ridge National Laboratory and was motivated by the construction of the Stanford Linear Accelerator Center and the many physics and engineering problems that were anticipated as a result of high-energy electron beams showering in various devices and structures at that facility. This code had been used by Alsmiller and others[5, 6, 7, 2, 8, 3, 4] for a number of studies since its development[1].

The second code was originally developed by Nagel[115, 112, 113] and several adaptations had been reported[177, 128, 37]. The original Nagel version, which Ford and Nelson called SHOWER1, was a FORTRAN code written for high-energy electrons ($\leq$ 1000 MeV) incident upon lead in cylindrical geometry. Six significant electron and photon interactions (bremsstrahlung, electron-electron scattering, ionization-loss, pair-production, Compton scattering, and the photoelectric effect) plus multiple Coulomb scattering were accounted for. Except for annihilation, positrons and electrons were treated alike and were followed until they reached a cutoff energy of 1.5 MeV (total energy). Photons were followed down to 0.25 MeV. The cutoff energies were as low as or lower than those used by either Messel and Crawford or by Zerby and Moran.

The availability of Nagel's dissertation[112] and a copy of his original shower program provided the incentive for Nicoli[128] to extend the dynamic energy range and flexibility of the code in order for it to be made available as a practical tool for the experimental physicist. Nicoli's modifications of SHOWER1 fell into three categories:

1. High-energy extensions to the least-squares fits for total interaction probabilities and branching ratios.
2. Provisions for including boundary-condition interrogation in the transport cycle, allowing for particle marking and/or discarding and the use of generalized energy cutoffs for electrons and photons.
3. The handling of input/output requirements.

In August, 1966 the Nicoli version (SHOWER2) was brought to SLAC by Nagel, who had been working at MIT and had consulted with Nicoli on the above changes and extensions. The SLAC Computation Group undertook the task of getting the code running on the IBM-360 system and generalizing the program to run in elemental media other than just lead. The latter was facilitated by a set of hand-written notes—brought to SLAC in 1966 by Nagel[114]—on the best way to accomplish this task and V. Whitis was assigned the job. Whitis left SLAC in the summer of 1967 and his work, which consisted mainly of a series of fitting programs written in the ALGOL language, was passed on to one of us (WRN)[2]. Under Nelson's direction, a programmer (J. Ryder)

---

[1]According to Alsmiller[1], the Zerby and Moran source code vanished from ORNL and they were forced to work with an octal version.

[2]Nagel's computer program was recovered from a trash receptacle at SLAC by WRN. Although no printout of the code could be found, the punch cards had been sequenced in columns 73-80 and they were easily sorted by machine.

constructed SHOWER3 in modular form and wrote a pre-processing code called PREPRO that computed fit-coefficients for the cross-section and branching-ratio data needed by SHOWER3. The values of these constants depended on the material in which the shower was to be simulated. During the summer of 1972 the Ryder version of SHOWER3/PREPRO was successfully tested for several different elements by B. Talwar under the direction of Nelson, thus bringing SHOWER3/PREPRO into an operational status.

Meanwhile, interest in a computer code capable of simulating electromagnetic cascade showers had been developing for several years at the High Energy Physics Laboratory (HEPL) at Stanford University where a group led by R. Hofstadter and E. B. Hughes was continuing their development of large NaI(Tl) Total Absorption Shower Counters (TASC's)[60]. A method of accurately predicting shower behavior in these counters was needed. A version of Nagel's code (SHOWER2) was obtained from Nelson in the fall of 1970; however, efforts to scale from lead to NaI were uncertain and led to a growing conviction that a generalized code was necessary. Thus it was that Richard Ford undertook the task of generalizing SHOWER2 to run in any element, mixture, or compound in September, l971—an effort similar to the one already underway by Ryder, Talwar, and Nelson that resulted in the final version of SHOWER3/PREPRO.

Ford obtained a copy of a Ryder version of SHOWER3/PREPRO and Nagel's notes from Nelson. In addition to the references mentioned in Nagel's notes, the Messel and Crawford "shower book"[103], as well as the review by Scott[147] on multiple scattering, were found to be very useful sources of information. The essential physics was formulated and the coding was completed by February, 1972. At that time the HEPL version was called SHOWER (now referred to as SHOWER4) and the corresponding preprocessor was completely new and was called SHINP (for **SH**ower **INP**ut). Both codes were in FORTRAN and were made operational on the IBM-7700 machine at HEPL—a second generation experimental data-acquisition computer. A number of interesting studies were subsequently performed, including calculations of detector resolutions and expected self-vetoes in gamma detectors due to backscattered photons from shower detectors downstream.

In January, 1974 it appeared likely that HEPL's computer would be sold. In addition, the Hofstadter group was involved in an experiment at SLAC that required shower simulations and the SHOWER4/SHINP codes were therefore made operational on the considerably faster and more efficient IBM-360/91 at SLAC. During the calculations that had been performed at HEPL, a couple of errors were found in the sampling routines that would have been detected earlier if it had been possible to test this in a more systematic way. Therefore, it was decided to incorporate into the new version being brought to SLAC test facilities to insure the correctness of these sampling routines. In order to facilitate comparison between the sampled secondary spectra and the theoretical distributions, the preprocessing code was split up and modularized into subprograms.

1.2.2 EGS1

About this time Nelson became interested in being able to use Ford's version of the code and offered to help support its further development. One of Ford's objectives was to make the preprocessor code produce data for the shower code in a form that was <u>directly</u> usable by the shower code with a minimum of input required by the user. In SHOWER3/PREPRO and in SHOWER4/SHINP, whenever it was desired to create showers in a new medium, it was necessary to look-up the photon cross sections in the literature and keypunch them for the preprocessing code to use. Subsequent to this it was necessary to select from several fits produced by the preprocessing code and to include this new information, consisting of many data cards, with other data used by the shower program. Ford rewrote the preprocessor to automatically produce all of the data needed by the shower code in a readily acceptable form and, with the assistance of Nelson, obtained photon cross sections for elements 1 to 100 from Storm and Israel[167] on magnetic tape. Ford also separated the shower code's material-input from its control-input. For flexibility and ease of use, the `NAMELIST` read facility of FORTRAN-IV was utilized for reading-in control data in both the preprocessor and the shower codes. The resultant shower code was re-named EGS (<u>E</u>lectron-<u>G</u>amma-<u>S</u>hower) and its companion code was called PEGS (<u>P</u>reprocessor for <u>EGS</u>). This version, written completely in the FORTRAN-IV language, is referred to as Version 1 of the EGS Code System (or more simply EGS1 and PEGS1).

The sampling routines were tested using the internal test-procedure facility of EGS1 and, with the exception of the bremsstrahlung process, were found to be operating very nicely. In the bremsstrahlung case a ripple, amounting to only 5% but still noticeable, was observed when the sampled data were compared with the theoretical secondary distribution. This effect went away upon selection of another random number generator, and it was concluded that correlations in the original number generator were the cause. EGS1 was then tested against various experiments in the literature and with other Monte Carlo results that were then available and the authors found reasonable good agreement in all cases.

1.2.3 EGS2

By the fall of 1974 the Hofstadter group had obtained some hexagonal modular NaI detectors and the discovery of the J/ψ particle[12, 13] in November, 1974 opened up an exciting area of high-energy gamma-ray spectroscopy for which the modularized NaI detectors were ideally suited. EGS1, however, could not be readily used to simulate showers in complex geometries such as those presented by modular stacks of NaI. A good example of this was the Crystal Ball detector for which EGS1, under the direction of E. Bloom at SLAC, was modified to handle the particular geometry in question. Furthermore, Nelson had received a large number of requests from the growing list of EGS users, both at SLAC and elsewhere in the high-energy physics community, to improve further EGS1 so that complex geometries could be realized in the near future. Thus it was decided that EGS1, which was a one-region, one-medium code, should be generalized in order to handle many-region, many-media, complex, three-dimensional geometries.

5

It soon became clear that, in the time available at least, it would not be possible to construct a self-contained code that would have all of the control, scoring, and output options that might ever be wanted, as well as a geometry package that would automatically handle arbitrary complex geometries. Therefore, Ford decided to put in only the necessary multi-region structures, to replace all scoring and output code in EGS1 with a user interface, and to dispense with the EGS1 main control program completely. Thus EGS1 became a subprogram in itself with two <u>user-callable</u> subroutines (HATCH and SHOWER) that require two <u>user-written</u> subroutines (HOWFAR and AUSGAB) in order to define the geometry and do the scoring, respectively.

For added flexibility and portability, EGS1 and PEGS1 were rewritten in an extended FOR-TRAN language called MORTRAN2, which was translated by a (MORTRAN2) Macro Processor into standard FORTRAN. The part of EGS1 that was used to test the sampling routines was reconfigured into a separate main program called the TESTSR code, also written in MORTRAN2. These revisions were completed by the end of 1975 and the new versions of EGS, PEGS, and TESTSR comprise what is called Version 2 of the EGS Code System, or more simply EGS2, PEGS2, and TESTSR2.

1.2.4   EGS3

One part of EGS2 which seemed aesthetically displeasing was the complex control logic needed in the electron transport routine, ELECTR, in order to transport electrons by variable distances to interaction points or boundaries using only step lengths taken from a set of 16 discrete values. This procedure had been necessary in order to implement Nagel's discrete reduced-angle multiple-scattering scheme[115, 112, 113, 114] in a general multi-region environment. In addition, comparisons of backscattered photon fluence as computed by EGS2 versus unpublished HEPL data, as well as bremsstrahlung angular distribution calculations comparing EGS2 results with those of Berger and Seltzer using ETRAN[19], suggested that EGS2 might be predicting values in the backward direction that were low by up to a factor of two. For these reasons, and in order to achieve greater universality of application (*e.g.*, so that a monoenergetic beam of electrons impinging on a very thin slab would have a continuous angular distribution on exit), Ford decided in the summer of 1976 to try to implement a multiple scattering-scheme that would correctly sample the continuous multiple-scattering distribution for arbitrary step lengths. After some thought, an extension of the method used by Messel and Crawford[103] was devised. Most of the code for this addition was written by Ford at Science Applications, Inc., and was brought to SLAC in August 1977 where it was debugged and tested by Nelson and Ford. The implementation of this system required some once-only calculations which were made using a stand-alone code called CMS (<u>C</u>ontinuous <u>M</u>ultiple <u>S</u>cattering)[3]. It should be mentioned that the version of PEGS brought to SLAC at this time had the same physics in it as Version 2, but had been partly rewritten in order to be more machine independent (*e.g.*, IBM versus CDC), its main remaining machine dependency being its use of NAMELIST. (NAMELIST is a common extension to FORTRAN employed by many FORTRAN compilers but is not part of the FORTRAN-IV or FORTRAN 77 standards.) Another option was added to the TESTSR code to allow testing of the new EGS multiple-scattering sampling routine,

---

[3]Logically, the CMS code should have been included as an option of PEGS, but this has never been done.

`MSCAT.`

These versions of EGS, PEGS, and TESTSR comprise what was called Version 3 of the EGS code system (*i.e.*, EGS3, PEG3, and TESTSR3). Subsequent comparisons of EGS3 calculations against experiments and other Monte Carlo results were made by the authors (*e.g.*, see SLAC-210[61] and/or Jenkins and Nelson[127]) and others and the agreements clearly demonstrated the basic validity of the code.

The EGS3 Code System released in 1978 contained many features that distinguished it from Nagel's original code, SHOWER1, the most noteworthy being:

1. Showers could be simulated in any element (Z=1 through 100), compound, or mixture.
2. The energy range for transporting particles was extended so that showers could be initiated and followed from 100 GeV down to 1 keV for photons, and 1.5 MeV (total energy) for charged particles.
3. Photons and charged particles were transported in random rather than discrete steps, resulting in a much faster running code.
4. Positrons were allowed to annihilate either in-flight or at rest, and their annihilation quanta were followed to completion.
5. Electrons and positrons were treated separately using exact, rather than asymptotic, Møller and Bhabha cross sections, respectively.
6. Sampling schemes were made more efficient.
7. EGS3 became a subroutine package with user interface, allowing much greater flexibility and reducing the necessity for being familiar with the internal details of the code. This also reduced the likelihood that user edits could introduce bugs into the code.
8. The geometry had to be specified by the user by means of a <u>user-written</u> subprogram called `HOWFAR`. However, geometry utilities for determining intersections of trajectories with common surfaces (*e.g.*, planes, cylinders, cones, spheres and boxes) had also been developed and were made available.
9. The task of creating media data files was greatly simplified and automated by means of the PEGS3 preprocessing code, which created output data in a convenient form for direct use by EGS3.
10. PEGS3 constructed piecewise-linear fits over a large number of energy intervals of the cross-section and branching-ratio data, whereas PREPRO and SHINP both made high-order polynomial fits over a small number of intervals (as did SHOWER1 and SHOWER2).
11. In addition to the options needed to produce data for EGS3, options were made available in PEGS3 for plotting any of the physical quantities used by EGS3, as well as for comparing sampled distributions from the TESTSR user code with theoretical spectra. The `NAMELIST` read facility of FORTRAN was also introduced at this time.

<u>In particular, for Version 3 versus Version 2</u>

12. The multiple-scattering reduced angle was sampled from a continuous rather than discrete distribution. This was done for arbitrary step sizes provided that they were not too large to invalidate the theory. An immediate application of this was the following simplification to

7

subroutine `ELECTR`.

13. The control logic in the charged-particle transport routine, `ELECTR`, was greatly simplified and modifications were made to both `ELECTR` and the photon transport routine, `PHOTON`, to make interactions at a boundary impossible.

14. The above changes to the control logic then made it possible for the user to implement importance-sampling[4] techniques into EGS without any furthers "internal" changes to the system itself. Examples that come to mind include the production of secondary electron beams at large angles, photon energy deposition in relatively small (low-Z) absorbers, and deep penetration (radial and longitudinal) calculations associated with shower counter devices.

15. Provision was made for allowing the density to vary continuously in any given region.

16. A new subroutine (`PHOTO`) was added in order to treat the photoelectric effect in a manner comparable to the other interaction processes. The main interest in this was to facilitate the development of a more general photoelectric routine, such as one that could produce fluorescent photons and/or Auger electrons for subsequent transport by EGS.

17. Additional calls to `AUSGAB`, bringing the total from 5 to 23, were made possible in order to allow for the extraction of additional information without requiring the user to edit the EGS code itself. For example, the user could determine the number of collision types (*e.g.*, Compton *vs.*photoelectric, *etc.*).

Upon release in 1978, the EGS3 Code System soon became the "industry standard" for designing shower detectors in high-energy physics. Looking back at this period of time several reasons can explain why EGS became so popular so quickly. Leading the list was the fact that the other codes mentioned above simply were not available; whereas, anyone could get EGS, together with its documentation, free-of-charge from SLAC. Furthermore, the code had been successfully benchmarked and support was provided to anyone requesting help. These things provided the fuel for the fire. What ignited it, however, was the so-called November Revolution[12, 13] of particle physics and the resulting shift to the use of colliding-beam accelerators. In particular, there was an immediate need by the high-energy physics community for tools to aid in the design of shower counters for the large, vastly-complicated, 4π detector systems associated with the new colliding-beam storage-ring facilities under construction throughout the world. EGS was there at the right time and right place when this happened.

We would be remiss if we did not mention one other code that also was available during this time period, particularly since published results from it had been used as part of the benchmarking of EGS3 itself. We refer to the ETRAN Monte Carlo shower code written by Berger and Seltzer[19][5]. ETRAN treated the low-energy processes (down to 1 keV) in greater detail than EGS3. Instead of the Molière[107, 108] formulation, ETRAN made use of the Goudsmit-Saunderson[63, 64] approach to multiple scattering, thereby avoiding the small-angle approximations intrinsic to Molière. ETRAN also treated fluorescence, the effect of atomic binding on atomic electrons, and energy-loss straggling. Because of the special care taken at low energies, ETRAN, which was initially writ-

---

[4]For those who may be unfamiliar with the term, importance sampling refers to sampling the most important regions of a problem and correcting for this bias by means of weight factors (see, for example, the report by Carter and Cashwell[41].)

[5]A later version of this program, which contained a fairly general geometry package, was known as SANDYL[46] and was also available at this time.

ten for energies less than 100 MeV and later extended to 1 GeV[16], ran significantly slower than EGS3. However, in spite of its accuracy and availability, ETRAN went unnoticed in the world of high-energy physics during this period.

1.2.5 EGS4

EGS3 was designed to simulate electromagnetic cascades in various geometries and at energies up to a few thousand GeV and down to cutoff kinetic energies of 0.1 MeV (photons) and 1 MeV (electrons and positrons). However, ever since the introduction of the code in 1978 there had been an increasing need to extend the lower-energy limits—*i.e.*, down to 1 and 10 keV for photons and electrons, respectively. Essentially, EGS3 had become more and more popular as a general, low-energy, electron-photon transport code that could be used for a variety of problems in addition to those generally associated with high-energy electromagnetic cascade showers. It had many features that made it both general as well as versatile, *and it was relatively easy to use,* so there had been a rapid increase in the use of EGS3 both by those outside the high-energy physics community (*e.g.*, medical physics) and by those within. Even though other low-energy radiation transport codes were available, most notably ETRAN[15, 18, 19] and its progeny[46, 68], there had been many requests to extend EGS3 down to lower energies and this was a major, but not the only, reason for creating EGS4. The various corrections, changes and additions, and new features that were introduced in the 1985 release of the EGS4 Code System[126] are summarized below.

Summary of EGS3 to EGS4 conversion

As with any widely used code, there had been many extensions made to EGS3 and many small errors found and corrected as the code was used in new situations. The following lists the most significant differences between EGS3 and EGS4.

- Major Changes and Additions to EGS3.
    - Conversion from MORTRAN2 to MORTRAN3.
    - Corrections to logic and coding errors in EGS3.
    - Extension of electron transport down to 10 keV (kinetic energy).
    - Improved Sternheimer treatment of the density effect.
    - Improved definition of the radiation length at low atomic numbers.
- New Options and Macros.
    - Macro templates for introduction of weighting and biasing techniques.
    - Pi-zero option.
    - Rayleigh scattering option.
    - Compton electron stack position preference (macro).
    - Positron discard option (macro) for creation of annihilation gammas.
- Auxiliary Subprograms and Utilities.

- – Geometry subprograms and corresponding macro packages.
- – Miscellaneous energy conservation and event counter utility routines.
- – Fixed fractional energy loss subroutines.

- New Applications and Examples.

  - – Leading-particle biasing macro to increase efficiency.
  - – Fluorescent-photon transport capability.
  - – Charged-particle transport in magnetic fields.
  - – Combinatorial Geometry package.
  - – Coupling of hadronic and electromagnetic cascade codes.

The most significant changes were made to subroutine `ELECTR` to correct problems which occurred when lower-energy charged-particle transport was done. The most significant change in this regard was first brought to attention in the paper "Low energy electron transport with EGS" by Rogers[136]. Many of the difficulties with the low-energy transport related to the fact that electron transport sub-steps (multiple scattering and continuous energy loss are modeled at the endpoints of these steps) were too large and various approximations that were valid for high-energy transport (above 10–20 MeV) were invalid for low-energy. Rogers modified the EGS code to allow the user to control the electron step-size in two ways, one by specifying a maximum allowable energy loss to continuous energy-loss processes (`ESTEPE`) and a geometric step-size control (`SMAX`) that restricts the electron step-size to be no larger than some user-specified distance. This allowed low-energy electron transport to be calculated with some degree of confidence although the user was required to study the parametric dependence of applications on these two parameters, `ESTEPE` and `SMAX`.

1.3   Overview of the EGS4 Code System – Vintage 1985

The following is a summary of the main features of the EGS4 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. That is, the data preparation package, PEGS4, creates data to be used by EGS4, using cross section tables for elements 1 through 100.
- Both photons and charged particles are transported in random rather than in discrete steps.
- The dynamic range of charged-particle kinetic energies goes from a few tens of keV up to a few thousand 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 thousand GeV (see above statement).
- The following physics processes are taken into account by the EGS4 Code System:

  - – Bremsstrahlung production (excluding the Elwert correction at low energies).
  - – Positron annihilation in flight and at rest (the annihilation quanta are followed to completion).

10

- Molière multiple scattering (*i.e.*, Coulomb scattering from nuclei). The reduced angle is sampled from a continuous (rather than discrete) distribution. This is done for arbitrary step sizes, selected randomly, provided that they are not so large or so small as to invalidate the theory.
- Møller (e^-e^-) and Bhabha (e^+e^-) scattering. Exact rather than asymptotic formulas are used.
- Continuous energy loss applied to charged-particle tracks between discrete interactions.
  * Total 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.
- Pair production.
- Compton scattering.
- Coherent (Rayleigh) scattering can be included by means of an option.
- Photoelectric effect.
  * Neither fluorescent photons nor Auger electrons are produced or transported in the default version of subroutine `PHOTO`.
  * Other *user-written* versions of `PHOTO` can be created, however, that allow for the production and transport of K- and L-edge photons.

- 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 EGS4.
  - 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 EGS4, PEGS4 contains options to plot any of the physical quantities used by EGS4, as well as to compare sampled distributions produced by the UCTESTSR User Code with theoretical spectra.

  - 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.
  - EGS4 uses material cross section and branching ratio data created and fit by the companion code, PEGS4.

- 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 EGS4 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 EGS4MAC file) which, if used, generally result in a faster running simulation.
  - The MORSE-CG Combinatorial Geometry package can be incorporated into `HOWFAR` (*e.g.*, see the UCSAMPCG file on the EGS4 Distribution Tape). However, experience

11

indicates that a much slower simulation generally results (of the order of at least a factor of four).

- – Transport can take place in a magnetic field by writing a specially designed `HOWFAR` subprogram, or in a more general manner (*e.g.*, including electric field) by making use of MORTRAN3 macro templates that have been appropriately placed for that purpose in subroutine `ELECTR`.

- The user scores and outputs information in the user-written subroutine called `AUSGAB`.

  - – Auxiliary subprogram `ECNSV1` is provided in order to keep track of energy for conservation (or other) purposes.
  - – Auxiliary subprogram `NTALLY` is provided in order to keep track of the number of times energy has been scored into the `ECNSV1` arrays (*i.e.*, an *event counter*).
  - – Auxiliary subprogram `WATCH` is provided in order to allow an event-by-event or step-by-step tracking of the simulation.

- EGS4 allows for the implementation of importance sampling and other variance reduction techniques (*e.g.*, leading particle biasing, splitting, path length biasing, Russian roulette, *etc.*).

- Initiation of the radiation transport:

  - – An option exists for initiating a shower with two photons from pi-zero decay (*i.e.*, use `IQI` with a value of 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 (*e.g.*, a synchrotron radiation spectrum).
  - – Transport can also be initiated from sources that have spatial and/or angular distributions.

1.4   Improvements to EGS Since 1985

In this section the improvements to EGS since the Version 4 release in December 1985 are described briefly. Only marginal detail is provided and the interested reader is encouraged to consult the references cited for deeper explanation. Note that many of the updates described here were incorporated in versions of EGS4 that were somewhat generally available, while other improvements are unique to the current release of EGS5.

1.4.1   Version 5's Improvements and Enhancements to EGS Electron Physics Modeling

Electron transport mechanics

A dual random hinge approach, in which energy loss and multiple elastic scattering are fully decoupled[34], has been adopted for modeling the spatial transport of electrons and positrons in

EGS5. The primary advantages of this technique lie in that the random multiple scattering hinge preserves near second order spatial moments of the transport equation over long step lengths[88], and that the hinge mechanics can be formulated so as to permit transport across boundaries between regions of differing media. Additionally, we believe, but have yet to prove analytically, that the use of the dual hinges preserves the energy dependent spatial moments.

The algorithm begins with the selection of a distance t_E for an energy step and a distance t_θ to the next multiple scattering event. Generalizing the random hinge methodology of PENELOPE [56], both the energy loss and the multiple scattering steps are comprised of two segments. The lengths of the initial segments are determined by multiplying random numbers ζ_E and ζ_θ by t_E and t_θ, respectively, and the lengths of the second segments are simply the remaining distances. Unlike EGS4, in which multiple scattering deflection is applied at the end of each step, in EGS5, multiple scattering is applied at a random "hinge" point (a distance $\zeta_\theta t_\theta$ along the track) during each step. Likewise, the energy of the electron remains unchanged along its track until the energy "hinge" at $\zeta_E t_E$ is reached. At this point, the electron energy is decremented by the *full amount* prescribed in the continuous slowing down approximation (CSDA) model for traversing the total pathlength t_E. The two hinge mechanisms are completely independent, as several energy hinges may occur before a single multiple scattering is applied, and vice versa. In evaluating the multiple scattering distribution and sampling for the deflection, all parameters which have an energy dependence are computed using the most recently updated energy for the particle. There is thus an implicit averaging of the electron energies over the track length, as energy hinges will occur sometimes before scattering hinges and sometimes after.

Subsequent to reaching a hinge point and undergoing either energy loss or deflection, the electron must then be transported the distance remaining for the given step, $(1 - \zeta_E)t_E$ for energy hinges or $(1 - \zeta_\theta)t_\theta$ for multiple scattering hinges. Once the end of the step is reached, the next step begins with the determination of a new value of t_E or t_θ, respectively, a random distance to the next hinge point is determined and the process is repeated.

Elimination of PRESTA's boundary crossing logic

The PRESTA (**P**arameter **R**educed **E**lectron **S**tep **T**ransport **A**lgortithm) algortithm was implemented in EGS4[31, 32, 33] to permit accurate treatment of electron transport over long steps. PRESTA changed the EGS4 transport mechanics model in four principal areas:

1. A refined calculation of the average curvature of the electron sub-step.
2. A lateral correlation algorithm was introduced that added an extra lateral component to the sub-step, correlating it to the multiple-scattering angle selected at the end of the sub-step.
3. A boundary crossing algorithm was introduced. This algorithm causes electron sub-steps to become shorter in the vicinity of boundaries insuring that no transport artifacts will occur near interfaces.
4. More careful energy averaging over the electron sub-step was introduced with subsequent refinements[101, 138].

Although the transport mechanics algorithm in PRESTA contains good corrections for longitudinal and lateral displacements (as well as average energy), it has been abandoned in EGS5. This change was motivated primarily by the desire to eliminate PRESTA's boundary algorithm, which, though theoretically the best way for handling elastic-scattering ambiguities in the vicinity of boundaries, is difficult to couple to general purpose combinatorial geometry codes, which may contain quadric and higher-order surfaces[28]. In the case of quadric surfaces, the determination of the closest distance to a boundary may involve solving for the roots of a sixth-order polynomial. While such algorithms may be devised, specifically designed with this application in mind[132], they are too slow for routine application.

Thus, in addition to providing improved treatment of longitudinal and lateral displacement, the new transport mechanics of EGS5 also yields advantages in computational speed compared to most models using a boundary sensitive approach. Although applications requiring boundary sensitivity must exist, we have yet to encounter any in practice. This is certainly an area that would benefit from more attention, as a boundary sensitive component could be added without much effort. Moreover, the current EGS5 transport mechanics could be adapted easily to most tracking codes, which perform ray tracing in combinatorial geometries without boundary sensitivity.

Bremsstrahlung angular distribution

Bielajew *et al.*[29] modified EGS4 to allow for angular distributions employing the Schiff formula from a review article by Koch and Motz[91]. Standard EGS4 makes the approximation that the angle of the bremsstrahlung photon with respect to the initiating charged particle's direction is $\Theta = 1/E_0$ where E_0 is the initiating charged particle's energy in units of the electron rest mass energy. It was acknowledged that this might be a bad approximation for thin-target studies, but it was expected that there would be no effect in thick-target studies since multiple scattering would "wash-out" the initial bremsstrahlung angular distribution and that an average value would be sufficient. However, thick-target studies in the radiotherapy range showed dramatic evidence of this approximation as a calculation artifact[54]. Angular distributions near the central axis changed by as much as 40%! Thick-target studies at diagnostic energies also showed the artifact which was eliminated through use of the new sampling technique[124]. This new capability was carried over to EGS5.

ICRU37 collision and radiative stopping powers

Duane *et al.*[52] modified PEGS4 to give collision stopping powers identical to those of ICRU Report 37[20, 79]. The NBS (now NIST) database EPSTAR[148] which was used to create the ICRU tables was employed. The modifications also allow the user to input easily an arbitrary density-effect correction. This change is relatively small but crucial if one is doing detailed stopping-power-ratio studies[101, 137].

In a related work, Rogers *et al.*[139] adapted PEGS4 to make the radiative stopping powers

ICRU37-compliant using the NIST database ESPA[148]. Effectively, this modification globally renormalizes EGS4's bremsstrahlung cross section so that the integral of the cross section (the radiative stopping power) agrees with that of ICRU Report 37[20, 79]. This improvement can lead to noticeable changes in the bremsstrahlung cross section for particle energies below 50 MeV[54] and significant differences for energies below a few MeV where bremsstrahlung production is very small[124]. These new capabilities were carried over to EGS5.

Low-energy elastic electron cross section modeling

It has long been acknowledged that the Molière multiple scattering distribution used in EGS4 breaks down under certain conditions. In particular: the basic form of the cross section assumed by Molière is in error in the MeV range, when spin and relativistic effects are important; various approximations in Molière's derivation lead to significant errors at pathlengths less than 20 elastic scattering mean free paths; and the form of Molière's cross section is incapable of accurately modeling the structure in the elastic scattering cross section at large angles for low energies and high atomic number. It is therefore desirable to have available a more exact treatment, and in EGS5, we use (in the energy range from 1 keV to 100 MeV) elastic scattering distributions derived from a state-of-the-art partial-wave analysis (an unpublished work) which includes virtual orbits at sub-relativistic energies, spin and Pauli effects in the near-relativistic range and nuclear size effects at higher energies. Additionally, unlike the Molière formalism of EGS4, this model includes explicit electron-positron differences in multiple scattering, which can be pronounced at low energies.

The multiple scattering distributions are computed using the exact approach of Goudsmit and Saunderson (GS)[63, 64]. Traditionally, sampling from GS distributions has been either prohibitively expensive (requiring computation of several slowly converging series at each sample) or overly approximate (using pre-computed data tables with limited accuracy). We have developed here a new fitting and sampling technique which overcomes these drawbacks, based on a scaling model for multiple scattering distributions which has been known for some time[27]. First, a change of variables is performed, and a reduced angle $\chi = (1 - \cos(\theta))/2$ is defined. The full range of angles ($0 \leq \theta \leq \pi$, or $0 \leq \chi \leq 1$) is then broken into 256 intervals of equal probability, with the 256th interval further broken down into 32 sub-intervals of equal spacing. In each of the 287 intervals or sub-intervals, the distribution is parameterized as

$$f(\chi) = \frac{\alpha}{(\chi + \eta)^2} \left[1 + \beta(\chi - \chi_-)(\chi_+ - \chi)\right]$$

where α, β and η are parameters of the fit and χ_- and χ_+ are the endpoints of the interval.

By using a large number of angle bins, this parameterization models the exact form of the distribution to a very high degree of accuracy, and can be sampled very quickly (see Chapter 2 for the details of the implementation).

15

Photoelectron angular distribution

In standard EGS4, a photoelectron, the electron produced when a photon is absorbed by an atom by the photoelectric effect, is set in motion in the same direction as the incident photon. In order to try to refine the comparison with low-energy experiments using thermoluminescent dosimeters, Bielajew and Rogers[30] employed the theory of Sauter[145] and made it an option for EGS4. Although Sauter's theory is relativistic ($v \approx c$), it was adopted universally, even though Fischer's non-relativistic theory[58] was derived for use in the ($v << c$) region. Davisson and Evans[50] have shown that Sauter's theory is accurate down to 92 keV ($v/c = 0.092$), and Bielajew and Rogers found that it had no significant effect relative to the use of Sauter's formalism in several cases studied[30]. This option has been carried over into EGS5.

Pair angular distributions

Bielajew[26] also modified[6] EGS4 to sample the angular distribution of the electron and positron emanating from pair production from theoretical distributions taken from the review article by Motz *et al.*[111]. As a first approximation, pair angles could be sampled from the leading order term in the Sauter-Gluckstern-Hull distribution (Equation 3D-2000 of Motz *et al.*[111]). Additionally, a fast method for sampling the Schiff angular distribution (Motz *et al.*Equation 3D-2003 *et al.*[111]), similar to that employed by ETRAN[149, 150], was made available. Use of these angular distributions has noticeable effects at large energies where the pair interaction dominates the photon cross section and the scoring regions of interest are small enough that multiple scattering does not "wash out" the effect of the initial pair distribution. This capability has been carried over into EGS5.

1.4.2 Improvements and Enhancements to EGS Photon Physics Modeling in Version 5

K- and L-shell fluorescence

Standard EGS4 does not create or transport fluorescent photons. However, a substitute version of the EGS subroutine `PHOTO` allowing for the generation of K_{α_1} and K_{β_1} fluorescent photons was provided as part of a sample user code with the EGS4 distribution. This version of `PHOTO`, along with an auxiliary subroutine `EDGSET` (extended by Keith Weaver of the University of California at San Francisco to 100 elements) defining shell edge energies, was enabled in EGS4 using software switches, but has become the default in EGS5.

Treating atomic relaxation processes explicitly requires resolving the total photoelectric cross section into subshell-by-subshell components, which can be cumbersome for compounds containing

---

[6]Development of the pair-angle sampling scheme was motivated by a high-energy physics experiment at SLAC[158]. A secondary result of this effort was the discovery (and correction) of a bug in subroutine `UPHI` that occurs at energies above 50 GeV and shows up at small angles.

high Z materials. Del Guerra *et al.*[51] developed a K- and L-edge sampling method for compounds in EGS4, and Conti *et al.*[47] used this scheme in investigating the response of HgI_2 and CdTe photon detectors in the diagnostic range.

In 2000, Hirayama and Namito published a general treatment of photoelectric-related phenomena for compounds or mixtures in EGS4[72]. The energy dependent branching ratio of each sub-shell (L1-, L2- and L3-) was introduced in this improvement by fitting to a quadratic function in log-log from data provided for limited materials (Ag, Pb and U). For compounds and mixtures, the ratio of photoelectric effect cross section of each element to total photoelectric cross section was calculated in PEGS4 and output as piece-wise linear fits. In addition to K- and L- x-rays, K- and L- Auger electrons could be treated.

Very recently, Hirayama and Namito published an improvement of their previous generalized treatment [73]. In this work, K-, L1-, L2-, L3- and other sub-shell photoelectric cross sections taken from the PHOTX data base (see below) are fitted to cubic functions in log-log form and the fitted coefficients and other associated data are initialized in a `BLOCK DATA` subprogram of EGS. It thus becomes possible to calculate the branching ratios for each element of compounds and mixtures inside EGS, negating the need to use approximate piece-wise linear-fitted data from PEGS. This new method is included in the default version of EGS5.

Improved photon cross sections

The standard EGS4 photon cross section data was based on the library compiled by Storm and Israel[167]. Sakamoto[143] updated EGS4 to use the more recent photon cross sections of the PHOTX library[131]. As discussed by Sakamoto, the principal differences between the Storm and Israel data and that of PHOTX lie in the low energy photoelectric cross sections. It should be noted that although the attenuation coefficients in the two photon libraries are different, the practical effect in many applications, such as the calculation of exposure buildup factors, is small. PHOTX is the default library for EGS5.

Binding effects in Compton interactions

In its treatment of Compton scattering, standard EGS4 assumed the electrons in the atomic clouds of the target atoms to be "free," ignoring the binding of the atomic electrons to the nucleus. This is a good approximation for photon energies down to 10's of keV for most materials. The lower bound where this approximation works reasonably well is defined by the K-shell energy, although the effects can have significant influence above it, particularly for the low-Z elements. Namito and Hirayama included binding effects in the Compton interaction model in EGS4 taking into account the change in the total Compton cross section as well as the angular distribution of the emergent particles[119]. The bound Compton modeling was shown to have noticeable effect in low-energy gamma-ray buildup factors in various materials at low energy (40–200 keV) with the effect being stronger at the lower energies studied[76]. Explicit treatment of binding effects in Compton

scattering is included (as an option) in EGS5.

Doppler broadening and linearly-polarized photon scattering

In addition to neglecting the binding of the electrons in the Compton interaction, standard EGS4 also ignored the motion of the electrons in the atomic cloud. Since this motion constitutes a distribution of momenta, bound electrons ejected in Compton interactions emerge not, as assumed in EGS4, with a single energy dependent on the scattering angle, but with a distribution of possible energies. (This effect is usually called Doppler broadening.) Doppler broadening of Compton electron energy spectra has been taken into account in EGS4 by Namito *et al.*[117], who demonstrated improved calculations of the measured energy spectra for photon scattering at 40 keV. Explicit modeling of this effect (on a shell-by-shell basis) is available (as an option) in standard EGS5.

Standard EGS4 also considered all particles to be unpolarized, using cross sections that have been summed over incident and out-going particle polarizations. Namito *et al.*[116] introduced polarized Compton and Rayleigh scattering in EGS4 and made comparisons with low-energy experiments where gamma ray build-up factors and attenuation coefficients in water, iron and lead were studied in the energy range 40–250 keV[74]. The effects of binding, Doppler broadening and polarization are pronounced in the lower energy range studied [7]. Both effects can be modeled as options in EGS5.

Electron impact ionization

Following the work of Namito and Hirayama[120, 122] K-shell electron-impact ionization (EII) is included in EGS5. Six different representations of the cross sections for EII are supplied:

1. Casnati[43, 44]

2. Kolbenstvedt (original)[92]

3. Kolbenstvedt (revised)[105]

4. Jakoby[80]

5. Gryziński[66, 67, 65]

6. Gryziński (relativistic)[66, 67, 65]

---

[7]Recent paper by Flöttmann[59] and others[93, 98] introduced polarization into EGS4, by means of the additional-calls-to-`AUSGAB` feature, in order to investigate the development of high-intensity positron sources for future linear colliders.

One of these six cross sections is selected by the user by specifying a PEGS5 input parameter. EII is treated as being a part of Møller scattering in the EGS5 code, and neither the electron mean-free path nor the stopping power are modified by including EII.

After a K-shell vacancy creation by EII, the emission of a K x-ray is sampled using the K-shell fluorescence yield. This is identical to the calculation after a K-shell vacancy creation by the photoelectric effect, although no Auger electron or cascade particle is followed in the treatment of EII. For the case of K x-ray emission, the K x-ray energy is sampled from the ten K x-ray energies listed in [57] according to the relative yield of the K x-ray, and the difference between E_B (K shell binding energy) and the K x-ray energy is deposited locally. In the case that no K x-ray is generated (*i.e.*, Auger electron emission), E_B is deposited locally.

1.4.3 Other Improvements

Merging of EGS and PEGS

Historically, EGS, the simulation code, and PEGS, the data generation code, were completely separate entities. This left open the possibility that a simulation could work on inappropriate data, despite the built-in integrity checks. In EGS5, the two codes have been merged, and now EGS calls PEGS during its initialization procedures. Since the elastic scattering runtime databases can require large amounts of CPU time in certain cases, the capability to reuse PEGS data files still exists, but its use is discouraged unless the accurate low energy elastic scattering option is employed.

Long sequence random number generators

EGS5 employs the `RANLUX` random number generator[82], with the default "luxury level" set to one. `RANLUX` has a period of 10^{155} unique random numbers and at luxury level 1 has a fidelity approximately equal to that of `RANMAR`[81], which became the default random number generator for EGS4 in the interim period between major releases. The user may override the default luxury level and so employ sequences which pass more stringent tests of randomness, although at a significant cost in speed.

Chapter 2

RADIATION TRANSPORT IN EGS5

2.1   Description of Radiation Transport-Shower Process

Electrons[1], as they traverse matter, lose energy by two basic processes: collision and radiation. The collision process is one whereby either the atom is left in an excited state or it is ionized. Most of the time the ejected electron, in the case of ionization, has a small amount of energy that is deposited locally. On occasion, however, an orbital electron is given a significant amount of kinetic energy such that it is regarded as a secondary particle called a *delta-ray*.

Energy loss by radiation (bremsstrahlung) is farily uniformly distributed among secondary photons of all energies from zero up to the kinetic energy of the primary particle itself. At low electron energies the collision loss mechanism dominates the electron stopping process, while at high energies bremsstrahlung events are more important, implying that there must exist an energy at which the rate of energy loss from the two mechanisms are equivalent. This energy coincides approximately with the *critical energy* of the material, a parameter that is used in shower theory for scaling purposes[141]. Note that the energetic photons produced in bremsstrahlung collisions may themselves interact in the medium through one of three photon-processes, in relative probabilities depending on the energy of the photon and the nature of the medium. At high energies, the most likely photon interaction is materialization into an electron-positron pair, while at somewhat lower energies (in the MeV range), the most prevalent process is Compton scattering from atomic electrons. Both processes result in a return of energy to the system in the form of electrons which can generate additional bremsstrahlung photons, resulting in a multiplicative cycle known as an *electromagnetic cascade shower*. The third photon interactive process, the photoelectric effect, as well as multiple Coulomb scattering of the electrons by atoms, perturbs the shower to some degree

---

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

at low energies. The latter, coupled with the Compton process, gives rise to a lateral spread in the shower. The net effect in the forward (longitudinal) direction is an increase in the number of particles and a decrease in their average energy at each step in the process.

As electrons slow, radiation energy loss through bremsstrahlung collisions become less prevalent, and the energy of the primary electron is dissipated primarily through excitation and ionization collisions with atomic electrons. The so-called *tail* of the shower consists mainly of photons with energies near the minimum in the mass absorption coefficient for the medium, since Compton scattered photons predominate at large shower depths.

Analytical descriptions of this shower process generally begin with a set of coupled integro-differential equations that are prohibitively difficult to solve except under severe approximation. One such approximation uses asymptotic formulas to describe pair production and bremsstrahlung, and all other processes are ignored. The mathematics in this case is still rather tedious[141], and the results only apply in the longitudinal direction and for certain energy restrictions. Three dimensional shower theory is exceedingly more difficult.

The Monte Carlo technique provides a much better way for solving the shower generation problem, not only because all of the fundamental processes can be included, but because arbitrary geometries can be modeled. In addition, other minor processes, such as photoneutron production, can be modeled more readily using Monte Carlo methods when further generalizations of the shower process are required.

Another fundamental reason for using the Monte Carlo method to simulate showers is their intrinsic random nature. Since showers develop randomly according to the quantum laws of probability, each shower is different. For applications in which only averages over many showers are of interest, analytic solutions of average shower behaviors, if available, would be sufficient. However, for many situations of interest (such as in the use of large NaI crystals to measure the energy of a single high energy electron or gamma ray), the shower-by-shower fluctuations are important. Applications such as these would require not just computation of mean values, but such quantities as the probability that a certain amount of energy is contained in a given volume of material. Such calculations are much more difficult than computing mean shower behavior, and are beyond our present ability to compute analytically. Thus we again are led to the Monte Carlo method as the best option for attacking these problems.

2.2 Probability Theory and Sampling Methods—A Short Tutorial

There are many good references on probability theory and Monte Carlo methods (viz. Halmos [69], Hammersley and Handscomb [70], Kingman and Taylor [89], Parzen[130], Loeve[99], Shreider[156], Spanier and Gelbard[157], Carter and Cashwell[41]) and we shall not try to duplicate their effort here. Rather, we shall mention only enough to establish our own notation and make the assumption

that the reader is already acquainted with the elements of probability theory.

The primary entities of interest will be <u>random variables</u> which take values in certain subsets of their range with specified probabilities. We shall denote random variables by putting a "hat" (ˆ) above them (*e.g.*, $\hat{x}$). If E is a logical expression involving some random variables, then we shall write $Pr\{E\}$ for the probability that E is true. We will call F the <u>distribution function</u> (or cumulative distribution function) of $\hat{x}$ if

$$F(x) = Pr\{ \ \hat{x} < x \} \ . \tag{2.1}$$

When $F(x)$ is differentiable, then

$$f(x) = F'(x) \tag{2.2}$$

is the <u>density function</u> (or probability density of $\hat{x}$ and

$$Pr \ \{ \ a < \hat{x} < b \} \ = \int_a^b \ f(x)dx \ . \tag{2.3}$$

In this case $\hat{x}$ is called a <u>continuous random variable</u>.

In the other case that is commonly of interest, $\hat{x}$ takes on discrete values x_i with probabilities p_i and

$$F(x) = \sum_{x_i \leq x} \ p_i \ . \tag{2.4}$$

We call P the probability function of $\hat{x}$ if

$$P(x) \ = \ p_i \quad \text{if } x = x_i$$
$$= \ 0 \quad \text{if } x \text{ equals none of the } x_i \ . \tag{2.5}$$

Such a random variable is called a <u>discrete random variable</u>.

When we have several random variables x_i $(i = 1, n)$, we define a <u>joint distribution function</u> F by

$$F(x_1, \ldots, \ x_n) = Pr\{ \ \hat{x}_1 \leq x_1 \& \ \ldots \& \ \hat{x}_n \leq x_n \} \ . \tag{2.6}$$

The set of random variables is called <u>independent</u> if

$$Pr\{ \ \hat{x}_1 \leq x_1 \& \ \ldots \& \ \hat{x}_n \leq x_n \} \ = \prod_{i=1}^n \ Pr\{ \ \hat{x}_1 < x_i \ \} \ . \tag{2.7}$$

If F is differentiable in each variable, then we have a <u>joint density function</u> given by

$$f(x_1, \ldots, x_n) = \partial^n F(x_1, \ldots, x_n)/\partial x_1, \ldots, \ \partial x_n \ . \tag{2.8}$$

Then, if A is some subset of R^n (n-dimensional Euclidean Space),

$$Pr\{ \ \hat{\vec{x}} \in A \ \} \ = \int_A \ f(x)d^n x \ . \tag{2.9}$$

22

If E_1 and E_2 are expressions involving random variables and $Pr\{ E_2 \} \neq 0$, we define the conditional probability of E_1 given E_2 by[2]

$$Pr\{ E_1|E_2 \} = Pr\{ E_1 \ \& \ E_2 \} \, /Pr\{ E_2 \} , \tag{2.10}$$

Thus, we can define <u>conditional distribution functions</u>

$$F_3(x_1,\ldots,x_j|x_{j+1},\ldots,x_n) = F_1(x_1,\ldots,x_n)/F_2(x_{j+1},\ldots,x_n) . \tag{2.11}$$

For continuous random variables, we define conditional density functions by

$$f_3(x_1,\ldots,x_j|x_{j+1},\ldots,x_n) = f_1(x_1,\ldots,x_n)/f_2(x_{j+1},\ldots,x_n) . \tag{2.12}$$

With this preliminary introduction we move now to sampling methods. In practice almost all sampling is based on the possibility of generating (using computers) sequences of numbers which behave in many ways like sequences of random variables that are uniformly distributed between 0 and 1. We shall denote uniformly distributed random variables by $\hat{\zeta}_i$, and values sampled from the uniform distribution by ζ_i. Clearly, if F and f are distribution and density functions, respectively, of $\hat{\zeta}$, then they are given by

$$
\begin{aligned}
F(\zeta) \ &= \ 0, & \text{if } \zeta < 0, \\
&= \ \zeta, & \text{if } 0 \leq \zeta < 1, \\
&= \ 1, & \text{if } 1 \leq \zeta;
\end{aligned}
\tag{2.13}
$$

$$
\begin{aligned}
f(\zeta) \ &= \ 1, & \text{if } \zeta \in (0,1), \\
&= \ 0, & \text{otherwise} ,
\end{aligned}
$$

We assume, in what follows, that we have an unlimited number of uniform variables available.

Now, suppose that $\hat{x}$ and $\hat{y}$ are related by $\hat{y} = h(\hat{x})$ (with h monotonically increasing), and that $\hat{x}$ and $\hat{y}$ have distribution functions F and G; then,

$$
\begin{aligned}
F(x) = Pr\{ \hat{x} < x \} \ &= \ Pr\{ h^{-1}(\hat{y}) < x\} \\
&= \ Pr\{ \hat{y} < h(x)\} \\
&= \ G(h(x)) .
\end{aligned}
\tag{2.14}
$$

Thus, we can find F given G and h. In particular, if we define $\hat{x}$ by

$$\hat{x} = F^{-1}(\hat{\zeta}) , \tag{2.15}$$

then

$$G(y) = y$$

and

$$h(x) = F(x) ,$$

---

[2]The notation $Pr\{ E_1|E_2 \}$ reads "the probability of E_1 given E_2."

so that
$$Pr\{ \ \hat{x} < x \ \} \ = G(h(x)) = F(x) \ .$$

This is the basis of the so called "direct method" of sampling $\hat{x}$ in which we set

$$F(x) = \zeta$$

and solve for x. The x-values so chosen will have the distribution F.

Another method involves evaluating a function of several variables using uniform random variables for arguments. Thus we let

$$\hat{x} = h(\hat{\zeta}_1, \ \ldots, \ \hat{\zeta}_n) \ , \tag{2.16}$$

then

$$Pr\{ \ \hat{x} < x \} \quad = \textstyle\int_{h(\vec{\zeta}) < x} \ d^n \ \zeta \ . \tag{2.17}$$

This method has been used to find simple schemes for sampling many distributions. For example, suppose we pick $n + m + 1$ values of ζ and number them such that

$$\zeta_1 < \zeta_2 \ldots < \zeta_n \le \zeta_{n+1} < \zeta_{n+2}, \ \ldots < \zeta_{n+m+1} \ . \tag{2.18}$$

Then, let $x = \zeta_{n+1}$. Using Equations 2.16-2.18 it can be shown that

$$f(x) = (n + m + 1)! \ x^n (1 - x)^m / n! m! \ , \quad x \in (0, 1) \ .$$

A method that we shall frequently use is a combination of the "composition" and "rejection" techniques (Butler [40], Hammersley and Handscomb[70], Kahn[85]). Suppose f and f_i are density functions, α_i are positive real numbers, and $g_i(x) \in [0, 1]$. We now sample $\hat{x}$ as follows:

1. Pick ζ_1 and let i be such that

$$\sum_{j=1}^{i-1} \alpha_j < \zeta_1 \sum_{j=1}^{n} \alpha_j \le \sum_{j=1}^{n} \alpha_j \ . \tag{2.19}$$

2. Pick x from $f_i(x)$, possibly by solving

$$\int_{-\infty}^{x} f_i(x) \ dx = \zeta_2 \ . \tag{2.20}$$

3. Pick ζ_3. Terminate the algorithm and accept value of x if

$$\zeta_3 < g_i(x) \ . \tag{2.21}$$

4. Otherwise, go back to step 1.

The result of this algorithm is that $\hat{x}$ will have the density function f given by

$$f(x) = \sum_{i=1}^{n} \alpha_i \, f_i(x) \, g_i(x) \; . \tag{2.22}$$

Thus, if we have a function f which can be put in this form for which the f_i can be sampled easily and the g_i evaluated relatively easily, then we have a good method of sampling from f. It can be shown that the mean number of tries to accept a value is $\sum \alpha_i$. If all the $g_i = 1$, we have the pure "composition" method; and if $n = 1$, we have the pure "rejection" method. For short we will call this the "mixed" method. We shall sometimes use the mixed method for some of the f_i also. In these cases we will use the notation (for example)

$$f_2(x) = \sum_{i=1}^{n_2} \alpha_{2j} \, f_{2j}(x) \, g_{2j}(x) \; . \tag{2.23}$$

and so on.

Finally, we consider the problem of sampling from a joint density function $f(x_1, x_2, \ldots, x_n)$. Define the marginal density functions

$$
\begin{aligned}
g_m(x_1, \; x_2, \; \ldots, \; x_m) \;\; &= \;\; \int \; f(x_1, \; x_2, \; \ldots, \; x_n) \; dx_{m+1} \; dx_{m+2} \ldots dx_n \\
&= \;\; \int \; g_{m+1}(x_1, \; x_2, \; \ldots, \; x_m, \; x_{m+1}) \; dx_{m+1} \; .
\end{aligned}
$$
$$\tag{2.24}$$

We see that $g_n = f$. Now consider h_m given by

$$h_m(x_m | x_1, \; x_2, \; \ldots, \; x_{m-1}) \equiv g_m(x_1, \; x_2, \; \ldots, x_m) / g_{m-1}(x_1, \; x_2, \; \ldots, \; x_{m-1}) \; . \tag{2.25}$$

We see that

$$\int \; h_m(x_m | x_1, \; x_2, \; \ldots, \; x_{m-1}) \; dx_m = 1 \tag{2.26}$$

from the definition of the g's. We see that h_m is the conditional density function for x_m given the specified values for $x_1, \; x_2, \; \ldots, \; x_{m-1}$. It can be easily seen that f can be factored into a product of the h's; namely,

$$
\begin{aligned}
f(x_1, \; x_2, \; \ldots, \; x_n) \;\; = \;\; & h_1(x_1) \; h_2(x_2 | x_1) \; h_3(x_3 | x_1, \; x_2) \; \ldots \\
& \ldots \; h_n(x_n | x_1, \; x_2, \; \ldots, \; x_{n-1}) \; .
\end{aligned}
\tag{2.27}
$$

The procedure then is to get a sample value x_1 using density function h_1. Then use this value x_1 to determine a density function $h_2(x_2 | x_1)$ from which to sample x_2. Similarly, the previously sampled $x_1, \; x_2, \; \ldots, \; x_m$ determine the density function $h_{m+1}(x_{m+1} | x_1, \; x_2, \; \ldots, \; x_m)$ for x_{m+1}. This scheme is continued until all x_i have been sampled. Of course, if the $\hat{x}_i$ are independent random variables, the conditional densities will just be the marginal densities and the variables can be sampled in any order.

There are other methods analogous to the one-dimensional sampling methods which can be used for sampling joint distributions. The reader is referred to the references cited above for more details.

2.3   Simulating the Physical Processes—An Overview

In some approaches, the Boltzmann transport equation is written for a system and from it a Monte Carlo simulation of the system is derived. This method gives correct average quantities, such as fluences, but may not correctly represent fluctuations in the real situation when variance reduction techniques are employed. The reader is referred to Chapter 3 in the book by Carter and Cashwell[41] for details of this particular method.

In all versions of EGS we have taken a different and more simple-minded approach in that we attempt to simulate the actual physical processes as closely as possible. We have not introduced any inherent variance reduction techniques, so that fluctuations in the Monte Carlo results should truly be representative of real-life fluctuations. For the design of high energy particle detectors, this is an important consideration. On the other hand, fluctuations are not usually of interest in radiation shielding-type problems, and so EGS5 includes several variance reduction techniques which may be optionally invoked to make certain classes of calculations run more efficiently. None of the variance reduction techniques are invoked by default, however, and so the method of EGS5 can generally be described as *analog*[3] Monte Carlo.

The simulation of an electromagnetic cascade shower can be decomposed into a simulation of the transport and interactions of single high energy particle, along with some necessary bookkeeping. A last-in-first-out (LIFO) stack is used to store the properties of particles which have yet to be simulated. Initially, only the incident particle is on the stack (more correctly, the properties of the incident particle are stored in the first position of corresponding arrays). The basic strategy is to transport the top particle in the stack either until an interaction takes place, until its energy drops below a predetermined cutoff energy, or until it enters a particular region of space. In the latter two cases, the particle is taken off the stack and the simulation resumes with the new top particle. If an interaction takes place, and if there is more than one secondary product particle, the particle with the lowest available energy is put on the top of the stack. By "available energy" we mean the maximum energy which can be imparted by a given particle to new secondary particles in a collision: E for photons; $E - m$ for electrons; and $E + m$ for positrons, where E is the particle's total energy and m is the electron rest mass energy. By always tracking the lowest energy particle first, we ensure that the depth of the stack will never exceed $\log_2(E_{max}/E_{cut})$, where E_{max} is the largest incident energy to be simulated and E_{cut} is the lowest cutoff energy. When the final particle is removed from the stack and none remain, the simulation of the shower event is ended. The complete simulation of each individual shower event is commonly called a Monte Carlo "history".

---

[3]The electron transport model in EGS5 is not strictly "analog" in that all scattering collisions are not treated on an individual basis, but it is "analog" in the sense that the models of the aggregate effects of the large numbers of collisions which are grouped together are analytic and can in most circumstances preserve the random nature of the fluctuations in showers.

2.4 Particle Transport Simulation

The mean free path, λ, of a particle is given in terms of its total cross section, σ_t, or alternatively, in terms of its macroscopic total cross section, Σ_t, according to the expression

$$\lambda = \frac{1}{\Sigma_t} = \frac{M}{N_a \rho \sigma_t}, \tag{2.28}$$

where

$$
\begin{aligned}
N_a &= \text{Avogadro's number}, \\
\rho &= \text{density}, \\
M &= \text{molecular weight}, \\
\sigma_t &= \text{total cross section per molecule}.
\end{aligned}
$$

The probability of an interaction is given by

$$Pr\{\text{interaction in distance dx}\} = dx/\lambda.$$

In general, the mean free path may change as the particle moves from one medium to another, or when it loses energy. The number of mean free paths traversed will be

$$N_\lambda = \int_{x_0}^{x} \frac{dx}{\lambda(x)}. \tag{2.29}$$

If $\hat{N}_\lambda$ is a random variable denoting the number of mean free paths from a given point until the next interaction, then it can be shown that $\hat{N}_\lambda$ has the distribution function

$$Pr\{\hat{N}_\lambda < N_\lambda\} = 1 - \exp(-N_\lambda) \qquad \text{for } N_\lambda > 0. \tag{2.30}$$

Using the direct sampling method and the fact that $1 - \zeta$ is also uniform on $(0, 1)$ if ζ is, we can sample N_λ using

$$N_\lambda = -\ln \zeta. \tag{2.31}$$

This may be used in Equation 2.29 to obtain the location of the next interaction.

Let us now consider the application of the above to the transport of photons. Pair production, Compton scattering, and photoelectric processes are treated by default in EGS5. Explicit treatment of Rayleigh scattering is included as a non-default option. These processes all have cross sections which are small enough that all interactions may be simulated individually, so that photons in EGS5 travel in straight lines with constant energies between interactions. Thus, if the overall volume in which a simulation takes place is composed of a finite number of regions, each of which is a homogeneous material with uniform density, then the integral in Equation 2.29 reduces to a sum. If $x_0, x_1, \cdots$ are the region boundary distances between which λ is constant, then Equation 2.29 becomes

$$N_\lambda = \sum_{j=1}^{i-1} \left(\frac{x_j - x_{j-1}}{\lambda_j} \right) + \left(\frac{x - x_{i-1}}{\lambda_i} \right), \tag{2.32}$$

where $x \in (x_{i-1}, x_i)$. The photon transport procedure is then as follows. First, pick the number of mean free paths to the next interaction using Equation 2.31. Then perform the following steps:

1. Compute λ at the current location.

2. Let $t_1 = \lambda N_\lambda$.

3. Compute d, the distance to the nearest boundary along the photon's direction.

4. Let t_2 equal the smaller of t_1 and d. Transport by distance t_2.

5. Deduct t_2/λ from N_λ. If the result is zero (this happens when $t_2 = t_1$), then it is time to interact—jump out of the loop.

6. This step is reached if $t_2 = d$. Thus, a boundary was reached. Do the necessary bookkeeping. If the new region is a different material, go to Step 1. Otherwise, go to Step 2.

In regions where there is a vacuum, $\sigma = 0$ ($\lambda = \infty$) and special coding is used to account for this situation.

Now let us consider charged particle transport. The relevant interactions considered in EGS5 are elastic Coulomb scattering by nuclei, inelastic scattering by atomic electrons, positron annihilation, and bremsstrahlung. Modeling of charged particle transport is particularly challenging because analytical expressions for the cross sections for all of the above processes (with the exception of annihilation) become infinite as the transferred energy approaches zero (the infrared catastrophe, etc.). In actuality, these cross sections, when various corrections are taken into account (*i.e.*, screening for nuclear scattering, electron binding for electron scattering, and Landau-Pomeranchuk-Migdal effect corrections for bremsstrahlung), are not infinite, but they are very large and the exact values for the total cross sections are not well known. Therefore, it is not practical to try to simulate every interaction. Fortunately, the low momentum transfer events which give rise to the large total cross section values do not result in large fluctuations in the shower behavior itself. For this reason, large numbers of low momentum transfer collisions can be combined together and modeled as continually-occurring processes. Cutoff energies are defined to demark the threshold between where interactions are treated as discrete events (sometimes referred to as "hard collisions") or aggregated with other low transfer or "soft" interactions. The electron and photon threshold energiesused by EGS5 (as set up in PEGS) are defined by the variables `AE` and `AP`, respectively, so that any electron interaction which produces a delta-ray with total energy of at least `AE` or a photon with energy of at least `AP` is considered to be a discrete event. All other interactions are treated as contributing to the continuous processes of energy loss and deflection of the electron in between discrete interactions. Continuous energy losses are due to soft interactions with the atomic electrons (excitation and ionization loss) and to the emission of soft bremsstrahlung photons. Deflections are mostly due to multiple Coulomb elastic scattering from the nucleus, with some contribution coming from soft electron scattering.

Typically, Monte Carlo methods simulate charged particle transport as a series of "steps" of distance t over which hundred or thousands (or more) of low momentum transfer elastic and inelastic events may occur. It is usually assumed that the transport during each step can be modeled as a single, straight-line, mono-energetic translation, at the end of which the aggregate effects of the continuous energy loss and deflection incurred over the step t can be accounted for by sampling

from appropriate distribution functions. Discrete interactions can interrupt these steps at random distances, just as in the case of photon transport.

Several factors complicate this model. First, because of continuous nature of the energy loss that occurs over t, the discrete collision cross section varies along the path of the electron. Thus, mean free paths between between discrete electron interactions vary between the time they are first computed and the time the interaction position is reached. More importantly, electron paths over simulated transport distances t are not straight lines, and so in addition to changing particles' energies and directions, continuous collisions also displace them laterally from their initial paths. Relatedly, these lateral displacements result in the actual straight-line transport distances being something less than t. The models used in EGS5 to treat all of the implications of the continuous energy loss and multiple elastic scattering methodology is examined in section 2.15.

2.5   Particle Interactions

When a point of a discrete interaction has been reached it must be decided which of the competing processes has occurred. The probability that a given type of interaction occurred is proportional to its cross section. Suppose the types of interactions possible are numbered 1 to n. Then $\hat{i}$, the number of the interaction to occur, is a random variable with distribution function

$$F(i) = \frac{\sum\limits_{j=1}^{i} \sigma_j}{\sigma_t}, \tag{2.33}$$

where σ_j is the cross section for the jth type of interaction and σ_t is the total cross section $(= \sum_{j=1}^{n} \sigma_j)$. The $F(i)$ are the *branching ratios*. The number of the interaction to occur, i, is selected by picking a random number and finding the i which satisfies

$$F(i-1) < \zeta < F(i). \tag{2.34}$$

Once the type of interaction has been selected, the next step is to determine the parameters for the product particles. In general, the final state of the interaction can be characterized by, say, n parameters $\mu_1, \mu_2, \cdots, \mu_n$. The differential cross section is some expression of the form

$$d^n \sigma = g(\vec{\mu}) \, d^n \mu \tag{2.35}$$

with the total cross section being given by

$$\sigma = \int g(\vec{\mu}) \, d^n \mu. \tag{2.36}$$

Then $f(\vec{\mu}) = g(\vec{\mu}) / \int g(\vec{\mu}) \, d^n \mu$ is normalized to 1 and has the properties of a joint density function. This may be sampled using the method given in a previous section or using some of the more general methods mentioned in the literature. Once the value of $\vec{\mu}$ determines the final state, the properties of the product particles are defined and can be stored on the stack. As mentioned before, the particle with the least energy is put on top of the stack. The portion of code for transporting particles of the type corresponding to the top particle is then entered.

2.6 General Implementation Notes

We have seen in the preceding sections how it is possible, given the total cross sections, branching ratios, final state joint density functions, and an endless supply of random numbers, to simulate coupled electron and photon transport. Because of the statistical nature of the Monte Carlo method, the uncertainty in the results will depend on the number of histories run. Generally, statistical uncertainties are proportional to the inverse square root of the number of histories[156]. Thus, to decrease uncertainties by half it is necessary to run four times as many histories. Also, for given cutoff energies, the CPU time required to simulate a shower history is in general slightly more than linear in the energy of the incident particle. Because of all of these factors, low-uncertainty Monte Carlo calculations of high-energy shower simulations can be very time consuming.

The most effective way to ease the computational effort of a Monte Carlo simulation is to minimize the number of calculations required at the deepest levels inside the particle transport loops by precomputing the values of as many of the required variables as possible. It is for this reason that the computational task of almost all Monte Carlo programs is divided into two parts. First, a preprocessor code (PEGS, for the EGS5 code system) uses theoretical (and sometimes empirical) formulas to compute the various physical quantities needed and prepares them in a form for fast numerical evaluation. A second code (EGS itself, in this work) then uses this data, along with user-supplied subroutines which describe the problem geometry and control the scoring of results, to perform the actual simulation. The motivation is to perform in advance (in PEGS) as much computation as possible, so that amount of work done during actual particle transport simulation is minimized. In earlier versions of EGS, PEGS was a stand-alone program, because with the slower CPUs in previous generation computers, the data preparation burdens were sometimes on the same scale as simulation times. As faster computers have become available, users have typically applied them to longer simulations, and the ratio of data-preparation time to simulation time has shrunk dramatically. Therefore, beginning with EGS5, PEGS, has been an embedded as subroutine which can be called by EGS5 user code at the beginning of each simulation.

To aid in debugging and to help those interested in studying the various physics processes modeled in EGS, starting with EGS4, the EGS Code System was expanded beyond the minimum coding necessary to simulate radiation transport. With this in mind, PEGS was written in a modular form, and now contains almost 100 subprograms. These include functions to evaluate physical quantities which are either needed by PEGS or are of interest for other reasons. Other routines necessary for use of PEGS within EGS5 include fitting routines and routines to write the data for a given material onto a data file. Included among the PEGS subprograms not needed for the operation of EGS5 itself are routines to plot the functions on a lineprinter or a graphic device, and a routine to compare (on a lineprinter plot) the theoretical final state density functions with sampled final state distributions. Even though PEGS is now a subroutine in the overall EGS5 package, all of the plotting and density function computation routines have been preserved, and may be accessed by appropriate user code.

The new PEGS driving subroutine, `PEGS5`, calls some once-only initialization routines and then enters an option loop. After reading in the option that is desired, a `NAMELIST` read establishes other

parameters which may be needed. The action requested is then performed and control returns to the beginning of the loop. This continues until the control input has been exhausted.

The EGS5 code itself contains, among many other subprograms, four user-callable subroutines, BLOCK_SET, PEGS5, HATCH, and SHOWER. These routines call other subroutines in the EGS5 code, some of which call two user-written subroutines, HOWFAR and AUSGAB, which define the geometry and output (scoring), respectively. The user communicates with EGS5 by means of the subroutines described above accessing variables contained in various COMMON blocks.

To use EGS5, the user must write a MAIN program and the subroutines HOWFAR and AUSGAB. Typically, MAIN performs any initialization needed for the geometry routine HOWFAR and then sets the values of COMMON block variables which specify such things as the names of the media to be used, the desired cutoff energies, the unit of distance (*e.g.*, centimeters, radiation lengths, *etc.*) and so on. MAIN must also call the new initialization routine BLOCK_SET, which is used primarily to initialize data which is not defined in BLOCK DATA subprograms. At this point, MAIN may call the new PEGS subroutine PEGS5 to prepare material data for the simulation (this may be skipped if an existing data set is available). A call to the HATCH subroutine then "hatches" EGS5 by doing some once-only integrity checks and initializations and by reading from a data set the material data prepared by PEGS5 for the media requested. With the initialization completed, simulation of the shower may begin with MAIN calling subroutine SHOWER when desired. Each call to SHOWER results in the generation of a single EGS5 particle history, with the arguments to SHOWER specifying the parameters (energy, position, direction, *etc.*) of an individual initial particle. This permits the user the freedom to use any source distribution desired. The flow of control and data when a user-written MAIN program is linked with EGS5 is illustrated in Figure 2.1. Detailed information needed to write user programs is provided in the EGS5 User Manual (Appendix B of this report) As a beginning introduction, however, the reader may find it useful to first study the series of short tutorials provided in Chapter 3 of this document.

Both EGS5 and PEGS use MeV for the unit of energy, and all EGS5 variables reference the *total* energy (*i.e.*, kinetic plus rest mass) associated with particles. EGS5 primarily employs the CGS system of units, but PEGS scales its outputs in units of radiation lengths for subsequent (internal) use by EGS5. By default, the centimeter is the basic working unit of distance used in EGS5, but other transport distance unit may be chosen by setting the variable DUNIT before calling HATCH (see Appendix B for details).

Table 2.1 defines some of the mathematical and program symbols for entities in EGS and PEGS, as well as other symbols used in this discussion. The first column gives the item number, the second column shows the mathematical symbol used for the entities, and the third column shows the FORTRAN name for the same thing in the EGS or PEGS code. A 'P', 'E', or both in column four shows whether the item is used in PEGS, EGS, or both.[4] The fifth column contains the definition, explanation, or name of the item.

A number of physical, mathematical, and derived constants are used by the codes. We have

---

[4]Even though PEGS is integrated into EGS, PEGS and EGS have retained the majority of their prior distinctiveness, sharing only a handful of variables.
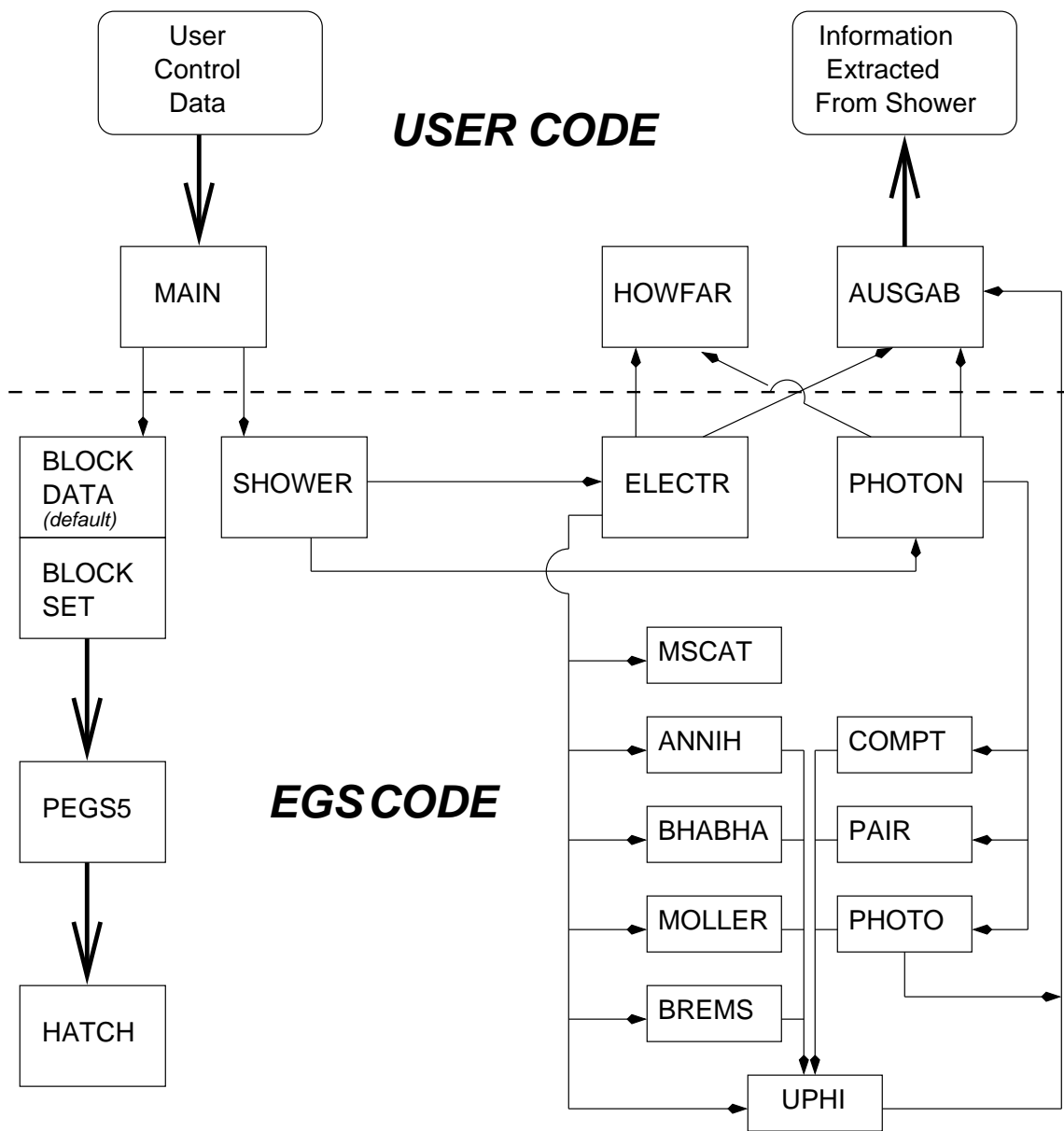
Figure 2.1: Program flow and data control in EGS5.

32

arrived at their values in a very mnemonic way by means of the PEGS subroutine called `PMDCON` (Physical, Mathematical, and Derived CONstants). These are items 1 through 16 in Table 2.1. As illustrated in items 17 through 27, cupped (˘) energy variables are in MeV, cupped distance variables are in radiation lengths, and uncupped quantities denote CGS units.

Materials are specified by giving density, ρ, numbers of elements, N_e, atomic symbols of the elements, and the proportions of the atoms which are of each type, p_i. PEGS maintains tables of atomic numbers, Z_i, and atomic weights, A_i, for elements 1 through 100. For cases in which materials are composed of single elements, PEGS has a table of default densities which it uses. For materials which have non-standard isotopes or densities, the values supplied by PEGS may be overridden. As an alternative to specifying the atomic proportions by number p_i, they may be given in terms of weight, by ρ_i. The parameters used to define media are summarized as items 28 through 33 in Table 2.1.

The PEGS subroutine, `MIX`, uses the material parameters described above to compute additional molecular parameters (items 34 through 53 in Table 2.1) useful in computing cross sections for mixtures, as will be seen in subsequent sections.

Table 2.1: Symbols used in EGS5 and PEGS5.

	Math	FORTRAN	Program	Definition
1.	π	PI	P	3.1415926536
2.	c	C	P	speed of light $= 2.997925 \times 10^{10}$ cm/sec
3.	m_e	RME	P	electron rest mass $= 9.1091 \times 10^{-28}$ g
4.	$\hbar$	HBAR	P	Planck's constant$/2\pi$ $= 1.05450 \times 10^{-27}$ erg-sec
5.	e_{CGS}	ECGS	P	electronic charge (CGS units) $= 4.80298 \times 10^{-10}$ esu
6.	e_{MKS}	EMKS	P	electronic charge (MKS units) $= 1.60210 \times 10^{-19}$ Coul
7.	N_a	AN	P	Avogadro's number $= 6.02252 \times 10^{23}$ mole^{-1}
8.	$(1 \text{ Rad})[°]$	RADDEG	P	radian expressed in degrees $= 180/\pi$
9.	α	FSC	P	fine structure constant $= e_{\text{CGS}}^2/(\hbar c)$
10.	$(1 \text{ MeV})[\text{erg}]$	ERGMEV	P	one MeV expressed in ergs $= e_{\text{MKS}} \times 10^{13}$
11.	r_0	RO	P	classical electron radius $= e_{\text{CGS}}^2/(m_e c^2)$
12.	m	RM	PE	electron rest energy in MeV $= m_e c^2/\texttt{ERGMEV}$
13.	2m	RMT2	PE	2m
14.	m^2	RMSQ	PE	m^2
15.	'22.9'	A22P9	P	constant in Nagel's formula for χ_c in degrees. Taken from Molière's paper in which χ_c was expressed in grads. So to get χ_c in degrees, should really use 22.9 $\times 360/400$. Defined as A22P9 = RADDEG $\sqrt{4\pi N_a}$ $\times e_{\text{CGS}}^2/\texttt{ERGMEV} = 22.6960$
16.	'6680'	A6680	P	constant in Bethe's formula for Molière's b $= 4\pi N_a(\hbar/m_e c)^2(0.885)^2$ $/(1.167 \times 1.13) = 6702.33$
17.	k, k_1		P	photon energy in ergs
18.	$\breve{k}$, $\breve{k}_1$		P	photon energy in MeV
19.	E, E_1		P	electron total energy in ergs
20.	$\breve{E}$, $\breve{E}_1$		P	electron total energy in MeV
21.	T		P	electron kinetic energy in ergs
22.	$\breve{T}$		P	electron kinetic energy in MeV
23.	t		P	distance in centimeters
24.	$\breve{t}$		P	distance in radiation lengths $= t/X_0$
25.	σ		P	cross section in cm^2/(atom or molecule)

Table 2.1 (cont.)

	Math	FORTRAN	Program	Definition
26.	Σ		P	macroscopic cross section (probability of interaction/cm) $= N_a \rho \sigma / M$
27.	$\breve{\Sigma}$		P	macroscopic cross section (in X_0^{-1} units) $= X_0 \Sigma$
28.	ρ	RHO	PE	material density (g/cm^3)
29.	N_e	NE	PE	number of elements in the material
30.	Z_i	Z(I)	P	atomic number of the ith element of the material
31.	A_i	WA(I)	P	atomic weight of the ith element of the material (g/mole)
32.	p_i	PZ(I)	P	proportion by number of the ith element in the material $\propto \rho_i / A_i$
33.	ρ_i	RHOZ(I)	P	proportion by weight of the ith element in the material $\propto p_i A_i$
34.	M	WM	P	molecular weight (g/mole) $= \sum_{i=1}^{N_e} p_i A_i$
35.	C_M	ZC	P	molecular charge (electrons/ molecule) $= \sum_{i=1}^{N_e} p_i Z_i$
36.	n	EDEN	P	electron density $(electrons/cm^3)$ $= N_a \rho C_M / M$
37.	X_0	RLC	PE	radiation length $= (N_a \rho 4 \alpha r_0^2 (Z_{AB} - Z_F)/M)^{-1}$
38.	Z_T	ZT	P	$\sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_i)$
39.	Z_B	ZB	P	$\sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_i) \ell n \, Z_i^{-1/3}$
40.	Z_F	ZF	P	$\sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_i) f_c(Z_i)$
41.	Z_S	ZS	P	$\sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_{MS})$
42.	Z_E	ZE	P	$\sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_{MS}) \ell n \, Z_i^{-2/3}$
43.	Z_X	ZX	P	$\sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_{MS})$ $\times \ell n \, (1 + 3.34(\alpha Z_i)^2)$
44.	Z_A	ZA	P	$Z_T \, \ell n \, 183$
45.	Z_G	ZG	P	Z_B / Z_T
46.	Z_P	ZP	P	Z_B / Z_A
47.	Z_V	ZV	P	$(Z_B - Z_F)/Z_T$
48.	Z_U	ZU	P	$(Z_B - Z_F)/Z_A$

Note: The next four are used in EGS directly and computed in MIX

	Math	FORTRAN	Program	Definition
49.	b_c	BLCC	PE	'6680' $\rho Z_S \exp(Z_E/Z_S) \, X_0$ $/(M \, \exp(Z_X/Z_S))$
50.	χ_{cc}	XCC	PE	('22.9'/RADDEG) $\sqrt{Z_S \rho X_0 / M}$
51.	t_{eff_0}	TEFF0	PE	$(\exp(B_{min})/B_{min})/b_c$
52.	χ_{r_0}	XR0	PE	$\chi_{cc} \sqrt{t_{eff_0} B_{min}}$
53.	ξ_{MS}	FUDGEMS	P	constant used to account for multiple scattering of atomic electrons

Table 2.1 (cont.)

	Math	FORTRAN	Program	Definition
54.	ξ_i	XSI(I)	P	constant for bremsstrahlung and pair production of atomic electrons for element Z_i $= L'_{rad}(Z_i)/[L_{rad}(Z_i) - f_c(Z_i)]$
55.	$f_c(Z_i)$	FCOUL(I)	P	bremsstrahlung and pair production Coulomb correction constant for element Z_i
56.	B_{min}	BMIN	P	
57.	λ	MFP	P	mean free path
58.	$A'(Z_i, E)$	APRIME(I)	P	empirical bremsstrahlung correction factor
59.	U(x)		P	unit step function
60.	$\phi_1(\delta)$		P	first screening function
61.	$\phi_2(\delta)$		P	second screening function
62.	δ_i		P	screening parameter for element Z_i
63.	δ'		P	average screening parameter for a mixture
64.	A_P	AP	PE	low energy threshold for soft bremsstrahlung production
65.	A_E	AE	PE	low energy threshold for knock-on electron production
66.	$\hat{\zeta}$, $\hat{\zeta}_i$, ζ, ζ_i,			uniform random variables and sampled value
67.	α_i			weight of ith element in sampling method
68.	λ_0			Compton wavelength of electron $= 2\pi r_0/\alpha = 2.4262 \times 10^{-10}$ cm
69.	$\bar{I}_{adj}$	IEV	P	average adjusted mean ionization energy
70.	δ	DELTA	P	density effect correction
Note: The following are Sternheimer (density effect) fit parameters				
71.	a	AFACT	P	
72.	m_s	SK	P	
73.	x_0	X0	P	
74.	x_1	X1	P	
75.	$-C$	CBAR	P	
Note: The following are used in the new definition of radiation length				
76.	'183'	A183	P	constant in L_{rad} which used to be 183, now 184.15
77.	'1440'	A1440	P	constant in L_{rad} which used to be 1440, now 1194.
78.	L_{rad}	ALRAD	P	a) $\ell n 184.15 Z^{-1/3}$ for $Z > 4$ b) tabulated value for $Z \le 4$
79.	L'_{rad}	ALRADP	P	a) $\ell n 1440 Z^{-2/3}$ for $Z > 4$ b) tabulated value for $Z \le 4$
80.	Z_{AB}	ZAB	P	$\sum_{i=1}^{N_e} p_i Z_i(Z_i + \xi_i)L_{rad}(Z_i)$
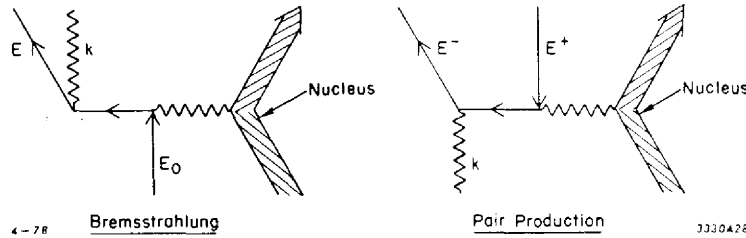
Figure 2.2: Feynman diagrams for bremsstrahlung and pair production.

2.7   Bremsstrahlung and Electron-Positron Pair Production

The bremsstrahlung and pair production processes are closely related, as can be seen from the Feynman diagrams in Figure 2.2 . In the case of bremsstrahlung, an electron or positron is scattered by two photons: a virtual photon from the atomic nucleus and another free photon which is created by the process. In the case of pair production, an electron traveling backward in time (a positron) is also scattered by two photons, one of which scatters it forward in time making it into an electron. The net effect is the absorption of the photon and creation of an electron-positron pair.

The discussions and descriptions of these processes which are given here use formulas taken from the review articles by Koch and Motz[91] on bremsstrahlung and by Motz, Olsen and Koch[111] on pair production. We also employ some ideas from Butcher and Messel[39] for mixing the cross sections for sampling of the secondary spectra. Below 50 MeV the Born approximation cross sections are used with empirical corrections added to get agreement with experiment. Above 50 MeV the extreme relativistic Coulomb corrected cross sections are used.

The "shower book" by Messel and Crawford[103] takes into account the Landau-Pomeranchuk-Migdal (LPM) "suppression effect"[95, 94, 106] which is important at electron energies greater than 100 GeV for bremsstrahlung and greater than a TeV or so for pair production. At these energies, the LPM effect, which had been demonstrated experimentally [9], manifests as significant reductions ("suppression") in the total bremsstrahlung and pair production cross sections. EGS5 does not currently model the LPM effect. In addition, an effect due to polarization of the medium (which apparently is effective even at ordinary energies) results in the cutoff of the bremsstrahlung differential cross section at secondary photon energies below a certain fraction of the incident electron energy. This has been quantified in terms of a factor F_P [103], given by

$$F_P = \left(1 + \frac{nr_0\lambda_0^2\,E_0^2}{\pi k^2}\right)^{-1} \tag{2.37}$$

where n is the electron density, r_0 is the classical electron radius, λ_0 is the Compton wavelength of an electron, and E_0 and k are the energies of the electron and photon, respectively. If we define a cutoff energy by

$$k_c = E_0\sqrt{nr_0\lambda_0^2/\pi} \tag{2.38}$$

then we see that for $k \gg k_c$, F_P goes to one; for $k \approx k_c$, it is about 1/2; and for $k \ll k_c$ it goes as k^2/k_c^2. In the latter case, the k^2 factor multiplied by the usual $1/k$ dependence results in an overall k dependence as $k \to 0$. Thus we have finite differential and total cross sections and the infrared catastrophe is averted. It can be seen that the ratio of the cutoff energy to the incident electron energy is independent of energy, and depends on the medium only through its electron density. For lead, the ratio is

$$
\begin{aligned}
k_c/E_0 &= \sqrt{nr_0\lambda_0^2/\pi} \\
&= 1.195 \times 10^{-4}
\end{aligned}
\tag{2.39}
$$

The natural log of the inverse of this ratio ($\simeq 9$) is then approximately equal to the total bremsstrahlung cross section in units of inverse radiation lengths (if one takes $d\sigma/dk\ [X_0^{-1}] = 1/k$ for $k > k_c$, $= 0$ for $k < k_c$).

As the corrections presented above have not been implemented in EGS5, it is instructive to investigate the magnitude of the error expected this introduces. First, it is clear that at energies above 100 GeV, ignoring the LPM suppression effect will have a significant impact on the predicted gross behavior of a shower because of the over-estimation of the bremsstrahlung and pair production cross sections. We therefore set 100 GeV as a safe upper limit to the present EGS5 version. Next, we see that by ignoring the bremsstrahlung cutoff k_c, EGS5 simulations of high energy electron transport produce too many low energy secondary photons. For example, a 10 GeV electron should not emit many photons below 1 MeV, whereas EGS5 would continue production down possibly as low as 1 keV. This should not disturb the general shower behavior much, as there will be many more low energy electrons than high energy electrons, so that the few extra low energy photons produced by the high energy electrons should be insignificant compared to the number of low energy photons produced by the lower energy electrons. It should be clear, however, that if the user were using EGS5 to determine thin target bremsstrahlung spectra from high energy electrons, the results would be in error below the cutoff k_c.

Neglecting possible crystal diffraction effects[55, 170], the macroscopic cross section for bremsstrahlung or pair production is given in terms of the microscopic cross sections, σ_i, for the atoms of type i by

$$
\Sigma = \frac{N_a\rho}{M} \sum_i p_i\sigma_i = N_a\rho \frac{\sum_i p_i\sigma_i}{\sum_i p_i A_i} .
\tag{2.40}
$$

We see that the macroscopic cross sections do not depend on the absolute normalization of the p_i's, only the ratios. With the exception of ionization losses (where polarization effects are important), Equation 2.40 is also valid for the other interactions that are considered (*e.g.*, Møller, Compton, etc.).

For conciseness in what follows we shall use the notation

$$
(E_1 \text{ if } B_1, \ \ldots, \ E_n \text{ if } B_n, \ E_{n+1})
\tag{2.41}
$$

to denote the conditional expression which takes E_i for its value if B_i is the first true expression, and takes E_{n+1} for its value if no B_i is true. We will also make use of the Kronecker delta function

defined by

$$\delta_{ij} = \begin{array}{ll} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{array} \tag{2.42}$$

Using this notation, we start with the following formulas for the bremsstrahlung and pair production differential cross sections:[5].

$$\frac{d\sigma_{Brem}(Z, \breve{E}_0, \breve{k})}{d\breve{k}} = \frac{A'(Z, \breve{E}_0) r_0^2 \; \alpha Z(Z + \xi(Z))}{\breve{k}}$$

$$\times \left\{ \left(1 + (\breve{E}/\breve{E}_0)^2\right) \left[\phi_1(\delta) - \frac{4}{3}\ell n \; Z - (4f_c(Z) \text{ if } \breve{E}_0 > 50, 0)\right] \right.$$

$$\left. - \frac{2}{3} \left(\frac{\breve{E}}{\breve{E}_0}\right) \left[\phi_2(\delta) - \frac{4}{3}\ell n \; Z - (4 \; f_c(Z) \text{ if } \breve{E}_0 > 50, 0)\right] \right\} \tag{2.43}$$

and

$$\frac{d\sigma_{Pair}(Z, \breve{k}, \breve{E}_+)}{d\breve{E}_+} = \frac{A'_p(Z, \breve{k}) r_0^2 \; \alpha Z(Z + \xi(Z))}{\breve{k}^3}$$

$$\times \left\{ \left(\breve{E}_+^2 + \breve{E}_-^2\right) \left[\phi_1(\delta) - \frac{4}{3}\ell n \; Z - (4f_c(Z) \text{ if } \breve{k} > 50, 0)\right] \right.$$

$$\left. + \frac{2}{3} \; \breve{E}_+ \breve{E}_- \left[\phi_2(\delta) - \frac{4}{3}\ell n \; Z - (4 \; f_c(Z) \text{ if } \breve{k} > 50, 0)\right] \right\} \tag{2.44}$$

where

$$\delta = 136 \; Z^{-1/3} \; 2\Delta \tag{2.45}$$

and

$$\Delta = \frac{\breve{k}m}{2\breve{E}_0\breve{E}} \quad \text{(for bremsstrahlung)} \tag{2.46}$$

$$= \frac{\breve{k}m}{2\breve{E}_+\breve{E}_-} \quad \text{(for pair production)} . \tag{2.47}$$

To avoid confusion it should be noted that our δ is the same as the δ of Butcher and Messel[39] but we use $\phi_i(\delta)$ to denote their $f_i(\delta)$. Rossi[141] and Koch and Motz[91] use a variable $\gamma = \frac{100}{136}\delta$. Also, note that our $\phi_i(\delta)$ has the same value as the $\phi_i(\gamma)$ of Koch and Motz[91] (e.g., see their Figure 1) provided "our δ" = $\frac{136}{100}$ times "their γ." For arbitrary screening, ϕ_1 and ϕ_2 are given by

$$\phi_1(\delta) = 4 \int_\Delta^1 (q - \Delta)^2 [1 - F(q, Z)]^2 \frac{dq}{q^3} + 4 + \frac{4}{3}\ell n \; Z , \tag{2.48}$$

$$\phi_2(\delta) = 4 \int_\Delta^1 \left[q^3 - 6\Delta^2 q \; \ell n \left(\frac{q}{\Delta}\right) + 3\Delta^2 q - 4\Delta^3\right]$$

$$\times \; [1 - F(q, Z)]^2 \frac{dq}{q^4} + \frac{10}{3} + \frac{4}{3}\ell n \; Z \tag{2.49}$$

---

[5]As described earlier, we have adopted the notation that cupped energy variables (e.g., $\breve{E}$) are in MeV and cupped distance variables (e.g., $\breve{t}$) are in radiation lengths. Uncupped variables are in CGS units and constants are defined explicitly

where $\delta = 272 \ Z^{-1/3}\Delta$ as before, and where $F(q, Z)$ is the atomic form factor for an atom with atomic number Z. Following Nagel[112], we have used the Thomas-Fermi form factors, for which ϕ_1 and ϕ_2 are Z independent and have already been evaluated. Butcher and Messel [39]have approximated the screening functions to within 1–2% by the formulas

$$\phi_1(\delta) = (20.867 - 3.242\delta + 0.625\delta^2 \text{ if } \delta \leq 1, \ 21.12 - 4.184 \ ln(\delta + 0.952)) \qquad (2.50)$$

and

$$\phi_2(\delta) = (20.029 - 1.930\delta - 0.086\delta^2 \text{ if } \delta \leq 1, \ 21.12 - 4.184 \ ln(\delta + 0.952)). \qquad (2.51)$$

The Thomas-Fermi screening is quite accurate for high atomic numbers, but at low atomic numbers its accuracy decreases. The Hartree form factors are better for low Z. Tsai[172] has given a review of bremsstrahlung and pair production cross sections including best estimates of form factors and screening functions, and Seltzer and Berger[153, 154] have reviewed and presented new cross section data for bremsstrahlung production. EGS has not been modified to reflect these more accurate cross sections except that it redefines the radiation length and ξ_i to be consistent with the definitions by Tsai[172] (see discussion of these changes below). We have also checked that, for example, the values of ϕ_1 and ϕ_2, given by Equations 2.50 and 2.51, agree with Tsai's values within 0.4% for $Z > 4$ and within 5% for hydrogen.

$A'(Z, \breve{E}_0)$ in Equation 2.43 is an empirical correction factor evaluated by the function `APRIM` in PEGS. For $\breve{E}_0 > 50$ MeV, PEGS takes $A' = 1$ since it uses the Coulomb corrected formulas , which are accurate to about 3% in this energy range. For $\breve{E}_0 < 50$ MeV, PEGS5 uses values of A' generated by Rogers *et al.* [139]. This effectively renormalizes the bremsstrahlung cross sections to assure that the total radiative stopping powers (see section 2.13) agree with those published in ICRU Report 37 [79]. As an option, the user may request the PEGS4 corrections, which were interpolated in Z from the curves of Koch and Motz[91] (see their Figure 23, *e.g.*). With the availability of the better cross section data mentioned above, this methodology is somewhat more approximate than need be, and an improved treatment awaits development by some fresh, energetic EGS5 user.

The pair production empirical correction factor $A'_p(Z, \breve{k})$ in Equation 2.44 is defined as: $A'_p(Z, \breve{k})$ is equal to ("The Best Empirical Estimate of the Total Pair Production Cross Section for given $Z, \breve{k}$") divided by ("The Total Pair Production Cross Section obtained by integrating Equation 2.44 over all allowed $\breve{E}_+$ values, with $A'_p = 1$"). For $\breve{k} < 50$ MeV, we take this best estimate to be the data compiled by Storm and Israel[167] and in fact we use this data directly without resorting to Equation 2.44 whenever pair production total cross sections are needed for $\breve{k} < 50$. For $\breve{k} > 50$ MeV, an integration of Equation 2.44 with $A'_p = 1$ is used for the pair production total cross section. This agrees within a few percent with Storm and Israel[167] up to the limiting energy for which they present data but it does lead to a slight discontinuity in the photon cross section at 50 MeV. Thus $A'_p(Z, \breve{k} > 50)$ is taken to be 1, as it is for bremsstrahlung. Unlike as with bremsstrahlung, however, A'_p is never explicitly calculated since it is not needed in determining the total cross section, nor, as will be seen later, is it used in sampling the secondary particle energies.

The $f_c(Z)$ in Equations 2.43 and 2.44 is the Coulomb correction term that was derived by

Davies, Bethe and Maximon[49] (*e.g.*, see their formula 36, p. 791)) and is given by

$$f_c(Z) = a^2 \sum_{\nu=1}^{\infty} \frac{1}{\nu(\nu^2 + a^2)} \tag{2.52}$$

where

$$a = \alpha \ Z \ .$$

They also suggest a formula accurate to 4 digits up to $a = 2/3$ (which corresponds to Uranium); namely,

$$f_c(Z) = a^2 \left\{ (1 + a^2)^{-1} + 0.20206 - 0.0369a^2 + 0.0083a^4 - 0.002a^6 \right\} \tag{2.53}$$

which function `FCOULC` of PEGS uses to evaluate $f_c(Z)$.

$\xi(Z)$ is a function which is used to take into account bremsstrahlung and pair production in the field of the atomic electrons. Strictly speaking, these interactions are different from the corresponding nuclear interaction not only because the mass and charge of an electron are different from the nuclear mass and charge, but also because of the identity of the electrons. Because of the lightness of the electron, it may be ejected from the atom. In the bremsstrahlung case what we really have is radiative Møller or Bhabha scattering. In the case of pair production, if the atomic electron is ejected, we have three rather than two energetic electrons and the reaction is called triplet production. Because of the electron exchange effects and the $\gamma - e$ interactions between the external photon and the target electron, and also because the target can no longer be treated as infinitely heavy, the cross section calculations for these interactions are more complicated than for the corresponding nuclear cases and involve a larger number of approximations (see p. 631 of Motz *et al.* [111]). As will be seen below, the ratio of cross sections for the interaction in the electron fields to those in the nuclear field is of the order of $1/Z$. Thus, for medium-low to high Z, the contributions of the atomic electrons are rather minor. On the other hand, for low Z, such as beryllium and certainly for hydrogen, these interactions are very significant and a more accurate treatment of these interactions is warranted. Nevertheless, we have not treated the bremsstrahlung and pair production in the electronic fields in a special way, primarily because most applications of interest do not involve only very low Z elements. When low Z elements are involved, they have usually been mixed with higher Z elements, in which case the pair production and bremsstrahlung in the low Z elements are relatively unimportant. This does limit somewhat the universality of EGS.

For very high energy incident particles the screening can be considered complete. In this case, relatively simple formulas for the interaction in the atomic field can be obtained (Bethe and Ashkin[24] (formula 59 on p. 263 and formula 119 on p. 332), Koch and Motz[91] (formula III-8 on p. 949)). The relative values of the radiation integral

$$\phi_{rad} \equiv \frac{1}{E_0} \int_0^{k_{max}} k \ \frac{d\sigma_{Brem}}{dk} \ dk \tag{2.54}$$

can be used as an estimate of the relative magnitude of the interactions in the electron or nuclear fields. In the completely screened nuclear field, the radiation integral (formula 4CS of Koch and Motz[91]) is

$$\phi_{rad,nucleus} = 4\alpha r_0^2 \ Z^2 \ \left[\ell n \ (183 \ Z^{-1/3}) + \frac{1}{18} - f_c(Z) \right] \ . \tag{2.55}$$

41

For ϕ_{rad} in the completely screened electron field Koch and Motz[91] (formula III-8 on p. 949) give

$$\phi_{rad,electron} = 4\alpha r_0^2 \, Z \, \ell n \, (530 \, Z^{-2/3}) \; . \tag{2.56}$$

On the other hand, from the formulas of Bethe and Ashkin[24] mentioned above, one would expect

$$\phi_{rad,electron} = 4\alpha r_0^2 \, Z \, \ell n \, (1440 \, Z^{-2/3}) \; .$$

Tsai's work[172] has dealt with these problems more accurately. We have not switched to his method but continue to define a single parameter, ξ, which is used in a simple $Z(Z + \xi)$ correction to the cross sections to account for electron effects. However, we have redefined ξ making use of Tsai's radiation logarithms:

$$\xi(Z) = \frac{L'_{rad}(Z)}{L_{rad}(Z) - f_c(Z)} \tag{2.57}$$

where

$$L'_{rad} = \begin{cases} \ell n1194 \, Z^{-2/3} & \text{if } Z > 4 \\ 6.144 & \text{if } Z = 1 \\ 5.621 & \text{if } Z = 2 \\ 5.805 & \text{if } Z = 3 \\ 5.924 & \text{if } Z = 4 \end{cases}$$

$$L_{rad} = \begin{cases} \ell n184.15 \, Z^{-1/3} & \text{if } Z > 4 \\ 5.310 & \text{if } Z = 1 \\ 4.790 & \text{if } Z = 2 \\ 4.740 & \text{if } Z = 3 \\ 4.710 & \text{if } Z = 4 \end{cases}$$

These expressions are used by function XSIF of PEGS to compute $\xi(Z)$ for use in Equations 2.43 and 2.44. These definitions replace the simpler $\ell n1440 \, Z^{-2/3}$ and $\ell n183 \, Z^{-1/3}$ used in PEGS3. Note that for the rest of this chapter, "183" is a variable name representing the value 184.15, and similarly, "1440" represents 1194.

We have also changed the definition of radiation length to that of Tsai[172]:

$$X_0^{-1} = \frac{N_a \rho \alpha r_0^2}{A} \left[Z^2 [L_{rad}(Z) - f_c(Z)] + Z L'_{rad}(Z) \right] \; . \tag{2.58}$$

This change has a considerable effect on X_0 for very light elements (a 9% increase for hydrogen) but only a small effect for $Z \geq 5$.

We have done several comparisons between the pair production cross sections used in PEGS and the more accurate results of Tsai. In general, they agree to within a few percent except for very low Z ($<$ about 10) elements at energies below 1 GeV.

Now that we have discussed all items appearing in Equations 2.43 and 2.44, we need to mention some additional corrections to Equation 2.43 which are neglected in EGS5. First, the differential

cross section given in Equation 2.43 goes to zero at the maximum photon energy; whereas, in reality the bremsstrahlung cross section, differential in photon energy, is non-zero at the "high frequency limit" (Koch and Motz[91] (p. 933)). A more rigorous treatment would modify Equation 2.43 so as to smoothly approach the proper value at the high frequency limit (bremsstrahlung).

Another correction for Equation 2.43 ignored by EGS5 is the "Elwert factor", which Koch and Motz[91] recommend be applied below $\check{T} = 2$. For problems in which the bremsstrahlung from low energy electrons plays an important role, this correction factor is necessary to produce accurate results. This is therefore a restriction on the use of EGS. However, radiative yield rises rapidly with electron energy. For example, as an electron slows down from 20 MeV, it loses about 8% of its energy through radiative loses, but of this, only 0.5% is lost by electrons with energies below 5 MeV. Thus ignoring the Elwert correction factor introduces a negligible error into most problems, especially those dealing with high energy electrons.

We now discuss the methods we use to sample the secondary energies for bremsstrahlung and pair production interactions. Our methods are based on Equations 2.43 and 2.44 with a couple of approximations. For the purpose of our discussion let x and x_0 be the secondary and incident particle energies, respectively. Then Equations 2.43 and 2.44 are of the form

$$\frac{d\sigma_{corrected}(Z, x_0, x)}{dx} = A'(Z, x_0) \, \frac{d\sigma_{uncorrected}(Z, x_0, x)}{dx} \; . \tag{2.59}$$

According to Equation 2.40 the differential macroscopic cross section, properly weighted with the various constituent materials, will be

$$\frac{d\Sigma(x_0, x)}{dx} = \frac{N_a \rho}{M} \sum_{i=1}^{N_e} p_i \, \frac{d\sigma_{corrected}(Z_i, x_0, x)}{dx} \tag{2.60}$$

and the total macroscopic cross section will be

$$\Sigma(x_0) = \int_{x_{min}}^{x_{max}} \frac{d\Sigma(x_0, x)}{dx} \, dx \tag{2.61}$$

If the incident particle has undergone an interaction of this type, then the probability density function for x will be

$$f(x) = \frac{d\Sigma(x_0, x)}{dx} \bigg/ \Sigma(x_0) \; . \tag{2.62}$$

We now observe that any constant factors in the right hand side of Equation 2.60 will cancel out in Equation 2.62, and, in particular, for materials consisting of only one element the correction factor $A'(Z, x_0)$ will be an overall factor and so may be ignored. We now make the approximation that the factor $A'(Z, x_0)$ can be ignored for secondary sampling purposes even when there is more than one element. As will be seen, this will allow us to obtain energy independent screening factors. The way we could avoid making this approximation is to have PEGS generate branching ratios among all the constituents as a function of particle energy so that whenever an interaction took place, the first thing done would be to decide with which type of atom the interaction occurred. For complex mixtures this would take a significantly larger amount of data and running time and we have chosen to make the above approximation instead.

Now suppose that the $f(x)$ in Equation 2.62 is decomposed in a manner that allows for a combination of "composition" and "rejection" techniques (*e.g.*, see Equation 2.10; that is

$$\frac{d\Sigma(x_0, x)}{dx} \bigg/ \Sigma(x_0) = \sum_{i=1}^{N_e} \alpha_i'\, f_i(x)\, g_i(x) \tag{2.63}$$

or

$$\frac{d\Sigma(x_0, x)}{dx} = \sum_{i=1}^{N_e} \alpha_i\, f_i(x)\, g_i(x) \,, \tag{2.64}$$

where

$$\alpha_i = a_i'\Sigma(x_0) \,.$$

But it can be shown (*e.g.*, see Equation 2.7) that if all α_i are multiplied by the same factor, it will not change the function being sampled. We conclude that to properly sample from the p.d.f. (Equation 2.62), it is not necessary to obtain $\Sigma(x_0)$, but it is sufficient to decompose $d\Sigma(x_0, x)/dx$ as shown in Equation 2.64 without bothering to normalize. We seek decompositions of the form

$$\sum_{j=1}^{n} \alpha_j f_j(x) g_j(x) = \sum_{i=1}^{N_e} p_i \, \frac{d\sigma_{uncorrected}(Z_i, x_0, x)}{dx} \,. \tag{2.65}$$

Let us now do this for the bremsstrahlung process (in a manner similar to that of Butcher and Messel[39]) using Equation 2.43 for $d\sigma_i/dx$ with $A'(Z_i, \breve{E}_0) = 1$. The first point to be noted[114] is that $\phi_1(\delta)$ and $\phi_2(\delta) \to 0$ as $\delta \to \infty$, so that the expressions in square brackets in Equation 2.43 go to zero, and in fact go negative, at sufficiently large δ. As can be seen from Equation 2.51, $\phi_2(\delta) = \phi_1(\delta)$ for $\delta > 1$. By checking numerical values it is seen that the value of δ for which the expressions go to zero is greater than 1, so that both [ ] expressions in Equation 2.43 go to zero simultaneously. Clearly the differential cross section must not be allowed to go negative, so this imposes an upper kinematic limit $\delta_{max}(Z_i, \breve{E}_0)$ resulting in the condition (from Equation 2.50)

$$21.12 - 4.184\, \ell n(\delta_{max}(Z, \breve{E}_0) + 0.952) - \frac{4}{3}\ell n\, Z - (4f_c(Z) \text{ if } \breve{E}_0 > 50, 0) = 0. \tag{2.66}$$

Solving for $\delta_{max}(Z, \breve{E}_0)$ we obtain

$$\delta_{max}(Z, \breve{E}_0) = \exp\left[(21.12 - \frac{4}{3}\, \ell n\, Z - (4f_c(Z) \text{ if } \breve{E}_0 > 50, 0))/4.184\right] - 0.952 \,. \tag{2.67}$$

From PEGS routines `BREMDZ`, `BRMSDZ`, and `BRMSFZ`, which compute $d\sigma_{Brem}(Z, \breve{E}_0, \breve{k})/d\breve{k}$ as given by Equation 2.43, it may be seen that the result is set to zero if $\delta > \delta_{max}(Z, \breve{E}_0)$. Another way of looking at this is to define

$$\phi_i'(Z, \breve{E}_0, \delta) = \Big(\phi_i(\delta) \text{ if } \delta \leq \delta_{max}\ (Z, \breve{E}_0),\ \phi_i(\delta_{max}(Z, \breve{E}_0))\Big) \,. \tag{2.68}$$

We now define

$$\mathcal{E} = \breve{k}/\breve{E}_0 \tag{2.69}$$

44

and use this as the variable to be sampled instead of $\breve{k}$. The expressions for Δ and δ then become

$$\Delta = \frac{m\mathcal{E}}{2\breve{E}_0(1-\mathcal{E})} \tag{2.70}$$

and

$$\delta_i = \frac{136 \; Z_i^{-1/3} \; m\mathcal{E}}{\breve{E}_0(1-\mathcal{E})} \; . \tag{2.71}$$

Let us also define a variable (corresponding to DEL in EGS)

$$\Delta_E = \frac{\mathcal{E}}{(1-\mathcal{E})\breve{E}_0} \; , \tag{2.72}$$

so that

$$\delta_i = 136 \; mZ_i^{-1/3} \; \Delta_E \; . \tag{2.73}$$

Since overall factors do not matter let us factorize $X_0 d\Sigma_{Brem}/d\breve{k}$. From Equations 2.60 and 2.43 with $A' = 1$, and the definition of X_0 in Table 2.1, we obtain

$$
\begin{aligned}
\frac{d\breve{\Sigma}_{Brem}}{d\mathcal{E}} \quad &= \quad X_0 \frac{d\Sigma_{Brem}}{d\mathcal{E}} \\
&= \quad \left(\sum_{i=1}^{N_e} p_i \frac{d\sigma_{uncorrected}}{d\mathcal{E}} \right) \Big/ \left[4\alpha r_0^2 (Z_{AB} - Z_F) \right] \\
&= \quad \langle \sum_{i=1}^{N_e} p_i \frac{Z_i(Z_i + \xi(Z_i))}{\mathcal{E}} \; \Big\{ \; (1 + (1-\mathcal{E})^2) \\
&\quad \times \quad \left[\phi_1'(Z_i, \breve{E}_0, \delta_i) - \frac{4}{3} \; \ell n \; Z_i - \left(4f_c(Z_i) \; \text{if} \; \breve{E}_0 > 50, 0 \right) \right] - \frac{2}{3}(1-\mathcal{E}) \\
&\quad \times \quad \left[\phi_2'(Z_i, \breve{E}_0, \delta_i) - \frac{4}{3} \; \ell n \; Z_i - \left(4f_c(Z_i) \; \text{if} \; \breve{E}_0 > 50, 0 \right) \right] \Big\} \rangle \\
&\quad \times \quad \frac{1}{4(Z_{AB} - Z_F)} \; . \tag{2.74}
\end{aligned}
$$

For some brevity let us use

$$\phi_{ji}' \; \text{for} \; \phi_j'(Z_i\breve{E}_0, \delta_i), \; f_{ci}' \; \text{for} \; (f_c(Z_i) \; \text{if} \; \breve{E}_0 > 50, 0), \; \text{and} \; \xi_i \; \text{for} \; \xi(Z_i) \; .$$

Then after some rearrangement Equation 2.74 becomes

$$
\begin{aligned}
\frac{d\breve{\Sigma}_{Brem}}{d\mathcal{E}} \quad &= \quad \frac{1}{4(Z_{AB} - Z_F)} \sum_{i=1}^{N_e} p_i Z_i(Z_i + \xi_i) \Big\{ \left(\frac{2}{3} \right) [3\phi_{1i}' - \phi_{2i}' \tag{2.75} \\
&\quad + \quad 8(\ell n \; Z_i^{-1/3} - f_{ci}')] \left(\frac{1-\mathcal{E}}{\mathcal{E}} \right) + \left[\phi_{1i}' + 4(\ell n \; Z_i^{-1/3} - f_{ci}') \right] \mathcal{E} \Big\}
\end{aligned}
$$

Now define

$$\hat{\phi}_j(\Delta_E) = \sum_{i=1}^{N_e} p_i Z_i(Z_i + \xi_i)\phi_j'(Z_i, \breve{E}_0, \; 136 \; Z_i^{-1/3} \; m\Delta_E) \; \text{for} \; j = 1, 2 \; . \tag{2.76}$$

45

Also, recall from Table 2.1 the definition of Z_B and Z_F. We then can rewrite Equation 2.76 as

$$\frac{d\breve{\Sigma}_{Brem}}{d\breve{k}} = \left\{ \left(\tfrac{2}{3}\right) \left[3\hat{\phi}_1(\Delta_E) - \phi_2(\Delta_E) + 8\left(Z_B - (Z_F \text{ if } \breve{E}_0 > 50, 0)\right)\right] \right.$$
$$\times \left(\tfrac{1-\mathcal{E}}{\mathcal{E}}\right) + \left[\hat{\phi}_1(\Delta_E) + 4\left(Z_B - (Z_F \text{ if } \breve{E}_0 > 50, 0)\right)\right] \left.\mathcal{E}\right\}$$
$$\times \tfrac{1}{4(Z_{AB} - Z_F)} . \tag{2.77}$$

Now let

$$\hat{A}(\Delta_E, \breve{E}_0) = 3\hat{\phi}_1(\Delta_E) - \hat{\phi}_2(\Delta_E) + 8(Z_B - (Z_F \text{ if } \breve{E}_0 > 50, 0)) \tag{2.78}$$

and

$$\hat{B}(\Delta_E, \breve{E}_0) = \hat{\phi}_1(\Delta_E) + (Z_B - (Z_F \text{ if } \breve{E}_0 > 50, 0)). \tag{2.79}$$

One can show that $\phi_j(\delta)$ have their maximum values at $\delta = 0$, at which they take values[39]

$$\phi_1(0) = 4 \, \ell n \, 183 \tag{2.80}$$
$$\phi_2(0) = \phi_1(0) - 2/3 . \tag{2.81}$$

The $\hat{\phi}_j(\Delta_E)$ also are maximum at $\Delta_E = 0$ and, in fact (see Table 2.1),

$$\hat{\phi}_1(0) = \sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_i)\phi_1(0) = 4Z_A , \tag{2.82}$$

$$\hat{\phi}_2(0) = \sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_i)\phi_2(0) = 4Z_A - \frac{2}{3} Z_T . \tag{2.83}$$

The maximum values of $\hat{A}$ and $\hat{B}$ are now given by

$$\hat{A}_{max}(\breve{E}_0) = \hat{A}(0, \breve{E}_0) = 3(4Z_A) - \left(4Z_A - \frac{2}{3} Z_T\right) + 8\left(Z_B - (Z_F \text{ if } \breve{E}_0 > 50, 0)\right)$$

$$= \frac{2}{3} Z_T + 8(Z_A + Z_B - (Z_F \text{ if } \breve{E}_0 > 50, 0)) , \tag{2.84}$$
$$\hat{B}_{max}(\breve{E}_0) = \hat{B}(0, \breve{E}_0) = 4\left(Z_A + Z_B - (Z_F \text{ if } \breve{E}_0 > 50, 0)\right) . \tag{2.85}$$

Now define δ' as the weighted geometric mean of the δ_i; that is,

$$\delta' \equiv \left(\prod_{i=1}^{N_e} \delta_i^{p_i Z_i (Z_i + \xi_i)}\right)^{\left[\sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_i)\right]^{-1}}$$

$$= \exp\left\{ Z_T^{-1} \sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_i)\ell n \, \delta_i\right\} \tag{2.86}$$

$$= 136 \, m\Delta_E \, \exp\left\{ Z_T^{-1} \sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_i)\ell n \, Z_i^{-1/3}\right\}$$

$$= 136 \, m\Delta_E \, \exp\left(Z_B/Z_T\right) = 136 \, m \, e^{Z_G}\Delta_E .$$

Or if we define

$$\Delta_C = 136 \, m \, e^{Z_G} , \tag{2.87}$$

46

we have

$$\delta' = \Delta_C \Delta_E \ . \tag{2.88}$$

Now define

$$A(\delta') = \hat{A}(\delta'/\Delta_C, \breve{E}_0)/\hat{A}_{max}(\breve{E}_0) \ , \tag{2.89}$$
$$B(\delta') = \hat{B}(\delta'/\Delta_C, \breve{E}_0)/\hat{B}_{max}(\breve{E}_0) \ . \tag{2.90}$$

Then A and B have maximum values of 1 at $\delta' = 0$, and are thus candidate rejection functions.

We are now ready to explain the reason for introducing the parameter δ' and to introduce our final approximation; namely, we assume that the $\hat{\phi}_j$ can be obtained using

$$\hat{\phi}_j(\delta'/\Delta_c) = \sum_{i=1}^{N_e} p_i Z_i(Z_i + \xi_i)\phi_j'(Z_i, \breve{E}_0, \ 136 \ Z_i^{-1/3} \ m\delta'/\Delta_c)$$

$$\approx \phi_j(\delta') \sum_{i=1}^{N_e} p_i Z_i(Z_i + \xi_i) = \phi_j(\delta')Z_T \ . \tag{2.91}$$

In order to justify the reasonableness of this approximation, assume for all i that

$$1 \ll 136 \ Z_i^{-1/3} \ m\delta'/\Delta_c < \delta_{max}(Z_i, \breve{E}_0) \ . \tag{2.92}$$

Then using Equations 2.50 and 2.51 for the ϕ_j we obtain

$$\sum_{i=1}^{N_e} p_i Z_i(Z_i + \xi_i)\phi_j(136 \ Z_i^{-1/3} \ m\delta'/\Delta_c)$$

$$= \sum_{i=1}^{N_e} p_i Z_i(Z_i + \xi_i) \left[21.12 - 4.184 \ \ell n \left(\left(136 \ Z_i^{-1/3} \ m\delta'/\Delta_c\right) + 0.952\right)\right]$$

$$\approx \sum_{i=1}^{N_e} p_i Z_i(Z_i + \xi_i) \left[21.12 - 4.184 \ \ell n \ \left(136 \ Z_i^{-1/3} \ m\delta'/\Delta_c\right)\right]$$

$$= \left[21.12 - 4.184 \ \ell n \ (136 \ m\delta'/\Delta_c)\right] \ Z_T - 4.184 \sum_{i=1}^{N_e} p_i Z_i(Z_i + \xi_i)\ell n \ Z_i^{-1/3}$$

$$= \left[21.12 - 4.184 \ (\ell n \ (136 \ m\delta'/\Delta_c) + Z_G)\right] \ Z_T$$

$$= \left(21.12 - 4.184 \ \ell n \ \delta'\right) Z_T$$

$$\approx \left[21.12 - 4.184 \ \ell n \ (\delta' + 0.952)\right] \ Z_T$$

$$= \phi_j(\delta')Z_T \ . \qquad \text{Q.E.D} \tag{2.93}$$

Butcher and Messel[39] make this approximation, although they don't mention it explicitly. As with our previous approximation, this approximation could be avoided (i.e., if we were willing to have PEGS fit the A and B functions in some convenient way for EGS).

We proceed now by using Equation 2.91 to eliminate the $\hat{\phi}_j$ from Equation 2.78 and Equation 2.79 yielding

$$\hat{A}(\Delta_E, \breve{E}_0) \quad = \quad [3\phi_1(\delta') - \phi_2(\delta')] Z_T$$
$$+ 8\left(Z_B - (Z_F \text{ if } \breve{E}_0 > 50, 0)\right) , \qquad (2.94)$$

$$\hat{B}(\Delta_E, \breve{E}_0) \quad = \quad \phi_1(\delta') Z_T + 4\left(Z_B - (Z_F \text{ if } \breve{E}_0 > 50, 0)\right) . \qquad (2.95)$$

If these are now used in Equations 2.89 and 2.90 we obtain

$$A(\delta') \quad = \quad \frac{3\phi_1(\delta') - \phi_2(\delta') + 8(Z_V \text{ if } \breve{E}_0 > 50, Z_G)}{\frac{2}{3} + 8\left[\ell n\ 183 + (Z_V \text{ if } \breve{E}_0 > 50, Z_G)\right]} \qquad (2.96)$$

$$B(\delta') \quad = \quad \frac{\phi_1(\delta') + 4(Z_V \text{ if } \breve{E}_0 > 50, Z_G)}{4\left[\ell n\ 183 + (Z_V \text{ if } \breve{E}_0 > 50, Z_G)\right]} . \qquad (2.97)$$

We now return to Equation 2.77 which we were trying to factor. We have

$$\frac{d\breve{\Sigma}_{Brem}}{d\mathcal{E}} \quad = \quad \left\{ \frac{2}{3} A(\delta')\hat{A}_{max}(\breve{E}_0) \left(\frac{1-\mathcal{E}}{\mathcal{E}}\right) + B(\delta')\hat{B}_{max}(\breve{E}_0)\ \mathcal{E} \right\}$$

$$\times \quad \frac{1}{4(Z_{AB} - Z_F)}$$

$$= \quad \left\{ \frac{2}{3} A(\delta') \left(\frac{1-\mathcal{E}}{\mathcal{E}}\right) \left[\frac{2}{3} Z_T + 8\left(Z_A + Z_B - (Z_F \text{ if } \breve{E}_0 > 50, 0)\right)\right] \right.$$

$$+ \quad \left. B(\delta')\mathcal{E}\ [4(Z_A + Z_B - (Z_F \text{ if } E_0 > 50, 0))] \right\}\ \frac{1}{4(Z_{AB} - Z_F)}$$

$$= \quad \frac{[Z_A + Z_B - (Z_F \text{ if } \breve{E}_0 > 50, 0)]}{(Z_{AB} - Z_F)}$$

$$\times \quad \left\{ \left[\frac{1}{9}\ \frac{Z_T}{[Z_A + Z_B - (Z_F \text{ if } \breve{E}_0 > 50, 0)]} + \frac{4}{3}\right] A(\delta')\ \left(\frac{1-\mathcal{E}}{\mathcal{E}}\right) \right.$$

$$\left. + B(\delta')\mathcal{E} \right\}$$

$$= \quad \frac{[Z_A + Z_B - (Z_F \text{ if } \breve{E}_0 > 50, 0)]}{(Z_{AB} - Z_F)}$$

$$\times \quad \left\{ \left[\ell n\ 2\left(\frac{4}{3} + \frac{1}{9\ell n\ 183[1 + (Z_U \text{ if } \breve{E}_0 > 50, Z_P)]}\right)\right] \right.$$

$$\left. \times \left[\frac{1}{\ell n\ 2}\left(\frac{1-\mathcal{E}}{\mathcal{E}}\right)\right]\ [A(\delta')] + \left[\frac{1}{2}\right][2\mathcal{E}][B(\delta')] \right\} . \qquad (2.98)$$

We then see that for $\breve{E}_0 \leq 50$, the case dealt with in Butcher and Messel[39] , we have

$$\frac{d\breve{\Sigma}_{Brem}}{d\mathcal{E}} \quad = \quad \left\{ \left[\ell n\ 2\left(\frac{4}{3} + \frac{1}{9\ell n\ 183(1 + Z_P)}\right)\right] \left[\frac{1}{\ell n\ 2}\left(\frac{1-\mathcal{E}}{\mathcal{E}}\right)\right]\ [A(\delta')] \right.$$

$$\left. + \left[\frac{1}{2}\right][2\mathcal{E}][B(\delta')] \right\} \frac{(Z_A + Z_B)}{(Z_{AB} - Z_F)} . \qquad (2.99)$$

48

This agrees with formula (10) of Butcher and Messel[39] (except that they use $Z_{AB} = Z_A + Z_B$ and do not use Z_F in their X_0 definition), since our Z_A, Z_B, Z_P are the same as their a, b, p. Now ignoring the factor preceding the {} in Equation 2.98, noting that we require $\check{k} > A_P$ (the photon energy cutoff) and also that energy conservation requires $\check{k} < \check{E}_0 - m$, we obtain the factorization (see Equation 2.64)

$$\alpha_1 = \ell n \ 2 \left(\frac{4}{3} + \frac{1}{9 \ \ell n \ 183 \left[1 + (Z_U \ \text{if} \ \check{E}_0 > 50, Z_P) \right]} \right) \ , \tag{2.100}$$

$$f_1(\mathcal{E}) = \frac{1}{\ell n \ 2} \left(\frac{1 - \mathcal{E}}{\mathcal{E}} \right) \quad \text{for} \ \mathcal{E} \in (0, 1) \ , \tag{2.101}$$

$$g_1(\mathcal{E}) = \left(A \left(\delta'(\mathcal{E}) \right) \ \text{if} \ \mathcal{E} \check{E}_0 \ \in \left(A_P, \ \check{E}_0 - m \right), 0 \right) \ , \tag{2.102}$$

$$\alpha_2 = \frac{1}{2} \ , \tag{2.103}$$

$$f_2(\mathcal{E}) = 2\mathcal{E} \ \text{for} \ \mathcal{E} \in (0, 1) \ , \tag{2.104}$$

$$g_2(\mathcal{E}) = \left(B \left(\delta'(\mathcal{E}) \right) \ \text{if} \ \mathcal{E} \check{E}_0 \ \in \ (A_P, \check{E}_0 - m), 0 \right) \ . \tag{2.105}$$

We notice that $f_2(\mathcal{E})$ is properly normalized, but that $f_1(\mathcal{E})$ has infinite integral over (0,1). Instead we limit the range over which $f_1(\mathcal{E})$ is sampled to $\left(2^{-N_{Brem}}, 1 \right)$, where N_{Brem} is chosen such that

$$2^{-N_{Brem}} \leq \frac{A_P}{\check{E}_0} < 2^{-(N_{Brem}-1)} \ . \tag{2.106}$$

To sample $f_1(\mathcal{E})$ we further factor it to

$$f_1(\mathcal{E}) = \sum_{j=1}^{N_{Brem}} \alpha_{1j} \ f_{1j}(\mathcal{E}) \ g_{1j}(\mathcal{E}) \tag{2.107}$$

where

$$\alpha_{1j} = 1 \ , \tag{2.108}$$

$$f_{1j}(\mathcal{E}) = \left(\frac{1}{\ell n \ 2} \ 2^{j-1} \ \text{if} \ \mathcal{E} < 2^{-j} \ , \frac{1}{\ell n \ 2} \frac{(1 - \mathcal{E} 2^{j-1})}{\mathcal{E}} \ \text{if} \ \mathcal{E} \in \left(2^{-j}, \ 2^{-j+1} \right), 0 \right) \ , \tag{2.109}$$

$$g_{1j}(\mathcal{E}) = 1 \ . \tag{2.110}$$

The f_{1j} are properly normalized distributions.

We sample $f_2(\mathcal{E})$ by selecting the larger of two uniform random variables (see Section 2.2); namely,

$$\mathcal{E} = \max \left(\zeta_1, \ \zeta_2 \right) \ , \tag{2.111}$$

where ζ_1 and ζ_2 are two random numbers drawn uniformly on the interval $(0, 1)$. To sample $f_1(\mathcal{E})$, we first select the sub-distribution index

$$j = \text{Integer Part} \ (N_{Brem} \ \zeta_1) + 1 \ . \tag{2.112}$$

Then to sample from $f_{1j}(\mathcal{E})$, first let

$$p = 2^{1-j} , \quad \mathcal{E}' = \mathcal{E}/p . \tag{2.113}$$

We then relate the distributions of $\mathcal{E}$ and $\mathcal{E}'$, as follows. Suppose $\hat{x}$, $\hat{y}$ are random variables with probability density functions $f(x)$ and $g(y)$ and cumulative distribution functions $F(x)$ and $G(y)$. Further, suppose that $\hat{x}$ is related to $\hat{y}$ by

$$\hat{x} = h(\hat{y}) \tag{2.114}$$

with h monotonic increasing ($\uparrow$) or decreasing ($\downarrow$). Then clearly

$$\begin{aligned}
F(x) &= Pr\{ \hat{x} < x \} = Pr\{ h(\hat{y}) < x \} \\
&= \left(Pr\{ \hat{y} < h^{-1}(x) \} \text{ if } h \uparrow, \ Pr\{ \hat{y} > h^{-1}(x) \} \right) \\
&= \left(G(h^{-1}(x)) \text{ if } h \uparrow, \ 1 - G(h^{-1}(x)) \text{ if } h \downarrow \right) .
\end{aligned} \tag{2.115}$$

Letting

$$x = h(y), \quad y = h^{-1}(x) , \tag{2.116}$$

we have

$$F(h(y)) = (G(y) \text{ if } h \uparrow, \ 1 - G(y) \text{ if } h \downarrow) . \tag{2.117}$$

Differentiating with respect to y, we obtain,

$$f(h(y)) \frac{dh(y)}{dy} = (g(y) \text{ if } h \uparrow, \ -g(y) \text{ if } h \downarrow) . \tag{2.118}$$

Now

$$\frac{dh(y)}{dy} = \left(\left| \frac{dh(y)}{dy} \right| \text{ if } h \uparrow, \ -\left| \frac{dh(y)}{dy} \right| \text{ if } h \downarrow \right) . \tag{2.119}$$

Hence

$$f(h(y)) = g(y) / \left| \frac{dh(y)}{dy} \right| , \tag{2.120}$$

$$f(x) = g\left(h^{-1}(x) \right) / \left| \frac{dh(h^{-1}(x))}{dy} \right| . \tag{2.121}$$

As an example, we claim that if

$$g(\mathcal{E}') = \frac{1}{\ell n \ 2} \left(1 \text{ if } \mathcal{E}' \in \left(0, \frac{1}{2} \right), \ \frac{1 - \mathcal{E}'}{\mathcal{E}'} \text{ if } \mathcal{E}' \in \left(\frac{1}{2}, 1 \right) \right) , \tag{2.122}$$

and

$$\mathcal{E} = h(\mathcal{E}') = \mathcal{E}'p, \ \mathcal{E}' = h^{-1}(\mathcal{E}) = \mathcal{E}/p, \ \frac{dh(\mathcal{E}')}{d\mathcal{E}'} = p , \tag{2.123}$$

then

$$f(\mathcal{E}) = \frac{1}{\ell n \ 2} \left(\frac{1}{p} \text{ if } \mathcal{E} \in (0, p/2), \ \frac{1 - \mathcal{E}/p}{\mathcal{E}} \text{ if } \mathcal{E} \in (p/2, p) \right) . \tag{2.124}$$

50

That is, using Equations 2.122 and 2.123 in Equation 2.121, we obtain

$$
\begin{aligned}
f(\mathcal{E}) &= g\left(h^{-1}(\mathcal{E})\right) \Big/ \left|\frac{dh}{dy}(h^{-1}(x))\right| && (2.125)\\
&= g(\mathcal{E}/p)/p \\
&= \frac{1}{\ell n\ 2}\left(\frac{1}{p} \text{ if } \frac{\mathcal{E}}{p} \in \left(0,\frac{1}{2}\right),\ \frac{1-\mathcal{E}/p}{\mathcal{E}/p}\cdot\frac{1}{p} \text{ if } \mathcal{E}/p \in \left(\frac{1}{2},1\right)\right)\\
&= \frac{1}{\ell n\ 2}\left(\frac{1}{p} \text{ if } \mathcal{E} \in (0,p/2),\ \frac{1-\mathcal{E}/p}{\mathcal{E}} \text{ if } \mathcal{E} \in (p/2,p)\right) \quad \text{Q.E.D.}
\end{aligned}
$$

Thus we sample $f_{1j}(\mathcal{E})$ by first sampling $\mathcal{E}'$ from Equation 2.121 and then letting $\mathcal{E} = \mathcal{E}'p$.

The $g(\mathcal{E}')$ in Equation 2.122 may be decomposed according to

$$
g(\mathcal{E}') = \sum_{i=1}^{2} \alpha_i'\ f_i'(\mathcal{E}') \ , \tag{2.126}
$$

with

$$
\alpha_1' = \frac{1}{2\ \ell n\ 2}\ ,\quad f_1'(\mathcal{E}') = \left(2 \text{ if } \mathcal{E}' \in \left(0,\frac{1}{2}\right),0\right)\ , \tag{2.127}
$$

$$
\alpha_2' = \left(1-\frac{1}{2\ \ell n\ 2}\right),\quad f_2'(\mathcal{E}') = \left(\frac{1}{(\ell n\ 2)-\frac{1}{2}}\cdot\frac{1-\mathcal{E}'}{\mathcal{E}'} \text{ if } \mathcal{E}' \in \left(\frac{1}{2},1\right),0\right) \ . \tag{2.128}
$$

We sample $f_1'(\mathcal{E}')$ by letting $\mathcal{E}' = \zeta/2$, where ζ is a random number uniformly drawn on the interval $(0,1)$. To sample $f_2'(\mathcal{E}')$ we let $\mathcal{E}' = 1-\frac{1}{2}x$, $x \in (0,1)$, and sample x from the frequency distribution function

$$
h(x) = \alpha''\ f''(x)\ g''(x) \tag{2.129}
$$

where

$$
\alpha'' = \frac{1}{(4\ \ell n\ 2)-2},\ f''(x) = 2x,\ g''(x) = \frac{1}{2-x} = \frac{0.5}{\mathcal{E}'} \tag{2.130}
$$

We already know how to sample $f''(x)$ (e.g., see Equation 2.111), so this completes the details of sampling the bremsstrahlung spectrum.

Let us now consider the pair production interaction. The general developments are quite analogous to the bremsstrahlung case and we obtain the following formulas:

$$
\mathcal{E} \equiv \frac{\breve{E}}{\breve{k}} \ , \tag{2.131}
$$

where $\breve{E}$ is the energy of one of the secondary electrons, $\breve{k}$ is the incident photon energy,

$$
\Delta_E = \frac{1}{\breve{k}\mathcal{E}(1-\mathcal{E})} \ , \tag{2.132}
$$

$$
\delta' = \Delta_C\Delta_E \ , \tag{2.133}
$$

$$\frac{d\breve{\Sigma}_{Pair}}{d\mathcal{E}} = \left(\frac{Z_A + Z_B - (Z_F \text{ if } \breve{k} > 50, 0)}{Z_{AB} - Z_F}\right)$$

$$\times \left\{ \left[\frac{2}{3} - \frac{1}{36 \; \ell n \; 183[1 + (Z_U \text{ if } \breve{k} > 50, \; Z_P)]}\right] [1] \; C(\delta')\right.$$

$$+ \left[\frac{1}{12}\left(\frac{4}{3} + \frac{1}{9 \; \ell n \; 183[1 + (Z_U \text{ if } \breve{k} > 50, \; Z_P)]}\right)\right]$$

$$\left. \times \left[12\left(\mathcal{E} - \frac{1}{2}\right)^2\right] A(\delta')\right\} , \qquad (2.134)$$

$$A(\delta') \quad = \text{same as for bremsstrahlung case,} \qquad (2.135)$$

$$C(\delta') \quad = \frac{3\phi_1(\delta') + \phi_2(\delta') + 16(Z_V \text{ if } \breve{k} > 50, \; Z_G)}{-\frac{2}{3} + (16 \; \ell n(183 + (Z_V \text{ if } \breve{k} > 50, \; Z_G))} , \qquad (2.136)$$

$$\Delta_C \quad = \text{same as for bremsstrahlung case.} \qquad (2.137)$$

We sample Equation 2.134 by the following decompositions:

$$\alpha_1 = \left[\frac{2}{3} - \frac{1}{36 \; \ell n \; 183[1 + (Z_U \text{ if } \breve{k} > 50, \; Z_P)]}\right] , \qquad (2.138)$$

$$f_1(\mathcal{E}) = 1, \quad \mathcal{E} \in (0, 1) , \qquad (2.139)$$

$$g_1(\mathcal{E}) = \left(C(\delta'(\mathcal{E})) \text{ if } \breve{k}\mathcal{E} \in (m, \; \breve{k} - m), 0\right) , \qquad (2.140)$$

$$\alpha_2 = \frac{1}{12}\left(\frac{4}{3} + \frac{1}{9 \; \ell n \; 183[1 + (Z_U \text{ if } \breve{k} > 50, \; Z_P)]}\right) \qquad (2.141)$$

$$f_2(\mathcal{E}) = 12\left(\mathcal{E} - \frac{1}{2}\right)^2 , \quad \mathcal{E} \in (0, 1) , \qquad (2.142)$$

$$g_2(\mathcal{E}) = \left(A(\delta'(\mathcal{E})) \text{ if } \breve{k}\mathcal{E} \in (m, \breve{k} - m), 0\right) . \qquad (2.143)$$

As with bremsstrahlung, we ignore the factor ahead of the {} in Equation 2.134 when sampling. This actually has the effect of giving some effective Coulomb correction below 50 MeV, since the factor neglected would have to be larger than 1 for $\breve{k} < 50$

We summarize the "run-time" bremsstrahlung and pair production cross sections:

$$\frac{d\breve{\Sigma}_{Brem, \; Run-time}}{d\mathcal{E}} = \left[\ell n \; 2\left(\frac{4}{3} + \frac{1}{9 \; \ell n \; 183[1 + (Z_U \text{ if } \breve{E}_0 > 50, Z_P)]}\right)\right]$$

$$\times \left[\frac{1}{\ell n \; 2}\left(\frac{1-\mathcal{E}}{\mathcal{E}}\right)\right] A(\delta') + \left[\frac{1}{2}\right] [2\mathcal{E}] \; B(\delta') , \qquad (2.144)$$

$$
\frac{d\breve{\Sigma}_{Pair,\ Run-time}}{d\mathcal{E}} = \left[\frac{2}{3} - \frac{1}{36\ \ell n\ 183[1+(Z_U\ \text{if}\ \breve{k}>50,Z_P)]} \right]\ [1]\ C(\delta')
$$
$$
+ \left[\frac{1}{12} \left(\frac{4}{3} + \frac{1}{9\ \ell n\ 183[1+(Z_U\ \text{if}\ \breve{k}>50,Z_P)]} \right) \right]
$$
$$
\times \left[12 \left(\mathcal{E} - \tfrac{1}{2} \right)^2 \right]\ A(\delta')\ . \tag{2.145}
$$

When these are divided by $\breve{E}_0$ and $\breve{k}$, respectively, they become

$$
\frac{d\breve{\Sigma}_{Brem,\ Run-time}}{d\breve{k}} \qquad \text{and} \qquad \frac{d\breve{\Sigma}_{Pair,\ Run-time}}{d\breve{E}}
$$

which are computed by PEGS functions `BREMDR` and `PAIRDR`. In general the letter R, as the last letter of a PEGS cross section function, means "run-time" function. PEGS may be used to plot these for comparison with the more exact `BREMDZ` and `PAIRDZ` which evaluate Equations 2.43 and 2.44.

It will be observed that the pair production formulas are symmetric about $\mathcal{E} = 1/2$. One of the electrons will be given energy $\breve{k}\mathcal{E}$ and the other $\breve{k}(1-\mathcal{E})$. The choice of which one is a positron is made randomly. Since we need to know which particle has the lower energy so we can put it on the top stack position, we restrict the range of of $\mathcal{E}$ to $(0,1/2)$ and double $f_1(\mathcal{E})$ and $f_2(\mathcal{E})$, thus guaranteeing that any sampled value of $\mathcal{E}$ will correspond to the electron with the lower energy. We thus sample $f_1(\mathcal{E})$ by letting $\mathcal{E} = 1/2\zeta$, and we sample $f_2(\mathcal{E})$ by letting

$$
\mathcal{E} = \frac{1}{2} \left(1 - \max \left(\zeta_1,\ \zeta_2,\ \zeta_3 \right) \right) \tag{2.146}
$$

where the ζ values are drawn uniformly on the interval $(0,1)$ (see Section 2.2).

A special approximation which has been carried over from previous versions is that if the incident photon has energy less than 2.1 MeV, one of the electrons is made to be at rest and the other given the rest of the available energy. One reason for making this approximation is that the pair sampling routine becomes progressively more inefficient as the pair production threshold is approached. Perhaps a better approximation would be to pick the energy of the low energy electron uniformly from the interval $(m, \breve{k}/2)$.

We now conclude this section on bremsstrahlung and pair production with a few general remarks. We first note that for $\delta' \leq 1$, that $A(\delta')$, $B(\delta')$ and $C(\delta')$ are all quadratic functions of δ'. The three coefficients depend on whether the incident energy is above or below 50 MeV. For $\delta' > 1$, $A(\delta')$, $B(\delta')$ and $C(\delta')$ are equal and are given by

$$
A,\ B,\ C(\delta') = \frac{\phi_1(\delta') + 4\left(Z_V\ \text{if}\ \breve{E}_0,\ \breve{k} > 50, Z_G \right)}{\phi_1(0) + 4\left(Z_V\ \text{if}\ \breve{E}_0,\ \breve{k} > 50, Z_G \right)} \tag{2.147}
$$

which will have the form $c_1 + c_2\ \ell n(\delta' + c_3)$. The $A,\ B,\ C$ must not be allowed to go negative. PEGS computes a maximum allowed Δ_E above which the $A,\ B,\ C$ are considered to be zero.

The various parameters needed to sample the secondaries' energies for the bremsstrahlung and pair production interactions are computed in the PEGS routine `DIFFER`. The parameters are stored for each medium during execution of EGS in `COMMON/BREMPR/`.

One other use for the bremsstrahlung cross section is for computing the mean energy loss per unit length of an electron due to emission of soft photons (i.e., those with energy below the photon cutoff energy A_p). This is given by

$$-\left(\frac{d\breve{E}}{dx}\right)_{Soft\ Brem} = \int_0^{A_P} \breve{k}\left(\frac{d\breve{\Sigma}_{Brem}}{d\breve{k}}\right)\ d\breve{k} \tag{2.148}$$

which will be discussed in Section 2.13.

2.7.1 Bremsstrahlung Photon Angular Distribution

So far we have only discussed the selection of the energy of the secondaries. Since these are interactions with three body final states, the polar angles of the secondary particles are not uniquely determined by the secondary energies, and a complete simulation would sample from some appropriate distributions. However the angles at which products from these reactions are usually emitted are small compared to angular deviations resulting from multiple scattering. Previous versions of EGS therefore assumed that the direction of an electron emitting bremsstrahlung is unchanged, that a bremsstrahlung photon is emitted at an angle relative to the incident electron direction, $\theta = m/\breve{E}_0$, and that pair produced particles have production angles relative to the incident photon direction given by $\theta = m/\breve{k}$. The azimuthal angle for the first product particle is chosen randomly and the other product particle is given the opposite azimuth.

The above model of the angular distribution of newly created bremsstrahlung photons may be overly simple for some applications. The angle given by $m/\breve{k}$ is a good estimate of the expected *average* scattering angle, and at high energies, where the distribution is strongly peaked in the forward direction, more accurate modeling of scattering angles does not significantly improve the accuracy of shower simulations. Additionally, at low energies, particularly in thick targets, the effect of multiple scattering of the initiating electrons greatly overwhelms the impact of photon angular distributions in defining the development of the shower, and the extra effort and computing time necessary to implement bremsstrahlung angular distribution sample is not worthwhile.

While it was recognized that the above argument may break down for applications involving thin target bremsstrahlung spectra, it was discovered that the above assumption about multiple scattering dominance does not apply *even for thick targets at low energies* (10 MeV or so) for narrow beams such as those employed in some medical linacs for producing photon beams for radiotherapy[54]. Thus, options for more accurate modeling of bremsstrahlung scattering angles in EGS have been developed.

Angular distribution formulas

The formula employed for the angular sampling routine is 2BS of Koch and Motz[91], which is the cross section, differential in photon energy and angle,

$$d\sigma_{k,\Theta} = \frac{4Z^2 r_0^2}{137} \frac{dk}{k} y dy \left\{ \frac{16y^2 E}{(y^2+1)^4 E_0} - \frac{(E_0+E)^2}{(y^2+1)^2 E_0^2} + \left[\frac{E_0^2 + E^2}{(y^2+1)^2 E_0^2} - \frac{4y^2 E}{(y^2+1)^4 E_0} \right] \ln M(y) \right\},$$

(2.149)

where,

$$y = E_0\Theta; \quad \frac{1}{M(y)} = \left(\frac{k}{2E_0 E} \right)^2 + \left(\frac{Z^{1/3}}{111(y^2+1)} \right)^2,$$

and, the following definitions for the variables apply:

k energy of the photon in units of $m_e c^2$
Θ angle between the outgoing photon and the incoming electron direction (in radians)
Z atomic number of the target material
r_0 $\equiv e^2/m_e c^2$ (classical electron radius)
E_0, E initial and final electron energy in units of $m_e c^2$

The following table, copied from the Koch and Motz article, outlines the essential approximations employed in the derivation of Equation 2.149.

	Approximation	Condition of validity
i)	Approximate screening potential	$(Ze/r)e^{-r/a}$
ii)	Born approximation	$(2\pi Z/137\beta_0)$, $(2\pi Z/137\beta) \ll 1$
iii)	Extreme relativistic	$E_0, E, k \gg 1$
iv)	Small angles	$\sin\Theta = \Theta$
v)	Approximate e^- angular integration	$\Theta < (Z^{1/3}/111E_0)$

It should be noted that only the angular distribution part of Equation 2.149 is employed. The cross section differential in photon energy employed by the EGS5 code is far less restrictive (Approximation (iii) plus Thomas-Fermi screening factors). For the purposes of modeling electron linacs, the ultimate test of these approximations is comparison with experiment. In this regard, Koch and Motz present encouraging data (their Figure 17) which exhibits excellent agreement between experiment and Equation 2.149 for 4.54 MeV electrons on Au. Although use of Equation 2.149 violates constraints ii), iii) and iv) in the cases they showed, the deviation was at worst 10% (at large angles) and usually much better. In particular, violating the Born approximation constraints seemed not especially deleterious to the comparison. The conditions of this experiment are similar to those

used for medical linacs (6–50 MeV, high-Z targets) and therefore the employment of Equation 2.149 seems justified. At lower energies, use of Equation 2.149 still needs to be demonstrated. At higher energies, the constraints are not so badly violated except for the Born approximation when high-Z materials are used. Again, experimental data will judge the suitability of Equation 2.149 in this context.

Sampling procedure

To sample the photon angular distribution, a mixed sampling procedure is employed. Since it is the angular distribution that is required, the overall normalization of Equation 2.149 is unimportant including any overall energy-dependent factors. The following expression for $p(y)$ is proportional to Equation 2.149:

$$p(y)\mathrm{d}y = f(y^2)N_\mathrm{r}g(y^2)\mathrm{d}y^2. \tag{2.150}$$

Defining $x = y^2$,

$$f(x) = \frac{1 + 1/(\pi E_0)^2}{(x+1)^2}, \tag{2.151}$$

and,

$$g(x) = 3(1+r^2) - 2r - [4 + \ln m(x)][(1+r^2) - 4xr/(x+1)^2], \tag{2.152}$$

where,

$$r = E/E_0; \ m(x) = \left(\frac{1-r}{2E_0 r}\right)^2 + \left(\frac{Z^{1/3}}{111(x+1)}\right)^2.$$

Note that $1/E_0$(high frequency limit)$\leq r \leq 1$(low frequency limit). N_r is a normalization constant which will be discussed later.

The function $f(x)$ will be used for direct sampling. It can be easily verified that this function is normalized correctly, (*i.e.*, $\int_0^{(\pi E_0)^2} f(x)\mathrm{d}x = 1$) and the candidate scattering angle is easily found by inversion to be:

$$\hat{\Theta} = \frac{1}{E_0}\sqrt{\frac{\zeta}{1 - \zeta + 1/(\pi E_0)^2}}, \tag{2.153}$$

where ζ is a random number selected uniformly on the range (0,1) and the "hat" over Θ signifies that it is a quantity determined by random selection.

The function g is sampled using the rejection technique. In order to employ this technique, the optimum case is to have the location of the maximum of the function, x_max characterized allowing the most efficient determination $N_\mathrm{r} = g(x_\mathrm{max})^{-1}$. Failing this, the next best scenario is to overestimate $g(x_\mathrm{max})$. The closer this estimate is to the true maximum value, the more efficient the rejection technique will be. Unfortunately, characterizing g in complete generality proved to be very difficult. The following observations were made, however. The maximum value of $g(x)$ occurs

at either $x = 0$, $x = (\pi E_0)^2$ (*i.e.*, at the minimum or maximum values of x), or in the vicinity of $x = 1$. Therefore, the rejection function normalization was chosen to be:

$$N_{\mathrm{r}} = \{\max[g(0), g(1), g((\pi E_0)^2)]\}^{-1}. \tag{2.154}$$

A more complete discussion of bremsstrahlung angular distributions as adapted to EGS, may be found documented elsewhere[29].

2.7.2 Pair Angle Sampling

In previous versions of EGS, both particles in all newly created $e^- e^+$ pairs were set in motion at fixed angles $\Theta_\pm$ with respect to the initiating photon direction. $\Theta_\pm$, the scattering angle of the e^+ or e^- (in radians), is of the form $\Theta_\pm = 1/k$ where k is the energy of the initiating photon in units of $m_{\mathrm{o}} c^2$, the rest mass of the electron. Defined in this way, $\Theta_\pm$ provides an estimate of the expected *average* scattering angle[6].

The motivation for employing such a crude approximation is as follows: At high energies the distribution is so strongly peaked in the forward direction that more accurate modeling will not significantly improve the shower development. At low energies, particularly in thick targets, multiple scattering of the resultant pair as the particles slow will "wash out" any discernible distribution in the initial scattering angle. Therefore, the extra effort and computing time necessary to implement pair angular distributions was not considered worthwhile. It was recognized, however, that the above argument would break down for applications where the $e^+ e^-$ pair may be measured before having a chance to multiple scatter sufficiently and obliterate the initial distribution, and this was indeed found to be the case.

To address this shortcoming, two new options for sampling the pair angle were introduced, as described in the two following subsections. Procedures for sampling these formulas are given in the next sections. The formulas employed in this report were taken from the compilation by Motz, Olsen and Koch[111].

Leading order approximate distribution

As a first approximation, the leading order multiplicative term of the Sauter-Gluckstern-Hull formula (Equation 3D-2000 of Motz *et al.*[111]) was used:

$$\frac{\mathrm{d}P}{\mathrm{d}\Theta_\pm} = \frac{\sin\Theta_\pm}{2 p_\pm (E_\pm - p_\pm \cos\Theta_\pm)^2}, \tag{2.155}$$

---

[6]The extremely high-energy form of the leading order approximation discussed later implies that the distribution should peak at $\Theta_\pm = 1/(\sqrt{3} E_\pm)$. However, the Bethe-Heitler cross section used in EGS5 peaks at $E_\pm = k/2$ and the approximation $\Theta_\pm = 1/k$ is a reasonable one on average, given the highly approximate nature of the angular modeling.

where $\Theta_\pm$ is the $e^\pm$ scattering angle (in radians), $E_\pm$ and $p_\pm$ are the $e^\pm$ total energy and momentum in units of the electron rest-mass energy, $m_\mathrm{o}c^2$.

The Sauter-Gluckstern-Hull formula, which is used in ETRAN-based codes[149], was derived under the following approximations:

	Approximation	Condition of validity
i)	No screening	Low-Z elements
ii)	First order Born approximation	$(2\pi Z/137\beta_\pm) \ll 1$
iii)	Negligible nuclear recoil	$k \gg (1/m_\mathrm{n}), k \ll m_\mathrm{n}$ (large angles)

where m_n is the rest mass energy of the nucleus in units of $m_\mathrm{o}c^2$ and $\beta_\pm$ is the magnitude of the velocity of the $e^\pm$ in units of the speed of light, c.

The full Sauter-Gluckstern-Hull formula differs from the above by a modulating factor which varies between 0 and approximately 2 and thus it is to be regarded as a crude approximation even in its region of validity.

The Schiff distribution

This formula employed for the angular sampling routine is Equation 3D-2003 of Motz *et al.*[111], which is the cross section, differential in photon energy and angle,

$$\frac{\mathrm{d}\sigma^2}{\mathrm{d}E_\pm \mathrm{d}\Omega_\pm} = \frac{2\alpha Z^2 r_0^2}{\pi} \frac{E_\pm^2}{k^3} \left\{ -\frac{(E_+ - E_-)^2}{(u^2 + 1)^2} - \frac{16u^2 E_+ E_-}{(u^2 + 1)^4} + \left[\frac{E_+^2 + E_-^2}{(u^2 + 1)^2} + \frac{4u^2 E_+ E_-}{(u^2 + 1)^4} \right] \ln M(y) \right\},$$
(2.156)

where,

$$u = E_\pm \Theta_\pm \ ; \ \frac{1}{M(y)} = \left(\frac{k}{2E_+ E_-} \right)^2 + \left(\frac{Z^{1/3}}{111(u^2 + 1)} \right)^2,$$

and, the following definitions for the variables apply:

k energy of the photon in units of $m_\mathrm{o}c^2$
E_+, E_- final $e^\pm$ total energy in units of $m_\mathrm{o}c^2$ ($k = E_+ + E_-$)
$\Theta_\pm$ angle between the outgoing $e^\pm$ and the incoming photon direction (in radians)
$\mathrm{d}\Omega_\pm$ Differential solid angle of the outgoing $e^\pm$
Z atomic number of the target material
r_0 $\equiv e^2/m_\mathrm{o}c^2$ (classical electron radius)
α $\equiv e^2/(\hbar c) = 1/137...$ (fine structure constant)

The following table, derived from the Motz *et al.* article, outlines the essential approximations involved in the development of Equation 2.156.

	Approximation	Condition of validity
i)	Approximate screening potential	$(Ze/r)e^{-r/a}$
ii)	First order Born approximation	$(2\pi Z/137\beta_\pm) \ll 1$
iii)	Extreme relativistic	$E_\pm, k \gg 1$
iv)	Small angles	$\Theta_\pm = O(E_\pm)$
v)	Negligible nuclear recoil	$k \gg (1/m_n), k \ll m_n$ (large angles)

It should be noted that only the angular distribution part of Equation 2.156 is employed. The cross section differential in electron energy employed by the EGS code is far more widely applicable, as it was derived under just Approximation (iii) above with Thomas-Fermi screening factors.

Sampling procedure

The distribution expressed by Equation 2.155 is normalized as

$$\int_0^\pi d\Theta_\pm \left(\frac{dP}{d\Theta_\pm} \right) = 1,$$

and may be sampled from the formulas

$$\sin \Theta_\pm = \frac{2\sqrt{\zeta(1-\zeta)}}{p_\pm(2\zeta - 1) + E_\pm} \; ; \; \cos \Theta_\pm = \frac{E_\pm(2\zeta - 1) + p_\pm}{p_\pm(2\zeta - 1) + E_\pm}, \tag{2.157}$$

where ζ is a random number uniform on the range $[0, 1]$. The $\sin \Theta_\pm$ and $\cos \Theta_\pm$ forms are related by trigonometric identities but both forms expressed by Equation 2.157 are useful.

To sample the angular distribution expressed by Equation 2.156, a rejection technique is applied following a change of variables. Consider the change of variables:

$$\xi = \frac{1}{u^2 + 1} \; ; \; \frac{1}{1 + E_\pm^2 \pi^2} \leq \xi \leq 1.$$

Then, applying the small angle approximation, $d\Omega_\pm \longrightarrow 2\pi\Theta_\pm d\Theta_\pm$, the angular distribution rejection function, g, has the form:

$$\frac{dg}{d\xi} = N_g\{2 + 3(r + r^{-1}) - 4[(r + r^{-1} + 4\xi - 4\xi^2][1 + \ln m(\xi)/4]\}, \tag{2.158}$$

where

$$m(\xi) = \left(\frac{(1+r)(1+r^{-1})}{2k} \right)^2 + \left(\frac{Z^{1/3}\xi}{111} \right)^2 \; ; \; r = E_-/E_+,$$

59

and N_g is a normalization factor that should be chosen so that

$$g(\xi) \leq 1 \ \forall \ \xi \in \left[\frac{1}{1 + E_{\pm}^2 \pi^2}, 1 \right].$$

The position of the maximum of the function g proved difficult to characterize accurately and so a two-step iterative scheme was developed based upon the slow variation of the logarithmic term $[1 + \ln m(\xi)/4]$ in Equation 2.158 and the observation that the position of the maximum is relatively independent of the value of r. For the purposes of estimating the maximum value of g, r is set equal to 1, and a satisfactory algorithm for calculating N_g is

$$N_g = 1.02 \max \left[g \left(\frac{1}{1 + E_{\pm}^2 \pi^2} \right), g(\xi_{\text{max}}^{(1)}) \right], \tag{2.159}$$

where $\xi_{\text{max}}^{(1)}$ is an estimate of the position of the maximum of g after a one-step iteration. The zeroth-order estimate is:

$$\xi_{\text{max}}^{(0)} = \max \left\{ 0.01, \max \left[\frac{1}{1 + E_{\pm}^2 \pi^2}, \min \left(0.5, \frac{222}{k^2 Z^{1/3}} \right) \right] \right\},$$

and the second iteration yields:

$$\xi_{\text{max}}^{(1)} = \max \left\{ 0.01, \max \left[\frac{1}{1 + E_{\pm}^2 \pi^2}, \min \left(0.5, \frac{1}{2} - \frac{\alpha'}{3\beta'} + \text{sgn}(\alpha') \sqrt{\left(\frac{\alpha'}{3\beta'} \right)^2 + \frac{1}{4}} \right) \right] \right\}$$

where

$$\alpha' = 1 + \ln m(\xi_{\text{max}}^{(0)})/4 - \beta'(\xi_{\text{max}}^{(0)} - 1/2),$$

$$\beta' = \frac{\xi_{\text{max}}^{(0)}(Z^{1/3}/111)^2}{2m(\xi_{\text{max}}^{(0)})}.$$

and

$$\text{sgn}(\alpha') = \left\{ \begin{array}{ll} +1 & \text{if } \alpha' \geq 0 \\ -1 & \text{if } \alpha' < 0 \end{array} \right.$$

The extra 1.02 in Equation 2.159 is a "safety factor".

The Schiff threshold

The Schiff distribution breaks down mathematically for $E_\gamma < 4.14$ MeV. To prevent non-physical modeling, if a user has requested the Schiff distribution, the lowest order approximate distribution of Equation 2.155 is used when the photon energy is less than the 4.14 MeV threshold.

A more complete discussion of pair angular distributions as adapted to the EGS code, may be found documented elsewhere[26].
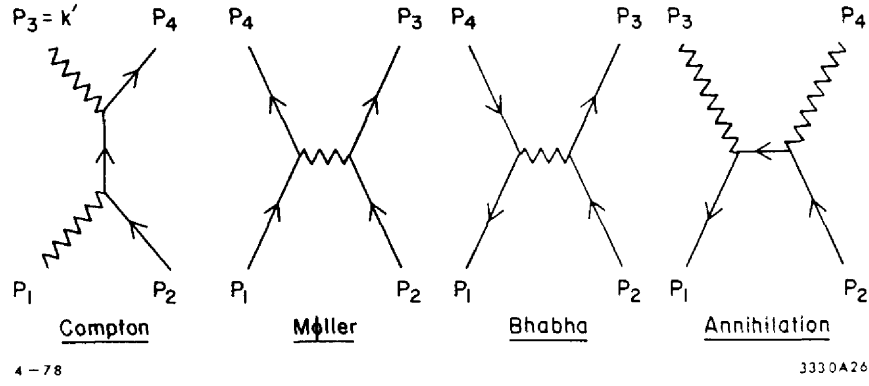
Figure 2.3: Feynman diagrams for two body interactions with electrons.
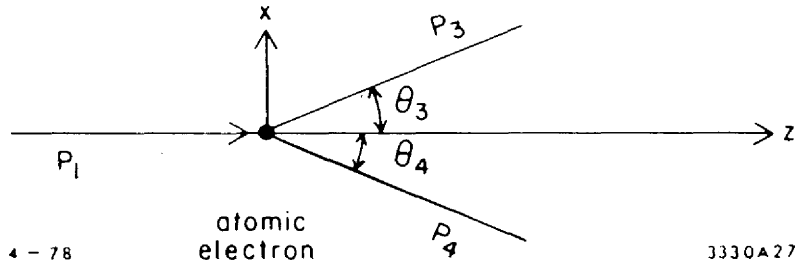
Figure 2.4: Definition of two-body scattering angles.

2.8   Interactions With Atomic Electrons – General Discussion

In this section we consider some general aspects of two-body interactions with atomic electrons. The details of two-body interactions involving specific sets of particles will be examined in subsequent sections. The reactions we will investigate, Compton scattering, Møller scattering, and Bhabha scattering, and two photon positron-electron annihilation, are illustrated in the Feynman diagrams in Figure 2.3.

The kinematics of these interactions are described by the four-vector equation

$$P_1 + P_2 = P_3 + P_4 \ , \tag{2.160}$$

where we employ the convention that P_1 is the incident particle, P_2 is the atomic electron (assumed free and at rest), P_3 is the particle whose energy will be calculated, and P_4 is the other final state particle. Since such collisions are two body reactions, they take place in planes, with scattering angles illustrated in Figure 2.4 , where we take the incident particle direction to be the z axis and the interaction occurs in the $x - z$ plane.

Letting m_i, E_i, p_i denote the mass, energy, and three-momentum of particle i, the four-momenta

61

are then given by

$$P_1 = (E_1, 0, 0, p_1) \tag{2.161}$$
$$P_2 = (m, 0, 0, 0) \tag{2.162}$$
$$P_3 = (E_3, p_3 \sin\theta_3, 0, p_3 \cos\theta_3) \tag{2.163}$$
$$P_4 = (E_4, -p_4 \sin\theta_4, 0, p_4 \cos\theta_4) \ . \tag{2.164}$$

If now we want to find the scattering angle θ_3, assuming we have determined all energy and momenta, we solve Equation 2.160 for P_4 and take its invariant square to get

$$P_4^2 = P_1^2 + P_2^2 + P_3^2 - 2(P_1 + P_2) \cdot P_3 + 2P_1 \cdot P_2 \ . \tag{2.165}$$

Making use of the relation $P_i^2 = m_i^2$ and Equations 2.161 and 2.164, we obtain

$$m_4^2 = m_1^2 + m^2 + m_3^2 + 2[E_1 m - (E_1 + m)E_3 + p_1 p_3 \ \cos\theta_3] \ , \tag{2.166}$$

and for $\cos\theta_3$ we arrive at

$$\cos\theta_3 = \frac{m_4^2 - m_1^2 - m^2 - m_3^2 + 2(E_1 + m)E_3 - 2E_1 m}{2p_1 p_3} \ . \tag{2.167}$$

Clearly, by symmetry, the above equation is still true if we interchange the indices 3 and 4 to obtain a relation for $\cos\theta_4$. Thus, we see that the scattering angles are uniquely determined by the final energies. Azimuthal angles are uniformly distributed, provided, of course, that the two particles have the opposite azimuth.

In the sections that follow we shall focus on the computation of the differential and total cross sections for these four processes, and we will provide the sampling methods used to determine the secondary energies. We shall also reduce Equation 2.167 to the specific reactions and derive the expressions used in EGS to determine the cosines of the scattering angles. In most cases, we get the sines of the scattering angles using the formulas

$$\sin\theta_3 = \sqrt{1 - \cos^2\theta_3} \tag{2.168}$$
$$\sin\theta_4 = -\sqrt{1 - \cos^2\theta_4} \ . \tag{2.169}$$

The reason for making $\sin\theta_4$ negative is that this effectively achieves the opposite azimuth within the frame work of the EGS routine UPHI, as discussed later. Note that we drop the subscripts on the scattering angles for simplicity in the discussions below.

2.9 Compton Scattering

The differential and total Compton scattering cross sections are given by formulas originally due to Klein and Nishina [90]:

$$\frac{d\breve{\Sigma}_{Compt}(\breve{k}_0)}{d\breve{k}} = \frac{X_0 n \ \pi r_0^2 m}{\breve{k}_0^2} \left[\left(\frac{C_1}{\mathcal{E}} + C_2 \right) \Big/ \mathcal{E} + C_3 + \mathcal{E} \right] \tag{2.170}$$

where

$$
\begin{aligned}
X_0 &= \quad \text{radiation length (cm),} \\
n &= \quad \text{electron density (electron/cm}^3\text{),} \\
r_0 &= \quad \text{classical electron radius (cm}^2\text{),} \\
m &= \quad \text{electron rest energy (MeV),} \\
\breve{k}_0 &= \quad \text{incident photon energy (MeV),} \\
\breve{k} &= \quad \text{scattered photon energy (MeV),} \\
\mathcal{E} &= \quad \breve{k}/\breve{k}_0, \\
C_1 &= \quad (k_0')^{-2}, \\
k_0' &= \quad \breve{k}_0/m, \\
C_2 &= \quad 1 - 2(1 + k_0')/(k_0')^2, \\
C_3 &= \quad (1 + 2\,k_0')/(k_0')^2.
\end{aligned}
$$

The Compton cross section integrated over the energy range from $\breve{k}_1$ to $\breve{k}_2$ can be expressed as

$$
\int_{\breve{k}_1}^{\breve{k}_2} \frac{d\breve{\Sigma}_{Compt}(\breve{k}_0)}{d\breve{k}} \, d\breve{k} = \frac{X_0 n \pi r_0^2}{k_0'} \left[C_1 \left(\frac{1}{\mathcal{E}_1} - \frac{1}{\mathcal{E}_2} \right) + \right.
$$
$$
\left. C_2 \, \ell n \, \frac{\mathcal{E}_2}{\mathcal{E}_1} + \mathcal{E}_2(C_3 + \mathcal{E}_2/2) - \mathcal{E}_1(C_3 + \mathcal{E}_1/2) \right] \tag{2.171}
$$

where

$$
\begin{aligned}
\mathcal{E}_1 &= \breve{k}_1/\breve{k}_0 \ , \\
\mathcal{E}_2 &= \breve{k}_2/\breve{k}_0 \ .
\end{aligned}
$$

The total scattering cross section is obtained from Equation 2.171 with $\breve{k}_1$ and $\breve{k}_2$ set to the minimum and maximum possible scattered photon energies. To see what these are, we use Equation 2.167, noting that $m_1 = m_3 = 0$, $m_4 = m$, $E_1 = p_1 = \breve{k}_0$, $E_3 = p_3 = \breve{k}$, $E_4 = \breve{E}$, and $p_4 = \breve{p}$, and arrive at

$$
\cos \theta = \frac{(\breve{k}_0 + m)\breve{k} - \breve{k}_0 m}{\breve{k}_0 \breve{k}} \ . \tag{2.172}
$$

Solving for $\breve{k}$ we get the well-known formula

$$
\breve{k} = \frac{\breve{k}_0}{1 + (1 - \cos \theta)\breve{k}_0/m} \ . \tag{2.173}
$$

The maximum and minimum values of $\breve{k}$ occur at $\cos \theta = 1$, -1, or

$$
\breve{k}_{max} = \breve{k}_0 \ , \tag{2.174}
$$

$$
\breve{k}_{min} = \frac{\breve{k}_0}{1 + 2\breve{k}_0/m} \ . \tag{2.175}
$$

63

Thus

$$\check{\Sigma}_{Compt,\ Total}\ (\check{k}_0) = \left(\text{Equation 2.171 with } \check{k}_2 = \check{k}_0,\ \check{k}_1 = \frac{\check{k}_0}{1 + 2\check{k}_0/m} \right) . \qquad (2.176)$$

PEGS functions `COMPDM`, `COMPRM`, and `COMPTM` evaluate Equations 2.170, 2.171, and 2.176, respectively. Note that in the discussion above, the atomic electrons are assumed to be unbound and at rest. In section 2.18 we will examine the effects of of this approximation.

We next consider techniques for sampling the energy of the scattered gamma ray. If we define the variable we wish to sample as

$$\mathcal{E} = \check{k}/\check{k}_0 , \qquad (2.177)$$

we can see from Equations 2.174 and 2.175 that $\mathcal{E}$ must be in the interval $(\mathcal{E}_0, 1)$, where

$$\mathcal{E}_0 = \frac{1}{1 + 2\ \check{k}_0/m} . \qquad (2.178)$$

We start with a form of the differential cross section similar to that given by Butcher and Messel[39]:

$$\frac{d\check{\Sigma}_{Compt}}{d\mathcal{E}} = \frac{X_0 n\pi r_0^2 m}{\check{k}_0} \left[\frac{1}{\mathcal{E}} + \mathcal{E} \right] \left[1 - \frac{\mathcal{E} \sin^2 \theta}{1 + \mathcal{E}^2} \right] \propto f(\mathcal{E})g(\mathcal{E}) . \qquad (2.179)$$

We will sample $f(\mathcal{E}) = \frac{1}{\mathcal{E}} + \mathcal{E}$ over $(\mathcal{E}_0, 1)$ and use $g(\mathcal{E}) = \left[1 - \frac{\mathcal{E}\ \sin^2 \theta}{1 + \mathcal{E}^2} \right]$ as a rejection function. We factorize $\left[\frac{1}{\mathcal{E}} + \mathcal{E} \right]$ over $(\mathcal{E}_0, 1)$ as follows

$$f(\mathcal{E}) = \frac{1}{\mathcal{E}} + \mathcal{E} = \sum_{i=1}^{2} \alpha_i f_i(\mathcal{E}) \qquad (2.180)$$

where

$$\alpha_1 = \ell n(1/\mathcal{E}_0),\ f_1(\mathcal{E}) = \frac{1}{\ell n(1/\mathcal{E}_0)} \left(\frac{1}{\mathcal{E}} \right),\ \mathcal{E} \in (\mathcal{E}_0, 1) \qquad (2.181)$$

$$\alpha_2 = (1 - \mathcal{E}_0^2)/2,\ f_2(\mathcal{E}) = \frac{2\mathcal{E}}{(1 - \mathcal{E}_0^2)},\ \mathcal{E} \in (\mathcal{E}_0, 1). \qquad (2.182)$$

We sample f_1 by letting

$$\mathcal{E} = \mathcal{E}_0\ e^{\alpha_1 \zeta} \qquad (2.183)$$

where ζ is a random number drawn uniformly on the interval $(0, 1)$. We could sample f_2 by taking the larger of two random numbers if we were willing to reject sampled values less than $\mathcal{E}_0$. but this would get very inefficient for low energy photons. Instead we make a change of variables. Let

$$\mathcal{E}' = \frac{\mathcal{E} - \mathcal{E}_0}{1 - \mathcal{E}_0} . \qquad (2.184)$$

Then in order to give $\mathcal{E}$ the proper distribution, $\mathcal{E}'$ must have the distribution

$$f_2'(\mathcal{E}') = f_2(\mathcal{E}) \frac{d\mathcal{E}}{d\mathcal{E}'} = \alpha_1' f_1''(\mathcal{E}') + \alpha_2' f_2''(\mathcal{E}') \qquad (2.185)$$

64

where

$$\alpha_1' = \frac{k_0'}{k_0' + 1} \ , f_1''(\mathcal{E}') = 2\mathcal{E}', \ \mathcal{E}' \in (0, 1) \tag{2.186}$$

$$\alpha_2' = \frac{1}{k_0' + 1} \ , f_2''(\mathcal{E}') = 1, \ \mathcal{E}' \in (0, 1) \ . \tag{2.187}$$

Both of these sub-distributions are easily sampled.

To compute the rejection function it is necessary to get $\sin^2 \theta$. Let

$$t = \frac{m(1 - \mathcal{E})}{\breve{k}_0 \mathcal{E}} \ . \tag{2.188}$$

Then using Equation 2.172, we have

$$\cos \theta = \frac{(\breve{k}_0 + m)\breve{k} - \breve{k}_0 m}{\breve{k}_0 \breve{k}} = 1 + \frac{m\mathcal{E} - m}{\breve{k}_0 \mathcal{E}} = 1 - t \ . \tag{2.189}$$

Thus

$$\sin^2 \theta = 1 - \cos^2 \theta = (1 - \cos \theta)(1 + \cos \theta) = t(2 - t) \ . \tag{2.190}$$

When the value of $\mathcal{E}$ is accepted, then $\sin \theta$ and $\cos \theta$ are obtained via

$$\sin \theta = \sqrt{\sin^2 \theta} \tag{2.191}$$

$$\cos \theta = 1 - t \ . \tag{2.192}$$

The sampling procedure is as follows:

1. Compute the parameters depending on $\breve{k}_0$, but not $\mathcal{E}$: k_0', $\mathcal{E}_0$, α_1, and α_2.

2. Sample $\mathcal{E}$ in the following way: If $\alpha_1 \geq (\alpha_1 + \alpha_2)\zeta_1$, use $\mathcal{E} = \mathcal{E}_0 \ e^{\alpha_1 \zeta_2}$. Otherwise, use $\mathcal{E} = \mathcal{E}_0 + (1 - \mathcal{E}_0)\mathcal{E}'$, where $\mathcal{E}'$ is determined from

$$\mathcal{E}' = \max \ (\zeta_3, \zeta_4) \text{ if } k_0' \geq (k_0' + 1)\zeta_2$$

   or from

$$\mathcal{E}' = \ \zeta_3 \text{ otherwise.}$$

3. Calculate t and the rejection function $g(\mathcal{E})$. If ζ_4(or ζ_5) $< g(\mathcal{E})$, reject and return to Step 2.

After determining the secondary energies, Equation 2.167 is used to obtain the scattering angles and UPHI is called to select random azimuth and to set up the secondary particles in the usual way. Note again that a more detailed treatment of Compton scattering is provided in section 2.18.

2.10 Møller Scattering

The form of the cross sections and the sampling methods that we use for modeling Møller and Bhabha scattering follow those given by Messel and Crawford[103] (*e.g.*, see p.13-14 therein) except that various misprints have been corrected.

The differential Møller[109] cross section is given by

$$\frac{d\breve{\Sigma}_{M\o ller}(\breve{E}_0)}{d\breve{E}} = \frac{X_0 n 2\pi r_0^2 m}{\beta^2 \breve{T}_0^2} \left[C_1 + \frac{1}{\mathcal{E}} \left(\frac{1}{\mathcal{E}} - C_2 \right) + \frac{1}{\mathcal{E}'} \left(\frac{1}{\mathcal{E}'} - C_2 \right) \right] \qquad (2.193)$$

where

$$
\begin{aligned}
\breve{E}_0 &= \text{ incident electron energy (MeV)}, \\
\breve{T}_0 &= \breve{E}_0 - m = \text{ incident kinetic energy (MeV)}, \\
\breve{E} &= \text{ energy of scattered electron (MeV)}, \\
\breve{T} &= \text{ kinetic energy of scattered electron (MeV)}, \\
\mathcal{E} &= \breve{T}/\breve{T}_0 = \text{ fraction of kinetic energy to scattered electron}, \\
\mathcal{E}' &= 1 - \mathcal{E} = \text{ fraction of kinetic energy remaining}, \\
\gamma &= \breve{E}_0/m, \\
C_1 &= [(\gamma - 1)/\gamma]^2, \\
C_2 &= (2\gamma - 1)/\gamma^2, \\
beta^2 &= 1 - 1/\gamma^2 = (v/c)^2,
\end{aligned}
$$

and where the other terms have been defined previously.

Because of the ambiguity in the identity of the initial and final electrons, the cross section is symmetric with respect to the interchange of $\mathcal{E}$ with $\mathcal{E}'$. Another consequence of this is that $\mathcal{E}$ is restricted to lie in the interval $(0, 1/2)$. It can be seen that Equation 2.193 is singular at $\mathcal{E} = 0$ (also at $\mathcal{E}' = 0$ but the range of $\mathcal{E}'$ is now restricted to $(1/2, 1)$), and the total Møller cross section is infinite. We get around this by considering as discrete collisions only Møller scattering interactions for which the scattered electron acquire at least some threshold energy, A_E (we also define the threshold kinetic energy of $T_E = A_E - m$). Since the incoming electron energy is $\breve{E}_0 + m$ and the minimum final energy is $2A_E$, we see that the threshold initial electron energy at which a discrete Møller scattering event can take place is given by

$$\breve{E}_{Th}^{M\o ller} = 2A_E - m = 2T_E + m . \qquad (2.194)$$

Møller scattering interactions in which secondary electrons (δ–rays) are ejected with energies than less A_E are treated as part of the continuous energy loss process (see Section 2.13).

The integral of the Møller cross section over some energy range can be expressed as

$$\int_{\breve{E}_1}^{\breve{E}_2} \frac{d\breve{\Sigma}_{M\o ller}(\breve{E}_0)}{d\breve{E}} \, d\breve{E} = \frac{X_0 n 2\pi r_0^2 m}{\beta^2 \breve{T}_0} \left[C_1 (\mathcal{E}_2 - \mathcal{E}_1) + \frac{1}{\mathcal{E}_1} - \frac{1}{\mathcal{E}_2} \right.$$

66

$$+ \quad \frac{1}{\mathcal{E}_2'} - \frac{1}{\mathcal{E}_1'} - C_2 \ell n \, \frac{\mathcal{E}_2 \mathcal{E}_1'}{\mathcal{E}_1 \mathcal{E}_2'} \Bigg] \tag{2.195}$$

where

$$\mathcal{E}_i = (\breve{E}_i - m)/\breve{T}_0, \quad i = 1, 2 , \tag{2.196}$$

$$\mathcal{E}_i' = 1 - \mathcal{E}_i, \quad i = 1, 2 , \tag{2.197}$$

and other symbols are the same as in Equation 2.193.

The minimum and maximum energies for the scattered electron (by convention, the scattered electron is taken to be the one which emerges with the lower energy) are A_E and $\frac{\breve{T}_0}{2} + m$, respectively. When these limits are used for $\breve{E}_1$ and $\breve{E}_2$ in Equation 2.195, we obtain for the total discrete Møller cross section (which we assume to non-zero only for $\breve{E}_0 > \breve{E}_{Th}^{Møller}$):

$$\breve{\Sigma}_{Møller, \; Total}(\breve{E}_0) == \text{Equation 2.195 with } \breve{E}_1 = A_E \text{ and } \breve{E}_2 = (\breve{T}_0/2) + m . \tag{2.198}$$

PEGS functions `AMOLDM`, `AMOLRM`, and `AMOLTM` evaluate Equations 2.193, 2.195, and 2.198, respectively.

In sampling for the resultant energy, we use the variable $\mathcal{E} = \breve{T}/\breve{T}_0$ and obtain

$$\frac{d\breve{\Sigma}_{Møller}(\breve{E}_0)}{d\mathcal{E}} = \frac{X_0 n 2\pi r_0^2 m}{\breve{T}_0 \mathcal{E}_0} \, f(\mathcal{E}) g(\mathcal{E}) \tag{2.199}$$

where

$$f(\mathcal{E}) = \frac{\mathcal{E}_0}{1 - 2\mathcal{E}_0} \frac{1}{\mathcal{E}^2}, \qquad \mathcal{E} \in (\mathcal{E}_0, 1/2), \tag{2.200}$$

$$g(\mathcal{E}) = g_1[1 + g_2 \mathcal{E}^2 + r(r - g_3)], \tag{2.201}$$

$$\mathcal{E}_0 = T_E/\breve{T}_0, \tag{2.202}$$

$$g_1 = (1 - 2\mathcal{E}_0)/\beta^2, \tag{2.203}$$

$$g_2 = (\gamma - 1)^2/\gamma^2, \tag{2.204}$$

$$g_3 = (2\gamma - 1)/\gamma^2, \tag{2.205}$$

$$r = \mathcal{E}/(1 - \mathcal{E}) . \tag{2.206}$$

The sampling procedure is as follows:

1. Compute parameters depending on $\breve{E}_0$, but not $\mathcal{E}$: $\mathcal{E}_0$, g_1, g_2, and g_3.

2. Sample $\mathcal{E}$ from $f(\mathcal{E})$ by using

$$\mathcal{E} = T_E/ [\breve{T}_0 - (\breve{E}_0 - \breve{E}_{Th}^{Møller})\zeta_1]). \tag{2.207}$$

3. Compute r and the rejection function $g(\mathcal{E})$. If $\zeta_2 > g(\mathcal{E})$, reject and return to Step 2.

After the secondary energies have been determined, Equation 2.167 can be used to obtain the scattering angles, and EGS routine `UPHI` can be called to select random azimuthal angles and set up the secondary particles in the usual way.

2.11 Bhabha Scattering

The differential Bhabha[25], cross section, as formulated in PEGS, is

$$\frac{d\breve{\Sigma}_{Bhabha}(\breve{E}_0)}{d\breve{E}_-} = \frac{X_0 n 2\pi r_0^2 m}{\breve{T}_0^2} \left[\frac{1}{\mathcal{E}} \left(\frac{1}{\mathcal{E}\beta^2} - B_1 \right) + B_2 + \mathcal{E} \left(\mathcal{E}B_4 - B_3 \right) \right] \qquad (2.208)$$

where

$$
\begin{aligned}
\breve{E}_0 &= \text{energy of incident positron (MeV),} \\
\breve{T}_0 &= \text{kinetic energy of incident positron (MeV),} \\
\beta &= v/c \text{ for incident positron,} \\
\gamma &= \breve{E}_0/m, \\
\breve{E}_- &= \text{energy of secondary electron (MeV),} \\
\mathcal{E} &= (\breve{E}_- - m)/\breve{T}_0 = \breve{T}_-/\breve{T}_0, \\
y &= 1/(\gamma + 1), \\
B_1 &= 2 - y^2, \\
B_2 &= (1 - 2y)(3 + y^2), \\
B_3 &= B_4 + (1 - 2y)^2, \\
B_4 &= (1 - 2y)^3.
\end{aligned}
$$

If Equation 2.208 is integrated between $\breve{E}_1$ and $\breve{E}_2$, we obtain

$$\int_{\breve{E}_1}^{\breve{E}_2} \frac{d\breve{\Sigma}_{Bhabha}(\breve{E}_0)}{d\breve{E}_-} \, d\breve{E}_- = \frac{X_0 n 2\pi r_0^2 m}{\breve{T}_0^2} \left[\frac{1}{\beta^2} \left(\frac{1}{\mathcal{E}_1} - \frac{1}{\mathcal{E}_2} \right) - B_1 \ell n \frac{\mathcal{E}_2}{\mathcal{E}_1} \right.$$
$$\left. + B_2(\mathcal{E}_2 - \mathcal{E}_1) + \mathcal{E}_2^2(\mathcal{E}_2 B_4/3 - B_3/2) - \mathcal{E}_1^2(\mathcal{E}_1 B_4/3 - B_3/2) \right]$$
$$(2.209)$$

where

$$\mathcal{E}_i = (\breve{E}_i - m)/\breve{T}_0, \qquad i = 1, 2 \qquad (2.210)$$

and other symbols are the same as in Equation 2.208.

Unlike in Møller scattering, in Bhabha scattering, the final state particles are distinguishable, so the upper limit for $\mathcal{E}$ is 1. Note that $\mathcal{E}$ is the fraction of the kinetic energy that the negative atomic electron gets. There is still a singularity at $\mathcal{E} = 0$ which is circumvented in the same way as for Møller by requiring that the energy transfered to the atomic electron be at least $T_E = A_E - m$. It should be noted that there is no singularity at $\mathcal{E} = 1$ as there was for Møller, and in fact, the final positron energy may be less than A_E (down to m). Thus, the threshold for a discrete Bhabha interaction is A_E, and as long as the positron is above the cutoff energy, it will have some non-zero Bhabha cross section. Using the minimum and maximum $\breve{E}_-$ for $\breve{E}_1$ and $\breve{E}_2$ in Equation 2.209, we obtain the total cross section as

$$\breve{\Sigma}_{Bhabha}(\breve{E}_0) = (\text{Equation 2.209 with } \breve{E}_1 = A_E \ \& \ \breve{E}_2 = \breve{E}_0 \text{ if } \breve{E}_0 > A_E, \ 0). \qquad (2.211)$$

PEGS functions BHABDM, BHABRM, and BHABTM evaluate Equations 2.208, 2.209, and 2.211, respectively.

In sampling the differential cross section to obtain the electron energy, we take $\mathcal{E} = \breve{T}_-/\breve{T}_0$ as the variable to be sampled and obtain

$$\frac{d\breve{\Sigma}_{Bhabha}(\breve{E}_0)}{d\mathcal{E}} = \frac{X_0 n 2\pi r_0^2 m}{\breve{T}_0 \mathcal{E}_0} \, f(\mathcal{E})g(\mathcal{E}) \tag{2.212}$$

where

$$f(\mathcal{E}) = \frac{\mathcal{E}_0}{1 - \mathcal{E}_0} \frac{1}{\mathcal{E}^2}, \ \mathcal{E} \in (\mathcal{E}_0, 1) \ , \tag{2.213}$$

$$g(\mathcal{E}) = (1 - \mathcal{E}_0)\left[\frac{1}{\beta^2} - \mathcal{E}\left(B_1 - \mathcal{E}\left(B_2 - \mathcal{E}(B_3 - \mathcal{E}B_4)\right)\right)\right] \ , \tag{2.214}$$

$$\mathcal{E}_0 = T_E/\breve{T}_0 \ , \tag{2.215}$$

$$y = 1/(\gamma + 1) \ , \tag{2.216}$$

$$B_4 = (1 - 2y)^3, \tag{2.217}$$

$$B_3 = B_4 + (1 - 2y)^2, \tag{2.218}$$

$$B_2 = (1 - 2y)(3 + y^2), \tag{2.219}$$

$$B_1 = 2 - y^2. \tag{2.220}$$

(Note that EGS uses variable YY to avoid conflict with the variable name of the y-coordinate of particle.)

The sampling method is as follows:

1. Compute parameters depending on $\breve{E}_0$ but not $\mathcal{E}$: $\mathcal{E}_0$, β, γ, B_1, B_2, B_3, and B_4.

2. Sample $\mathcal{E}$ from $f(\mathcal{E})$ using

$$\mathcal{E} = \mathcal{E}_0/[1 - (1 - \mathcal{E}_0)\zeta_1]. \tag{2.221}$$

3. Compute the rejection function $g(\mathcal{E})$. If $\zeta_2 > g(\mathcal{E})$, reject and return to Step 2.

The rest of the procedure is similar that used in sampling from the Møller cross section except that now the delta ray may have the most energy, in which case the contents of the two top locations of the EGS particle stack must be interchanged to ensure that the particle with the lower energy will be tracked first.

2.12 Two Photon Positron-Electron Annihilation

The two photon positron-electron annihilation cross sections in EGS are taken from Heitler[71] (see p. 268-270 therein). Using Heitler's formula 6 (on p. 269), translating to the laboratory frame,

integrating over azimuth, and changing from an angle variable to an energy variable, we obtain the following form of the differential cross section, which is used in PEGS:

$$\frac{d\breve{\Sigma}_{Annih}(\breve{E}_0)}{d\breve{k}} = S_1(k') + S_1(A - k') \tag{2.222}$$

where

$$
\begin{aligned}
\breve{E}_0 &= \text{energy of incident positron (MeV)}, & (2.223) \\
\breve{k} &= \text{energy of secondary photon of lower energy (MeV)}, & \\
\gamma &= \breve{E}_0/m, & (2.224) \\
A &= \gamma + 1 = \text{(available energy)}/m, & (2.225) \\
T_0' &= \gamma - 1 = \text{(kinetic energy)}/m, & (2.226) \\
p_0' &= \breve{p}_0/m = \sqrt{\gamma^2 - 1} = \sqrt{AT_0'}, & (2.227) \\
k' &= \breve{k}/m, & (2.228) \\
S_1(x) &= C_1[-1 + (C_2 - 1/x)/x], & (2.229) \\
C_1 &= \frac{X_0 n\pi r_0^2}{AT_0'm}, & (2.230) \\
C_2 &= A + 2\gamma/A. & (2.231)
\end{aligned}
$$

We see that Equation 2.222 satisfies, in a manifest way, the symmetry under exchange of the annihilation photons.

Integrating Equation 2.222 between $\breve{k}_1$ and $\breve{k}_2$ we obtain

$$\int_{\breve{k}_1}^{\breve{k}_2} \frac{d\breve{\Sigma}_{Annih}}{d\breve{k}} \, d\breve{k} = S_2(k_2') - S_2(k_1') + S_2(A - k_1') - S_2(A - k_2') \tag{2.232}$$

where

$$k_i' = \breve{k}_i/m, \qquad i = 1, \ 2, \tag{2.233}$$

$$S_2(x) = m \int S_1(y)dy = m \, C_1[-x + C_2 \, \ell n \, x + 1/x] . \tag{2.234}$$

For the total annihilation cross section we use Heitler's formula with appropriate changes to take into account units and notation, and we have

$$\breve{\Sigma}_{Annih}(\breve{E}_0) = \frac{X_0 n\pi r_0^2}{\gamma + 1} \left[\frac{\gamma^2 + 4\gamma + 1}{\gamma^2 - 1} \ell n(\gamma + \sqrt{\gamma^2 - 1}) - \frac{\gamma + 3}{\sqrt{\gamma^2 - 1}} \right] . \tag{2.235}$$

PEGS functions `ANIHDM`, `ANIHRM`, and `ANIHTM` evaluate Equations 2.222, 2.232, and 2.235, respectively.

In computing the energies of the secondary photons, we sample the parameter $\mathcal{E}$, which is defined by

$$\mathcal{E} = \frac{k'}{A} = \frac{\breve{k}}{\breve{E}_0 + m} . \tag{2.236}$$

To find the limits of $\mathcal{E}$, we first compute the limits for k'. Using Equation 2.167 with $m_1 = m$, $E_1 = \breve{E}_0$, $p_1 = \breve{p}_0$, $m_3 = m_4 = 0$, $E_3 = p_3 = \breve{k}$, which yields

$$\cos\theta = \frac{(\breve{E}_0 + m)\breve{k} - \breve{E}_0 m - m^2}{\breve{p}_0 \breve{k}} \quad . \tag{2.237}$$

Solving for $\breve{k}$ and dividing by m we obtain

$$k' = \frac{A}{A - (\breve{p}_0/m)\cos\theta} = \frac{1}{1 - \sqrt{T_0'/A}\cos\theta} \quad . \tag{2.238}$$

Setting $\cos\theta_3 = \pm 1$ in Equation 2.238 we then see that

$$k'_{min} = \frac{1}{1 + \sqrt{T_0'/A}} = \frac{A}{A + p_0'} \quad , \tag{2.239}$$

$$k'_{max} = \frac{1}{1 - \sqrt{T_0'/A}} = \frac{A}{A - p_0'} \quad . \tag{2.240}$$

We also have

$$
\begin{aligned}
k'_{min} + k'_{max} &= \frac{1}{1 + \sqrt{T_0'/A}} + \frac{1}{1 - \sqrt{T_0'/A}} \\
&= \frac{1 - \sqrt{T_0'/A} + 1 + \sqrt{T_0'/A}}{1 - T_0'/A} = \frac{2A}{\gamma + 1 - (\gamma - 1)} \\
&= A \quad .
\end{aligned}
\tag{2.241}
$$

That is, when one photon is at the minimum, the other is at the maximum and the sum of their energies is equal to the available energy (divided by m).

Because of the indistinguishability of the two photons, we have the restriction $\mathcal{E} < 1/2$. Moreover, we have $\mathcal{E} > \mathcal{E}_0$, where

$$\mathcal{E}_0 = \frac{k'_{min}}{A} = \frac{1}{A + \sqrt{AT_0'}} = \frac{1}{A + p_0'} \quad . \tag{2.242}$$

Given our change of variables, the cross section to be sampled is given as

$$\frac{d\breve{\Sigma}_{Annih}}{d\mathcal{E}} = mA\left[S_1(A\mathcal{E}) + S_1\left(A(1 - \mathcal{E})\right)\right] \quad . \tag{2.243}$$

Because of the symmetry in $\mathcal{E}$, we can expand the range of $\mathcal{E}$ to be sampled from $(\mathcal{E}_0, \ 1/2)$ to $(\mathcal{E}_0, \ 1 - \mathcal{E}_0)$ and we can ignore the second S_1, as we can use $1 - \mathcal{E}$ if we sample a value of $\mathcal{E}$ greater than $1/2$. We can then express the distribution to be sampled as

$$\frac{d\breve{\Sigma}_{Annih}}{d\mathcal{E}} = mAC_1\left[-1 + \left(C_2 - \frac{1}{A\mathcal{E}}\right)\Big/ A\mathcal{E},\right] \ , \mathcal{E} \in (\mathcal{E}_0, \ 1 - \mathcal{E}_0) \quad . \tag{2.244}$$

71

Using the value of C_2 and rearranging, we obtain

$$\frac{d\breve{\Sigma}_{Annih}}{d\mathcal{E}} = \frac{X_0 n \pi r_0^2 m}{\breve{T}_0} \ \ell n \ [(1 - \mathcal{E}_0)/\mathcal{E}_0] \ f(\mathcal{E})g(\mathcal{E}) \tag{2.245}$$

where

$$f(\mathcal{E}) \ = \ \frac{1}{\ell n[(1 - \mathcal{E}_0)/\mathcal{E}_0]} \frac{1}{\mathcal{E}} \ , \tag{2.246}$$

$$g(\mathcal{E}) \ = \ 1 - \mathcal{E} + \frac{1}{A^2}\left(2\gamma - \frac{1}{\mathcal{E}}\right) \ . \tag{2.247}$$

To see that $g(\mathcal{E})$ is a valid rejection function we find its extrema, as below:

$$g'(\mathcal{E}) \ = \ -1 + \frac{1}{A^2 \mathcal{E}^2} = 0 \Rightarrow \mathcal{E} = \frac{1}{A} \tag{2.248}$$

$$g''(\mathcal{E}) \ = \ \frac{-2}{A^2 \mathcal{E}^3} < 0 \ .$$

Thus, we see that $g(\mathcal{E})$ has a maximum at $\mathcal{E} = \frac{1}{A} \geq \mathcal{E}_0$ with maximum value

$$g\left(\frac{1}{A}\right) = 1 - \frac{2}{A^2} < 1 \ . \tag{2.249}$$

The sampling procedure is then as follows:

1. Compute parameters depending on $\breve{E}_0$ only:

$$A, \ \gamma, \ T_0', \ p_0', \ \mathcal{E}_0, \ \ell n\left(\frac{1 - \mathcal{E}_0}{\mathcal{E}_0}\right) \ .$$

2. Sample $\mathcal{E}$ using

$$\mathcal{E} = \mathcal{E}_0 \exp\left[\zeta_1 \ell n \ \left(\frac{1 - \mathcal{E}_0}{\mathcal{E}_0}\right)\right] \ . \tag{2.250}$$

3. Compute $g(\mathcal{E})$. Then reject $\mathcal{E}$ and return to Step 2 if $\zeta_2 > g(\mathcal{E})$.

4. Get larger and smaller $\mathcal{E}$ values using

$$\mathcal{E}_{Bigger} \ = \ max(\mathcal{E}, 1 - \mathcal{E}) \ , \tag{2.251}$$

$$\mathcal{E}_{Smaller} \ = \ 1 - \mathcal{E}_{Bigger} \ . \tag{2.252}$$

5. Use these to set up the photons.

The polar angles are determined by the photon energies using Equation 2.167, and UPHI is called to select random azimuths.

   A special case arises when a positron falls below the cutoff energy set in EGS before annihilating. When this occurs, the particle is assumed to come to rest before annihilating, and each gamma
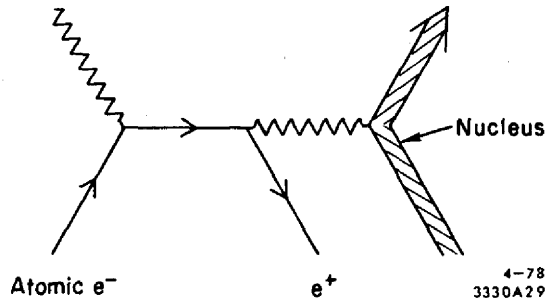
Figure 2.5: Feynman diagram for single photon e$^+$e$^-$ annihilation.

is assumed to have energy $\check{k} = m$. The gamma angular distribution is taken to be isotropic the photons moving in opposite directions.

It should be noted that a positron may annihilate and give off only one photon via the process shown in Figure 2.5. Messel and Crawford[103] make the point that the ratio of one photon annihilation to two photon annihilation is small until higher energies are reached, at which point the absolute value of the cross section is small. Thus, EGS ignores one photon positron annihilation. Note that it is also possible to have annihilation accompanied by three or more photons, but these interactions are even less likely than one photon annihilation and are not considered in EGS.

2.13   Continuous Electron Energy Loss

Recall the distinction made earlier between collisions contributing to discrete and to continuous energy losses in electron and positron scattering. In discrete collisions, secondary particles with energies above their cutoff energies are created and subsequently transported. Continuous energy losses result from interactions in which the energy transfered to the secondary particles is below the discrete transport energy thresholds. The resulting secondary particles, which are either soft bremsstrahlung photons or low energy atomic electrons. The mean total energy loss per unit length in these continuous loss collisions is thus given by

$$-\left(\frac{dE_\pm}{dx}\right)_{\substack{Total \\ Continuous}} = -\left(\frac{dE_\pm}{dx}\right)_{\substack{Soft \\ Bremsstrahlung}} - \left(\frac{dE_\pm}{dx}\right)_{\substack{Sub\text{-}Cutoff \\ Atomic\ Electrons}} \qquad (2.253)$$

where $\pm$ denotes positive or negative electrons. The first term on the right-hand side of Equation 2.253 is the same for electrons and positrons (to the accuracy with which we treat bremsstrahlung) and is given by Equation 2.148.

The second term may be expressed as the integral of the differential cross section for transferring a specified amount of energy, T, to an atomic electron, times the amount of the energy transfer

73

over the range of energy transfers which still give rise to soft final state secondary electrons, as in:

$$-\left(\frac{dE_\pm}{dx}\right)_{\substack{Sub\text{-}Cutoff \\ Atomic\ Electrons}} = \int_0^{T_{max}} T\ \frac{d\Sigma_\pm}{dT}\ dT\ .$$ (2.254)

For values of T on the order of the atomic excitation levels, the frequencies and strengths of the atomic oscillators must be taken into account and the evaluation of this integral is quite complicated. For values of T large enough that the atomic electrons may be considered free, the Møller or Bhabha cross sections can be used to describe the scattering. If we let T_{med} be a value of T which is sufficiently above the atomic excitation level, but is still small compared to T_{max}, we have

$$\left(-\frac{dE_-}{dx}\right) = \int_0^{T_{med}} T\ \frac{d\Sigma_-}{dT}\ dT + \int_{T_{med}}^{T_{max}} T\ \frac{d\Sigma_{M\text{ø}ller}}{dE}\ dT$$ (2.255)

and

$$\left(-\frac{dE_+}{dx}\right) = \int_0^{T_{med}} T\ \frac{d\Sigma_+}{dT}\ dT + \int_{T_{med}}^{T_{max}} T\ \frac{d\Sigma_{Bhabha}}{dE}\ dE$$ (2.256)

When appropriate approximations are made, Equations 2.224 and 2.225 are independent of T_{med}. We use the formulas recommended by Berger and Seltzer [17] for restricted stopping power[7] which are based on the Bethe-Bloch formula [21, 22, 36]. Note that we have corrected typographic errors in Berger and Seltzer's Equations 22-24 for $F^+(\tau, \Delta)$). Other useful references regarding this topic are Rohrlich and Carlson[140], Jauch and Rohrlich[83], Turner[173], Sternheimer[159, 160, 161, 162, 163, 165], Evans[53] (for Bhabha and Møller cross sections), Armstrong and Alsmiller[10], and Messel and Crawford[103]. The formula used in PEGS for the restricted stopping power (i.e., due to sub-cutoff electrons) is

$$\left(-X_0\ \frac{d\check{E}_\pm}{dx}\right)_{\substack{Sub\text{-}Cutoff \\ Atomic\ Electrons}} = \frac{X_0 n 2\pi r_0^2 m}{\beta^2}\ \left[\ell n\ \frac{2(\tau+2)}{(\bar{I}_{adj}/m)} + F^\pm(\tau, \Delta) - \delta\right]$$ (2.257)

$$\text{where} \quad \gamma \quad = \quad \check{E}_0/m = \text{usual relativistic factor,} \tag{2.258}$$
$$\eta \quad = \quad \sqrt{\gamma^2 - 1} = \beta\gamma = \check{p}_0 c/m, \tag{2.259}$$
$$\beta \quad = \quad \sqrt{1 - \gamma^{-2}} = v/c \text{ for incident particle,} \tag{2.260}$$
$$T_E' \quad = \quad T_E/m = \text{ K.E. cutoff in electron mass units,} \tag{2.261}$$
$$\tau \quad = \quad \gamma - 1, \tag{2.262}$$
$$y \quad = \quad (\gamma + 1)^{-1} \text{ (See Bhabha formula),} \tag{2.263}$$
$$T_{max}' \quad = \quad \text{maximum energy transfer}$$
$$= \quad (\tau \text{ if positron, } \tau/2 \text{ if electron),} \tag{2.264}$$
$$\Delta \quad = \quad \text{restricted maximum energy transfer}$$
$$= \quad min(T_E',\ T_{max}'), \tag{2.265}$$

---

[7]The concept of "restricted stopping power" is discussed in detail in the book by Kase and Nelson[87].

$$\bar{I}_{adj} \quad = \quad \text{average adjusted mean ionization energy,} \tag{2.266}$$

$$\delta \quad = \quad \text{density effect correction,} \tag{2.267}$$

$$F^-(\tau, \Delta) \quad = \quad -1 - \beta^2 + \ell n[(\tau - \Delta)\Delta] + \tau/(\tau - \Delta)$$
$$+ \quad \left[\frac{\Delta^2}{2} + (2\tau + 1)\ell n \left(1 - \frac{\Delta}{\tau} \right) \right] / \gamma^2 \tag{2.268}$$

$$F^+(\tau, \Delta) \quad = \quad \ell n(\tau\Delta) - \frac{\beta^2}{\tau} \left\{ \tau + 2\Delta - \frac{3\Delta^2 y}{2} \right.$$
$$- \quad \left. (\Delta - \Delta^3/3)y^2 - (\Delta^2/2 - \tau\Delta^3/3 + \Delta^4/4)y^3 \right\} \quad . \tag{2.269}$$

Since EGS3 was released there has been considerable effort devoted to establishing accurate values for the parameters in the above equations and considerable changes have occurred, especially in the recommended I values. The values recommended in 1983 by Berger and Seltzer[20] have also been recommend by the International Commission on Radiation Units and Measurements[79]. PEGS contains a table of I values for all elements and these are presented in Table 2.2 along with the default atomic weights and elemental densities used in PEGS (any of which can be overridden by the user).

The values of $\bar{I}_{adj}$ for mixtures and compounds are derived from

$$\ell n \ \bar{I}_{adj} = \left(\sum_{i=1}^{N_e} p_i Z_i \left(\ell n \ I_{adj}(Z_i) \right) \right) \Big/ \left(\sum_{i=1}^{N_e} p_i Z_i \right) \tag{2.270}$$

or taken from Table 2.3 for a select group of materials as discussed below.

Table 2.2: Default atomic numbers, symbols, weights, densities and I values in PEGS.

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
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
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

Table 2.2 (cont.)

Z	Symbol	Atomic Weight	Density (g/cm^3)	I(eV)
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
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

Table 2.3: Default Sternheimer density effect coefficients in PEGS.

LABEL	a	m_s	x_0	x_1	I(eV)	$-$C	δ_0
H2-GAS	0.14092	5.7273	1.8639	3.2718	19.2	9.5835	0.00
H2-LIQUID	0.13483	5.6249	0.4759	1.9215	21.8	3.2632	0.00
HE-GAS	0.13443	5.8347	2.2017	3.6122	41.8	11.1393	0.00
LI	0.95136	2.4993	0.1304	1.6397	40.0	3.1221	0.14
BE	0.80392	2.4339	0.0592	1.6922	63.7	2.7847	0.14
B	0.56224	2.4512	0.0305	1.9688	76.0	2.8477	0.14
C-2.265 G/CM**3	0.26142	2.8697	-0.0178	2.3415	78.0	2.8680	0.12
C-2.00 G/CM**3	0.20240	3.0036	-0.0351	2.4860	78.0	2.9925	0.10
C-1.70 G/CM**3	0.20762	2.9532	0.0480	2.5387	78.0	3.1550	0.14
N2-GAS	0.15349	3.2125	1.7378	4.1323	82.0	10.5400	0.00
O2-GAS	0.11778	3.2913	1.7541	4.3213	95.0	10.7004	0.00
F	0.11083	3.2962	1.8433	4.4096	115.0	10.9653	0.00
NE-GAS	0.08064	3.5771	2.0735	4.6421	137.0	11.9041	0.00
NA	0.07772	3.6452	0.2880	3.1962	149.0	5.0526	0.08
MG	0.08163	3.6166	0.1499	3.0668	156.0	4.5297	0.08
AL	0.08024	3.6345	0.1708	3.0127	166.0	4.2395	0.12
SI	0.14921	3.2546	0.2014	2.8715	173.0	4.4351	0.14
P	0.23610	2.9158	0.1696	2.7815	173.0	4.5214	0.14
S	0.33992	2.6456	0.1580	2.7159	180.0	4.6659	0.14
CL	0.19849	2.9702	1.5555	4.2994	174.0	11.1421	0.00
AR-GAS	0.19714	2.9618	1.7635	4.4855	188.0	11.9480	0.00
K	0.19827	2.9233	0.3851	3.1724	190.0	5.6423	0.10
CA	0.15643	3.0745	0.3228	3.1191	191.0	5.0396	0.14
SC	0.15754	3.0517	0.1640	3.0593	216.0	4.6949	0.10
TI	0.15662	3.0302	0.0957	3.0386	233.0	4.4450	0.12
V	0.15436	3.0163	0.0691	3.0322	245.0	4.2659	0.14
CR	0.15419	2.9896	0.0340	3.0451	257.0	4.1781	0.14
MN	0.14973	2.9796	0.0447	3.1074	272.0	4.2702	0.14
FE	0.14680	2.9632	-0.0012	3.1531	286.0	4.2911	0.12
CO	0.14474	2.9502	-0.0187	3.1790	297.0	4.2601	0.12
NI	0.16496	2.8430	-0.0566	3.1851	311.0	4.3115	0.10
CU	0.14339	2.9044	-0.0254	3.2792	322.0	4.4190	0.08
ZN	0.14714	2.8652	0.0049	3.3668	330.0	4.6906	0.08
GA	0.09440	3.1314	0.2267	3.5434	334.0	4.9353	0.14
GE	0.07188	3.3306	0.3376	3.6096	350.0	5.1411	0.14
AS	0.06633	3.4176	0.1767	3.5702	347.0	5.0510	0.08
SE	0.06568	3.4317	0.2258	3.6264	348.0	5.3210	0.10
BR	0.06335	3.4670	1.5262	4.9899	343.0	11.7307	0.00
KR-GAS	0.07446	3.4051	1.7158	5.0748	352.0	12.5115	0.00
RB	0.07261	3.4177	0.5737	3.7995	363.0	6.4776	0.14
SR	0.07165	3.4435	0.4585	3.6778	366.0	5.9867	0.14
Y	0.07138	3.4585	0.3608	3.5542	379.0	5.4801	0.14
ZR	0.07177	3.4533	0.2957	3.4890	393.0	5.1774	0.14
NI	0.13883	3.0930	0.1785	3.2201	417.0	5.0141	0.14
MO	0.10525	3.2549	0.2267	3.2784	424.0	4.8793	0.14
TC	0.16572	2.9738	0.0949	3.1253	428.0	4.7769	0.14
RU	0.19342	2.8707	0.0599	3.0834	441.0	4.7694	0.14

78

Table 2.3 (cont.)

LABEL	a	m_s	x_0	x_1	I(eV)	$-C$	δ_0
RH	0.19205	2.8633	0.0576	3.1069	449.0	4.8008	0.14
PD	0.24178	2.7239	0.0563	3.0555	470.0	4.9358	0.14
AG	0.24585	2.6899	0.0657	3.1074	470.0	5.0630	0.14
CD	0.24609	2.6772	0.1281	3.1667	469.0	5.2727	0.14
IN	0.23879	2.7144	0.2406	3.2032	488.0	5.5211	0.14
SN	0.18689	2.8576	0.2879	3.2959	488.0	5.5340	0.14
SB	0.16652	2.9519	0.3189	3.3489	487.0	5.6241	0.14
TE	0.13815	3.0354	0.3296	3.4418	485.0	5.7131	0.14
I	0.23766	2.7276	0.0549	3.2596	491.0	5.9488	0.00
XE-GAS	0.23314	2.7414	1.5630	4.7371	482.0	12.7281	0.00
CS	0.18233	2.8866	0.5473	3.5914	488.0	6.9135	0.14
BA	0.18268	2.8906	0.4190	3.4547	491.0	6.3153	0.14
LA	0.18591	2.8828	0.3161	3.3293	501.0	5.7850	0.14
CE	0.18885	2.8592	0.2713	3.3432	523.0	5.7837	0.14
PR	0.23265	2.7331	0.2333	3.2773	535.0	5.8096	0.14
ND	0.23530	2.7050	0.1984	3.3063	546.0	5.8290	0.14
PM	0.24280	2.6674	0.1627	3.3199	560.0	5.8224	0.14
SM	0.24698	2.6403	0.1520	3.3460	574.0	5.8597	0.14
EU	0.24448	2.6245	0.1888	3.4633	580.0	6.2278	0.14
GD	0.25109	2.5977	0.1058	3.3932	591.0	5.8738	0.14
TB	0.24453	2.6056	0.0947	3.4224	614.0	5.9045	0.14
DY	0.24665	2.5849	0.0822	3.4474	628.0	5.9183	0.14
HO	0.24638	2.5726	0.0761	3.4782	650.0	5.9587	0.14
ER	0.24823	2.5573	0.0648	3.4922	658.0	5.9521	0.14
TM	0.24889	2.5469	0.0812	3.5085	674.0	5.9677	0.14
YB	0.25295	2.5141	0.1199	3.6246	684.0	6.3325	0.14
LU	0.24033	2.5643	0.1560	3.5218	694.0	5.9785	0.14
HF	0.22918	2.6155	0.1965	3.4337	705.0	5.7139	0.14
TA	0.17798	2.7623	0.2117	3.4805	718.0	5.5262	0.14
W	0.15509	2.8447	0.2167	3.4960	727.0	5.4059	0.14
RE	0.15184	2.8627	0.0559	3.4845	736.0	5.3445	0.08
OS	0.12751	2.9608	0.0891	3.5414	746.0	5.3083	0.10
IR	0.12690	2.9658	0.0819	3.5480	757.0	5.3418	0.10
PT	0.11128	3.0417	0.1484	3.6212	790.0	5.4732	0.12
AU	0.09756	3.1101	0.2021	3.6979	790.0	5.5747	0.14
HG	0.11014	3.0519	0.2756	3.7275	800.0	5.9605	0.14
TL	0.09455	3.1450	0.3491	3.8044	810.0	6.1365	0.14
PB	0.09359	3.1608	0.3776	3.8073	823.0	6.2018	0.14
BI	0.09410	3.1671	0.4152	3.8248	823.0	6.3505	0.14
PO	0.09282	3.1830	0.4267	3.8293	830.0	6.4003	0.14
RN-GAS	0.20798	2.7409	1.5368	4.9889	794.0	13.2839	0.00
RA	0.08804	3.2454	0.5991	3.9428	826.0	7.0452	0.14
AC	0.08567	3.2683	0.4559	3.7966	841.0	6.3742	0.14
TH	0.08655	3.2610	0.4202	3.7681	847.0	6.2473	0.14
PA	0.14770	2.9845	0.3144	3.5079	878.0	6.0327	0.14
U	0.19677	2.8171	0.2260	3.3721	890.0	5.8694	0.14
NP	0.19741	2.8082	0.1869	3.3690	902.0	5.8149	0.14

Table 2.3 (cont.)

LABEL	a	m_s	x_0	x_1	I(eV)	$-$C	δ_0
PU	0.20419	2.7679	0.1557	3.3981	921.0	5.8748	0.14
AM	0.20308	2.7615	0.2274	3.5021	934.0	6.2813	0.14
CM	0.20257	2.7579	0.2484	3.5160	939.0	6.3097	0.14
BK	0.20192	2.7560	0.2378	3.5186	952.0	6.2912	0.14
A 150-PLASTIC	0.10783	3.4442	0.1329	2.6234	65.1	3.1100	0.00
ACETONE	0.11100	3.4047	0.2197	2.6028	64.2	3.4341	0.00
ACETYLENE	0.12167	3.4277	1.6017	4.0074	58.2	9.8419	0.00
ADENINE	0.20908	3.0271	0.1295	2.4219	71.4	3.1724	0.00
ADIPOSE TISSUE	0.10278	3.4817	0.1827	2.6530	63.2	3.2367	0.00
AIR-GAS	0.10914	3.3994	1.7418	4.2759	85.7	10.5961	0.00
ALANINE	0.11484	3.3526	0.1354	2.6336	71.9	3.0965	0.00
ALUMINIUM OXIDE	0.08500	3.5458	0.0402	2.8665	145.2	3.5682	0.00
AMBER	0.11934	3.4098	0.1335	2.5610	63.2	3.0701	0.00
AMMONIA	0.08315	3.6464	1.6822	4.1158	53.7	9.8763	0.00
ANILINE	0.13134	3.3434	0.1618	2.5805	66.2	3.2622	0.00
ANTHRACENE	0.14677	3.2831	0.1146	2.5213	69.5	3.1514	0.00
B-100 BONE-EQ. PLASTIC	0.05268	3.7365	0.1252	3.0420	85.9	3.4528	0.00
BAKELITE	0.12713	3.3470	0.1471	2.6055	72.4	3.2582	0.00
BARIUM FLUORIDE	0.15991	2.8867	-0.0098	3.3871	375.9	5.4122	0.00
BARIUM SULFATE	0.11747	3.0427	-0.0128	3.4069	285.7	4.8923	0.00
BENZENE	0.16519	3.2174	0.1710	2.5091	63.4	3.3269	0.00
BERYLLIUM OXIDE	0.10755	3.4927	0.0241	2.5846	93.2	2.9801	0.00
BGO	0.09569	3.0781	0.0456	3.7816	534.1	5.7409	0.00
BLOOD (ICRP)	0.08492	3.5406	0.2239	2.8017	75.2	3.4581	0.00
BONE, COMPACT (ICRU)	0.05822	3.6419	0.0944	3.0201	91.9	3.3390	0.00
BONE CORTICAL (ICRP)	0.06198	3.5919	0.1161	3.0919	106.4	3.6488	0.00
BORON CARBIDE	0.37087	2.8076	0.0093	2.1006	84.7	2.9859	0.00
BORN OXIDE	0.11548	3.3832	0.1843	2.7379	99.6	3.6027	0.00
BRAIN (ICRP)	0.08255	3.5585	0.2206	2.8021	73.3	3.4279	0.00
BUTANE	0.10852	3.4884	1.3788	3.7524	48.3	8.5633	0.00
N-BUTYL ALCHOL	0.10081	3.5139	0.1937	2.6439	59.9	3.2425	0.00
C-552 AIR-EQ. PLASTIC	0.10492	3.4344	0.1510	2.7083	86.8	3.3338	0.00
CADMIUM TELLURIDE	0.24840	2.6665	0.0438	3.2836	539.3	5.9096	0.00
CADMIUM TUNGSTATE	0.12861	2.9150	0.0123	3.5941	468.3	5.3594	0.00
CALCIUM CARBONITE	0.08301	3.4120	0.0492	3.0549	136.4	3.7738	0.00
CAF2	0.06942	3.5263	0.0676	3.1683	166.0	4.0653	0.00
CALCIUM OXIDE	0.12128	3.1936	-0.0172	3.0171	176.1	4.1209	0.00
CALCIUM SULFATE	0.07708	3.4495	0.0587	3.1229	152.3	3.9388	0.00
CALCIUM TUNGSTATE	0.06210	3.2649	0.0323	3.8932	395.0	5.2603	0.00
CARBON DIOXIDE	0.11768	3.3227	1.6294	4.1825	85.0	10.1537	0.00
CARBON TETRACHLORIDE	0.19018	3.0116	0.1773	2.9165	166.3	4.7712	0.00
CELLOPHANE	0.11151	3.3810	0.1580	2.6778	77.6	3.2647	0.00
CELLULOSE ACETATE BUTYRA	0.11444	3.3738	0.1794	2.6809	74.6	3.3497	0.00
CELLULOSE NITRATE	0.11813	3.3237	0.1897	2.7253	87.0	3.4762	0.00
CERIC SURFARE DOSIMETER	0.07666	3.5607	0.2363	2.8769	76.7	3.5212	0.00
CESIUM FLUORIDE	0.22052	2.7280	0.0084	3.3374	440.7	5.9046	0.00
CSI	0.25381	2.6657	0.0395	3.3353	553.1	6.2807	0.00

Table 2.3 (cont.)

LABEL	a	m_s	x_0	x_1	I(eV)	$-C$	δ_0
CHLOROBENZENE	0.09856	3.3797	0.1714	2.9272	89.1	3.8201	0.00
CHLOROFORM	0.16959	3.0627	0.1786	2.9581	156.0	4.7055	0.00
CONCRETE, PORTLAND	0.07515	3.5467	0.1301	3.0466	135.2	3.9464	0.00
CYCLOHEXANE	0.12035	3.4278	0.1728	2.5549	56.4	3.1544	0.00
1,2-DICHLOROBENZENE	0.16010	3.0836	0.1587	2.8276	106.5	4.0348	0.00
DICHLORODIETHYL ETHER	0.06799	3.5250	0.1773	3.1586	103.5	4.0135	0.00
1,2-DICHLOROETHANE	0.13383	3.1675	0.1375	2.9529	111.9	4.1849	0.00
DIETHYL ETHER	0.10550	3.4586	0.2231	2.6745	60.0	3.3721	0.00
N,N-DIMETHYL FORMAMIDE	0.11470	3.3710	0.1977	2.6686	66.6	3.3311	0.00
DIMETHYL SULFOXIDE	0.06619	3.5708	0.2021	3.1263	98.6	3.9844	0.00
ETHANE	0.09627	3.6095	1.5107	3.8743	45.4	9.1043	0.00
ETHYL ALCOHOL	0.09878	3.4834	0.2218	2.7052	62.9	3.3699	0.00
ETHYL CELLULOSE	0.11077	3.4098	0.1683	2.6257	69.3	3.2415	0.00
ETHYLENE	0.10636	3.5387	1.5528	3.9327	50.7	9.4380	0.00
EYE LENS (ICRP)	0.09690	3.4550	0.2070	2.7446	73.3	3.3720	0.00
FERRIC OXIDE	0.10478	3.1313	-0.0074	3.2573	227.3	4.2245	0.00
FERROBORIDE	0.12911	3.0240	-0.0988	3.1749	261.0	4.2057	0.00
FERROUS OXIDE	0.12959	3.0168	-0.0279	3.2002	248.6	4.3175	0.00
FERROUS SULFATE DOSIMETE	0.08759	3.4923	0.2378	2.8254	76.4	3.5183	0.00
FREON-12	0.07978	3.4626	0.3035	3.2659	143.0	4.8251	0.00
FREON-12B2	0.05144	3.5565	0.3406	3.7956	284.9	5.7976	0.00
FREON-13	0.07238	3.5551	0.3659	3.2337	126.6	4.7483	0.00
FREON-13B1	0.03925	3.7194	0.3522	3.7554	210.5	5.3555	0.00
FREON-13I1	0.09112	3.1658	0.2847	3.7280	293.5	5.8774	0.00
GADOLINIUM OXYSULFIDE	0.22161	2.6300	-0.1774	3.4045	493.3	5.5347	0.00
GALLIUM ARSENIDE	0.07152	3.3356	0.1764	3.6420	384.9	5.3299	0.00
GEL IN PHOTOGRAPHIC EMUL	0.10102	3.4418	0.1709	2.7058	74.8	3.2687	0.00
PYREX-GLASS	0.08270	3.5224	0.1479	2.9933	134.0	3.9708	0.00
GLASS, LEAD	0.09544	3.0740	0.0614	3.8146	526.4	5.8476	0.00
GLASS, PLATE	0.07678	3.5381	0.1237	3.0649	145.4	4.0602	0.00
GLUCOSE	0.10783	3.3946	0.1411	2.6700	77.2	3.1649	0.00
GLUTAMINE	0.11931	3.3254	0.1347	2.6301	73.3	3.1167	0.00
GLYCEROL	0.10168	3.4481	0.1653	2.6862	72.6	3.2267	0.00
GUANINE	0.20530	3.0186	0.1163	2.4296	75.0	3.1171	0.00
GYPSUM, PLASTER OF PARIS	0.06949	3.5134	0.0995	3.1206	129.7	3.8382	0.00
N-HEPTANE	0.11255	3.4885	0.1928	2.5706	54.4	3.1978	0.00
N-HEXANE	0.11085	3.5027	0.1984	2.5757	54.0	3.2156	0.00
KAPTON, POLYIMIDE FILM	0.15972	3.1921	0.1509	2.5631	79.6	3.3497	0.00
LANTHANUM OXYBROMIDE	0.17830	2.8457	-0.0350	3.3288	439.7	5.4666	0.00
LANTHANUM OXYSULFIDE	0.21501	2.7298	-0.0906	3.2664	421.2	5.4470	0.00
LEAD OXIDE	0.19645	2.7299	0.0356	3.5456	766.7	6.2162	0.00
LITHIUM AMIDE	0.08740	3.7534	0.0198	2.5152	55.5	2.7961	0.00
LITHIUM CARBONATE	0.09936	3.5417	0.0551	2.6598	87.9	3.2029	0.00
LIF	0.07593	3.7478	0.0171	2.7049	94.0	3.1667	0.00
LITHIUM HYDRIDE	0.90567	2.5849	-0.0988	1.4515	36.5	2.3580	0.00
LII	0.23274	2.7146	0.0892	3.3702	485.1	6.2671	0.00
LITHIUM OXIDE	0.08035	3.7878	-0.0511	2.5874	73.6	2.9340	0.00

81

Table 2.3 (cont.)

LABEL	a	m_s	x_0	x_1	I(eV)	$-C$	δ_0
LITHIUM TETRABORATE	0.11075	3.4389	0.0737	2.6502	94.6	3.2093	0.00
LUNG (ICRP)	0.08588	3.5353	0.2261	2.8001	75.3	3.4708	0.00
M3 WAX	0.07864	3.6412	0.1523	2.7529	67.9	3.2540	0.00
MAGNESIUM CARBONATE	0.09219	3.5003	0.0860	2.7997	118.0	3.4319	0.00
MANESIUM FLUORIDE	0.07934	3.6485	0.1369	2.8630	134.3	3.7105	0.00
MAGNESIUM OXIDE	0.08313	3.5968	0.0575	2.8580	143.8	3.6404	0.00
MAGNESIUM TETRABORATE	0.09703	3.4893	0.1147	2.7635	108.3	3.4328	0.00
MERCURIC IODIDE	0.21513	2.7264	0.1040	3.4728	684.5	6.3787	0.00
METHANE	0.09253	3.6257	1.6263	3.9716	41.7	9.5243	0.00
METHANOL	0.08970	3.5477	0.2529	2.7639	67.6	3.5160	0.00
MIX D WAX	0.07490	3.6823	0.1371	2.7145	60.9	3.0780	0.00
MS20 TISSUE SUBSTITUTE	0.08294	3.6061	0.1997	2.8033	75.1	3.5341	0.00
MUSCLE, SKELETAL (ICRP)	0.08636	3.5330	0.2282	2.7999	75.3	3.4809	0.00
MUSCLE, STRIATED (ICRU)	0.08507	3.5383	0.2249	2.8032	74.7	3.4636	0.00
MUSCLE-EQ. LIQ. W SUCROS	0.09481	3.4699	0.2098	2.7550	74.3	3.3910	0.00
MUSCLE-EQ. LIQ. W/O SUCR	0.09143	3.4982	0.2187	2.7680	74.2	3.4216	0.00
NAPTHALENE	0.14766	3.2654	0.1374	2.5429	68.4	3.2274	0.00
NITROBENZENE	0.12727	3.3091	0.1777	2.6630	75.8	3.4073	0.00
NITROUS OXIDE	0.11992	3.3318	1.6477	4.1565	84.9	10.1575	0.00
NYLON, DU PONT	0.11513	3.4044	0.1503	2.6004	64.3	3.1250	0.00
NYLON, TYPE 6 AND 6/6	0.11818	3.3826	0.1336	2.5834	63.9	3.0634	0.00
NYLON, TYPE 6/10	0.11852	3.3912	0.1304	2.5681	63.2	3.0333	0.00
NYLON, TYPE 11	0.14868	3.2576	0.0678	2.4281	61.6	2.7514	0.00
OCTANE, LIQUID	0.11387	3.4776	0.1882	2.5664	54.7	3.1834	0.00
PARAFFIN WAX	0.12087	3.4288	0.1289	2.5084	55.9	2.9551	0.00
N-PENTANE	0.10809	3.5265	0.2086	2.5855	53.6	3.2504	0.00
PHOTOEMULSION	0.12399	3.0094	0.1009	3.4866	331.0	5.3319	0.00
PLASTIC SCINT.	0.16101	3.2393	0.1464	2.4855	64.7	3.1997	0.00
PLUTONIUM DIOXIDE	0.20594	2.6522	-0.2311	3.5554	746.5	5.9719	0.00
POLYCRYLONITRILE	0.16275	3.1975	0.1504	2.5159	69.6	3.2459	0.00
POLYCARBONATE	0.12860	3.3288	0.1606	2.6225	73.1	3.3201	0.00
POLYCHLOROSTYRWNE	0.07530	3.5441	0.1238	2.9241	81.7	3.4659	0.00
POLYETHYLENE	0.12108	3.4292	0.1370	2.5177	57.4	3.0016	0.00
MYLAR	0.12679	3.3076	0.1562	2.6507	78.7	3.3262	0.00
LUCITE	0.11433	3.3836	0.1824	2.6681	74.0	3.3297	0.00
POLYOXYMETHYLENE	0.10808	3.4002	0.1584	2.6838	77.4	3.2514	0.00
POLYPROPYLENE	0.15045	3.2855	0.1534	2.4822	59.2	3.1252	0.00
POLYSTYRENE	0.16454	3.2224	0.1647	2.5031	68.7	3.2999	0.00
TEFLON	0.10606	3.4046	0.1648	2.7404	99.1	3.4161	0.00
POLYTRIFLUOROCHLOROETHY.	0.07727	3.5085	0.1714	3.0265	120.7	3.8551	0.00
POLYVINYL ACETATE	0.11442	3.3762	0.1769	2.6747	73.7	3.3309	0.00
POLYVINYL ALCOHOL	0.11178	3.3893	0.1401	2.6315	69.7	3.1115	0.00
POLYVINYL BUTYRAL	0.11544	3.3983	0.1555	2.6186	67.2	3.1865	0.00
POLYVINYL CHLORIDE	0.12438	3.2104	0.1559	2.9415	108.2	4.0532	0.00
SARAN	0.15466	3.1020	0.1314	2.9009	134.3	4.2506	0.00
PLOYVINYLIDENE FLUORIDE	0.10316	3.4200	0.1717	2.7375	88.8	3.3793	0.00
POLYVINYL PYRROLIDONE	0.12504	3.3326	0.1324	2.5867	67.7	3.1017	0.00

Table 2.3 (cont.)

LABEL	a	m_s	x_0	x_1	I(eV)	$-$C	δ_0
POTASSIUM IODINE	0.22053	2.7558	0.1044	3.3442	431.9	6.1088	0.00
POTASSIUM OXIDE	0.16789	3.0121	0.0480	3.0110	189.9	4.6463	0.00
PROPANE	0.09916	3.5920	1.4326	3.7998	47.1	8.7878	0.00
PROPANE, LIQUID	0.10329	3.5620	0.2861	2.6568	52.0	3.5529	0.00
N-PROPYL ALCOHOL	0.09644	3.5415	0.2046	2.6681	61.1	3.2915	0.00
PYRIDINE	0.16399	3.1977	0.1670	2.5245	66.2	3.3148	0.00
RUBBER, BUTYL	0.12108	3.4296	0.1347	2.5154	56.5	2.9915	0.00
RUBBER, NATURAL	0.15058	3.2879	0.1512	2.4815	59.8	3.1272	0.00
RUBBER, NEOPRENE	0.09763	3.3632	0.1501	2.9461	93.0	3.7911	0.00
SIO2	0.08408	3.5064	0.1385	3.0025	139.2	4.0029	0.00
AGBR	0.24582	2.6820	0.0352	3.2109	486.6	5.6139	0.00
AGCL	0.22968	2.7041	-0.0139	3.2022	398.4	5.3437	0.00
SILVER HALIDES IN EMUL.	0.24593	2.6814	0.0353	3.2117	487.1	5.6166	0.00
SILVER IODIDE	0.25059	2.6572	0.0148	3.2908	543.5	5.9342	0.00
SKIN (ICRP)	0.09459	3.4643	0.2019	2.7526	72.7	3.3546	0.00
SODIUM CARBONATE	0.08715	3.5638	0.1287	2.8591	125.0	3.7178	0.00
NAI	0.12516	3.0398	0.1203	3.5920	452.0	6.0572	0.00
SODIUM MONOXIDE	0.07501	3.6943	0.1652	2.9793	148.8	4.1892	0.00
SODIUM NITRATE	0.09391	3.5097	0.1534	2.8221	114.6	3.6502	0.00
STILBENE	0.16659	3.2168	0.1734	2.5142	67.7	3.3680	0.00
SUCROSE	0.11301	3.3630	0.1341	2.6558	77.5	3.1526	0.00
TRRPHENYL	0.14964	3.2685	0.1322	2.5429	71.7	3.2639	0.00
TESTES (ICRP)	0.08533	3.5428	0.2274	2.7988	75.0	3.4698	0.00
TETRACHLOROETHTLENE	0.18595	3.0156	0.1713	2.9083	159.2	4.6619	0.00
THALLIUM CHLORIDE	0.18599	2.7690	0.0705	3.5716	690.3	6.3009	0.00
TISSUE, SOFT (ICRP)	0.08926	3.5110	0.2211	2.7799	72.3	3.4354	0.00
ICRU FOUR-COMP. TISSUE	0.09629	3.4371	0.2377	2.7908	74.9	3.5087	0.00
TISSUE-EQ. GAS (METHANE)	0.09946	3.4708	1.6442	4.1399	61.2	9.9500	0.00
TISSUE-EQ. GAS (PROPANE)	0.09802	3.5159	1.5139	3.9916	59.5	9.3529	0.00
TITANIUM DIOXIDE	0.08569	3.3267	-0.0119	3.1647	179.5	3.9522	0.00
TOLUEN	0.13284	3.3558	0.1722	2.5728	62.5	3.3026	0.00
TRICHLOROETHYLENE	0.18272	3.0137	0.1803	2.9140	148.1	4.6148	0.00
TRIETHYL PHOSPHATE	0.06922	3.6302	0.2054	2.9428	81.2	3.6242	0.00
TUNGSTEN HEXAFLUORIDE	0.03658	3.5134	0.3020	4.2602	354.4	5.9881	0.00
URANIUM DICARBIDE	0.21120	2.6577	-0.2191	3.5208	752.0	6.0247	0.00
URANIUM MONOCARBIDE	0.22972	2.6169	-0.2524	3.4941	862.0	6.1210	0.00
URANIUM OXIDE	0.20463	2.6711	-0.1938	3.5292	720.6	5.9605	0.00
UREA	0.11609	3.3461	0.1603	2.6525	72.8	3.2032	0.00
VALINE	0.11386	3.3774	0.1441	2.6227	67.7	3.1059	0.00
VITON	0.09965	3.4556	0.2106	2.7874	98.6	3.5943	0.00
H2O	0.09116	3.4773	0.2400	2.8004	75.0	3.5017	0.00
H2O-VAPOR	0.08101	3.5901	1.7952	4.3437	71.6	10.5962	0.00
XYLENE	0.13216	3.3564	0.1695	2.5675	61.8	3.2698	0.00

The density effect has been treated extensively by Sternheimer [159, 160, 161, 162, 163], Sternheimer and Peierls[165], and recently by Berger, Seltzer, and Sternheimer in various collabora-

tions [20, 152, 151, 166, 164]. According to Sternheimer, the density effect is given by the formulas

$$\delta = \begin{cases} 0 & \text{if } x < x_0 \\ 2(\ell n \ 10)x + C + a(x_1 - x)^{m_s} & \text{if } x \in (x_0, x_1) \\ 2(\ell n \ 10)x + C & \text{if } x > x_1 \end{cases} \tag{2.271}$$

where

$$x = \log_{10}(\check{p}c/m) = \ell n \ \eta/\ell n \ 10, \tag{2.272}$$

$$C = -2 \ \ell n(\bar{I}_{adj}/h\nu_P) - 1, \tag{2.273}$$

$$\nu_P = \text{plasma frequency} = \sqrt{nr_0^2 c^2/\pi} \tag{2.274}$$

and x_0, x_1, a, m_s are parameters obtained by fits made to values of δ as a function of energy which were explicitly calculated from atomic oscillator strengths for specific materials. Two sets of parameters are available [164, 166], both of which derive from the same program for calculating δ [20]. As of 1995, PEGS has employed the more extensive data from [164], which includes 98 elements and 180 compounds. A list of the available materials and their parameters is found in Table 2.3. The composition of the materials is given elsewhere (*e.g.*, Table 7 of Seltzer and Berger[151]).

Since there are such a variety of density effect parameters available, version 4 of PEGS was modified to allow the user to override the PEGS defaults and to specify directly the six parameters needed for any material being considered. However, as the six parameters used in PEGS merely parameterize δ, the stopping powers calculated with these values will be slightly different ($\leq \pm 0.5\%$) from those calculated directly from Berger and Seltzer's original δ values (*e.g.*, see Figure 1 of Seltzer and Berger[152]). Because some benchmarking application require stopping power values with extremely fine precision (see, *e.g.*, [101] or [137]), a further override option was developed for PEGS by Duane *et al.*[52] in which the values of δ and $\bar{I}_{adj}$ used in [20] can be used in PEGS. When this option is invoked (through the PEGS input flag `EPSTFL`), PEGS reproduces exactly values of collision stopping power found in ICRU Report [79]. Provided with the implementation of this option in PEGS are values of δ and $\bar{I}_{adj}$ for 100 elements and 345 compounds.

For materials not included in Table 2.3 or for which the density effect parameters or values are not input, the general prescription of Sternheimer and Peierls[165] is used as follows:

1. The value of $\bar{I}_{adj}$ is determined as described above.

2. m_s is always taken as 3.

3. C is given by Equations 2.273 and 2.274.

4. For solids and liquids:

   (a) If $\bar{I}_{adj} < 100$ eV, then $x_1 = 2.0$ and
       $x_0 = 0.2$ when $-C < 3.681$ or $x_0 = -0.326C - 1$ otherwise.
   (b) If $\bar{I}_{adj} > 100$ eV, then $x_1 = 3.0$ and
       $x_0 = 0.2$ if $-C < 5.215$ or $x_0 = -0.326C - 1.5$ otherwise.

84

5. For gases:

   (a) If $-C < 10.0$ then $x_0 = 1.6$
   and $x_1 = 4.0$.

   (b) If $10.0 \leq -C < 10.5$ then $x_0 = 1.7$
   and $x_1 = 4.0$.

   (c) If $10.5 \leq -C < 11.0$ then $x_0 = 1.8$
   and $x_1 = 4.0$.

   (d) If $11.0 \leq -C < 11.5$ then $x_0 = 1.9$
   and $x_1 = 4.0$.

   (e) If $11.5 \leq -C < 12.25$ then $x_0 = 2.0$
   and $x_1 = 4.0$.

   (f) If $12.25 \leq -C < 13.804$ then $x_0 = 2.0$
   and $x_1 = 5.0$.

   (g) If $-C \geq 13.804$ then $x_0 = -0.326C - 2.5$
   and $x_1 = 5.0$.

6. $a = [-C - (2\ell n\ 10)x_0]/(x_1 - x_0)^3$.

The final stage in determining the density effect correction for gases is to scale the results in terms of the gas pressure since the tabulated values are all given for NTP (0°C, 760mm Hg). If the pressure correction factor is

GASP = actual pressure/pressure at NTP

then

$$C(GASP) = C(NTP) + \ell n(GASP)$$

and

$$x_0(GASP) = x_0(NTP) - \tfrac{1}{2}\log_{10}(GASP)$$

$$x_1(GASP) = x_1(NTP) - \tfrac{1}{2}\log_{10}(GASP) .$$

PEGS routine `SPINIT` initializes the stopping power routines for a particular medium. Routine `SPIONB`, which is a function of $\breve{E}_0$, `AE`, and `POSITR`, evaluates Equation 2.257 for a positron if `POSITR` is true, and for a electron if `POSITR` is false.

Note that limitations on the applicability of EGS5 for simulating electron transport at low energies derive primarily from the breakdown of the Bethe-Bloch formula for the stopping power. The presence in Equation 2.257 of a term containing the logarithm of the ratio of the electron energy and the mean ionization energy implies that there must exist some energy (dependent on

Z) at which the Bethe-Bloch expression will yield physically unrealistic negative values. In addition, Equation 2.257 was derived under approximations which are strictly valid for high energy particles, large energy losses and large recoil energies, so the accuracy of the Bethe-Bloch formula begins to wane at energies ranging from around 1 keV for low Z materials to several tens of keV for high Z materials.

As a final comment on continuous energy loss, we note that when an electron is transported a given distance, it is assumed that its energy loss due to sub-cutoff collisions is equal to the distance traveled times the mean loss per unit length as evaluated using Equation 2.253. In actuality, the energy loss over a transported distance is subject to fluctuations and gives rise to a restricted Landau distribution. Fluctuations due to discrete interactions can be properly accounted for in EGS in most applications by setting the cutoff energy for charged particle production sufficiently low. For example, for 20 MeV electrons passing through a thin slab of water, Rogers[136] has shown that the energy straggling predicted by considering all secondaries down to 1 keV (*i.e.*, AE=512 keV) is in good agreement with the Blunck-Leisegang modification of the energy loss straggling formalism of Landau (except for large energy loss events where the modified Landau distribution appears to be wrong). The creation of large numbers of low-energy secondary electrons, even when they are discarded immediately, adds significant inefficiency to EGS shower simulations, however, and so a correct energy loss fluctuation model is currently under development for EGS5.

2.14   Multiple Scattering

When an electron passes through matter, it undergoes a large number of elastic collisions with the atomic nuclei. These have the effect of changing the electron's direction, but do not significantly change its energy. As noted previously, the number of such collisions is so great that direct simulation of individual scattering events is almost never practical[8]. Instead, elastic scattering is typically treated by bundling large numbers of collisions together into large "steps" and then assuming that the electron transport over these larger steps can be characterized by the particle's longitudinal translation and lateral displacement during the step plus its aggregate scattering angle Θ over the step (the azimuthal angle is assumed to be uniform), which is taken from an appropriate "multiple scattering" distribution function. The details of the current implementation of this approach are provided in the next section. The remainder of this section is devoted to discussion of multiple scattering probability density function (p.d.f.) for Θ. EGS5 currently offers two choices for the multiple scattering p.d.f., one based on the theory of Molière, and a new one based on the approach of Goudsmit and Saunderson (GS) [63, 64].

---

[8]Even in PENELOPE [14], which was originally developed for modeling the transport of electrons at low energies, only large angle collisions are treated explicitly.

2.14.1 The Molière Multiple Scattering Distribution

All previous versions of EGS have treated electron elastic scattering using Molière's [107]theory of multiple scattering as formulated by Bethe [23]. The details of computing multiple scattering in mixtures, and a good introduction to the subject is given in the the review article by Scott [147], to which we make frequent reference in the discussion that follows.

In Versions 1 and 2 of EGS the method of sampling scattering angles was based on a scheme of Nagel's whereby one of 29 discrete representative reduced angles was selected and then used to obtain the real scattering angle. EGS3 departed from this scheme to use a method similar to that of Messel and Crawford[103], whereby the scattering angles are chosen in a truly continuous way. This method also allows us to transport over variable step lengths (usually denoted by t) while still taking multiple scattering properly into account.

The cross section for elastic scattering off the nucleus is proportional to Z^2. Scattering from atomic electrons is taken into account by replacing Z^2 by $Z(Z + \xi_{MS})$. Scattering from atomic electrons which results in discrete delta-rays is already properly taken to account, so the ξ_{MS} need only account for the sub-cutoff scatterings. Scott[147] has outlined the procedures for taking into account scattering from atomic electrons in a more rigorous way, but we have not implemented it here. Instead we treat ξ_{MS} more as a "fudge factor" to get our multiple scattering as consistent with experiment as possible. In the developments to follow we shall need the parameters

$$Z_S = \sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_{MS}) \ , \tag{2.275}$$

$$Z_E = \sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_{MS}) \ell n \ Z_i^{-2/3} \ , \tag{2.276}$$

and

$$Z_X = \sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_{MS}) \ell n \left[1 + 3.34(\alpha Z_i)^2 \right] \ . \tag{2.277}$$

One of the advantages of Molière's theory is that the energy-dependent p.d.f. of Θ can be expressed in terms of an energy-independent p.d.f. of a reduced angle θ, where

$$\theta = \frac{\Theta}{\chi_c B^{1/2}} \ , \tag{2.278}$$

and where χ_c and B are parameters that depend on energy, material, and pathlength t. The p.d.f. of Θ is given by

$$f(\Theta) = f_M(\Theta)(\sin\Theta/\Theta)^{1/2} \ , \tag{2.279}$$

which is like Bethe's formula (58) except that we define our $f_M(\Theta)$ to be their $f_M(\Theta)$ times Θ; that is, we include the phase space factor in ours. The factor $(\sin\Theta/\Theta)^{1/2}$ is less than one and is used as a rejection function to correct the Molière distribution at large angles. In addition, we reject all

sampled $\Theta > 180°$. The p.d.f. $f_M(\Theta)$ is sampled by first sampling θ, the reduced angle, from its p.d.f., $f_r(\theta)$, and then using Equation 2.278 to get Θ. This is equivalent to saying that

$$f_M(\Theta)d\Theta = f_r(\theta)d\theta .\qquad(2.280)$$

For the reduced angle p.d.f. we use the first three terms of Bethe's Equation 25; namely,

$$f_r(\theta) = \left[f^{(0)}(\theta) + \frac{1}{B}f^{(1)}(\theta) + \frac{1}{B^2}\ f^{(2)} \right](\theta) .\qquad(2.281)$$

The general formula for the $f^{(i)}$ is (Bethe, Equation 26)

$$f^{(n)}(\theta) = (n!)^{-1} \int_0^\infty udu\ J_0(\theta u) \times \exp(-u^2/4) \left[1/4\ u^2\ \ell n\ (u^2/4) \right]^n .\qquad(2.282)$$

For $n = 0$ this reduces to

$$f^{(0)}(\theta) = 2e^{-\theta^2} .\qquad(2.283)$$

Instead of using the somewhat complicated expressions when $n = 1$ and 2, we have elected to use a) the numerical values presented in Bethe's paper (for 29 selected values of θ from 0 to 10), b) the fact that $f^{(i)}(\theta)$ behaves as θ^{-2i-2} for large θ, and c) the fact that $f^{(1)}(\theta)$ goes over into the single scattering law at large θ. That is,

$$\lim_{\theta\to\infty}\ f^{(1)}(\theta)\theta^4 = 2.\qquad(2.284)$$

This also implies that

$$\lim_{\theta\to\infty}\ f^{(2)}(\theta)\theta^4 = 0.\qquad(2.285)$$

The $f^{(i)}(\theta)$ functions are not needed in EGS directly, but rather PEGS needs the $f^{(i)}(\theta)$ to create data that EGS does use. Let

$$\eta = 1/\theta\qquad(2.286)$$

and

$$f_\eta^{(i)}(\eta) = f^{(i)}(1/\eta)\eta^{-4} = f^{(i)}(\theta(\eta))\theta(\eta)^4 .\qquad(2.287)$$

As a result of Equations 2.285 and 2.286 we see that $f_\eta^{(1)}(0) = 2$ and $f_\eta^{(2)}(0) = 0$. We now do a cubic spline fit to $f^{(i)}(\theta)$ for $\theta\epsilon(0,10)$ and $f_\eta^{(i)}(\eta)$ for $\eta\epsilon(0,5)$. If we use $\hat{f}^{(i)}(\theta)$ and $\hat{f}_\eta^{(i)}(\eta)$ to denote these fits, then we evaluate the $f^{(i)}(\theta)$ as

$$f^{(i)}(\theta) = \left(\hat{f}^{(i)}(\theta) \text{ if } \theta < 10,\ \frac{1}{\theta^4}\ \hat{f}_\eta^{(i)}(1/\theta) \right) .\qquad(2.288)$$

Similarly if we want $f_\eta^{(i)}(\eta)$ for arbitrary η we use

$$f_\eta^{(i)}(\eta) = \left(\hat{f}_\eta^{(i)}(\eta) \text{ if } \eta < 5,\ \frac{1}{\eta^4}\ \hat{f}^{(i)}(1/\eta) \right) .\qquad(2.289)$$

88

To complete the mathematical definition of $f(\Theta)$ we now give additional formulas for the evaluation of χ_c and B. We have

$$B - \ell n\ B = b, \tag{2.290}$$

$$b = \ell n\ \Omega_0, \tag{2.291}$$

$$\Omega_0 = b_c t/\beta^2, \tag{2.292}$$

$$b_c = \frac{`6680'\rho Z_S e^{Z_E/Z_S}}{M e^{Z_X/Z_S}} \tag{2.293}$$

(Note, PEGS computes $\breve{b}_c = X_0 b_c$),

$$`6680' = 4\pi N_a \left(\frac{\hbar}{m_e c}\right)^2 \left[\frac{(0.885)^2}{1.167 \times 1.13}\right] = 6702.33, \tag{2.294}$$

$$\rho = \text{material mass density (g/cm}^3)\ , \tag{2.295}$$

$$M = \text{molecular weight} = \sum_{i=1}^{N_e} p_i A_i, \tag{2.296}$$

$$\chi_c = \frac{\chi_{cc}\sqrt{t}}{\breve{E}_{MS}\beta^2}, \tag{2.297}$$

$$\chi_{cc} = \frac{`22.9'}{(180/\pi)}\sqrt{\frac{\rho Z_S}{M}}(cm^{-1/2}\ MeV) \tag{2.298}$$

$$\left(\text{Note, PEGS computes } \chi_{cc} = \chi_{cc}\sqrt{X_0}(r.\ell.^{-1/2}\ MeV)\right)\ ,$$

$$`22.9' = (180/\pi)\sqrt{4\pi N_a}\ r_0 m = 22.696\ (cm\ MeV)\ . \tag{2.299}$$

$\breve{E}_{MS}$ is the energy (in MeV) of the electron that is scattering and may be set equal to the energy at the beginning or end of the step (or something in between) to try to account for ionization loss over the step. Equations 2.297 through 2.299 are based on formula 7.4 of Scott[147] which is equivalent to

$$\chi_c^2 = \frac{N_a \rho}{M}\ 4\pi r_0^2 \left[\sum_{i=1}^{N_e} p_i Z_i(Z_i + \xi_{MS})\right] \int_0^t \frac{m^2 dt'}{\breve{E}(t')^2 \beta(t')^4}\ . \tag{2.300}$$

From this we see that, to be proper, we should replace $\sqrt{t}/\breve{E}_{MS}\beta^2$ using

$$\frac{\sqrt{t}}{\breve{E}_{MS}\beta^2} = \left(\int_0^t \frac{dt'}{\breve{E}(t')^2 \beta(t')^4}\right)^{1/2}\ , \tag{2.301}$$

where $\breve{E}(t')$ is the particle's energy (in MeV) after going a distance t' along its path. Likewise, $\beta(t') = \sqrt{1 - m^2/\breve{E}(t')^2}$ is the particle's velocity, at the same point, divided by the speed of light. We assume that our steps are short enough and the energy high enough that Equations 2.297 through 2.299 are sufficiently accurate.

For completeness, we give a derivation of Equations 2.292. We start with the definition of Ω_0, (which differs somewhat from Scott's definition),

$$\Omega_0 \equiv e^b \ . \tag{2.302}$$

According to Bethe's formula (22)

$$e^b = \frac{\chi_c^2}{\chi_{\alpha'}^2} = \frac{\chi_c^2}{`1.167'\chi_\alpha^2} \ . \tag{2.303}$$

From the derivation in Bethe it is seen that

$$`1.167' = e^{2C-1} \tag{2.304}$$

where

$$C = 0.577216 \text{ is Euler's constant.} \tag{2.305}$$

Scott's formula (7.25) for χ_α is

$$\chi_c^2 \ \ell n \ \chi_\alpha = 4\pi \int_0^t \frac{dt'}{k^2(t')} \ \sum_{i=1}^{N_e} \ N_i \alpha_i^2 \left[\ell n \ \chi_{\alpha_1} + \frac{\ell n \ \chi_i^{e\ell}}{Z_i} \right] \ . \tag{2.306}$$

where

$$
\begin{aligned}
k &= p/\hbar, & \text{(2.307)} \\
p &= \text{particle momentum,} \\
N_i &= \frac{N_a\rho}{M} p_i = \text{density of atoms of type } i, & \text{(2.308)} \\
\alpha_i &= \alpha Z_i/\beta = Z_i e^2/\hbar v \ , & \text{(2.309)} \\
\chi_{\alpha_i} &= \text{the screening angle for atoms of type } i, \\
&= \left[\chi_0^2(1.13 + 3.76\alpha_i^2)\right]^{1/2} \ , & \text{(2.310)} \\
\chi_0 &= \frac{\lambdabar_0}{r_{\text{TF}}} \ , & \text{(2.311)} \\
\lambdabar_0 &= \lambda_0/2\pi = \hbar/p = \text{wavelength of electron}/2\pi, & \text{(2.312)} \\
r_{\text{TF}} &= \text{Thomas} - \text{Fermi radius of atom} \\
&= 0.885 \ a_0 \ Z_i^{-1/3} \ , & \text{(2.313)} \\
a_0 &= \text{Bohr radius} = \hbar^2/m_e e^2 & \text{(2.314)} \\
\chi_i^{e\ell} &= \text{screening angle for the atomic} \\
&\quad \text{atoms of type } i.
\end{aligned}
$$

The next step is to let $\beta = 1$ in the α_i that are in the χ_{α_i}, to delete the term with $\chi_i^{e\ell}$, and to let $Z_i^2 \rightarrow Z_i(Z_i + \xi_{MS})$. Recalling that $p = E\beta/c$, Equation 2.306 now becomes

$$\chi_c^2 \ \ell n \ \chi_\alpha = \frac{N_a\rho}{M} \ 4\pi e^4 \left[\sum_{i=1}^{N_e} p_i Z_i(Z_i + \xi_{MS}) \ \ell n \ \chi_{\alpha_i} \right] \int_0^t \frac{dt'}{E^2\beta^4} \ . \tag{2.315}$$

Since $e^2 = r_0 m_e c^2$, and using Equation 2.301, we obtain

$$\chi_c^2 \, \ell n \, \chi_\alpha = \frac{N_a \rho}{M} \, 4\pi r_0^2 \left[\sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_{MS}) \, \ell n \, \chi_{\alpha_i} \right] \frac{m^2 t}{\breve{E}_{MS}^2 \beta^4} \, . \tag{2.316}$$

Dividing by Equation 2.300 and multiplying by 2, we get

$$\ell n \, \chi_\alpha^2 = \left[\sum_{i=1}^{N_e} p_i Z_i (Z_i + \xi_{MS}) \, \ell n \, \chi_{\alpha_i}^2 \right] Z_S^{-1} \, . \tag{2.317}$$

But using Equations 2.310 through 2.314,

$$\ell n \, \chi_{\alpha_i}^2 = \ell n \, \chi_0^2 + \ell n \left[1.13 + 3.76 (\alpha Z_i)^2 \right] \, , \tag{2.318}$$

$$\ell n \, \chi_0^2 = \ell n \left[\frac{\hbar^2 m_e^2 e^4}{p^2 (0.885)^2 \hbar^4 Z_i^{-2/3}} \right] = \ell n \left[\frac{m_e^2 e^4}{p^2 \hbar^2 (0.885)^2} \right] - \ell n \, Z_i^{-2/3} \, , \tag{2.319}$$

$$\ell n \left[1.13 + 3.76 (\alpha Z_i)^2 \right] = \ell n \, 1.13 + \ell n [1 + 3.34 (\alpha Z_i)^2] \, . \tag{2.320}$$

Hence,

$$\ell n \, \chi_\alpha^2 = \left[Z_S \, \ell n \left(\frac{m_e^2 e^4 \, 1.13}{p^2 \hbar^2 (0.885)^2} \right) + Z_X - Z_E \right] Z_S^{-1} \, , \tag{2.321}$$

so that

$$\chi_\alpha^2 = \frac{m_e^2 e^4 \, 1.13 \, e^{Z_X/Z_S}}{p^2 \hbar^2 (0.885)^2 e^{Z_E/Z_S}} \, , \tag{2.322}$$

Now, recalling that Equations 2.297 through 2.299 are equivalent to

$$\chi_c^2 = \frac{N_a \rho}{M} \, 4\pi r_0^2 Z_S t \left(m_e^2 c^2 / E^2 \beta^4 \right) \, , \tag{2.323}$$

and using Equations 2.302, 2.303, 2.322 and 2.323, we obtain

$$
\begin{aligned}
\Omega_0 &= \frac{\frac{N_a \rho}{M} \, 4\pi r_0^2 Z_S t m_e^2 c^4 (E^2 \beta^2 / c^2) \hbar^2 (0.885)^2 e^{Z_E/Z_S}}{m_e^2 (r_0^2 m_e^2 c^4) e^{Z_X/Z_S} E^2 \beta^4 (1.167)(1.13)} \\
&= \text{`6680'} \left[\frac{\rho Z_S e^{Z_E/Z_S}}{M e^{Z_X/Z_S}} \right] \frac{t}{\beta^2} \\
&= b_c t / \beta^2 \, . \qquad \text{Q.E.D.}
\end{aligned}
\tag{2.324}
$$

Molière's B parameter is related to b by the transcendental Equation 2.290. For a given value of b, the corresponding value of B may be found using Newton's iteration method. As a rough estimate, $B = b + \ell n \, b$. It can be seen that b, and hence B, increases logarithmically with increasing transport distance.

The intuitive meaning of Ω_0 is that it may be thought of as the number of scatterings that take place in the slab. If this number is too small, then the scattering is not truly multiple scattering and
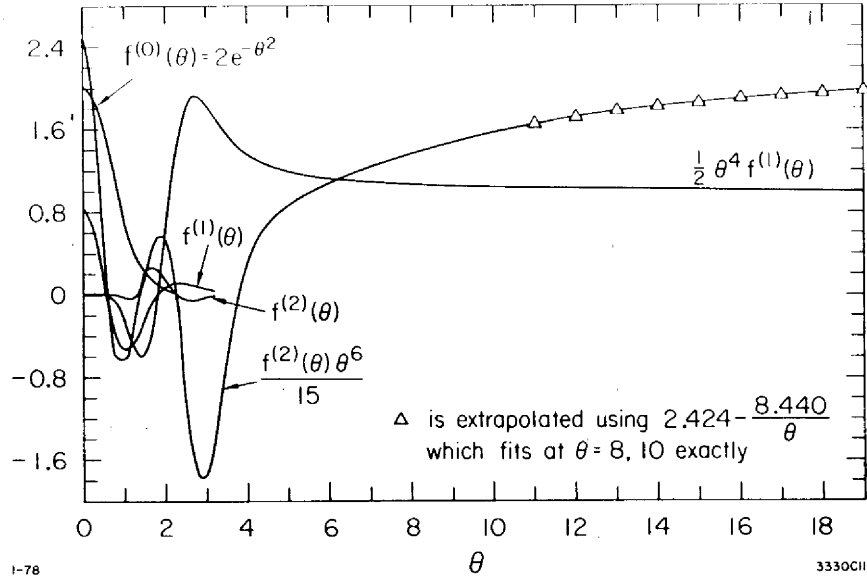
Figure 2.6: Plots of Molière functions $f^{(0)}$, $f^{(1)}$, and $f^{(2)}$.

various steps in Molière's derivation become invalid. In Molière's original paper[107], he considered his theory valid for

$$\Omega_0(\text{his } \Omega_b) \geq 20 \ , \tag{2.325}$$

which corresponds to

$$B \geq 4.5, \quad \text{and} \quad b \geq 3 \ . \tag{2.326}$$

From Equations 2.325 and 2.325 we arrive at the condition

$$t/\beta^2 \geq 20/b_c = (t_{eff})_0 \ .$$

Another restriction on the validity of Equation 2.279 is mentioned by Bethe[23], namely,

$$\chi_c^2 B < 1 \ . \tag{2.327}$$

Resuming our presentation of the method used to sample Θ, we return now to the problem of sampling θ from $f_r(\theta)$ given by Equation 2.281. It might at first appear that $f_r(\theta)$ is already decomposed into sub-distribution functions. However, $f^{(1)}(\theta)$ and $f^{(2)}(\theta)$ are not always positive, and thus, are not candidate distribution functions. Graphs of $f^{(1)}(\theta)$ and $f^{(2)}(\theta)$ are shown in Figure 2.6. We now adopt a strategy similar to that used by Messel and Crawford[103]; namely, mix enough of $f^{(0)}(\theta)$ with $f^{(1)}(\theta)$ and $f^{(2)}(\theta)$ to make them everywhere positive. Unlike Messel and Crawford, who dropped the term involving $f^{(2)}(\theta)$, we have been able to retain all of the first three terms in the expansion.

92

The factorization we use is

$$f_r(\theta) = \sum_{i=1}^{3} \alpha_i f_i(\theta) g_i(\theta) \ , \tag{2.328}$$

where

$$\alpha_1 = 1 - \lambda/B \ , \tag{2.329}$$

$$f_1(\theta) = 2e^{-\theta^2}\theta \quad \text{for} \quad \theta\epsilon(0,\infty) \ , \tag{2.330}$$

$$g_1(\theta) = 1 \ , \tag{2.331}$$

$$\alpha_2 = \mu g_{2,Norm}/B \ , \tag{2.332}$$

$$f_2(\theta) = 1/\mu \quad \text{for} \quad \theta\epsilon(0,\mu) \ , \tag{2.333}$$

$$g_2(\theta) = \frac{\theta}{g_{2,Norm}} \left(\lambda f^{(0)}(\theta) + f^{(1)}(\theta) + f^{(2)}(\theta)/B \right), \tag{2.334}$$

$$\alpha_3 = g_{3,Norm}/2\mu^2 B \ , \tag{2.335}$$

$$f_3(\theta) = 2\mu^2\theta^{-3} \quad \text{for} \quad \theta\epsilon(\mu,\infty) \ , \tag{2.336}$$

$$g_3(\theta) = \frac{\theta^4}{g_{3,Norm}} \left(\lambda f^{(0)}(\theta) + f^{(1)}(\theta) + f^{(2)}(\theta)/B \right) \ . \tag{2.337}$$

When the third sub-distribution function is selected, we first sample $\eta = 1/\theta$ using $f_{\eta 3}(\eta)$ and $g_{\eta 3}(\eta)$ given by

$$f_{\eta 3}(\eta) = 2\mu^2\eta \quad \text{for} \quad \eta\epsilon(0,1/\mu) \ , \tag{2.338}$$

$$g_{\eta 3}(\eta) = \frac{\eta^{-4}}{g_{3,Norm}} \left(\lambda f^{(0)}(1/\eta) + f^{(1)}(1/\eta) + f^{(2)}(1/\eta)/B \right) \ . \tag{2.339}$$

Then we let $\theta = 1/\eta$.

As presented above, this scheme contains four parameters, λ, μ, $g_{2,Norm}$ and $g_{3,Norm}$; the latter two are so chosen that $g_2(\theta)$ and $g_{\eta 3}(\eta)$ have maximum values (over the specified ranges) which are not greater than 1. The first sub-distribution is the Gaussian (actually exponential in θ^2) distribution that dominates for large B (thick slabs). The third sub-distribution represents the " single scattering tail." The second sub-distribution can be considered as a correction term for central θ values. The parameter μ separates the central region from the tail. The parameter λ determines the admixture of $f^{(0)}$ in the second and third sub-distribution functions. It must be large enough to ensure that $g_2(\theta)$ and $g_3(\theta)$ are always positive. It will also be noted that α_1 becomes negative if $B < \lambda$ so that this case must be specifically treated. After studying the variation of the theoretical sampling efficiency with the variation of these parameters, the values

$$\lambda = 2, \quad \mu = 1, \ g_{2,Norm} = 1.80, \ g_{3,Norm} = 4.05 \tag{2.340}$$

were chosen. These values do not give the absolute optimum efficiency, but the optimum μ values were usually close to one, so we chose $\mu = 1$ for simplicity. λ could not have been chosen much lower while still maintaining positive rejection functions. Furthermore it was desired to keep λ as low as possible since this would allow Molière's distribution to be simulated for as low values of

B as possible. Although Molière's theory becomes less reliable for $B < 4.5$, it was felt that it was probably as good an estimate as could easily be obtained even in this range.

Since $\alpha_1 < 0$ for $B < \lambda$, some modification of the scheme must be devised in this case. What we have done is to use the computed values of B in computing $\chi_c\sqrt{B}$, but for sampling we set '$1/B$' = '$1/\lambda$'. This has the effect of causing the Gaussian not to be sampled.

Our next point is best made by means of Figure 2.7 which is a graph of Equation 2.290, the transcendental equation relating B and b. It will be observed that when viewed as defining a function of b the resulting function is double valued. We of course reject the part of the curve for $B < 1$. We would, however, like to have a value of B for any thickness of transport distance (i.e., any value of b). In order to obtain a smooth transition to zero thickness we join a straight line from the origin, $(B = 0, \ b = 0)$, to the point on the curve $(B = 2, \ b = 2 - \ell n\ 2)$. B is then determined by

$$B = \left\{ \begin{array}{l} \frac{2}{2-\ell n 2}\ b \quad \text{if } b < 2 - \ell n\ 2, \\ \text{the } B > 1 \text{ satisfying } B - \ell n\ B = b, \\ \quad\quad \text{if } b > 2 - \ell n 2\ . \end{array} \right\} \tag{2.341}$$

For rapid evaluation, B has been fit using a piecewise quadratic fit for $b\epsilon(2, 30)$; $b = 30$ corresponding roughly to a thickness of 10^7 radiation lengths, which should be sufficient for any application.

Actually, $b = 0$ does not correspond to $t = 0$, but rather to $t \approx 2 \times 10^{-6} X_0$. We nevertheless set $\theta = 0$ if $b \leq 0$. The case where $b\epsilon(0,3)$ is not too likely either, since $b = 3$ roughly corresponds to $t \approx 10^{-4} X_0$, and is not very important since the scattering angles should be small. However, Rogers[136] has found that in low energy applications, it is possible to take too small a step, thereby running into this constraint and effectively turning off the multiple scattering.

To complete our discussion on sampling we note that $f_1(\theta)$ is sampled directly by means of

$$\theta = \sqrt{-\ell n\ \zeta}\ . \tag{2.342}$$

The p.d.f. of $f_2(\theta)$ is sampled by merely choosing a uniformly distributed random number. The p.d.f. of $\hat{f}_3(\eta)$ is sampled by taking the larger of two uniformly distributed random numbers. Finally, $g_2(\theta)$ and $g_{\eta 3}(\eta)$ are divided into "B-independent" parts

$$g_2(\theta) = g_{21}(\theta) + g_{22}(\theta)/B \quad \text{for} \quad \theta\epsilon(0,1)\ , \tag{2.343}$$

$$g_{\eta 3}(\theta) = g_{31}(\eta) + g_{32}(\eta)/B \quad \text{for} \quad \eta\epsilon(0,1)\ . \tag{2.344}$$

The functions g_{21}, g_{22}, g_{31}, and g_{32} have been fit by PEGS over the interval (0,1) using a piecewise quadratic fit. This completes our discussion of the method used to sample θ.

Note that the Bethe condition, $\chi_c^2 B < 1$, places a limit on the length of the electron step size which can be accurately modeled using Molière's multiple scattering p.d.f. We can determine the maximum total step size consistent with this constraint, t_B, by starting with

$$\chi_c^2(t_B)B(t_B) = 1\ . \tag{2.345}$$
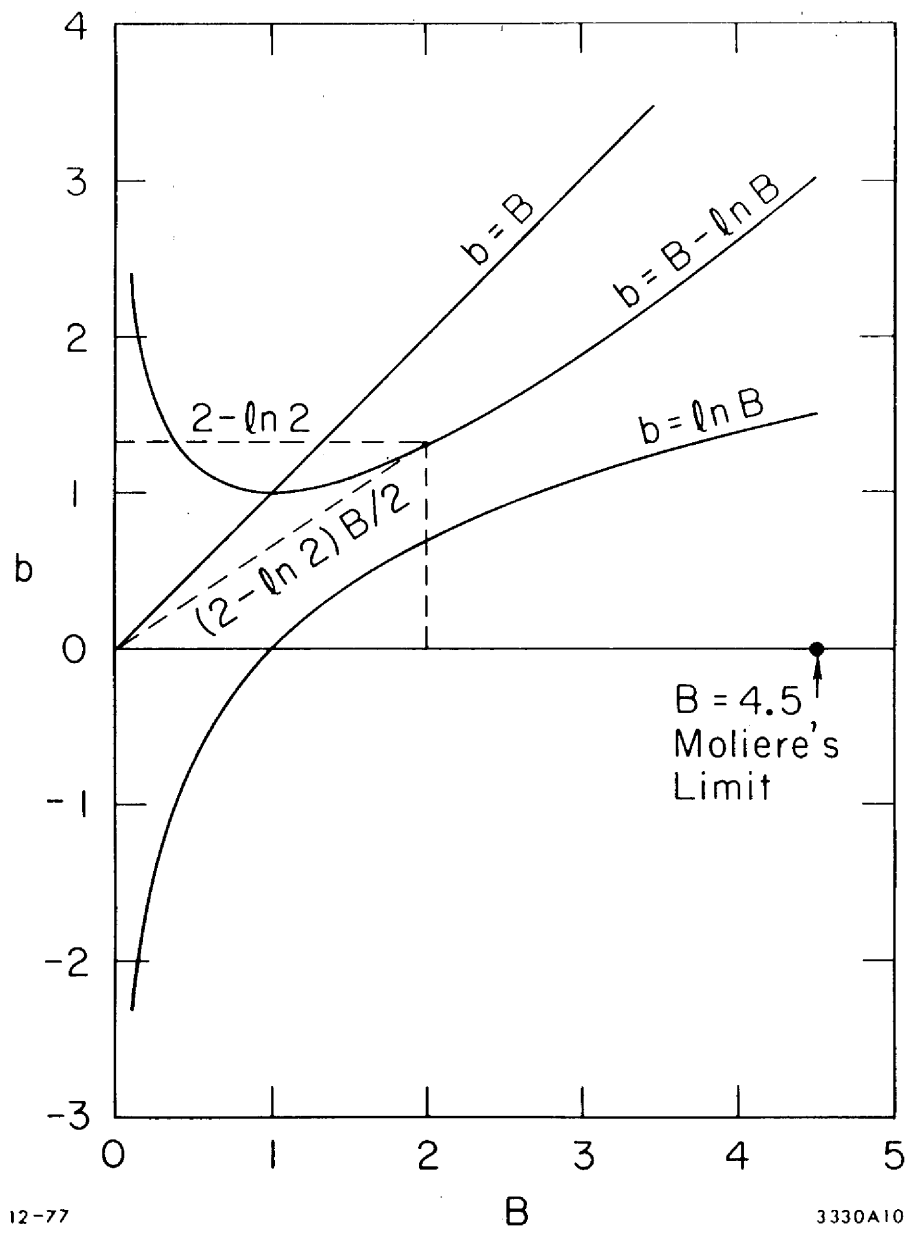
Figure 2.7: Plot of Equation 2.290 ($B - \ell n\ B = b$).

From Equation 2.290 we can write

$$e^b = e^B/B \ , \tag{2.346}$$

and using Equations 2.291, 2.292, 2.297 with Equations 2.345 and 2.346, we have

$$\frac{b_c t_B}{\beta^2} = \frac{\exp[(\breve{E}_{MS}\beta^2)^2/\chi_{cc}^2 t_B]\chi_{cc}^2 t_B}{(\breve{E}_{MS}\beta^2)^2} \ . \tag{2.347}$$

Solving for t_B we obtain

$$t_B = \frac{(\breve{E}_{MS}\beta^2/\chi_{cc})^2}{\ell n[b_c(\breve{E}_{MS}\beta/\chi_{cc})^2]} \ . \tag{2.348}$$

2.14.2   The Goudsmit-Saunderson Multiple Scattering Distribution

It has long been acknowledged that Molière's multiple scattering distribution breaks down under certain conditions in addition to that given above in 2.348. In particular: the basic form of the cross section assumed by Molière is in error in the MeV range, when spin and relativistic effects are important; various approximations in Molière's derivation lead to significant errors at pathlengths less than 20 elastic scattering mean free paths (recall Equation 2.325); and the form of Molière's cross section is incapable of accurately modeling the structure in the elastic scattering cross section at large angles for low energies and high atomic number. It is therefore desirable to have available a more exact treatment, and in EGS5, we use in the energy range from 100 eV to 100 MeV elastic scattering distributions derived from a state-of-the-art partial-wave analysis (unpublished work) which includes virtual orbits at sub-relativistic energies, spin and Pauli effects in the near-relativistic range and nuclear size effects at higher energies. Additionally, unlike the Molière formalism, this model includes explicit electron-positron differences in multiple scattering, which can be pronounced at low energies.

The multiple scattering distributions[9] are computed using the exact approach of Goudsmit and Saunderson (GS) [63, 64]. We consider electrons or positrons with kinetic energy $\breve{T}_0$ moving in a hypothetical infinite homogeneous medium, with $N_a\rho/M$ scattering centers per unit volume, in which they experience only elastic collisions. We assume that the single-scattering differential cross section (DCS), $d\sigma(\Theta)/d\Omega$, depends only on the polar scattering angle Θ, *i.e.* it is axially symmetrical about the direction of incidence. This assumption is satisfied as long as the scattering centers are spherically symmetrical atoms or randomly oriented molecules. Moreover, interference effects resulting from coherent scattering by several centers are assumed to be negligible. As a consequence, the theory is applicable only to amorphous materials and to polycrystalline solids. For the sake of simplicity, we limit our considerations to single-element materials; the generalization to compounds is straightforward (additivity approximation).

---

[9]The derivation given here is graciously provided by Dr. F. Salvat of Institut de Tècniques Energètiques at Universitat Politècnica de Catalunya, Barcelona, Spain. Dr. Salvat also provided the source code for computing the GS distribution which we have adapted for use in PEGS.

Denoting σ as the total elastic cross section, the angular distribution $f_1(\Theta)$ after a single scattering event is

$$f_1(\Theta) = \frac{1}{\sigma} \frac{\mathrm{d}\sigma(\Theta)}{\mathrm{d}\Omega}, \tag{2.349}$$

The probability of having a polar scattering angle between Θ and $\Theta + \mathrm{d}\Theta$ in a single collision is given by $2\pi f_1(\Theta) \sin\Theta \mathrm{d}\Theta$. It is convenient to write $f_1(\Theta)$ in the form of a Legendre series

$$f_1(\Theta) = \sum_{\ell=0}^{\infty} \frac{2\ell+1}{4\pi} F_\ell \, P_\ell(\cos\Theta), \tag{2.350}$$

where P_ℓ are the Legendre polynomials and

$$F_\ell \equiv 2\pi \int_{-1}^{1} P_\ell(\cos\Theta) f_1(\Theta) \, \mathrm{d}(\cos\Theta) \equiv \langle P_\ell(\cos\Theta) \rangle. \tag{2.351}$$

The quantities

$$g_\ell \equiv 1 - F_\ell = 2\pi \int_{-1}^{1} [1 - P_\ell(\cos\Theta)] \, f_1(\Theta) \, \mathrm{d}(\cos\Theta) \equiv \langle 1 - P_\ell(\cos\Theta) \rangle \tag{2.352}$$

will be referred to as the transport coefficients. Notice that $F_0 = 1$ and $g_0 = 0$. The value of F_ℓ decreases with ℓ due to the increasingly faster oscillations of $P_\ell(\cos\Theta)$ and, hence, g_ℓ tends to unity when ℓ goes to infinity. The transport mean free paths λ_ℓ are defined by

$$\lambda_\ell \equiv \lambda/g_\ell. \tag{2.353}$$

Assume that an electron starts off from a certain position, which we select as the origin of our reference frame, moving in the direction of the z-axis. Let $f(\mathbf{r}, \hat{\mathbf{d}}; t)$ denote the probability density of finding the electron at the position $\mathbf{r} = (x, y, z)$, moving in the direction given by the unit vector $\hat{\mathbf{d}}$ after having traveled a path length t. The diffusion equation for this problem is given by Lewis [97] as

$$\frac{\partial f}{\partial t} + \hat{\mathbf{d}} \cdot \nabla f = N \int \left[f(\mathbf{r}, \hat{\mathbf{d}}'; t) - f(\mathbf{r}, \hat{\mathbf{d}}; t) \right] \frac{\mathrm{d}\sigma(\chi)}{\mathrm{d}\Omega} \, \mathrm{d}\Omega, \tag{2.354}$$

where $\chi \equiv \cos^{-1}(\hat{\mathbf{d}} \cdot \hat{\mathbf{d}}')$ is the scattering angle corresponding to the angular deflection $\hat{\mathbf{d}}' \to \hat{\mathbf{d}}$. This equation has to be solved under the boundary condition $f(\mathbf{r}, \hat{\mathbf{d}}; 0) = (1/\pi)\delta(\mathbf{r})\delta(1 - \cos\Theta)$, where Θ is the polar angle of the direction $\hat{\mathbf{d}}$. By expanding $f(\mathbf{r}, \hat{\mathbf{d}}; t)$ in spherical harmonics, Lewis obtained general expressions for the angular distribution and for the first moments of the spatial distribution after a given path length t. The angular distribution is given by

$$F_{\mathrm{GS}}(\Theta; t) \equiv \int f(\mathbf{r}, \hat{\mathbf{d}}; t) \, \mathrm{d}\mathbf{r} = \sum_{\ell=0}^{\infty} \frac{2\ell+1}{4\pi} \exp(-tg_\ell/\lambda) P_\ell(\cos\Theta). \tag{2.355}$$

It is worth noticing that $F_{\mathrm{GS}}(\Theta; t)\mathrm{d}\Omega$ gives the probability of having a final direction in the solid angle element $\mathrm{d}\Omega$ around a direction defined by the polar angle Θ. Evidently, the distribution of Equation 2.355 is symmetrical about the z-axis, *i.e.*, independent of the azimuthal angle of the final direction.

The result given by Equation 2.355 coincides with the distribution obtained by Goudsmit and Saunderson [63] in a more direct way, which we sketch here to make the physical meaning clearer. Using the Legendre expansion given by Equation 2.350 and a folding property of the Legendre polynomials, the angular distribution after exactly n collisions is found to be

$$f_n(\Theta) = \sum_{\ell=0}^{\infty} \frac{2\ell+1}{4\pi} (F_\ell)^n P_\ell(\cos\Theta). \qquad (2.356)$$

The probability distribution of the number n of collisions after a path length t is Poissonian with mean t/λ, i.e.

$$P(n) = \exp(-t/\lambda) \frac{(t/\lambda)^n}{n!}. \qquad (2.357)$$

Therefore, the angular distribution after a path length t can be obtained as

$$F_{\mathrm{GS}}(\Theta; t) = \sum_{n=0}^{\infty} P(n) f_n(\Theta) = \sum_{\ell=0}^{\infty} \frac{2\ell+1}{4\pi} \left[\exp(-t/\lambda) \sum_{n=0}^{\infty} \frac{(t/\lambda)^n}{n!} (F_\ell)^n \right] P_\ell(\cos\Theta), \qquad (2.358)$$

which coincides with expression given in Equation 2.355.

From the orthogonality of the Legendre polynomials, it follows that

$$\langle P_\ell(\cos\Theta) \rangle_{\mathrm{GS}} \equiv 2\pi \int_{-1}^{1} P_\ell(\cos\Theta) F_{\mathrm{GS}}(\Theta; t)\, \mathrm{d}(\cos\Theta) = \exp(-tg_\ell/\lambda). \qquad (2.359)$$

In particular, we have

$$\langle \cos\Theta \rangle_{\mathrm{GS}} = \exp(-t/\lambda_1) \qquad (2.360)$$

and

$$\langle \cos^2\Theta \rangle_{\mathrm{GS}} = \frac{1}{3} \left[1 + 2\exp(-t/\lambda_2) \right]. \qquad (2.361)$$

These expressions can be used to check the accuracy of the calculated Goudsmit-Saunderson distribution.

The Goudsmit and Saunderson expansion of the multiple scattering distribution given in Equation 2.355 is exact for pure elastic scattering without energy loss. To compute it for a given single scattering DCS, we have to evaluate the transport coefficients g_ℓ as defined in Equation 2.352. The number of terms needed to make the Goudsmit-Saunderson series converge increases as the path length becomes shorter. In the case of small path lengths, the convergence of the series can be improved by separating the contributions from electrons that have had no collisions,

$$\begin{aligned} F_{\mathrm{GS}}(\Theta; t) \quad &\equiv \quad \exp(-t/\lambda) \frac{\delta(\cos\Theta - 1)}{2\pi} \\ &+ \sum_{\ell=0}^{\infty} \frac{2\ell+1}{4\pi} \left[\exp(-tg_\ell/\lambda) - \exp(-t/\lambda) \right] P_\ell(\cos\Theta). \end{aligned} \qquad (2.362)$$

The first term on the right-hand side represents unscattered electrons. In the current implementation, angular distributions are expressed in terms of the variable

$$\mu \equiv (1 - \cos\Theta)/2. \qquad (2.363)$$

We have

$$F_{\text{GS}}(\mu;t) = \exp(-t/\lambda)\delta(\mu) + \sum_{\ell=0}^{\infty}(2\ell+1)\left[\exp(-tg_\ell/\lambda) - \exp(-t/\lambda)\right]P_\ell(2\mu-1). \qquad (2.364)$$

Differential cross sections for elastic scattering of electrons and positrons by neutral atoms have been calculated for the elements $Z = $1-95 and projectile kinetic energies from 100 eV to 100 MeV. The calculations were performed by using the Dirac partial-wave program ELSEPA. The scattering potentials were obtained from self-consistent atomic electron densities calculated with Desclaux's multiconfiguration Dirac-Fock program. A finite nucleus, with Fermi model charge distribution, was assumed. Exchange effects in electron scattering were described by means of the local exchange potential of Furness and McCarthy. The DCSs computed in this way account for screening, finite nuclear size, exchange, spin and relativistic effects in a consistent way.

Electron and positron elastic DCS data is presented in tabulated form using a mesh of kinetic energies and scattering angles that is dense enough to allow cubic spline interpolation of $\ln(d\sigma/d\Omega)$ in $\ln T$ and μ. The size of the complete data set, referred to as DCSLIB, is about 125 MB. A series of new PEGS subroutines (ELASTINO being the main one) evaluates elastic DCSs for electrons and positrons in elements and compounds (using the additivity approximation) and computes Goudsmit-Saunderson multiple scattering distributions for specified path lengths. The transport coefficients g_ℓ are obtained by means of Gauss-Legendre quadrature, with the angular range split into a number of subintervals to improve accuracy.

At small path lengths t, a very large number of coefficients g_ℓ may be required to ensure convergence of the Legendre series given in Equation 2.362. The default value of the number of coefficients is set in ELASTINO to 1000, which should yield convergence for reasonably short paths in most cases. Note that since the time to compute a single distribution function depends on the square of the number of coefficients, and, typically, a very large number of distributions must be computed (the distribution function space is three dimensional, depending on material, energy, and path length, for both positrons and electrons), the total time to compute a full set of distribution functions for some problems can be very long. To speed the computation, ELASTINO works from low energy and short paths to higher energies and longer paths, re-setting at each new step the number of terms to be computed based on the number required at the previous step. Even so, computation times can be extremely long, even when error tolerances have been set fairly high, and users should be mindful of this whenever invoking the use of this distribution.

Traditionally, sampling from GS distributions has been either prohibitively expensive (requiring computation of the slowly converging series) or overly approximate (using very large pre-computed data tables with limited accuracy). We have developed here new fitting and sampling techniques that overcome these drawbacks, using a scaling model that have been known for some time [27]. First, the change of variables of Equation 2.363 is performed, and a reduced angle now called $\chi = (1 - \cos(\Theta))/2$ is defined. The full range of angles ($0 \leq \Theta \leq \pi$, or $0 \leq \chi \leq 1$) is then broken into 128 intervals of equal probability, with the 128th interval further broken down into 32 sub-intervals of equal spacing. Because the distribution is so heavily forward peaked, in most cases the last 32 sub-intervals cover the entire range from approximately $\pi/2$ to π, which is why this region

is treated specially. In each of the 287 intervals or sub-intervals, the distribution is parameterized as

$$f(\chi) = \frac{\alpha}{(\chi + \eta)^2} \left[1 + \beta(\chi - \chi_-)(\chi_+ - \chi) \right] \tag{2.365}$$

where α, β and η are parameters of the fit (which can be determined analytically if it is assumed that the distribution function conforms the shape given above over each given interval) and χ_- and χ_+ are the endpoints of the interval.

Using the large number of angle bins which we do, this parameterization models the exact form of the distribution to a very high degree of accuracy, and, in addition, it can be sampled very quickly. As the 128 bins are equally probable, initial angle intervals can be determined trivially. When the last (128th) interval is selected, a linear table search is used to determine which of the 32 equally spaced sub-intervals to use. This table search is usually very fast because the first several of the 32 sub-intervals typically cover 90% of the probable angles in the final interval. Once the correct bin is chosen, the distribution can be sampled quickly. We begin by inverting the first term of Equation 2.365 to produce a value of χ from

$$\chi = \frac{\eta\zeta + \chi_-(\chi_+ + \eta)/(\chi_+ - \chi_-)}{(\chi_+ + \eta)/(\chi_+ - \chi_-) - \zeta}, \tag{2.366}$$

and we then use $1 + \beta(\chi - \chi_-)(\chi_+ - \chi)$ as a rejection function. Rejection efficiencies on the order of 95% are typical. Energy and pathlength interpolation is done by randomly selecting either the upper or lower bounding grid point in the tabulated mesh.

2.15 Transport Mechanics in EGS5

As noted in earlier sections, because of the very large number of scattering collisions per unit path which electrons undergo as they pass through matter, Monte Carlo simulation of individual electron collisions (sometimes referred to as "analog Monte Carlo electron transport") is computationally feasible only in limited situations. Computationally realistic simulations of electron transport must therefore depart from the physical situation of linear, point-to-point, translation between individual scattering collisions (elastic or inelastic) and instead transport particles through long "multiple-scattering steps," over which thousands of collisions may occur, a technique commonly referred to as the "condensed history" [15] method. If we assume that most electron Monte Carlo simulation algorithms exhibit Larsen convergence[10] [96], then all algorithms exhibit an accuracy/speed trade-off which is driven solely by the length of the multiple scattering step they can take while still remaining faithful to the physical processes which might occur during those long steps. Models describing energy loss, angular deflection, and secondary electron production over long steps are quite well known. Descriptions of the longitudinal and transverse displacement coupled to energy loss and deflection over a long transport step, even in homogeneous media, however, require complete solutions of the transport equation, and so models which are sometimes quite approximate

---

[10]Larsen noted that electron transport algorithms using multiple-scattering steps ought to converge to the analog result as the number of collisions in the multiple-scattering steps decreases. It must be noted that in general, this is *not* the case.
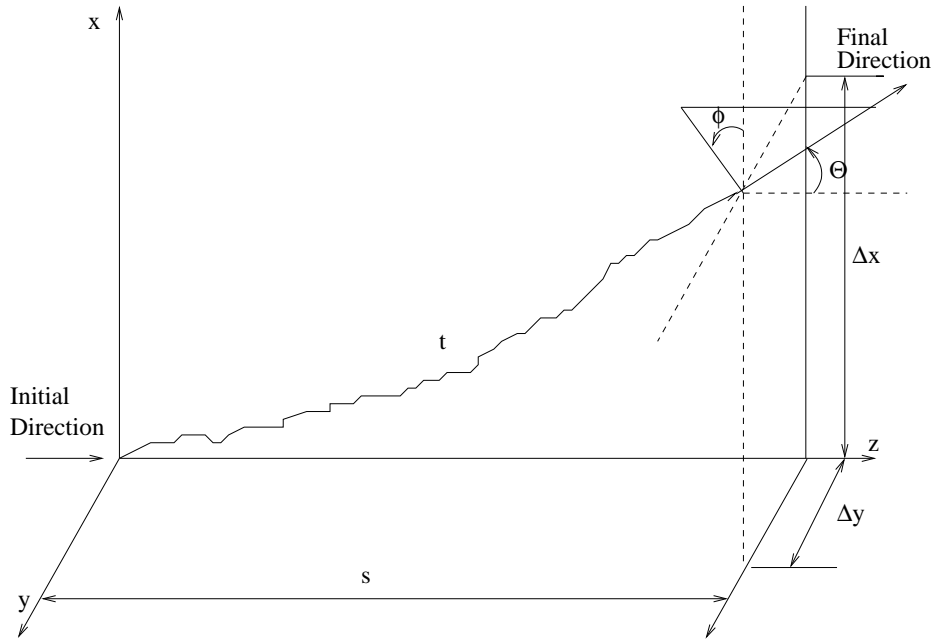
Figure 2.8: Schematic of electron transport mechanics model.

are used. The methods employed in Monte Carlo programs for selecting electron spatial coordinates after transport through multiple-scattering steps have come to be referred to as "transport mechanics." This is illustrated schematically in Figure 2.8.

As an electron moving in the z direction passes through a semi-infinite region of thickness s, it will have traveled a total distance of t, will have undergone lateral displacements (relative to its initial direction) of Δx and Δy, and will be traveling in a direction specified by Θ and ϕ. That is, relative to its initial position and direction, the particle's final position is $(\Delta x,\ \Delta y,\ s)$, and the final direction vector given by $(\sin\Theta\ \cos\phi,\ \sin\Theta\ \sin\phi,\ \cos\Theta)$. The angle Θ can be determined from an appropriate multiple scattering distribution, the angle ϕ is assumed to be uniformly distributed, and the selection of Δx, Δy and s are determined from the transport mechanics of a given simulation algorithm.

The fidelity with which an algorithm's transport mechanics model reproduces true distributions for displacement is almost the only factor driving the speed/accuracy trade-off of the program, and hence a substantial effort was made to improve the transport mechanics model in EGS5 relative to that in EGS4. In this section, we will describe the motivation and development of the EGS5 transport mechanics model, discuss the parameters used to control the multiple scattering steps sizes in EGS5, and then present a novel approach for automatically determining nearly optimal values of the parameters based on a single, geometry-related parameter.

Until the introduction of the PRESTA [31] algorithm, previous versions of EGS corrected for the difference between s and t, but ignored the lateral deflections Δx and Δy. Thus, the procedure

101

used to simulate the transport of the electron was to translate it in a straight line a distance s along its initial direction and then determine its new direction by sampling the scattering angle Θ from a multiple scattering p.d.f. dependent upon the material, the total distance traveled, t, and the particle energy.

Ignoring lateral deflections introduces significant errors, however, unless restrictions (often quite severe) are placed on the maximum sizes of the electron transport tracklengths t. These restrictions were greatly eased by the introduction of the PRESTA algorithm, which treats Δx and Δy explicitly during the transport simulation and which also includes a more accurate prescription for relating the straight line transport distance s to the actual pathlength t. It should be noted here that because of the random nature of the particle trajectory, s, Δx, and Δy are actually random variables, dependent upon the scattering angle Θ and the tracklength t. In PRESTA, Δx and Δy range between 0 and $t/2$ and s is given by some fraction of t. There are two major drawbacks to the PRESTA formalism, however. First, in situations where an electron is traveling close to a region boundary, translating it lateral distances Δx and Δy perpendicular to its initial direction can sometimes result in moving it across the boundary and into a region with different material properties. Thus PRESTA required computationally expensive interrogation of the problem geometry and sometimes resulted in very small steps when particles were traveling roughly parallel to nearby region boundaries. Second, PRESTA is not adept at modeling backscattering. Electron backscattering in general results from a single, very large angle collision and not as the aggregate effect of a large number of small-angle collisions. It is clear from Figure 2.8 that physically, if an electron were to experience a 180 degree collision immediately at $t = 0$, it could potentially travel a distance $s = -t$ in the backward direction. Thus the set of all possible final positions for an electron traveling a pathlength t is a sphere of radius t (this is sometimes referred to as the "transport sphere"). Because it always sets s to be a positive value, the PRESTA formalism tends to overestimate the penetration prior to the large-angle backscatter event, which can lead to significant errors in computations of energy distributions of backscattered electrons, among other quantities.

2.15.1 Random Hinge Transport Mechanics

In EGS5 a new transport mechanics model, the random hinge, is introduced to address these shortcomings of PRESTA and thus permit very long step-sizes. The random hinge model, first used in PENELOPE [14], derives a large part of its success from being formulated so that the set of all possible termination points is in fact the full transport sphere of radius t. In this methodology, instead of transporting the particle a distance s and then displacing the particle by Δx and Δy and updating its direction according to the sampled scattering angles Θ and ϕ, the particle track t is first split randomly into two parts of lengths ζt and $(1 - \zeta)t$, where ζ is a random number. The electron is transported along its initial direction a distance ζt, at which point (the hinge point) its direction is updated (using Θ and ϕ as the scattering angles) and the particle is then translated in this new direction the remainder of it original tracklength, given by $(1 - \zeta)t$.

It is clearly seen from the schematic of Figure 2.9 that the case of $\zeta = 0$ (*i.e.*, the hinge point
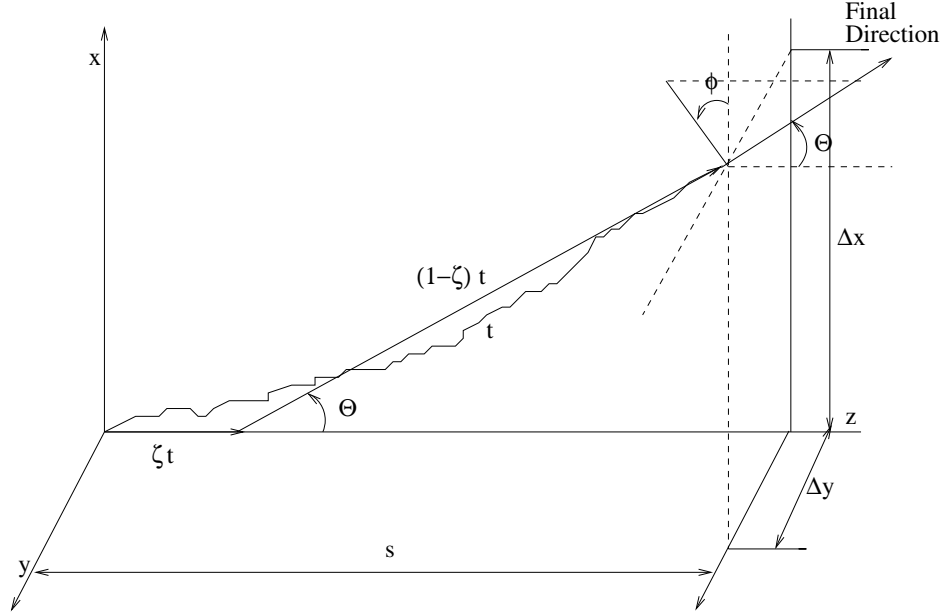
Figure 2.9: Random hinge transport mechanics schematic.

falling at the initial position) provides a mechanism to simulate track termination points over the full transport sphere. Formally, the advantages of this method can be demonstrated by examining the moments of the random hinge model. Ignoring energy losses, Kawrakow and Bielajew [88] have shown that this version of the random hinge model yields the correct values of the average straight-line path $\langle s \rangle / t$ and lateral displacement $\langle \Delta x^2 + \Delta y^2 \rangle / t$, and also comes very near to preserving many higher angle spatial moments.

Energy losses, because they result in changes in the single scattering cross section along t, reduce the accuracy of the random hinge. It has been shown by Fernández-Varea *et al.* [56] that when the single elastic scattering cross section is dependent on t, the first moments for axial and lateral displacements in the random hinge model differ from the exact moments in that they effectively evaluate first and second transport cross sections at slightly different distances along t. The transport cross sections (or inverse transport mean free paths), G_ℓ, are given by the Legendre moments of the single scattering cross sections, as in

$$G_\ell(t) = 2\pi \int_{-1}^{1} d\mu \Sigma(\mu; t)[1 - P_\ell(\mu)], \tag{2.367}$$

where $\Sigma(\mu; t)$ is the spatially dependent macroscopic single elastic scattering cross section and μ is $\cos(\Theta)$. Whereas the correct expressions for $\langle s \rangle / t$ and $\langle \Delta x^2 + \Delta y^2 \rangle / t$ in the energy dependent case are

$$\frac{\langle s \rangle}{t} = 1 - \frac{t}{2} G_1\left(\frac{t}{3}\right) \tag{2.368}$$

and

$$\frac{\langle \Delta x^2 + \Delta y^2 \rangle}{t} = \frac{2t}{9} G_2\left(\frac{t}{4}\right), \tag{2.369}$$

103

the average axial and lateral displacements in the random hinge model are

$$\frac{\langle s \rangle}{t} = 1 - \frac{t}{2}G_1(\frac{t}{2})$$ (2.370)

and

$$\frac{\langle \Delta x^2 + \Delta y^2 \rangle}{t} = \frac{2t}{9}G_2(\frac{t}{2})$$ (2.371)

respectively [56]. Since the $G_\ell(t)$'s increase as t increases (and energy decreases), the random hinge model slightly underestimates the average straight-line path s and overestimates the lateral deflections.

2.15.2 Modified Random Hinge Transport Mechanics

The analysis above suggests a modification to correct the random hinge method which was first employed in the Monte Carlo program DPM [155]. In the original random hinge methodology, energy loss is accounted for by simply evaluating the multiple scattering p.d.f. at the hinge point, which, on average, will occur at $t/2$. In the new methodology, the location of the random hinge is not a randomly selected fraction of the total distance to be traveled, t, but instead it is based on a randomly selected fraction of the integral over t of G_1, the first transport mean free path, commonly referred to as the "scattering power." This integral quantity is commonly called the "scattering strength," and denoted by $K_1(t)$, as in

$$K_1(t) = \int_0^t dt' G_1(t').$$ (2.372)

Thus, instead of the hinge consisting of steps ζt and $(1 - \zeta)t$, it consists of steps of the distances corresponding to the accumulation of scattering strengths $\zeta K_1(t)$ and $(1 - \zeta)K_1(t)$. In this way, the total scattering strength $K_1(t)$ over the step is preserved, and the average location of the hinge will correspond to that location which preserves the first moments of spatial displacements. The modified random hinge transport mechanics of EGS5, illustrated in Figure 2.10, is therefore capable of simulating accurately the average final positions of electrons moving long pathlengths through materials, even when energy loss occurs continuously along the track. The length of an electron step is therefore limited only by the accuracy of the multiple scattering p.d.f., any need to higher order spatial moments (as would be required for some differential tallies), and the accuracy of the method used to compute the integral in Equation 2.372 along the paths between hinge points.

Values of the scattering powers G_1 for electrons and positrons as a function of energy are computed in PEGS in a new function G1E. For kinetic energies less than 100 MeV, values are taken from cubic spline fits of data provided in the DCSLIB package described in the previous section. Above 100 MeV, the Wentzel-shaped scattering cross section used by Molière is assumed, in which case Equation 2.367 for $\ell = 1$ can be integrated analytically to yield

$$G_1(E) = \frac{N_a \rho}{M} \frac{2\pi Z^2 e^4}{p^2 v^2} \left[\ln \left(\frac{\pi^2 + \chi_a^2}{\chi_a^2} - \frac{\pi^2}{\pi^2 + \chi_a^2} \right) \right],$$ (2.373)
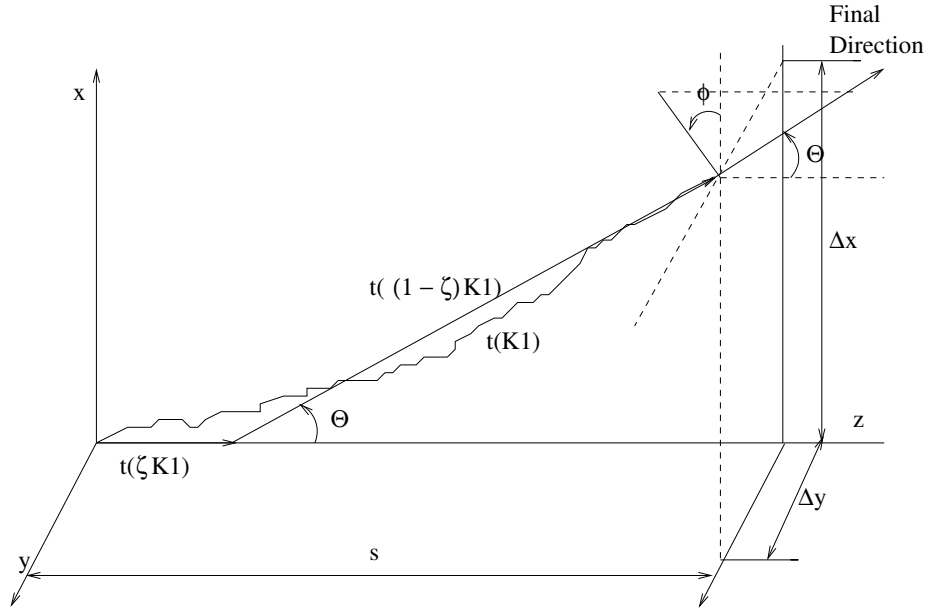
Figure 2.10: Modified random hinge transport mechanics schematic.

where χ_a^2 is taken from Equation 2.322. Discontinuities at 100 MeV between values of G_1 computed from the partial-wave cross section data of DCSLIB and from the Wentzel cross section above are usually small (on the order of fractions of 1 percent), and hence are smoothed only crudely.

Since the Goudsmit-Saunderson multiple scattering distributions are pre-computed in PEGS, and as we noted earlier, they are dependent on step size, the modified random hinge methodology requires that they be tabulated in terms of K_1 instead of t. When this option is invoked, a new PEGS routine MAKEK1, which uses a Gauss-Legendre quadrature to solve Equation 2.372 with G_1 expressed in terms of energy, is called to prepare a table of values of K_1 ranging between the maximum and minimum seen in the current problem. Note that because in EGS5 we march across boundaries rather than stop and apply multiple scattering first, it is never the case that an arbitrarily small multiple scattering step will be taken, and the minimum step is known in advance.

2.15.3 Dual Random Hinge Approach

The accuracy with which the integral over the scattering power G_1 is computed over the hinge can be stated more plainly as the accuracy of the computation of the hinge distances in cm given the scattering strength, ζK_1 and $(1-\zeta)K_1$. Inverting integrals over pathlength such as Equation 2.372 is a common process in Monte Carlo electron transport programs, analogous to the computation in EGS4, for example, of the pathlength to be taken given a specified fractional energy loss to occur over the step. In typical Monte Carlo programs this type of calculation is done by some form of the trapezoid rule, which requires that the integrand (stopping power, scattering power, *etc.*) vary
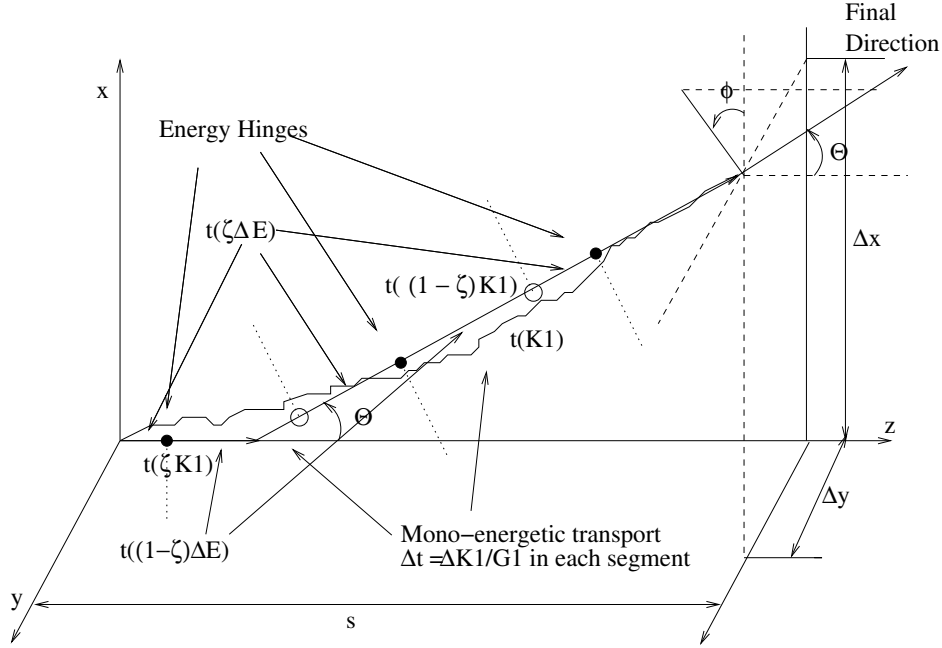
Figure 2.11: Dual (energy and angle) hinge transport mechanics schematic.

no more than linearly through the step.

EGS5 employs a different approach to this problem in an effort to retain as much as possible the very large steps permitted with the multiple scattering hinge as long as average values of K_1 are preserved over the steps. Thus, rather than require small steps to assure accurate integration of G_1 over energy, EGS5 completely decouples energy loss and multiple scattering by employing a second random hinge to describe energy loss, relying on the random position of the energy loss hinges to yield the correct energy-averaged computation of K_1. A schematic of the dual random hinge approach is presented in Figure 2.11. Independent energy loss steps $t_E(\Delta E)$ (the distance being determined from some initially specified fractional energy loss) and multiple scattering steps $t_\Theta(K_1)$ (here the distance is determined from an initially specified scattering strength) are simultaneously processed, using random hinges. The lengths of the initial segments are determined by multiplying random numbers ζ_E and ζ_Θ by ΔE and K_1 respectively, as before. A particle is then transported linearly until it reaches a hinge point at either $t_E(\zeta_E \Delta E)$ or $t_\Theta(\zeta_\Theta K_1)$. When the multiple scattering hinge point is reached, the deflection is modeled assuming that the entire multiple scattering path $t_\Theta(K_1)$ has been traversed, and transport continues. Similarly, the energy of the electron remains unchanged along its track until the energy hinge point $t_E(\zeta_E \Delta E)$ is reached, at which time the electron energy is decremented by the *full amount* ΔE prescribed. Subsequent to a reaching a hinge point and undergoing either energy loss or deflection, the electron must then be transported the distance remaining for the given step, $t_E[(1 - \zeta_E)\Delta E]$ for energy hinges or $t_\Theta[(1 - \zeta_\Theta)K_1]$ for multiple scattering hinges. The two hinge mechanisms are completely independent, as several energy hinges may occur before a single multiple scattering is applied, and vice versa.

106

In evaluating the multiple scattering distribution and sampling for the deflection, all parameters which have an energy dependence are computed using *the most recently updated energy* for the particle. Since energy hinges will occur sometimes before scattering hinges and sometimes after, there is thus an implicit averaging of the electron energies over the full hinge distance. It is in this way that the random energy hinge provides accurate integration of K_1 by, on average, using G_1 at the correct energy. This same methodology is employed when computing the energy loss from the integral of the stopping power: rather than attempt to approximate the average stopping power over the step, EGS5 relies on the random energy hinge to provide the correct average result for the total energy loss.

Note that the quantities that remain fixed over the hinges are ΔE and K_1, and not the distances t_E and t_Θ. Whenever an event occurs in which values of the scattering power or stopping power change, such as an energy hinge, the energy loss and scattering strength remain fixed, and any unused hinge distances t_E and t_Θ must be updated using the new values of the scattering and stopping power. Energy deposition along the track is done using the continuous slowing down approximation (CSDA) model, with updates made whenever a hinge point is reached or any other event occurs.

2.15.4   Boundary Crossing

One extremely important implication of the use of the modified random hinge for multiple scattering is that it lends itself to a method for seamlessly crossing region boundaries. In all previous versions of EGS, including EGS4 with PRESTA, when a particle boundary was reached multiple elastic scattering was applied and transport started anew in the new region. With the modified random hinge however, multiple scattering occurs only at hinge points. If a boundary is crossed during either the pre-hinge ($\zeta K_1(t)$) or post-hinge ($(1-\zeta)K_1(t)$) portion of the step, the value of G_1 used in updating the accumulated scattering strength is simply changed to reflect the new value of the scattering power in the new media. Thus it is not necessary to apply multiple scattering at region boundaries, and the expensive re-interrogation of the problem geometry required by PRESTA is completely avoided. Inherent in this is the implication that multiple scattering distributions are equivalent for different materials at a given energy and for pathlengths which correspond to the same scattering strength K_1. This, of course, is not strictly true. It can be shown formally, however, that for a multiple scattering distribution expressed as a sum of Legendre polynomials in Θ,

$$\langle \cos(\Theta) \rangle = \exp(-K_1) \tag{2.374}$$

so that for small K_1

$$K_1 \simeq 1 - \langle \cos(\Theta) \rangle. \tag{2.375}$$

Thus, in preserving K_1 for cross boundary transport, the EGS5 method also roughly preserves the average cosine of the scattering angle over the boundary.

Inspection of the implementation details reveals that the boundary crossing in EGS5 is analogous to an energy hinge without energy loss. All step-size variables (rates and distances) need to be updated, but otherwise transport to the next event is uninterrupted, as shown in Figure 2.12.
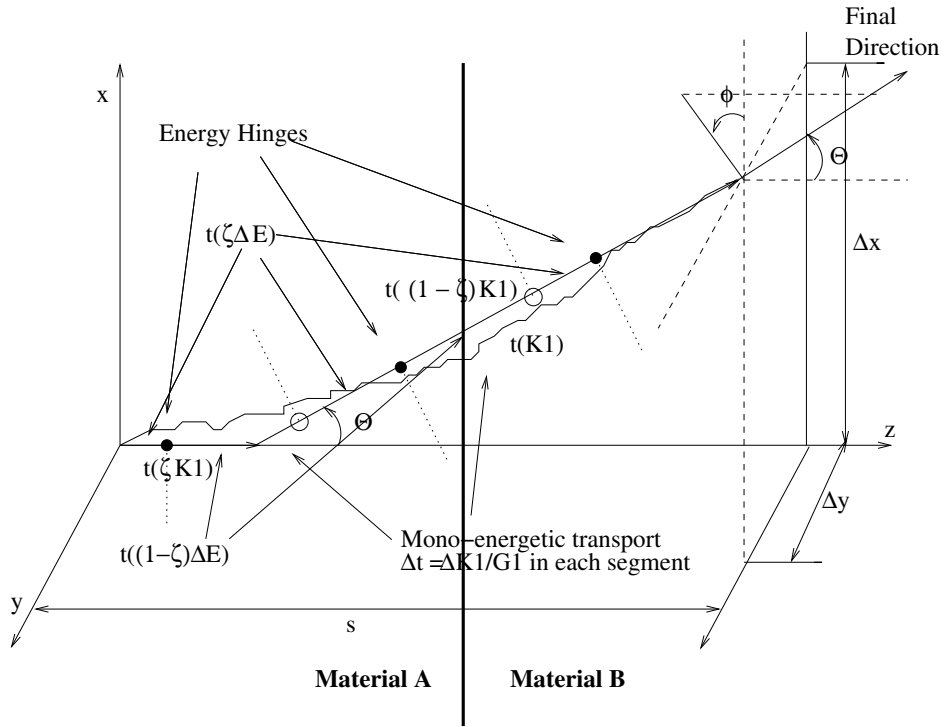
Figure 2.12: Electron transport across region boundaries.

2.15.5 EGS5 Transport Mechanics Algorithm

The transport between hard collisions (bremsstrahlung or delta-ray collisions) is superimposed on the decoupled hinge mechanics as an independent, third possible transport process. To retain the decoupling of geometry from all physics processes, for hard collisions EGS5 holds fixed over all boundary crossing an initially sampled number of mean free paths before the next hard collision, updating the corresponding distance (computed from the new total cross section), when entering a new region. Again, while the random energy hinge preserves the average distance between hard collisions, it does not preserve the exact distribution of collision distances if the hard collision cross section exhibits an energy dependence between the energy hinges. In practice, however, this leads to only small errors in cases where the energy hinge steps are very large and the hard collision mean free path is sharply varying with energy.

Thus the dual random hinge transport mechanics can be described as follows: at the beginning of the particle simulation, four possible events are considered: an energy loss hinge (determined by a hinge on specified energy loss ΔE); a multiple scattering hinge (determined by a hinge on a specified scattering strength K_1); a hard collision (specified by a randomly sampled number of mean free paths); and boundary crossing (specified by the problem geometry). The distances to each of these possible events is computed using the appropriate stopping power, scattering power, total cross section or region geometry, and the particle is transported linearly through the shortest of those 4 distances. The appropriate process is applied, values of the stopping power, scattering power, cross section and boundary condition are updated if need be, and new values of the distances to the varies events are computed to reflect any changes. Transport along all hinges then continues through to the next event. Effectively, there are four transport processes occurring simultaneously at each single translation of the electron.

The details of the implementation of the dual random hinge, because it is such a radical departure from other transport mechanics models, can sometimes lead to some confusion (and, in any case, the energy hinge definitely leads to important implications for many common tallies), and so we present an expanded explication here. Ignoring hard collisions and boundary crossings for the moment and generalizing here for the sake of brevity, we note that a step t involves transport over the initial step prior to the hinge a distance t_{init} given by (ζt) followed by transport through the residual step distance t_{resid} by $((1 - \zeta)t)$. In practice, once a particle reaches the hinge point at t_{init}, we do not simply transport the particle through t_{resid} to the end of the current step, because nothing actually occurs there, as the physics was applied at the hinge point. So instead, immediately after each hinge the distance to the *next* hinge point is determined and total step that the particle must be transported before it reaches its next hinge is given by $t_{step} = t_{resid}^1 + t_{init}^2$, where the superscripts refer to the 1st and 2nd hinges steps. We thus have the somewhat counter-intuitive situation in that when a particle is translated between two hinge points, it is actually being moved a distance which corresponds to the residual part of one transport step plus the initial part of the second transport step. Thus we distinguish between *translation hinge steps*, over which particles are moved from one hinge point to the next, linearly and with constant energy, and *transport steps*, which refer to the conventional condensed history Monte Carlo distances over which energy losses and multiple scattering angles are computed and applied. Translation hinge steps and transport

Transport Steps, $\Delta E = E \times$ **ESTEPE**

transport step 1   **transport step 2**

$\Delta E1 = $ DEINITIAL1+ DERESID1 $\Delta E2 = $ DEINITIAL2 + DERESID2

DEINITIAL1   DERESID1   DEINITIAL2   DERESID2

energy hinge 1   energy hinge 2

$KE = KE - \Delta E1$ $KE = KE - \Delta E2$

initial translation step,   **translation step 2,**   **translation step 3,**
DEINITIAL1   DERESID1 + DEINITIAL2   DERESID2 + DEINITIAL3

Translation Steps, between random hinge points
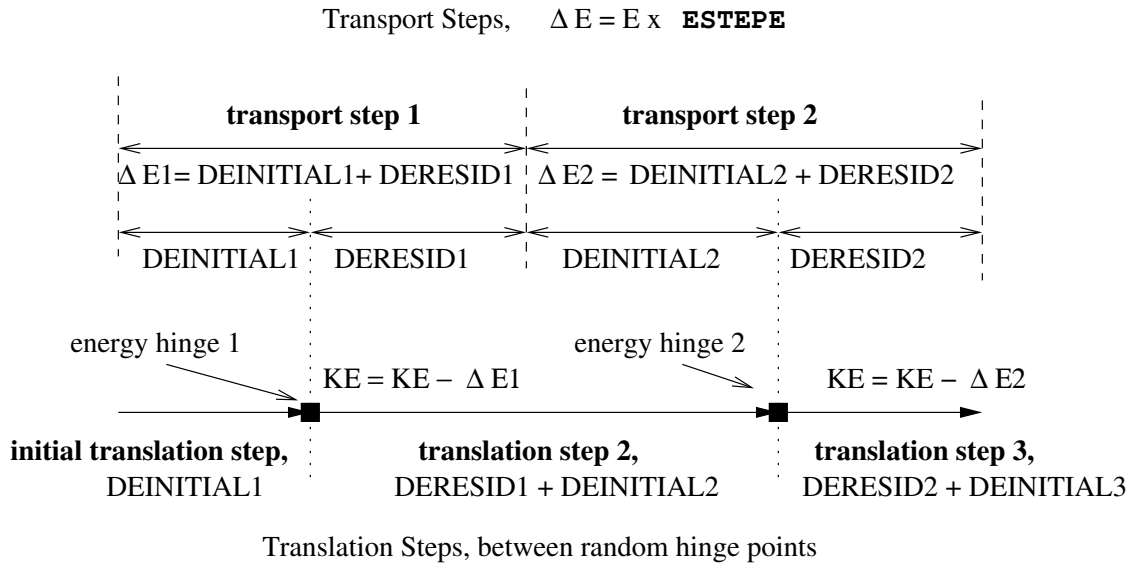
Figure 2.13: Translation steps and transport steps for energy loss hinges. The top half of the figure illustrates the step size (in terms of energy loss) for consecutive conventional Monte Carlo transport steps, with the energy loss set at a constant fraction of the current kinetic energy (E times ESTEPE as in EGS4, for example). The lower schematic shows how these steps are broken into a series of translation steps between randomly determined hinge distances. Transport through the translation steps is mono-energetic, with full energy loss being applied at the hinge points. Note that the second translation step, for which the electron kinetic energy is constant, actually involves moving the electron through pieces of 2 different transport steps. Multiple scattering could interrupt this energy step translation at any point or at several points, but does not impact the energy transport mechanics.

steps thus overlap rather than correspond, as illustrated in Figure 2.13.

Variables which contain information about the full distance to the next hinge (the translation step), the part of that distance which is the initial part of the current transport step, and the residual (post-hinge) distance remaining to complete the current transport step, for both energy loss and deflection, must now be stored while the particle is being transported. Again, it is not the distances themselves but rather the energy losses and scattering strengths which matter, and in EGS5, these variables are called DENSTEP, DEINITIAL and DERESID, for the energy loss hinge and K1STEP, K1INIT and K1RSD, for the the multiple scattering hinge. For reasons discussed below, only the scattering strength variables become part of the particle stack; the energy loss hinge variables are all local to the current particle only.

Several interesting consequences arise from the use of energy hinge mechanics. Even though the electron energy is changed only at the energy hinge point, energy deposition is modeled as occurring along the entire electron transport step, and the EGS4 energy loss variable EDEP, which has been retained in EGS5, is computed along every translation step, and passed to the user for

110

tally, as in EGS4. Depositing energy continually in this manner, rather than only in discrete, large, chunks at the hinge points, obviates requiring very small values of DEINITIAL to attain small statistical uncertainty in most energy deposition problems. It does introduce some artifacts, however, in that while translating a particle to an energy hinge point, energy is being deposited into the medium *without* being decremented from the particle. Similarly, along the track segment between the hinge and the end of the energy transport step (the part described by DERESID), the full transport step energy loss will have been decremented from the electron, but not all of that will have been deposited via continuous stopping loss until the end of the transport step. Thus energy is not strictly conserved, in that a particle's total energy plus the energy of its daughter particles plus the energy that it has deposited do not sum to the particle's initial energy except at the exact transitions between hinge steps (which are not stopping points on the translation steps). This has some interesting consequences. For instance, if an electron escapes the problem geometry before reaching its next energy hinge, a hinge must be imposed at the boundary to and the amount of energy deposited prior to the escape of the particle much be decremented from the particle before it is tallied as an escaped particle. Similarly, if an electron escapes during the residual part of a previous hinge (*i.e.*, before the full DERESID has been deposited), its kinetic energy must adjusted upward to account for the fact that the full hinge energy has already been decremented from the particle, but a portion of it has yet to be deposited, because the end of the hinge step had yet to be reached. See Figure 2.14 for a schematic illustrating this problem.

Likewise, when a hard collision occurs, a particle will be somewhere in the midst of some transport step, on either the residual side of the previous energy hinge or the initial side of the upcoming hinge, and so the kinetic energy available for the collision must be adjusted to account for the energy deposited during the translation step. If the electron has yet to reach the end of the previous hinge still (*i.e.*, it's still on the residual side), we must adjust the particle kinetic energy by increasing it to account for energy decremented but not deposited before proceeding the collision analysis. And, as with the case of escaping particles, if the electron has passed the end of the previous hinge step and is on the initial part of the next hinge step, we must impose a hinge and decrement the already deposited energy from the electron. These adjustments are necessary not only to preserve energy deposition, but also to determine the kinetic energy available for the hard collision. In terminating hinges before hard collisions, we adjust the electron energy exactly as was done for escaping electrons, by setting $E = E -$ DEINITIAL $+$ DENSTEP.

A final interesting problem occurs at final electron energy hinge. Because the electron energy is set to be equivalent to ECUT at the final hinge point, which is reached *before* the electron has been translated through the DERESID portion of its transport step, some provision must be made for tracking electrons during the residual parts of their final hinge steps. This requirement is complicated somewhat in that such electrons will have total energy which is *exactly equal* to the problem cutoff energy. In the present EGS5 algorithm, when the last energy hinge point is reached, the electron energy is set to ECUT, DENSTEP is set to DERESID, and DEINITIAL is set to zero (DERESID is then set to zero as well). The multiple scattering hinge step at this point is set to infinity, as is the distance to the next hard collision. The particle is thus transported linearly through the distance corresponding to DERESID, depositing the appropriate CSDA energy along the way.
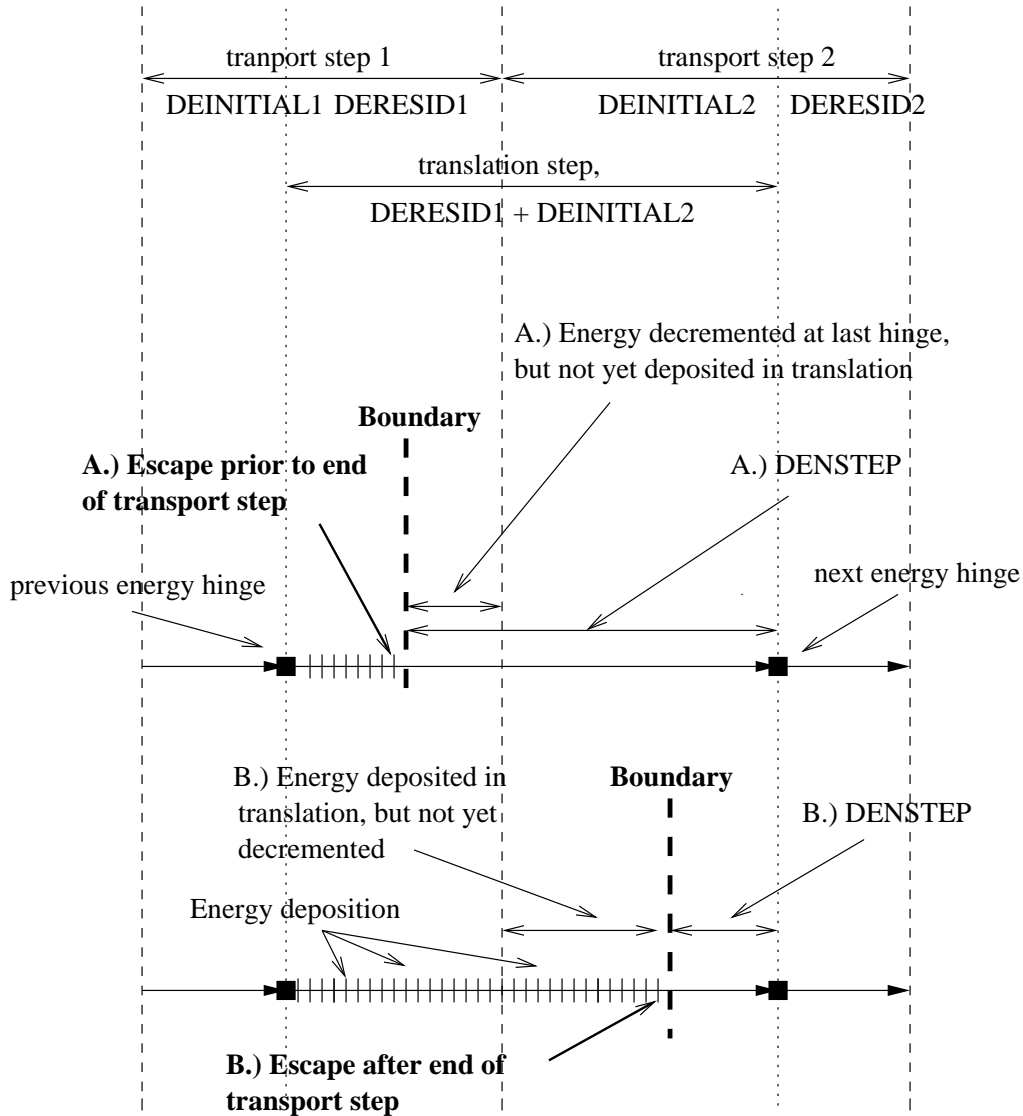
Figure 2.14: Electron boundary crossing during translation steps. In the top schematic (case A), the electron crosses escapes the problem volume before traversing the full distance corresponding to the initial transport step (*i.e.*, through both legs of the hinge), while in the bottom schematic (case B), the particle escapes the volume after the completion of initial transport step. In case A, the full transport step CSDA energy loss is decremented from the particle's energy at the time it passes through the hinge point, but as the particle has not traveled the distance corresponding to that amount of energy loss, some kinetic energy must be *added* back to the particle. In case B, the particle crosses the boundary during the first leg of the second energy hinge step, thus it has deposited energy into the system even though its energy has not been updated, so some kinetic energy must be *subtracted* at that point. In either instance, the final energy of the electron can be shown to be $E = E - \texttt{DEINITIAL} + \texttt{DENSTEP}$.

2.15.6   Electron Step-Size Selection

User control of multiple scattering step-sizes in EGS4 was accomplished through the specification of the variable `ESTEPE`, the fractional kinetic energy loss desired over the steps. Because of the complete decoupling of the energy loss and elastic scattering models in EGS5, however, there are two separate step-size mechanisms, one based on energy loss and controlling energy hinge steps, and one based on scattering strength and controlling multiple-scattering hinge steps. Of the two, the multiple-scattering step is almost always the more limiting because it is what drives the accuracy of the transport mechanics model. As stated earlier, energy hinges are required primarily to provide accurate numerical integration of energy-dependent quantities. Because hard collisions impose *de facto* energy hinges whenever they are encountered, for situations in which their cross sections are large because the PEGS thresholds `AE` and `AP` are small, the numerous hard collisions often provide the full accuracy required in energy integration, thus rendering energy hinges superfluous. Only in cases where hard collision probabilities are small and material cross sections, stopping powers, *etc..*, vary rapidly with energy do energy hinge steps actually need to be imposed, and for such instances, a mechanism has been developed by which these hinge steps are automatically determined in PEGS, as described below.

2.15.7   Energy Hinge Step-Size Determination in PEGS

As noted above, in the dual hinge formalism of EGS5, the energy steps provide no function other than assuring accurate numerical integration over energy-dependent variables (such as energy loss, scattering strength, *etc.*). For any such quantity f which varies with energy through a step of total length t, if the energy hinge occurs at a distance h, the EGS5 random energy hinge methodology gives for the integration of f over distance variable s through t

$$
\begin{aligned}
F(t\!:\!h) &= \int_0^t ds\, f(s) \qquad\qquad\qquad\qquad\qquad (2.376)\\
&= h f(E_0) + (t - h) f(E_1)
\end{aligned}
$$

where E_0 is the initial energy and E_1 the energy at the end of the energy hinge step t. As implemented in EGS5, the energy hinges distances are uniformly distributed in energy, so if ζ is a random number, we have $h = \zeta \Delta E |\frac{dE}{dx}|^{-1}$ and $(t - h) = (1 - \zeta)\Delta E |\frac{dE}{dx}|^{-1}$, giving us in practice

$$
F(t\!:\!h) = \zeta \Delta E\, f(E_0) \left|\frac{dE}{dx}\right|_{E_0}^{-1} + (1 - \zeta)\Delta E\, f(E_1) \left|\frac{dE}{dx}\right|_{E_1}^{-1} \qquad (2.377)
$$

Since the energy hinges are uniformly distributed over t, the average values of the integrated quantities are given by

$$
\begin{aligned}
\overline{F(t)} &= \int_0^t dh\, F(t\!:\!h)\, p(h) \qquad\qquad\qquad\qquad\qquad (2.378)\\
&= \int_0^1 d\zeta \left(\zeta \Delta E\, f(E_0) \left|\frac{dE}{dx}\right|_{E_0}^{-1} + (1 - \zeta)\Delta E\, f(E_1) \left|\frac{dE}{dx}\right|_{E_1}^{-1} \right)
\end{aligned}
$$

113

$$= \frac{\Delta E}{2} \left(f(E_0) \left| \frac{dE}{dx} \right|_{E_0}^{-1} + f(E_1) \left| \frac{dE}{dx} \right|_{E_1}^{-1} \right)$$

Thus, the random energy hinge step distances are limited by the accuracy which can be achieved in numerically integrating energy dependent quantities of interest using the trapezoid rule. This limit suggests a prescription for determining the energy hinge step-sizes in EGS5: we take t as the longest step-size which assures that Equation 2.379 is accurate to within a given tolerance ϵ_{fE} when applied to the integration of the following: the stopping power to compute the energy loss; the scattering power to compute the scattering strength; and the hard collision cross section to compute the hard collision total scattering probability and mean free path. Thus, in general, if $\overline{\Delta F}|_{anal}$ is the analytic integral of one of our functions f over t, we wish to satisfy

$$\overline{\Delta F}|_{anal} - \overline{F(t)} \le \epsilon_{fE} \tag{2.379}$$

or

$$\int_0^t ds\, f(t) \bigg|_{anal} - \left[\frac{t}{2} \left(f(E_0) + f(E_1) \right) \right] \le \epsilon_{fE} \tag{2.380}$$

or

$$\int_{E_0}^{E_1} dE f(E) \left| \frac{dE}{dx} \right|^{-1} \bigg|_{anal} - \left[\frac{R_C(E_0) - R_C(E_1)}{2} \left(f(E_0) + f(E_1) \right) \right] \le \epsilon_{fE} \tag{2.381}$$

where $R_C(E)$ is the CSDA range for an electron with energy E.

For the scattering strength, the function f is the scattering power G_1, and for energy loss f is the stopping power, in which case the analytical expression reduces simply to ΔE. Note that in the case of the electron mean free path and total scattering probability, the expressions for both the analytical function and the random hinge results are somewhat different from the results described above, as the integrands for those quantities contain the spatial distribution of the collision distances. For the random energy hinge methodology, the probability per unit path of an interaction taking place over a step of length t is given by

$$p(s\!:\!h) = \begin{array}{ll} \Sigma_0 e^{-s\Sigma_0} & s \le h, \\ \Sigma_1 e^{-h\Sigma_0} e^{-(s-h)\Sigma_1} & s > h \end{array} \tag{2.382}$$

where h is the hinge distance, Σ_0 the cross section at the initial energy, and Σ_1 the cross section after the energy hinge. The random hinge mean free path over t is then given as

$$
\begin{aligned}
\lambda_{Eh}(t) &= \frac{1}{P_{Eh}(t)} \int_0^t ds\, s\, p(s) \\
&= \frac{1}{P_{Eh}(t)} \int_0^t ds\, s \int_0^s dh\, p(s\!:\!h)\, p(h) \\
&= \frac{1}{P_{Eh}(t)} \int_0^t ds\, s \int_0^s dh \left[\frac{\Sigma_0(t-s)}{st} e^{-s\Sigma_0} + \frac{\Sigma_1}{t} e^{-h\Sigma_0} e^{-(s-h)\Sigma_1} \right] \\
&= \frac{1}{P_{Eh}(t)} \int_0^t ds\, s \left[\frac{\Sigma_0(t-s)}{t} e^{-s\Sigma_0} + \frac{\Sigma_1 e^{-s\Sigma_1} \left(1 - e^{-s(\Sigma_0 - \Sigma_1)} \right)}{t(\Sigma_0 - \Sigma_1)} \right] \\
&= \frac{1}{P_{Eh}(t)} \left[\frac{1}{\Sigma_0} - \frac{e^{-t\Sigma_1} \left(1 - e^{-t(\Sigma_0 - \Sigma_1)} \right)}{(\Sigma_0 - \Sigma_1)} \left(1 + \frac{1}{t\Sigma_1} \right) + \frac{(\Sigma_0 - \Sigma_1) \left(1 - e^{-t\Sigma_0} \right)}{t\Sigma_0^2 \Sigma_1} \right]
\end{aligned}
\tag{2.383}
$$

where $P_{Eh}(t)$ is the probability that there is a hard collision of any kind over t, $p(h)$ is uniform and given by $1/s$, with the probability for a given s that we have yet to encounter the hinge given by $(t-s)/t$ and the probability that we are past the hinge given by s/t. $P_{Eh}(t)$ is given by

$$
\begin{aligned}
P_{Eh}(t) &= \int_0^t ds \int_0^s dh\, p(s{:}h)\, p(h) & (2.384)\\
&= \int_0^t ds \int_0^s dh \left[\frac{\Sigma_0(t-s)}{st} e^{-s\Sigma_0} + \frac{\Sigma_1}{t} e^{-h\Sigma_0} e^{-(s-h)\Sigma_1} \right]\\
&= \int_0^t ds \left[\frac{\Sigma_0(t-s)}{t} e^{-s\Sigma_0} + \frac{\Sigma_1 e^{-s\Sigma_1}\left(1 - e^{-s(\Sigma_0-\Sigma_1)}\right)}{t(\Sigma_0 - \Sigma_1)} \right]\\
&= 1 - \frac{e^{-t\Sigma_1}\left(1 - e^{-t(\Sigma_0-\Sigma_1)}\right)}{t(\Sigma_0 - \Sigma_1)}
\end{aligned}
$$

Note that the distribution of collision distances, $p(s)$, for the random energy hinge can be seen from the integrand in the above expressions to be

$$
p(s) = \frac{\Sigma_0(t-s)}{t} e^{-s\Sigma_0} + \frac{\Sigma_1 e^{-s\Sigma_1}\left(1 - e^{-s(\Sigma_0-\Sigma_1)}\right)}{t(\Sigma_0 - \Sigma_1)} \tag{2.385}
$$

In the exact case for electrons passing through media with varying cross sections, we have, of course,

$$
\lambda(t) = \frac{1}{P(t)} \int_0^t ds\, s\, \Sigma(s) \exp\left\{ -\int_0^s ds' \Sigma(s') \right\} \tag{2.386}
$$

with the expression for $P(t)$, the probability of any scatter,

$$
P(t) = \int_0^t ds\, \Sigma(s) \exp\left\{ -\int_0^s ds' \Sigma(s') \right\} \tag{2.387}
$$

Expressed in terms of energy loss steps rather than distance these become

$$
\lambda(t) = \int_{E_0}^{E_1} dE\, (R_C(E_0) - R_C(E)) \left|\frac{dE}{dx}\right|^{-1} \Sigma(E) \exp\left\{ -\int_{E_0}^{E} dE' \left|\frac{dE'}{dx}\right|^{-1} \Sigma(E') \right\} \tag{2.388}
$$

and

$$
P(t) = \int_{E_0}^{E_1} dE \left|\frac{dE}{dx}\right|^{-1} \Sigma(E) \exp\left\{ -\int_{E_0}^{E} dE' \left|\frac{dE'}{dx}\right|^{-1} \Sigma(E') \right\} \tag{2.389}
$$

Note that in the above, we have described energy hinge steps in both terms of the change in energy loss (from E_0 to E_1) and also in terms of distance traveled t, as convenient. In EGS5 we use a simple prescription for relating the two and for switching back and forth. For a given initial energy E_0 and a pathlength t, E_1 is given as $E_0 - \Delta E(t)$ with $\Delta E(t)$ computed as follows. A table of electron CSDA ranges $R_C(E)$ is constructed as a function of energy as

$$
R_C(E) = \int_0^E dE' \left|\frac{dE}{dx}\right|^{-1}. \tag{2.390}
$$

Since the CSDA range is uniquely defined monotonic function of energy, its inverse, the energy of an electron with a given CSDA range, $E_C(R)$, can be trivially determined. Thus we have

$$\Delta E(t) = E_0 - E_C(R_C(E_0) - t) \tag{2.391}$$

By interpolating tabulated values of $R_C(E)$ and $E_C(R)$, relating energy loss to distance traveled is straightforward.

Implementation We begin with the energy loss integration, for which case we are looking for the largest ΔE for which

$$1 - \frac{1}{t}\frac{\Delta E}{2}\left(\left|\frac{dE}{dx}(E_0)\right|^{-1} + \left|\frac{dE}{dx}(E_1)\right|^{-1}\right) < \epsilon_E \tag{2.392}$$

where $t = R_C(E_0) - R_C(E_1)$, the pathlength as determined from the range tables, and represents the analytical value we wish to preserve within a relative error tolerance given by ϵ_E. We use an iterative process, beginning with a value of ΔE that is 50% of E_0 and step down in 5% increments until the inequality is satisfied. We next look at scattering power, starting with the value of ΔE required by the stopping power integration. In this case, we numerically compute the integral of stored values of $G_1(E_0)$ times the stopping power for $K_1(\Delta E)$ and compare that value to that from the energy hinge trapezoidal integration,

$$\frac{\Delta E}{2}\left(\left|\frac{dE}{dx}\right|^{-1}(E_0)G_1(E_0) + \left|\frac{dE}{dx}\right|^{-1}(E_1)G_1(E_1)\right)$$

If the difference is greater than ϵ_E, we reduce ΔE by 5% and continue until the difference is less than ϵ_E.

A treatment for the maximum hinge steps which preserve the mean free path (using Equations 2.386 and 2.384) and total scattering probability (using Equations 2.385 and 2.387) for hard collisions to within ϵ_E is still being developed.

2.15.8 Multiple Scattering Step-size Specification Using Fractional Energy Loss Parameters

To control multiple scattering step-sizes, it would seem logical for EGS5 to require specification of $\cos\Theta$, because K_1 is very close to $1 - \langle\cos\Theta\rangle$ (see Equation 2.375). However, because electron scattering power changes (increases) much more rapidly than stopping power as an electron slows below an MeV or so, fixing K_1 for the entire electron trajectory results in taking much, much smaller steps for lower energy electrons than the fixed fractional energy loss method using ESTEPE of EGS4, given the same step size at the initial (higher) energy. For example, the value of K_1 corresponding to a 2% energy loss for a 10 MeV electron in water corresponds to a 0.3% energy loss at 500 keV. Thus a step-size control mechanism based on constant scattering strength would force so many small steps at lower energies that EGS5 could be slower than EGS4 for certain problems,

despite being able to take much longer steps at high energies because of the random hinge transport mechanics. To circumvent this problem, EGS5 reverts to specifying the multiple scattering step-size based on a fractional energy loss over the step, which is now called `EFRACH`, to distinguish it from `ESTEPE` in EGS4. `EFRACH` is material dependent and specified in the PEGS input NAMELIST.

One consequence of the work done to assure that EGS5 expends most of its computational effort where it was most important (*i.e.*, at higher energies), was the recognition that additional efficiency might be achieved by allowing the fractional energy loss of the multiple scattering steps to vary over the energy range of the problem. Thus, in addition to `EFRACH`, which is now defined to be the fractional energy loss over a multiple scattering step at the highest problem energy, a second parameter `EFRACL`, corresponding to the fractional energy loss over a multiple scattering step at the lowest problem energy, is used. Like `EFRACH`, `EFRACL` is also material dependent and is specified in the PEGS input NAMELIST, and the fractional energy loss permitting for multiple-scattering steps swings logarithmically over the energy range of the problem.

Since the computational effort required to solve electron transport problems by Monte Carlo is directly related to the number (and therefore the size) of the multiple scattering steps, an "optimal" step-size in a simulation would therefore be the longest one for which the transport mechanics error is less than the desired accuracy. Users thus need to judiciously select `EFRACH` and `EFRACL` to glean maximum peformance.

In practice, however, optimal selection of step-sizes is difficult because the intricate interplay between electron step-size and accuracy almost always depends on not only the particular quantity of interest being computed, but also on the fineness or granularity of desired output tallies. For example, a simulation of the spatial energy deposition distribution in a voxelized geometry is a much more fine grain simulation than a computation of bulk energy deposited in a large detector module. Even given that optimization of step-size selection necessarily requires some input from the user because of it is inherently problem-dependent, it is clear that even expert users would be unduly burdened by the need to optimally select both `EFRACH` and `EFRACL` in order to maximize the performance of EGS5. To address this problem, a method has been devised which will automatically select step-sizes based on a single user input parameter based on the problem geometry, as described below.

2.15.9 Multiple Scattering Step-Size Optimization Using Media "Characteristic Dimensions"

As optimization always involves a speed/accuracy trade-off, development of an automated method for optimization of step-size selection first requires the adoption of a practical standard defining "accuracy." Perhaps the most severe test of a Monte Carlo program's electron transport algorithm is the "broomstick" problem, in which the tracks of electrons normally incident on the planar faces of semi-infinite right-circular cylinders of progressive smaller radii (to isolate the effects of the transport mechanics, hard-collisions are usually ignored in this problem) are simulated. For cylinders of radii approaching infinitesimal thinness, the average total tracklength of incident electrons before
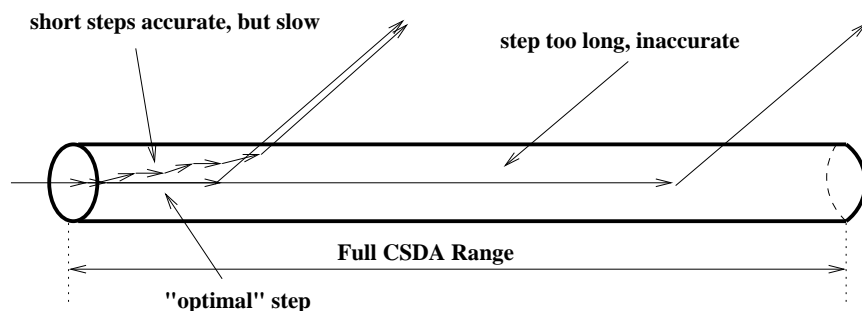
Figure 2.15: Schematic illustrating the "broomstick" problem.

they scatter out of the cylinder is given by the single elastic scattering mean-free path, and any Monte Carlo algorithm using a larger step-size would over-estimate the penetration (and hence average total tracklength) in the cylinder. It should be clear then that for cylinders with arbitrary radii, if we begin with results generated by using very small step-sizes and then gradually increase the steps, we will eventually encounter divergence in the computed average electron track-length inside the cylinder, as our model will at some point over-estimate the penetration prior to deflection out toward the sides of the "broomstick." This is illustrated in Figure 2.15.

Since spatial energy deposition profiles are essentially maps of region-dependent electron tracklengths, and because most problem tallies will be correct if the spatial distribution of electron tracklengths is correct, the largest value of scattering strength K_1 which produces converged results for energy deposition for a given material in a reference geometric volume was initially chosen to be the standard for the EGS5 step-size selection algorithm. Subsequent studies showed, however, that many problems require a more stringent characterization of electron tracks than average length, and so a standard based on the average position of track end points for electrons traversing geometric volumes of given reference sizes was adopted as the standard. In this new adaptation, instead of computing energy deposition in semi-infinite cylinders, we look at the average lateral deflection $\langle r \rangle$ of electrons emerging from the far face of cylinders with lengths L equal to their diameters D. Values of $\langle r \rangle$ over finite cylinders are clearly more sensitive to multiple scattering step-sizes than values of $\langle t \rangle$, leading to more conservative estimates of the maximum permitted scattering strength and thus better assuring accurate results for problems other than those involving energy deposition.

Using this criteria for defining accuracy, tables of the material and energy dependent values of the largest scattering strengths K_1 yielding, for volumes of a given size, values of $\langle r \rangle$ which are within 1% of the converged results for small K_1 were compiled. Given this data, EGS5 is able to provide a step-size control mechanism based on a single user input parameter in units of length which characterizes the geometric granularity of the problem tallies. This value is called the "characteristic dimension" for the problem, and is set in the user's MAIN program by specifying a non-zero value of the variable CHARD, which is material dependent. Given this parameter, EGS5 automatically selects the optimal energy dependent values of K_1 by interpolating the compiled data tables of maximum scattering strengths in dimension, material and energy. The tables range in
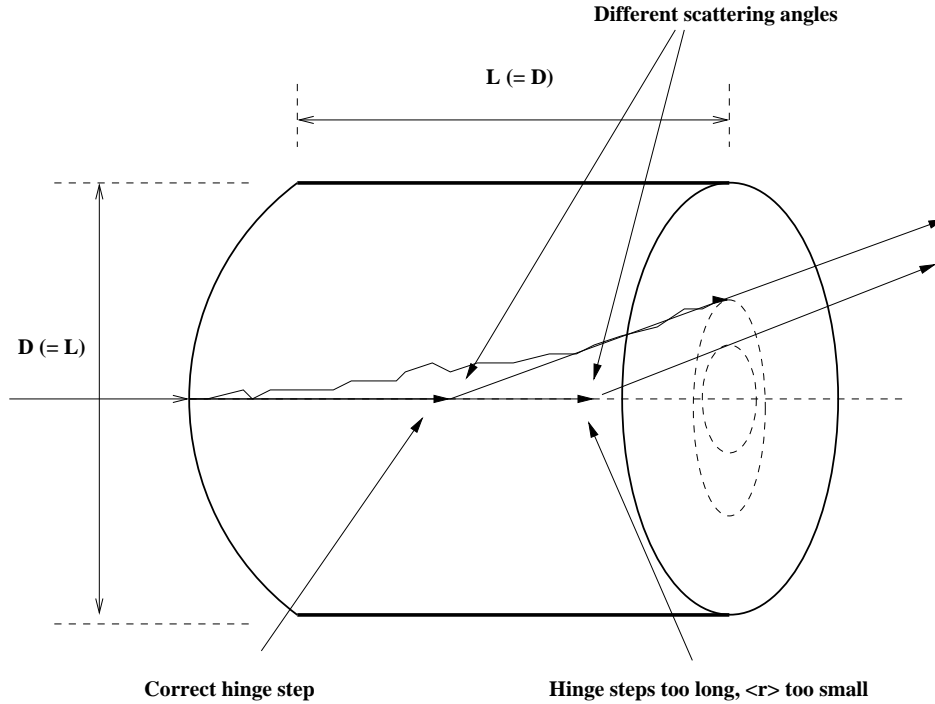
Figure 2.16: Schematic illustrating the modified "broomstick" problem as used in EGS5.

energy from 2 keV[11] up to 1 TeV at values of 2, 3, 5, 7 and 10 in each of the 9 energy decades spanning the energy range. The characteristic geometric dimensions in the data sets range from 10^{-6} times the electron CSDA range at the low end, and up to half of the CSDA range at the high end[12]. (Note that ignoring hard collisions and using unrestricted stopping powers at the upper end of the energy range in question is physically unrealistic. Computations in this energy range were made nonetheless to fill out the tables with overly-conservative estimates of the appropriate step-size.)

Because K_1 is the integral over distance of scattering power, which is proportional to $\rho Z^2/A$ times the integral of the shape of the differential elastic scattering cross section, K_1 should be roughly proportional to $t\rho Z^2/A$, if t is the distance, and that was generally found to be the case. Interpolation in the geometric dimension variable is therefore done in terms of $t\rho$ so that inter-polation between materials can be performed in terms of Z^2/A. (To account for the effect of soft collision electron scattering, interpolations are actually done in terms of $Z(Z+1)$ instead of Z^2.) Positron K_1 values are determined by scaling electron scattering strength by the ratio of the positron and electron scattering power. The list of reference materials is given in Table 2.4.

---

[11]For high Z materials for which the Bethe stopping power formula is inaccurate at 2 keV, the tables stop at 10 keV.

[12]An additional constraint on the minimum characteristic dimension in EGS5 is the smallest pathlength for which the Molière multiple scattering distribution produces viable results. Bethe [23] has suggested that paths which encompass at least 20 elastic scattering collisions are necessary, though EGS5 will compute the distribution using as few as e collisions, which is a numerical limit that simply assures positivity.

Table 2.4: Materials used in reference tables of scattering strength vs. characteristic dimension at various energies.

Material	Z	$Z(Z+1)$	A	ρ	$Z(Z+1)/A$
Li	3	12	6.93900	0.5340	1.7294
C	6	42	12.01115	2.2600	3.4968
H_2O	10	76	18.01534	1.0000	4.2186
Al	13	182	26.98150	2.7020	6.7454
S	16	272	32.06435	2.0700	8.4829
Ti	22	506	47.90000	4.5400	10.5637
Cu	29	870	63.54000	8.9333	13.6922
Ge	32	1056	72.59000	5.3600	14.5475
Zr	40	1640	91.22000	6.4000	17.9785
Ag	47	2256	107.87000	10.5000	20.9141
La	57	3306	138.91000	6.1500	23.7996
Gd	64	4160	157.25000	7.8700	26.4547
Hf	72	5256	178.49000	11.4000	29.4470
W	74	5550	183.85000	19.3000	30.1877
Au	79	6320	196.98700	19.3000	32.0833
U	92	8556	232.03600	18.9000	36.8736

To generate the data sets, then, for each of the 16 reference materials, 45 reference energies, and 29 broomstick lengths and diameters (*i.e.*, characteristic dimension) a series of Monte Carlo simulations were performed, using up to 25 different values of fractional energy loss (called `EFRACH` in EGS5), covering the range from 30% to 0.001% (except when such steps were less than the theoretical lower limits of the Molière distribution). Energy loss hinges were set to the lesser of `EFRACH` and 4% fractional energy loss, and 100,000 histories were simulated, resulting in relative statistical uncertainties in the computed values of $\langle r \rangle$ at 2σ of around 0.3%. Tallies were made of the the average track length inside the volume, the average lateral displacement of the particles escaping the end of the volume, the average longitudinal displacement of particles escaping the sides of the broomstick, and the fractional energy deposited, backscattered and escaping from the side of the broomstick. Computations of the number of hinges expected for the scattering strength being tested given the broomstick dimension, were also made for each run, and the anticipated number of collisions per hinge were also determined and stored.

Illustrative plots showing the divergence in the results as step-sizes are increased in Copper at 5 MeV for several different broomstick thicknesses are shown in Figures 2.17 (results of energy deposition) and 2.18 (results of lateral spread).

Approximately 20,000 such plots were generated from over 500,000 simulations to encompass the desired ranges of materials, energies, and characteristic dimensions. The data was then analyzed to determine the maximum fractional energy loss which showed convergence within the statistical uncertainty of the data, using a least-squares fit to a line with slope zero and intercept given by the
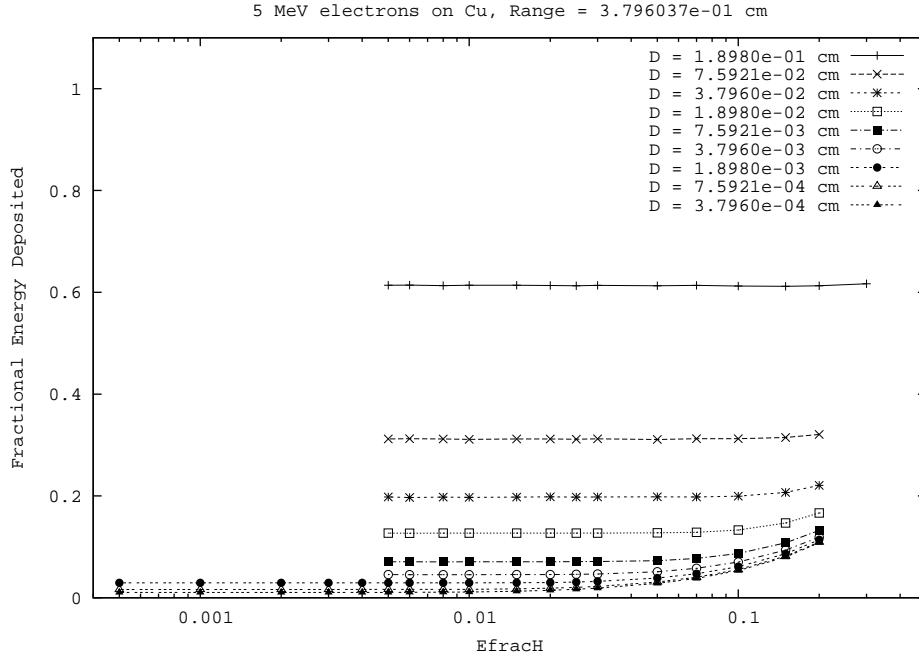
Figure 2.17: Convergence of energy deposition as a function of step-size (in terms of fractional energy loss) for the broomstick problem with varying diameters D in copper at 5 MeV.

converged value at short paths. The initial scattering strengths K_1 corresponding to the determined maximum fractional energy losses were computed using Equation 2.372 cast in terms of an integral over energy instead of over pathlength,

$$K_1(E_0) = \int_{E_1}^{E_0} dE' G_1(E') \left| \frac{dE}{dx} \right|^{-1}, \qquad (2.393)$$

where E_0 is the initial energy, E_1 the energy after the determined maximum fractional energy loss, G_1 the scattering strength and $\left| \dfrac{dE}{dx} \right|$ the stopping power of the medium. Plots of these maximum values of K_1 to assure convergence as a function of broomstick diameter for several energies in titanium are shown in Figure 2.19.

The expected nearly linearly relationship between K_1 and t and the appropriate scaling of K_1 with E is clearly evident in Figure 2.19. We also see in that figure, however, several artifacts of our method. First, we see that our estimation process did not always produce monotonic results, primarily because of noise in the data due to the Monte Carlo statistics (given the large number of runs, some outlier points were to be expected). Additionally, the plots exhibit some discrete jumps because of the finite number of possible K_1 values tested in the parameter study. An additional artifact can be seen in the top plot of Figure 2.20, which shows converged scattering strength data vs. characteristic dimension for a variety of elements at 100 MeV. Because we limited the test runs to a maximum of 30% fractional energy loss, we see a plateauing of the plots for high energies. Other data sets show a corresponding artifact caused by numerical limits on the minimum step-size
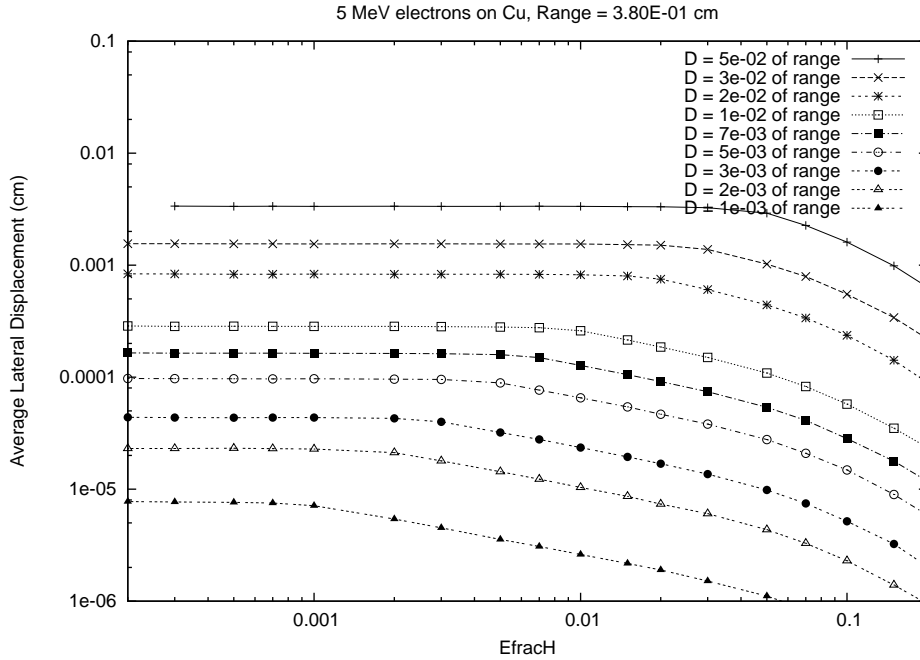
Figure 2.18: Convergence of average lateral displacement as function of step-size (in terms of fractional energy loss) for the broomstick problem with varying diameters D in copper at 5 MeV.

at low energies. Note also from Figure 2.20 that the scaling of K_1 in $Z(Z+1)/A$ rather than Z is evident in the comparison of the plots for water and carbon.

Despite the approximations involved in the definition and determination of convergence, the plots of our computed values of the maximum K_1 which still assures accurate $\langle r \rangle$ as a function of the broomstick dimension exhibits for the most part the behavior we expected. Especially for a given element at a given energy, the log-log K_1 plots can generally be described as being roughly linear in t, though possibly plateauing at either end. Thus each curve (representing one energy for a given element) can be wholly defined by the characteristic dimensions corresponding to the onset (if any) of plateaus at either end, the values of K_1 at those points, and the slope of line on a log-log between those plateau points. All of the curves were inspected numerically and corrected to assure monotonicity, to eliminate the more significant artifacts caused by having a limited set of discrete data points and also to eliminate any physically unrealistic trends in K_1 as a function of t or E for the same material. Using these corrected plots, the five parameters defining each line were determined, with a least squares fit applied to calculate the slopes. The full set of these parameters has been compiled into a single data file K1.dat provided with the EGS5 distribution. Thus, for any material, characteristic dimension, and energy, an easy three-way linear interpolation can be performed to determine the appropriate value of K_1. This is done in a new EGS5 routine RK1 which is called by HATCH, and which maps a piece-wise linear fit of $K_1(E)$ onto the same global energy ladder used by the other electron data variables in EGS.
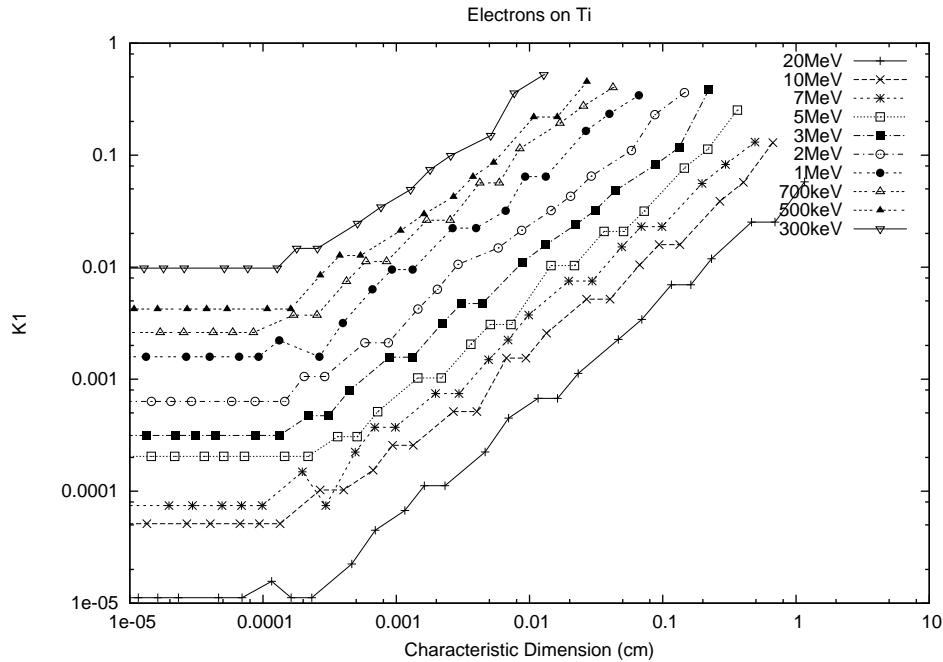
Figure 2.19: Optimal initial scattering strength K_1 vs. broomstick diameter (equivalent to the characteristic dimension) in titanium at various energies.

The sample problem `tutor4` described in the next chapter investigates and illustrates the effectiveness of the automated step-size selection method based on characteristic dimension.

2.15.10 Treatment of Initial Steps of Primary Electrons

It must be noted that not all classes of problems are guaranteed to be modeled accurately whenever the average tracklength or average lateral deflection is modeled correctly in given region volumes. In particular, problems using tallies which have a spatial or directional dependence on secondary particle production occurring prior to the first multiple scattering hinge point (such as deep penetration shower simulations) can exhibit step-size artifacts not present in EGS4, since high energy bremsstrahlung directions can be correlated with electron directions. In such cases, while EGS4 always imposes multiple scattering prior to secondary particle production, the random hinge methodology of EGS5 does not always assure at least some deflection prior to secondary particle generating, sometimes leading to over-estimation of particles in the forward direction.

To counter this problem and still permit EGS5 to take very very long steps as often as possible, for all problems in which secondary particles are being produced and the primary source particles are electrons, a mechanism has been introduced in EGS5 to force very small initial multiple scattering hinge steps. If and only if an electron is determined by EGS subroutine `SHOWER` to be a primary
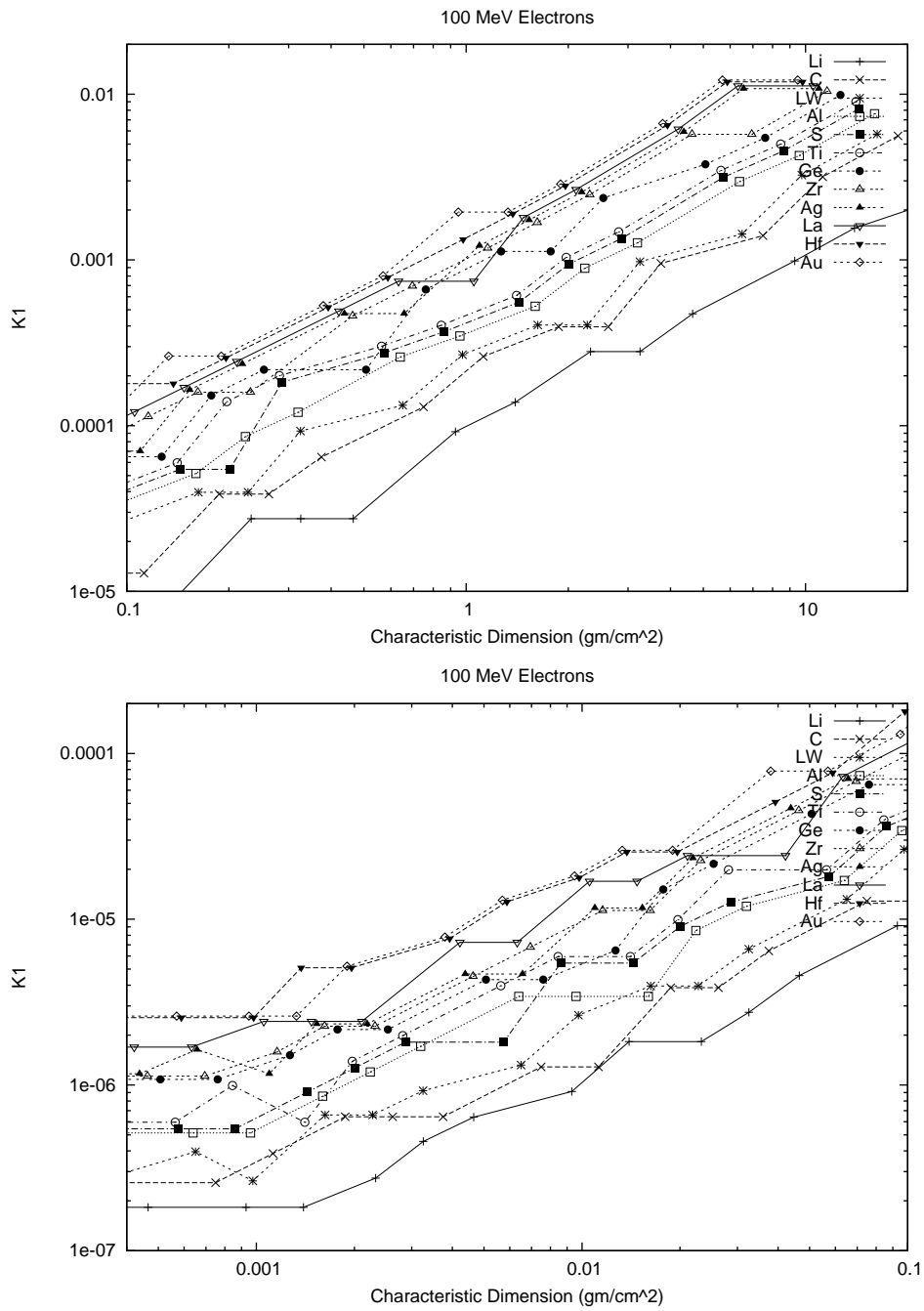
Figure 2.20: Optimal initial scattering strength K_1 vs. broomstick diameter for various elements at 100 MeV. The upper figure is for values of ρt greater than 0.1, and the lower figure for smaller characteristic dimensions.

particle on its very first track, its initial multiple scattering step is automatically set to be that used for the smallest characteristic dimension treated in the data set. (This in effect is usually the distance corresponding to the smallest value of K_1 for which the Molière distribution is defined.) Subsequent multiple scattering steps for such particles are then taken to be the minimum of twice the previous step and the default step given the characteristic dimension of the problem. This approach is still approximate, however, and may be replaced by a single scattering model.

2.16 Photoelectric Effect

The total photoelectric cross sections used in the standard version PEGS4 were taken from Storm and Israel[167]. In PEGS5 we use the more recent compilation in the PHOTX library[131], as originally implemented by Sakamoto[143] as a modification to PEGS4. PHOTX provides data for elements 1 through 100 in units of barns/atom, and PEGS subroutine `PHOTTZ` computes

$$\breve{\Sigma}_{photo,partial}(Z,\breve{k}) = \frac{N_a \rho}{M} \; X_0 \left(\frac{1 \times 10^{-24} \text{ cm}^2}{barn}\right) \; \sigma_{photo}(Z,\breve{k}) \;\; \text{(barns)} \;, \qquad (2.394)$$

where $\sigma_{photo}(Z,\breve{k})$ is obtained by using PEGS function `AINTP` to do a log-log interpolation in energy of the cross sections in the data base. The total cross section, as computed by PEGS routine `PHOTTE`, is given by

$$\breve{\Sigma}_{photo}(\breve{k}) = \sum_{i=1}^{N_e} \; p_i \; \breve{\Sigma}_{photo,partial}(Z_i,\breve{k}) \;. \qquad (2.395)$$

This total photoelectric cross section is then used in the computation of the photon mean free path.

The run-time model of the photoelectric effect was rather simple in early versions of EGS, which treated photoelectric event directly within subroutine `PHOTON`. Starting with EGS3, however, subroutine `PHOTO` was created to provide flexibility in modeling the energies and angles of the ejected secondary electrons. Since some of the photon energy imparted in a photoelectric absorption is consumed in ejecting the electron from its orbit, the kinetic energy of a photoelectron is given by the difference between the incident photon energy and the edge energy of the electron's sub-shell. For applications involving high energy gammas, the edge energy is negligible, but for applications involving photons with energies on the order of several hundred keV or less, treating sub-shell edge energies can be important. Thus the default version of EGS4 provided the weighted average K-edge energy given by

$$\breve{E}_{K-edge} = \frac{\sum_{i=1}^{N_e} \; p_i \; \breve{\Sigma}_{photo}(A_P)\breve{E}_{K-edge}(Z_i)}{\sum_{i=1}^{N_e} \; p_i \; \breve{\Sigma}_{photo}(A_P)} \;. \qquad (2.396)$$

In this implementation, photoelectrons are created with total energy

$$\breve{E} = \breve{k} - \breve{E}_{K-edge} + m, \qquad (2.397)$$

provided, of course, that the initial photon energy is greater than $\breve{E}_{K-edge}$. To preserve the energy balance, a photon of energy $\breve{E}_{K-edge}$ is created and then forcibly discarded. Photoelectric inter-

actions involving photons with energies below $\breve{E}_{K-edge}$ are treated as being completely absorbed, discarded by a call to user routine `AUSGAB` with `IARG=4`.

For applications involving energies on the order of the K-edge energy of the materials being modeled, this treatment is not suitable. Thus, provided with the EGS4 distribution as part of a sample user code, was a substitute version of `PHOTO`[45] which allowed for more explicit modeling of K-shell interactions, including the generation of K_{α_1} and K_{β_1} fluorescent photons. This version of `PHOTO` is the basis for the much more generalized version of `PHOTO` which has become the default in EGS5.

The microscopic photoelectric absorption cross section $\sigma_{photo}(Z, \breve{k})$ of Equation 2.394 is actually the sum over all the constituent atomic sub-shells of the cross section for each sub-shell s which has edge energy less than $\breve{k}$, as in

$$\sigma_{photo}(Z, \breve{k}) = \sum_s \sigma^s_{photo}(Z, \breve{k}) \tag{2.398}$$

where $\sigma^s_{photo}(Z, \breve{k})$ is the photoelectric cross section for sub-shell s of element Z at energy $\breve{k}$.

Evaluation of cross sections near absorption edges   The energy dependence of gamma cross section is modeled in PEGS and EGS using a piece-wise linear fit, which can result in large errors in the vicinity of photon absorption edges. For example, material data created by PEGS for element copper with `UP=1.0` MeV,`AP=0.001` MeV and 200 energy bins produces errors of 60% and 79% in the gamma mean-free path (GMFP) at the energies of the Cu $K_{\beta 1}$ and $K_{\beta 2}$ x-rays as shown in Table 2.5. From Figure 2.21, we can see that the linearly fitted gamma mean free path (GMFP) differs significantly from its exact value in the 64th energy interval in the fitted data, which contains the absorption edge. This GMFP error leads to an underestimation of K_β x-ray production by a factor of 2, as shown in Figure 2.22.

To circumvent this problem, a method called the "local extrapolation method" (LEM), has been devised to specially treat energy intervals containing absorption edges [121]. For such energy intervals, an extrapolation is performed using either the next higher or next lower energy bin, depending on whether the gamma energy is higher or lower than the edge energy. also employed. The gamma mean free path of Cu at $K_{\beta 1}$ and $K_{\beta 2}$ energies evaluated using LEM agree with exact values to within 1% as shown in Table 2.5. Figure 2.21 shows that accurate prediction of the intensity of the characteristic x-rays can be achieved by using LEM. EGS5 employs the LEM method for K-, L1-, L2- and L3- edges by default.

2.16.1   General Treatment of Photoelectric-Related Phenomena

Accurate modeling of the energy of ejected photoelectrons in the general case thus requires resolving the total photoelectric cross section $\sigma_{photo}(Z, \breve{k})$ for a given species into the appropriate sub-shell cross sections, so that the correct sub-shell atomic binding energy can be substituted for $\breve{E}_{K-edge}$ in
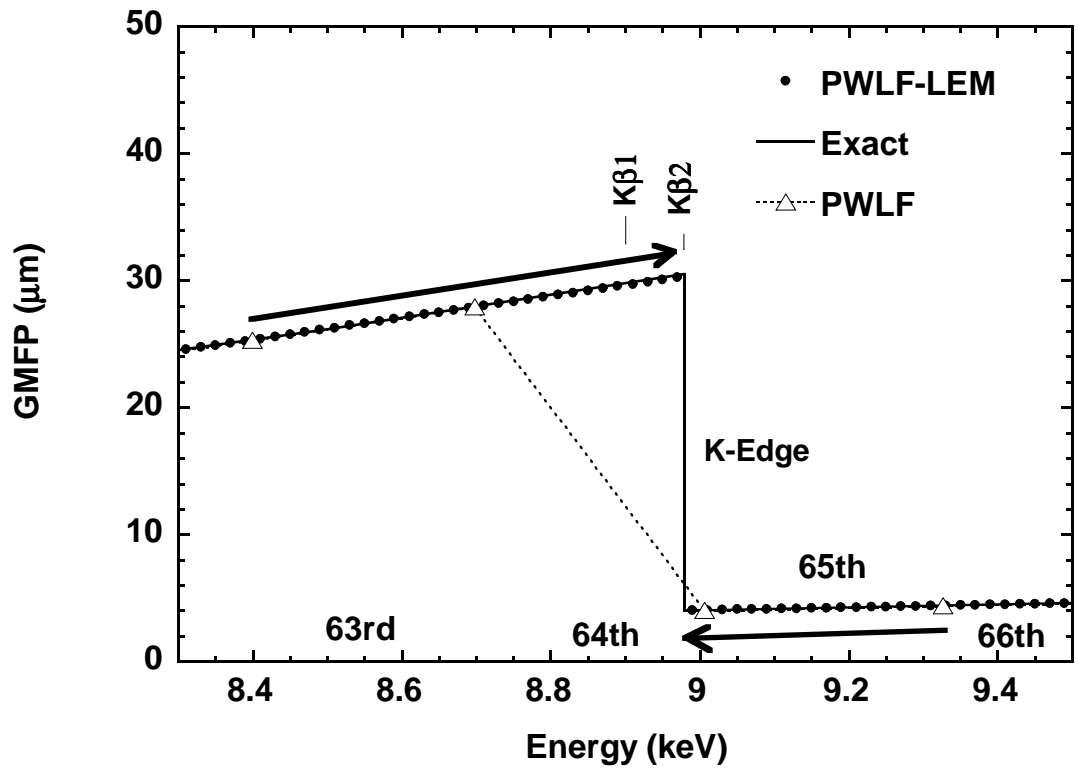
Figure 2.21: Cu GMFP values evaluated by PWLF and LEM
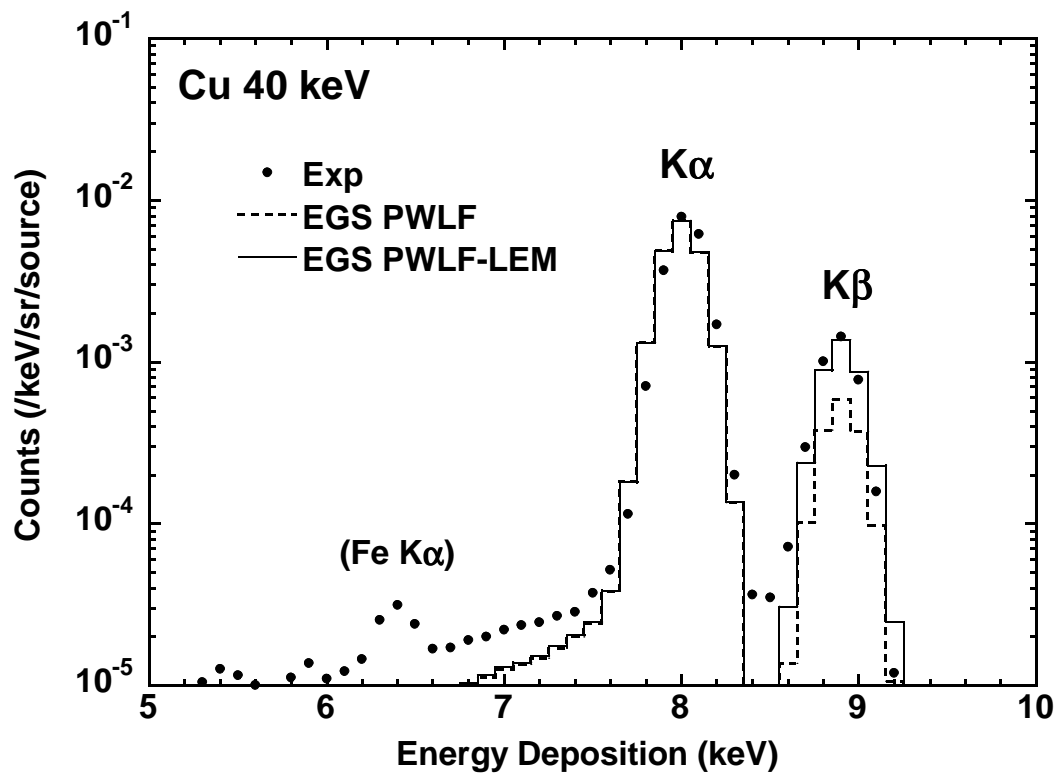
Figure 2.22: Comparison of measured and calculated intensity of K x-rays.

Equation 2.397. Additionally, the vacancies created in atomic sub-shells subsequent to the ejection of photoelectrons can give rise to either characteristic x-rays or Auger electrons (and additional vacancies in lower energy sub-shells) when the atom de-excites. Modeling the photoelectric effect with this level of detail is crucial in many low-energy applications, such as the simulation of detector response in at low energies.

A general treatment of photoelectric-related phenomena in elements, compounds and mixtures was introduced into EGS4, and an improved method has been implemented in EGS5 by Hirayama and Namito[72, 73]. K-, L1-, L2-, L3- and other sub-shell photoelectric cross sections taken from the PHOTX data base are fitted to cubic functions in log-log form,

$$\ln(\sigma_{photo}^s(Z, \breve{k})) = M_0^s(Z) + M_1^s(Z)\ln(\breve{k}) + M_2^s(Z)\ln(\breve{k})^2 + M_3^s(Z)\ln(\breve{k})^3. \tag{2.399}$$

It thus becomes possible to calculate the ratios of sub-shell photoelectric cross sections for each sub-shell of each constituent element of any compound or mixture quickly and accurately inside EGS, rather than approximately via the piece-wise linear fits supplied by PEGS. Once the correct element and sub-shell have been determined (by sampling the discrete distributions of the branching ratios), the photoelectron energy is given by

$$\breve{E} = \breve{k} - \breve{E}_{s-edge}(Z) + m \ , \tag{2.400}$$

given that $\breve{E}_{s-edge}(Z)$ is the binding energy of the s-shell of element Z. Since the sub-shell vacancy is thus known, atomic relaxation can be modeled and additional secondary particles generated based on fluorescence and Auger transition probabilities and energies.

The fitted coefficients $M_0^s(Z), M_1^s(Z), M_2^s(Z)$ and $M_3^s(Z)$ and the other associated data required to model sub-shell level photoelectric effect and secondary particles from atomic relaxation are initialized in a new `BLOCK DATA` subprogram of EGS5. The full data set required and the sources for the data in EGS5 are given in Table 2.6.

Of the more than 50 possible transitions which may occur during the relaxation of L-shell vacancies, 20 of the most important can be modeled in EGS5. All have relative intensities larger than 1% of the $L_{\alpha1}$ transition for Fermium ($Z = 100$). Table 2.7 lists the atomic transitions which produce these x-rays, along with their energies and their intensities relative to the $L_{\alpha1}$ line for lead.

Table 2.5: GMFP of Cu at K_{β_1}(8.905 keV) and K_{β_2}(8.977 keV) energies.

	GMFP (μm)			
	K_{β_1}	(Error)	K_{β_2}	(Error)
Exact*	29.84		30.53	
PWLF	11.80	(-60%)	6.295	(-79%)
PWLF-LEM	29.69	(-0.5%)	30.30	(-0.8%)

* Obtained using CALL option of PEGS.

129

Table 2.6: Data sources for generalized treatment of photoelectric-related phenomena in EGS5.

Data	Source
K-edge energies	Table 2 of Table of Isotopes, Eighth Edition [57]
Probabilities of x-ray emission at K- and L-shell absorption	Table 3 of Table of Isotopes, Eighth Edition
K x-ray energies	Table 7 Table of Isotopes, Eighth Edition
K x-ray emission probabilities	Table 7 Table of Isotopes, Eighth Edition, Adjusted to experimental data by Salem *et al.*[144]
L1, L2, and L3 edge energies	Table 2 of Table of Isotopes, Eighth Edition
L x-ray energies	Table 7b of Table of Isotopes, Eighth Edition (Storm and Israel [167] is used if data is not available in [57])
Probability of Coster-Kronig L1- and L2-shell absorption	Table 3 of Table of Isotopes, Eighth Edition
L x-ray emission probabilities	Theoretical data by Scofield[146], adjusted to experimental data by Salem *et al.*
Average M edge energies	Calculated from sub-shell binding energy in Table 2 of Table of Isotopes, Eighth Edition
Auger electron energies	Calculated neglecting correction term by using atomic electron binding energy in Table 2 of Table of Isotopes, Eighth Edition
K-Auger intensities	$Z = 12 - 17$, Table 1 of Assad[11] $Z > 17$ Table 8 of Table of Isotopes, Eighth Edition
L-Auger intensities	Table 2 from McGuire[102]
$M_0^s(Z), M_1^s(Z), M_2^s(Z)$ and $M_3^s(Z)$	PHOTX[131]

Table 2.7: L x-ray energies and representative intensities (relative to $L_{\alpha 1}$) for lead. The relative intensities in this table are taken from Storm and Israel [167], and were derived using a representative energy. EGS5 explicitly models the energy dependence of the relative frequency at which L-sub-shell vacancies are created.

L X-ray	Energy (keV)	Intensity (%)
L_1-M_2=β_4	12.307	31.6
L_1-M_3=β_3	12.794	34.6
L_1-M_9=β_{10}	13.275	1.15
L_1-M_5=$\beta_{9/1}$	13.377	1.71
L_1-N_2=γ_2	15.097	8.13
L_1-N_3=γ_3	15.216	9.67
L_1-O_2=$\gamma_{4/1}$	15.757	1.59
L_1-O_3=$\gamma_{4/2}$	15.775	1.86
L_2-M_1=η	11.349	3.56
L_2-M_4=β_1	12.614	130.
L_2-N_1=γ_5	14.309	.917
L_2-N_4=γ_1	14.765	26.7
L_2-O_4=γ_6	15.178	3.26
L_3-M_1=l	9.184	5.91
L_3-M_4=α_2	10.450	11.4
L_3-M_5=α_1	10.551	100.
L_3-N_1=β_6	12.142	1.45
L_3-N_4=β_{15}	12.600	2.14
L_3-N_5=β_2	12.622	19.3
L_3-$O_{4,5}$=β_5	13.015	2.57

2.16.2　Photoelectron Angular Distribution

In previous versions of EGS newly created photoelectrons were set in motion in the same direction as the initiating photon. This proved to be too approximate for some applications, and so to address this shortcoming, EGS4 was modified by Bielajew and Rogers to use the theory of Sauter[145] to determine photoelectron angles. Empirical justification for the use of this distribution has been given by Davisson and Evans[50], who showed that it applies even in the non-relativistic realm, despite being derived for relativistic electrons. The implementation of a non-relativistic formula due to Fischer[58] did not significantly impact simulation results, and so Sauter's formula has been applied universally in the EGS code.

Sampling the Sauter angular distribution

The Sauter distribution[145] as given by Davisson and Evans[50] may be integrated over the azimuthal angle and cast in the form:

$$f(\mu)\mathrm{d}\mu = \frac{1 - \mu^2}{(1 - \beta\mu)^4}[1 + K(1 - \beta\mu)]\mathrm{d}\mu \tag{2.401}$$

where

$$
\begin{aligned}
\mu &= \cos\Theta \\
\beta &= v/c \\
\gamma &= 1/\sqrt{1 - \beta^2} \\
K &= (\gamma/2)(\gamma - 1)(\gamma - 2).
\end{aligned}
$$

Here μ is the cosine of the angle that the electron is ejected (with respect to the initiating photon direction), β is the speed of the electron relative to the speed of light and γ is the familiar relativistic factor. Although Equation 2.401 may be integrated easily, its integral can not be inverted analytically and so a direct sampling approach is not feasible. In addition, equation 2.401 may also be very sharply peaked in the forward direction, making rejection sampling inefficient. Therefore, we employ a mixed technique to sample for photoelectron angles. We make the separation:

$$f(\mu) = g(\mu)h(\mu) \tag{2.402}$$

where

$$g(\mu) = \frac{1}{(1 - \beta\mu)^3}[1 + K(1 - \beta\mu)] \tag{2.403}$$

is the directly sampled part, and

$$h(\mu) = \frac{1 - \mu^2}{1 - \beta\mu} \tag{2.404}$$

is sampled via the rejection method. $g(\mu)$ and $h(\mu)$ may be easily normalized so that $\int_{-1}^{1} g(\mu)\mathrm{d}\mu = 1$ and $h(\mu) \leq 1$ for all μ, resulting in efficient sampling. A more complete discussion of photoelectron angular distributions as adapted for EGS may be found elsewhere[30].

Table 2.8: Total cross section (10^{-24} cm^2/molecule) for coherent scattering from water.

Photon Energy (keV)	Free O + 2 Free H$^{(b)}$	Free H$_2$O Molecule	Liquid Water
20	2.65	2.92	2.46
60	0.417	0.444	0.392
100	0.161	0.170	0.151

$^{(a)}$ From Johns and Yaffe [84]. Note that effects on scattering angle are more dramatic than for total cross sections.

$^{(b)}$ Default values used in EGS/PEGS.

2.17 Coherent (Rayleigh) Scattering

The total coherent (Rayleigh) scattering cross sections used in EGS are from Storm and Israel[167] and are available for elements 1 through 100. As with the photoelectric effect cross sections, the data file is in units of barns/atom. The PEGS routine `COHETZ` computes

$$\check{\Sigma}_{coher,partial}(Z,\check{k}) = \frac{N_a \rho}{M} \; X_0 \left(\frac{1 \times 10^{-24} \text{ cm}^2}{\text{barn}} \right) \; \sigma_{coher}(Z,\check{k}) \;\; \text{(barns)} , \qquad (2.405)$$

where $\sigma_{coher}(Z,\check{k})$ is obtained by using the PEGS function `AINTP` to do a log-log interpolation in energy of the Storm and Israel cross sections. To obtain the total cross section, we treat all the atoms as if they act independently. That is, in PEGS routine `COHETM` we compute

$$\check{\Sigma}_{coher}(\check{k}) = \sum_{i=1}^{N_e} p_i \; \check{\Sigma}_{coher,partial}(Z_i,\check{k}) . \qquad (2.406)$$

To permit PEGS to retain complete generality when treating compounds and mixtures, the independent atom approximation is also used when calculating form factors for coherent scattering. However, as shown in Table 2.8, this assumption is known to be poor, as both the molecular structure (row 3 of Table 2.8) and the structure of the medium (row 4 of Table 2.8) can affect coherent scattering [110, 84]. For simulations involving problems in which the correlation effect between molecules on coherent scattering is important, PEGS5 provides for the direct input of interference coherent cross sections and form factors (if such data is available). Interference coherent cross sections and form factors for selected materials, including water, are provided in the EGS5 distribution, and the procedure for invoking this option is described in Appendix C of this report, The PEGS5 User Manual.

Rather than passing $\check{\Sigma}_{coher}(\check{k})$ to the EGS code, PEGS passes the ratio $(\check{\Sigma}_{tot} - \check{\Sigma}_{coher})/\check{\Sigma}_{tot}$, which EGS uses as a correction factor to include coherent scattering only if the user requests it.

Let us now develop a method for sampling the coherent scattering angle. The differential coherent scattering cross section is given by

$$\frac{d\sigma_R(\theta)}{d\Omega} = \frac{r_0^2}{2}(1 + \cos^2 \theta)[F_T(q)]^2 , \qquad (2.407)$$

where r_0 is the classical electron radius. $F_T(q)$ is the total molecular form factor calculated under the independent atoms assumption discussed above. That is,

$$[F_T(q)]^2 = \sum_{i=1}^{N_e} p_i [F(q, Z_i)]^2 \tag{2.408}$$

where $F(q, Z_i)$ is the atomic form factor for element Z_i and the momentum transfer, q, is given by

$$q = 2k \sin \frac{\theta}{2} = \sqrt{2} k (1 - \cos \theta)^{1/2} \; . \tag{2.409}$$

Using $d\Omega = 2\pi d(\cos \theta)$, defining $\mu = \cos \theta$, and with $q^2 = 2k^2(1 - \mu)$, we can write

$$
\begin{aligned}
\frac{d\sigma_R(q^2)}{dq^2} &= \frac{\pi r_0^2}{k^2} \left(\frac{1 + \mu^2}{2} \right) [F_T(q)]^2 \\
&= \frac{\pi r_0^2}{k^2} A(q_{max}^2) \left(\frac{1 + \mu^2}{2} \right) \frac{[F_T(q)]^2}{A(q_{max}^2)} \; ,
\end{aligned}
\tag{2.410}
$$

where

$$A(q_{max}^2) = \int_0^{q_{max}^2} [F_T(q)]^2 \, d(q^2) \; . \tag{2.411}$$

Using this decomposition, we take $[F_T(q)]^2 / A(q_{max}^2)$ as a probability density function and $(1 + \mu^2)/2$ as a rejection function (*e.g.*, see Section 2.2). The variable q^2 is sampled from

$$\zeta_1 = \frac{A(q^2)}{A(q_{max}^2)} \; , \tag{2.412}$$

where ζ_1 is a random number drawn uniformly on the interval $(0, 1)$ and the μ value corresponding to the q^2 value is obtained from

$$\mu = 1 - \frac{q^2}{2k^2} \; . \tag{2.413}$$

If a second random number, ζ_2, is chosen such

$$\frac{1 + \mu^2}{2} \geq \zeta_2 \; , \tag{2.414}$$

then the value of μ is accepted for the scattering angle. Otherwise μ is rejected and the sampling process is repeated. Tabulated values of $F(q, Z)$ given by Hubbell and Overbø[77] have been used in PEGS.

2.18 Binding Effects and Doppler Broadening in Compton Scattering

The treatment of Compton scattering presented earlier in this chapter assumed that atomic electrons in two-body collisions are unbound and at rest. For high energy photons these assumptions

are reasonable, but at lower initial energies, atomic electron binding has the effect of decreasing the Compton scattering cross section given by Equation 2.170, particularly in the forward direction. Additionally, because bound atomic electrons are in motion, they emerge from Compton interactions not at energies wholly defined by the scattering angle as given in Equation 2.173, but with a distribution of possible energies (this effect is usually referred to as "Doppler broadening.") Treatments of both atomic binding effects and Doppler broadening were introduced into EGS4 by Namito and co-workers [119, 117], and all of those methods have been incorporated into the default version of EGS5, as options initiated through flags specified by the user.

To examine the effects of atomic binding and electron motion, we start with a more generalized treatment of photon scattering than that of Klein and Nishina. Ribberfors derived a doubly differential Compton scattering cross section for unpolarized photons impingent on bound atomic electrons using the relativistic impulse approximation [134]. His result can be expressed as

$$\left(\frac{d^2\sigma}{d\Omega d\breve{k}}\right)_{bC,i} = \frac{r_0^2}{2}\left(\frac{\breve{k}_c\breve{k}}{\breve{k}_0^2}\right)\frac{dp_z}{d\breve{k}}\left(\frac{\breve{k}_c}{\breve{k}_0} + \frac{\breve{k}_0}{\breve{k}_c} - \sin^2\theta\right)J_i(p_z) \tag{2.415}$$

where

$$p_z = -137\frac{\breve{k}_0 - \breve{k} - \breve{k}_0\breve{k}(1-\cos\theta)/m}{\hbar c|\vec{\breve{k}}_0 - \vec{\breve{k}}|}, \tag{2.416}$$

$$\frac{dp_z}{d\breve{k}} = \frac{137\breve{k}_0}{\hbar c|\vec{\breve{k}}_0 - \vec{\breve{k}}|\breve{k}_c} - \frac{p_z(\breve{k} - \breve{k}_0\cos\theta)}{(\hbar c)^2|\vec{\breve{k}}_0 - \vec{\breve{k}}|^2}, \tag{2.417}$$

$$\breve{k}_c = \frac{\breve{k}_0}{1 + \frac{\breve{k}_0}{m}(1-\cos\theta)}, \tag{2.418}$$

and

$$\hbar c|\vec{\breve{k}}_0 - \vec{\breve{k}}| = \sqrt{\breve{k}_0^2 + \breve{k}^2 - 2\breve{k}_0\breve{k}\cos\theta}. \tag{2.419}$$

Here, the subscript "bC" denotes *Compton scattering by a bound electron*; subscript "i" denotes the sub-shell number corresponding to the (n, l, m)-th sub-shells; r_0 is the classical electron radius as before; $\breve{k}_0$ and $\breve{k}$ are the incident and scattered photon energies, respectively, and $\breve{k}_c$ is the Compton scattered photon energy for an electron at rest (Equation 2.173); p_z is the projection of the electron pre-collision momentum on the photon scattering vector in atomic units; $J_i(p_z)$ is the Compton profile of the i-th sub-shell[35]; θ is the scattering polar angle; and m is the electron rest mass. Note that as we are dealing with bound electrons, it is implicit in the above that the cross section given by Equation 2.415 is 0 when $\breve{k} > \breve{k}_0 - I_i$, where I_i is the binding energy of an electron in the i-th shell. Note also that by substituting Equation 2.417 into Equation 2.415 after eliminating the second term on the right-hand side of Equation 2.417, one obtains an equivalent formula to Ribberfors' Equation 3 [134].

The singly-differential Compton cross section (in solid angle) for the scattering from a bound electron is obtained by integrating Equation 2.415 over $\breve{k}$ with the assumption that $\breve{k} = \breve{k}_c$ in the

second term on the right-hand side to yield

$$\left(\frac{d\sigma}{d\Omega}\right)_{bC,i} = \frac{r_0^2}{2}\left(\frac{\breve{k}_c}{\breve{k}_0}\right)^2\left(\frac{\breve{k}_c}{\breve{k}_0}+\frac{\breve{k}_0}{\breve{k}_c}-\sin^2\theta\right)S_i^{IA}(\breve{k}_0,\theta,Z),\tag{2.420}$$

where

$$S_i^{IA}(\breve{k}_0,\theta,Z) = \int_{-\infty}^{p_{i,max}} J_i(p_z)dp_z.\tag{2.421}$$

Here, Z is the atomic number and $S_i^{IA}(\breve{k}_0,\theta,Z)$ is the called the incoherent scattering function of the i-th shell electrons in the impulse approximation calculated by Ribberfors and Berggren[135], and $p_{i,max}$ is obtained by putting $\breve{k}=\breve{k}_0-I_i$ in Equation 2.416. Note that $S_i^{IA}(\breve{k}_0,\theta,Z)$ converges to the number of electrons in each sub-shell when $p_{i,max}\to\infty$. The singly-differential Compton cross section of a whole atom is obtained by summing Equation 2.420 for all of the sub-shells,

$$\left(\frac{d\sigma}{d\Omega}\right)_{bC}^{IA} = \frac{r_0^2}{2}\left(\frac{\breve{k}_c}{\breve{k}_0}\right)^2\left(\frac{\breve{k}_c}{\breve{k}_0}+\frac{\breve{k}_0}{\breve{k}_c}-\sin^2\theta\right)S^{IA}(\breve{k}_0,\theta,Z),\tag{2.422}$$

where

$$S^{IA}(\breve{k}_0,\theta,Z) = \sum_i S_i^{IA}(\breve{k}_0,\theta,Z).\tag{2.423}$$

Here, $S^{IA}(\breve{k}_0,\theta,Z)$ is the incoherent scattering function of the atom in the impulse approximation. Note that an alternative computation of the incoherent scattering function based on Waller-Hartree theory[178] and denoted as $S^{WH}(x,Z)$ has been widely used in modeling electron binding effects on the angular distribution of Compton scattered photons. In this representation of the incoherent scattering function, x is the momentum transfer in Å, given by

$$x = \frac{\breve{k}_0(\text{keV})}{12.399}\sin\left(\frac{\theta}{2}\right),\tag{2.424}$$

and equivalent to q of Equation 2.409. Using $S^{WH}(x,Z)$ as the incoherent scattering function, the differential Compton scattering cross section is given by

$$\left(\frac{d\sigma}{d\Omega}\right)_{bC}^{WH} = \frac{r_0^2}{2}\left(\frac{\breve{k}_c}{\breve{k}_0}\right)^2\left(\frac{\breve{k}_c}{\breve{k}_0}+\frac{\breve{k}_0}{\breve{k}_c}-\sin^2\theta\right)S^{WH}(x,Z),\tag{2.425}$$

which is the simply the Klein-Nishina cross section from before multiplied by the incoherent scattering function. Close agreement between $S^{IA}(\breve{k}_0,\theta,Z)$ and $S^{WH}(x,Z)$ for several atoms has been shown by Ribberfors [135], though Namito *et al.*[118] have pointed out differences between S^{IA} and S^{WH} at low energies. As noted earlier, as $S^{WH}(x,Z)$ increases from a value of 0 at $x=0$ to Z as $x\to\infty$, the net effect of atomic binding as defined through the incoherent scattering function is to decrease the Klein-Nishina cross section in the forward direction for low energies, especially for high Z materials.

The total bound Compton scattering cross section of an atom can be obtained by integrating Equation 2.425 over the solid angle (Ω),

$$\sigma_{bC}^{WH} = \int^{4\pi}\left(\frac{d\sigma}{d\Omega}\right)_{bC}^{WH}d\Omega.\tag{2.426}$$

Implementation in EGS5   If the user requests that binding effects be taken into account, The total Compton scattering cross section σ_{bC} from Equation 2.426 is used in computing the total photon scattering cross section, with values taken from the DLC-99/HUGO[142, 78, 77] library. If the user further requests that incoherent scattering functions be used in determining Compton scattering angles, θ is sampled according to Equation 2.425, using $S^{WH}(x, Z)$ (taken from HUGO [142] as a rejection function. (Note that the methodology presented earlier for sampling the Klein-Nishina distribution sampled the scattered photon energy, rather than the scattering angle.)

If the user further requests that shell-wise Compton profiles be used to simulate Doppler broadening, θ is determined as above and then the electron sub-shell number i is randomly selected, taken in proportion to the number of electrons in each sub-shell. Next, $p_{i,max}$ is calculated by setting $\breve{k} = \breve{k}_0 - I_i$ in Equation 2.416 and using the sampled θ. Then, p_z is sampled in the interval $(0, 100)$ from the normalized cumulative density function of $J_i(x)$ for the i-th sub-shell, according to

$$\zeta = F_i(p_z) = \frac{\int_o^{p_z} J_i(p_z')dp_z'}{\int_0^{100} J_i(p_z')dp_z'}. \tag{2.427}$$

where ζ is a random number between $(0, 1)$. PEGS computes and prints piece-wise linear fits of $F_i^{-1}(\zeta)$, the inverse of $F_i(p_z)$, so that p_z can be determined trivially as $p_z = F_i^{-1}(\zeta)$. The value of "100" in Equation 2.427 comes from the upper limit of the p_z values taken from the Compton profile data in Biggs [35]. The speed penalty involved in using shell-wise Compton scattering in EGS5 is negligible, as the sampling of p_z is rapid. Note that when $p_z > p_{i,max}$, it is rejected, and the sub-shell number i and p_z are sampled again. Lastly, another rejection by $\frac{\breve{k}}{\breve{k}_0}$, which corresponds to the second term on the right-hand side of Equation 2.415, is performed.

Limitations of EGS5 in modeling bound Compton scattering and Doppler broadening
The following theoretical restrictions apply to the detailed model of Compton scattering in EGS5:

- The Compton profile of a free atom is used. To treat compounds, an amorphous mixture of free atoms is assumed, and any molecular effects on the Compton profiles are ignored.

- The Compton scattered electron energy and direction are calculated using the energy and momentum conservation laws assuming that no energy absorption by atoms in the Compton scattering occurs. The electron binding energy is then subtracted from the recoil electron energy. Electron binding energy is deposited locally with `IARG=4`.

2.19   Scattering of Linearly Polarized Photons

All of the treatments of Compton and Rayleigh photon scattering presented earlier in this chapter have assumed that the incoming photons are unpolarized. Since the scattering of polarized photons is not isotropic in the azimuthal angle, simulations involving polarized photons will not be accurate using the methods described above. To overcome this limitation, Namito *et al.* [116] developed a
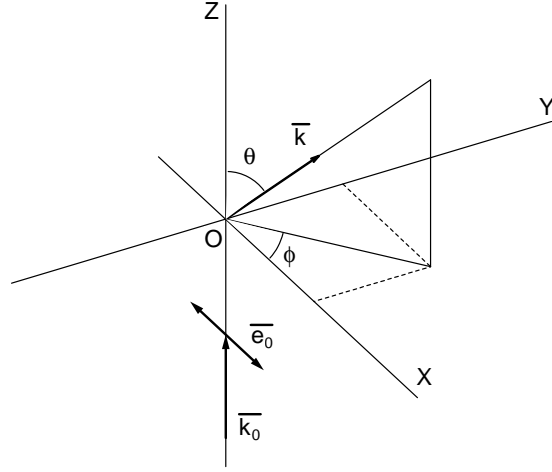
Figure 2.23: Photon scattering system.   An incident photon toward the Z-direction is scattered at point O. The propagation vector $\vec{k_0}$ and polarization vector $\vec{e_0}$ of an incident photon are parallel to $\vec{e_z}$ and $\vec{e_x}$, respectively. Here, $\vec{e_z}$ and $\vec{e_x}$ are unit vectors along the z- and x-axis. The scattering polar angle is θ and the scattering azimuth angle from the plane of $\vec{e_0}$ is ϕ. The scattered propagation vector is $\vec{k}$.

method for modeling the scattering of linearly polarized photons in EGS4, and that treatment has been included in EGS5.

Consider a photon scattering system of Figure 2.23, in which a completely linearly polarized photon, whose propagation vector and polarization vector are $\vec{k_0}$ and $\vec{e_0}$, is scattered at point O, The propagation vector of the scattered photon is $\vec{k}$, and the polar and azimuth scattering angles are θ and ϕ. Using the methodology of Heitler [71], we consider two components of the direction vector $\vec{e}$, one in the same plane as $\vec{e_0}$ (which we denote as $\vec{e_\parallel}$) and the other component perpendicular to the plane of $\vec{e_0}$, (called $\vec{e_\perp}$). Figure 2.24 shows $\vec{e_\parallel}$ in the plane S defined by $\vec{k}$ and $\vec{e_0}$, and $\vec{e_\perp}$ perpendicular to the plane S. Under the condition that $\vec{k_0} \parallel \vec{e_z}$ and $\vec{e_0} \parallel \vec{e_x}$ (as shown in Figures 2.23 and 2.24), these two polarization vectors, $\vec{e_\parallel}$ and $\vec{e_\perp}$, can be determined to be the following functions of θ and ϕ :

$$\vec{e_\parallel} = N\vec{e_x} - \left(\frac{1}{N}\sin^2\theta\cos\phi\sin\phi\right)\vec{e_y} - \left(\frac{1}{N}\cos\theta\sin\theta\cos\phi\right)\vec{e_z} \tag{2.428}$$

and

$$\vec{e_\perp} = \left(\frac{1}{N}\cos\theta\right)\vec{e_y} - \left(\frac{1}{N}\sin\theta\sin\phi\right)\vec{e_z}. \tag{2.429}$$

Here, $N = \sqrt{\cos^2\theta\cos^2\phi + \sin^2\phi}$, $\vec{e_x}, \vec{e_y}, \vec{e_z}$ are unit vectors along the x-, y- and z-axis, respectively, and $\vec{e_\parallel}$ and $\vec{e_\perp}$ are treated as normalized vectors.

The Compton scattering cross section for linearly polarized photons   Ribberfors derived a doubly differential Compton scattering cross section for an unpolarized photon using the relativis-

138
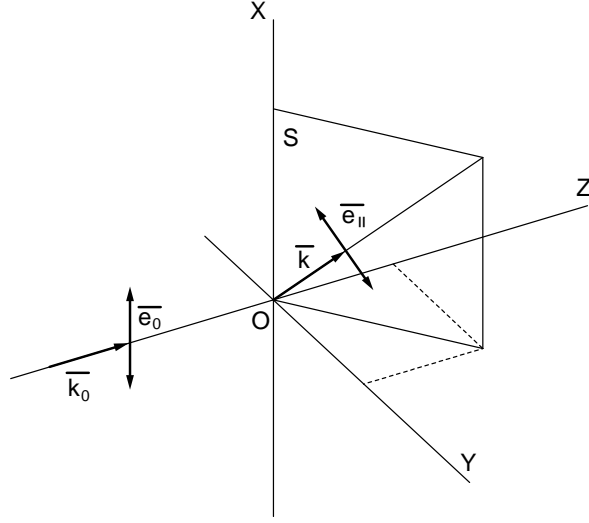
Figure 2.24: Direction of the polarization vector of the scattered photon. Plane S contains $\vec{e_0}$ and $\vec{k}$. $\vec{e_\parallel}$ is in plane S, and is perpendicular to $\vec{k}$. $\vec{e_\perp}$ is perpendicular to plane S.

tic impulse approximation [134]. By modifying Ribberfors' formula, a doubly differential Compton scattering cross section for a linearly polarized photon can be derived as:

$$\left(\frac{d^2\sigma}{d\Omega dk}\right)_{bC,i} = \frac{r_0^2}{4}\left(\frac{k_c k}{k_0^2}\right)\left(\frac{k_c}{k_0} + \frac{k_0}{k_c} - 2 + 4\cos^2\Theta\right) \times \frac{dp_z}{dk}J_i(p_z), \qquad (2.430)$$

where Θ is the angle between the incident polarization vector $\vec{e_0}$ and the scattered polarization vector $\vec{e}$. Note that Equation 2.415 can be obtained by integrating Equation 2.430 over the azimuthal angle, and that the Compton-scattering cross section of a free electron (*i.e.*, ignoring binding effects) for linearly polarized photons is obtained by integrating Equation 2.430 over $\check{k}$ and putting $I_i = 0$,

$$\left(\frac{d\sigma}{d\Omega}\right)_{fC} = \frac{1}{4}r_0^2\left(\frac{\check{k}_c}{\check{k}_0}\right)^2\left(\frac{\check{k}_c}{\check{k}_0} + \frac{\check{k}_0}{\check{k}_c} - 2 + 4\cos^2\Theta\right). \qquad (2.431)$$

The Rayleigh scattering cross section for linearly polarized photons The Rayleigh scattering cross section for linearly polarized photons is the product of the square of the atomic form factors (as defined earlier in Equation 2.408) and the single electron elastic scattering (Thomson) cross section for linear polarized photons. The polarized Thomson scattering cross section per electron is given by [86]

$$\left(\frac{d\sigma}{d\Omega}\right)_T = r_0^2\cos^2\Theta, \qquad (2.432)$$

and so the polarized Rayleigh scattering cross section is thus

$$\left(\frac{d\sigma_R}{d\Omega}\right)_R = r_0^2\cos^2\Theta[F_T(q)]^2. \qquad (2.433)$$

139

Note the similarities and differences between this expression and Equation 2.407 for the unpolarized Rayleigh cross section.

The parallel and perpendicular components of the single electron Compton and Thomson cross sections are given by

$$
\left(\frac{d\sigma}{d\Omega}\right)_{fC,\|} = \frac{1}{4}r_0^2\left(\frac{\breve{k}_c}{\breve{k}_0}\right)^2\left\{\frac{\breve{k}_c}{\breve{k}_0} + \frac{\breve{k}_0}{\breve{k}_c} - 2\right.
$$
$$
\left. +4\left(1 - \sin^2\theta\cos^2\phi\right)\right\}, \tag{2.434}
$$

$$
\left(\frac{d\sigma}{d\Omega}\right)_{fC,\perp} = \frac{1}{4}r_0^2\left(\frac{\breve{k}_c}{\breve{k}_0}\right)^2\left(\frac{\breve{k}_c}{\breve{k}_0} + \frac{\breve{k}_0}{\breve{k}_c} - 2\right), \tag{2.435}
$$

and

$$
\left(\frac{d\sigma}{d\Omega}\right)_{T,\|} = r_0^2(1 - \sin^2\theta\cos^2\phi), \tag{2.436}
$$

$$
\left(\frac{d\sigma}{d\Omega}\right)_{T,\perp} = 0. \tag{2.437}
$$

By adding Equations 2.434 and 2.435, Equations 2.436 and 2.437, respectively, the following Compton scattering cross section and the Thomson scattering cross section for θ and ϕ are obtained:

$$
\left(\frac{d\sigma}{d\Omega}\right)_{fC} = \frac{1}{2}r_0^2\left(\frac{\breve{k}_c}{\breve{k}_0}\right)^2\left(\frac{\breve{k}_c}{\breve{k}_0} + \frac{\breve{k}_0}{\breve{k}_c} - 2\sin^2\theta\cos^2\phi\right) \tag{2.438}
$$

and

$$
\left(\frac{d\sigma}{d\Omega}\right)_{T} = r_0^2(1 - \sin^2\theta\cos^2\phi). \tag{2.439}
$$

Sampling of the scattering azimuth angle and polarization vector The azimuthal angle ϕ is sampled for the determined θ according to Equations 2.438 and 2.439. The direction of the scattered polarization vector is then calculated. It is shown in Equations 2.434 and 2.435 that a completely linearly polarized photon is de-polarized in Compton scattering according to some de-polarization probability. This de-polarization probability, $1 - P$, is

$$
1 - P = \left(\frac{\breve{k}_c}{\breve{k}_0} + \frac{\breve{k}_0}{\breve{k}_c} - 2\right)\bigg/\left(\frac{\breve{k}_c}{\breve{k}_0} + \frac{\breve{k}_0}{\breve{k}_c} - 2\sin^2\theta\cos^2\phi\right). \tag{2.440}
$$

In Compton scattering, either a polarized or de-polarized photon is sampled according to this de-polarization probability. When the scattered photon is polarized, the direction of the polarization vector is calculated according to Equations 2.428. When the scattered photon is de-polarized, the direction of the polarization vector is sampled from the direction between $\vec{e}_\|$ and $\vec{e}_\perp$, shown as Equations 2.428 and 2.429, respectively, at random. In Rayleigh scattering, since there is no probability for de-polarization, the direction of $\vec{e}$ is always calculated according to Equation 2.428.

Transformation to the laboratory system In the scattering system used here, $\vec{k_0}$ and $\vec{e_0}$ are in the direction of $\vec{e_z}$ and $\vec{e_x}$, respectively, as shown in Figure 2.23, whereas $\vec{k_0}$ and $\vec{e_0}$ may be in an arbitrary direction in the laboratory system. The scattering and laboratory systems are connected via three rotations, which are calculated from the direction of $\vec{k_0}$ and $\vec{e_0}$ in the laboratory system. Using these three rotations, $\vec{k}$ and $\vec{e}$ are transformed from the scattering system to the laboratory system.

Here, we describe the relation of the laboratory system, which is used in the default EGS4 simulation, to the scattering system used in Compton and Rayleigh scattering routines for linearly polarized photons.

The laboratory system, in which $\vec{k_0}$ and $\vec{e_0}$ are in arbitrary directions, and the scattering system, in which $\vec{k_0}$ is parallel to $\vec{e_z}$ and $\vec{e_0}$ is parallel to $\vec{e_x}$, is transformed to each other by three rotations. $\vec{e_z}$ and $\vec{e_x}$ are unit vectors parallel to the z- and x-axes. Two rotations are necessary to make $\vec{k_0} \parallel \vec{e_z}$. These rotations were described by Cashwell and Everett [42]. The default version of EGS4 already treats these rotations. Using Cashwell's notation, this $\mathbf{A}^{-1}$ matrix is

$$\mathbf{A^{-1}} = \begin{pmatrix} \frac{uw}{\rho} & \frac{vw}{\rho} & -\rho \\ \frac{-v}{\rho} & \frac{u}{\rho} & 0 \\ u & v & w \end{pmatrix}. \tag{2.441}$$

Here, $\rho = \sqrt{1 - w^2}$. This matrix is written with an inverse sign, since $\mathbf{A}$ is mainly used for a transformation from the scattering system to the laboratory system. It is clear that

$$\mathbf{A}^{-1} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{2.442}$$

In the laboratory system, $\vec{e_0} \perp \vec{k_0}$; rotation by $\mathbf{A}^{-1}$ does not change this relation. As $\mathbf{A}^{-1}$ makes $\vec{k_0} \parallel \vec{e_z}$, $\mathbf{A}^{-1}$ moves $\vec{e_0}$ onto the x-y plane. In Figure 2.25, $\vec{k_0}$ and $\vec{e_0}$ after two rotations by the $\mathbf{A}^{-1}$ matrix is shown. Another rotation by an angle $(-\omega)$ along the z-axis is necessary to make $\vec{e_0} \parallel \vec{e_x}$. The $\cos \omega$ and $\sin \omega$ are calculated using

$$\mathbf{A}^{-1} \vec{e_0} = \begin{pmatrix} \cos \omega \\ \sin \omega \\ 0 \end{pmatrix}. \tag{2.443}$$

By these three rotations, $\vec{k_0}$ and $\vec{e_0}$ in the laboratory system are transferred to those in the scattering system. The scattered photon propagation vector ($\vec{k}$) and the polarization vector ($\vec{e}$) are transferred from the scattering system to the laboratory system by an inverse of these three rotations after Compton or Rayleigh scattering.

The relationship of $\vec{k_0}$, $\vec{k}$, $\vec{e_0}$ and $\vec{e}$ in laboratory system and those in scatter system are:

$$\begin{aligned} \vec{k_0}(\text{lab}) &= \mathbf{A} \cdot \mathbf{B} \cdot \vec{k_0}(\text{scatter}), \\ \vec{e_0}(\text{lab}) &= \mathbf{A} \cdot \mathbf{B} \cdot \vec{e_0}(\text{scatter}), \\ \vec{k}\,(\text{lab}) &= \mathbf{A} \cdot \mathbf{B} \cdot \vec{k}\,(\text{scatter}), \\ \vec{e}\,(\text{lab}) &= \mathbf{A} \cdot \mathbf{B} \cdot \vec{e}\,(\text{scatter}), \end{aligned} \tag{2.444}$$

141
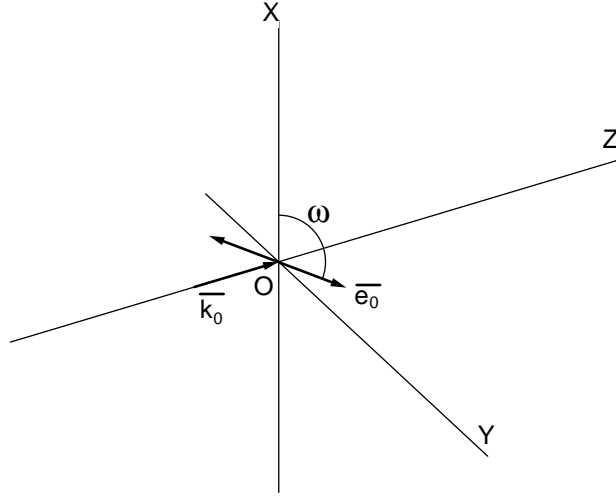
Figure 2.25: Direction of $\vec{k_0}$ and $\vec{e_0}$ after two rotations by $\mathbf{A}^{-1}$. While $\vec{k_0}$ is already parallel to $\vec{e_z}$, another rotation along the z-axis is necessary to make $\vec{e_0} \parallel \vec{e_x}$.

where

$$\mathbf{B} = \begin{pmatrix} \cos\omega & -\sin\omega & 0 \\ \sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.445}$$

Limitations of the present code for modeling linearly polarized photon scattering

- Circularly polarized photon scattering is ignored and elliptically polarized photon scattering is treated as partially linearly polarized photon scattering.

- Characteristic x-rays and bremsstrahlung photons are assumed to be unpolarized.

The formulas used in each simulation mode are summarized in Table 2.9.

2.20 Electron Impact Ionization

Because of the interest in modeling the generation in the production of characteristic radiation, it is desirable to treat explicit δ-ray collisions involving inner shells atomic electrons. Inelastic electron scattering collisions which result in the ejection of a bound atomic electron is typically called electron impact ionization (EII), and a modification to EGS4 by Namito *et al.* [120] allowing the treatment of K-shell electron impact ionization has been retained in EGS5. Six different cross sections describing EII are available, as given below. A detail discussion of the cross sections and a guide for selecting the most appropriate one for given applications can be found in [120].

Table 2.9: Formulas used in various simulation modes employing detailed treatment of Compton and Rayleigh scattering.

Equation	Simulation mode
2.430	Compton scattering with LP, σ_{bC}, $S(x, Z)$ and DB.
2.431	Compton scattering with LP.
2.432	Rayleigh scattering with LP.
2.415	Compton scattering with σ_{bC}, $S(x, Z)$ and DB.
2.425	Compton scattering with σ_{bC} and $S(x, Z)$.
2.426	Compton scattering with σ_{bC}.

1. Casnati [43, 44]

2. Kolbenstvedt-revised [105]

3. Kolbenstvedt-original [92]

4. Jakoby [80]

5. Gryziński [65] Equation 21

6. Gryziński-relativistic [65] Equation 23

EII is treated as a subset of Møller scattering in EGS, so neither the electron mean-free path nor the stopping power are modified when EII is treated. Molecular binding effects are ignored, and electron impact ionization in L and higher shells is treated as free electron Møller scattering.

The ratio of the K-shell EII cross section of J-th element in a material to the Møller scattering cross section is calculated by the following equation:

$$R(E, J) = \frac{\sum_{j=1}^{J} \Sigma_{\text{EII},j}(E)}{\Sigma_{\text{Moller}}(E)}, \tag{2.446}$$

$$\Sigma_{\text{EII},j}(E) = p_j \, \sigma_{\text{EII},j}(E) \, \rho \frac{N_0}{W}, \tag{2.447}$$

where
$R(E, J)$ = the cumulative distribution function of the ratio of the K-shell EII cross section of the J-th element in a material to the Møller scattering cross section at electron energy E,
$\Sigma_{\text{Moller}}(E)$ = macroscopic Møller scattering cross section at electron energy E,
$\Sigma_{\text{EII},j}(E)$ = macroscopic EII cross section of the j-th element at electron energy E,
$\sigma_{\text{EII},j}(E)$ = microscopic EII cross section of the j-th element at electron energy E,
p_j = proportion by number of the j-th element in the material,
ρ = density of a material,
N_0 = Avogadro's number,

W = Atomic, molecular and mixture weight for an element, for a compound, and for a mixture.

In EGS, K-shell vacancy creation by EII is sampled using $R(E, J)$ values calculated in PEGS and included in the material data file.

K-x ray emission and energy deposition   After K-shell vacancy creation by EII, emission of K-x rays is sampled using the K-shell fluorescence yield [57], as in the treatment of the photoelectric effect [73]. However, in the current EGS implementation, neither Auger electron emission nor atomic relaxation cascades are modeled for K-shell vacancies generated by EII. K-shell x-ray energies are sampled from the ten possible lines given in [57], and the difference between the K-shell binding energy E_B and the emitted x-ray energy is deposited locally. In the case when no x-ray is generated (*i.e.*, an Auger electron emission occurred), the full binding energy E_B is deposited locally.

Secondary electron energies and angles   The energy and direction of ejected electrons following EII are treated in an approximate manner. The binding energy E_B is subtracted at random from the energy of either one of the two electrons after energies have been determined from the standard Møller scattering analysis. In the case that neither of the two electron has kinetic energy greater than E_B, E_B is subtracted from the energy of both electrons while keeping the ratio of the kinetic energies unchanged. The directions of electrons after EII are those given from the Møller scattering collision analysis.

Chapter 3

A SERIES OF SHORT EGS5 TUTORIALS

EGS is a powerful system which can be used to produce very complex Monte Carlo simulations. In spite of some 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 the user interface in what we refer to as "EGS5user codes." In these user codes we will introduce some basic scoring techniques. Formal documentation in the form of EGS5 and PEGS user manuals can be found in Appendices B and C, respectively.

These tutorials are written under the assumption that the reader is generally familiar with the contents of the EGS5 and PEGS user manuals, although a complete understanding of the manuals 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 extensions (generally of a bookkeeping nature), a wide range of powerful programs can be constructed from these tutorial examples. For brevity, we sometimes present only partial source listings of these user codes in the following sections. The complete source code for each tutorial can be found in the EGS5 distribution. Note also that the results from these tutorial programs may be slightly different on machines with different word lengths, different floating-point hardware, or different compiler optimizations.

3.1   Tutorial 1 (Program tutor1.f)

The geometry of the first seven 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 traveling 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 EGS5 to answer the question "What comes out the far side of the plate?", we have created the user code (**tutor1.f**) shown below. Also provided is the PEGS5 input file required for this run (see Appendix C for a description of how to construct PEGS5 input files).

```
!**************************************************************************
!
!                      *************
!                      *           *
!                      *  tutor1.f *
!                      *           *
!                      *************
!
!  An EGS5 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 SLAC-R-730/KEK Report 2005-8: A simple example which 'scores'
!  by listing particles
!
!  The following units are used: unit 6 for output
!**************************************************************************
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
!------------------------------------------------------------------------
!---------------------------- main code ---------------------------------
!------------------------------------------------------------------------


!------------------------------------------------------------------------
! Step 1: Initialization
!------------------------------------------------------------------------

      implicit none

!       ------------
!       EGS5 COMMONs
!       ------------
      include 'include/egs5_h.f'                    ! Main EGS "header" file

      include 'include/egs5_bounds.f'
      include 'include/egs5_media.f'
      include 'include/egs5_misc.f'
      include 'include/egs5_thresh.f'
      include 'include/egs5_useful.f'
      include 'include/egs5_usersc.f'
      include 'include/randomm.f'

!       bounds contains ecut and pcut
!       media contains the array media
!       misc contains med
!       thresh contains ae and ap
!       useful contains RM
```

146

```fortran
!     usersc contains emaxe

      common/geom/zbound
      real*8 zbound
!     geom passes info to our howfar routine

      real*8 ein,xin,yin,zin,                       ! Arguments
     *       uin,vin,win,wtin
      integer iqin,irin

      integer i,j                                   ! Local variables
      character*24 medarr(1)

!     ----------
!     Open files
!     ----------
      open(UNIT= 6,FILE='egs5job.out',STATUS='unknown')

!     ===================
      call counters_out(0)
!     ===================


!-------------------------------------------------------------------------
! Step 2: pegs5-call
!-------------------------------------------------------------------------
!     =============
      call block_set              ! Initialize some general variables
!     =============

!     ---------------------------------
!     define media before calling PEGS5
!     ---------------------------------
      nmed=1
      medarr(1)='TA                      '

      do j=1,nmed
        do i=1,24
          media(i,j)=medarr(j)(i:i)
        end do
      end do

! nmed and dunit default to 1, i.e. one medium and we work in cm

      chard(1) = 0.1d0        !  optional, but recommended to invoke
                             !  automatic step-size control


!     -----------------------------------------------
!     Run KEK version of PEGS5 before calling HATCH
!     (method was developed by Y. Namito - 010306)
!     -----------------------------------------------
```

```
      write(6,100)
100   FORMAT(' PEGS5-call comes next'/)


!     ==========
      call pegs5
!     ==========


!-----------------------------------------------------------------------
! Step 3: Pre-hatch-call-initialization
!-----------------------------------------------------------------------
      nreg=3
!     nreg : number of region

      med(1)=0
      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


!     ------------------------------------------------------------
!     Random number seeds.  Must be defined before call hatch
!     or defaults will be used.  inseed (1- 2^31)
!     ------------------------------------------------------------
      luxlev = 1
      inseed=1
      write(6,120) inseed
120   FORMAT(/,' inseed=',I12,5X,
     *           ' (seed for generating unique sequences of Ranlux)')


!     =============
      call rluxinit  ! Initialize the Ranlux random-number generator
!     =============


!-----------------------------------------------------------------------
! Step 4:  Determination-of-incident-particle-parameters
!-----------------------------------------------------------------------
! Define initial variables for 20 MeV beam of electrons incident
! perpendicular to the slab
      iqin=-1
!           Incident charge - electrons
!           20 MeV kinetic energy
      ein=20.0d0 + RM
      xin=0.0
      yin=0.0
      zin=0.0
!     Incident at origin
```

148

```
      uin=0.0
      vin=0.0
      win=1.0
!           Moving along z axis
      irin=2
!           Starts in region 2, could be 1
!           weight = 1 since no variance reduction used
      wtin=1.0
!     Weight = 1 since no variance reduction used


!-------------------------------------------------------------------------
! Step 5:   hatch-call
!-------------------------------------------------------------------------
! Maximum total energy of an electron for this problem must be
! defined before hatch call
      emaxe = ein

      write(6,130)
130   format(/' Start tutor1'/' Call hatch to get cross-section data')

!     ----------------------------
!     Open files (before HATCH call)
!     ----------------------------
      open(UNIT=KMPI,FILE='pgs5job.pegs5dat',STATUS='old')
      open(UNIT=KMPO,FILE='egs5job.dummy',STATUS='unknown')

      write(6,140)
140   FORMAT(/,' HATCH-call comes next',/)

!     =========
      call hatch
!     =========

!     ----------------------------
!     Close files (after HATCH call)
!     ----------------------------
      close(UNIT=KMPI)
      close(UNIT=KMPO)

!    Pick up cross section data for ta
      write(6,150) ae(1)-RM, ap(1)
150   format(/' 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


!-------------------------------------------------------------------------
! Step 6:  Initialization-for-howfar
!-------------------------------------------------------------------------
      zbound=0.1
```

```
!     plate is 1 mm thick

!----------------------------------------------------------------------
! Step 7:  Initialization-for-ausgab
!----------------------------------------------------------------------
! Print header for output - which is all ausgab does in this case
      write(6,160)
160   format(/T19,'Kinetic energy(MeV)',T40,'charge',T48,
     *'angle w.r.t. z axis-degrees')

!----------------------------------------------------------------------
! Step 8:  Shower-call
!----------------------------------------------------------------------
! Initiate the shower 10 times
      do i=1,10
        write(6,170) i
170     format(' Start history',I4)
        call shower(iqin,ein,xin,yin,zin,uin,vin,win,irin,wtin)
!----------------------------------------------------------------------
! Step 9:  Output-of-results
!----------------------------------------------------------------------
!  Note output is at the end of each history in subroutine ausgab
      end do
      stop
      end
!------------------------last line of main code------------------------


!----------------------------ausgab.f----------------------------------
!----------------------------------------------------------------------
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
! --------------------------------------------------------------------
! Required subroutine for use with the EGS5 Code System
! --------------------------------------------------------------------
!**********************************************************************
!
!  In general, ausgab is a routine which is called under a series
!  of well defined conditions specified by the value of iarg (see the
!  egs5 manual for the list).  This is a particularly 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
!
!**********************************************************************
      subroutine ausgab(iarg)

      implicit none

      include 'include/egs5_h.f'              ! Main EGS "header" file

      include 'include/egs5_stack.f'     ! COMMONs required by EGS5 code
      include 'include/egs5_useful.f'
```

150

```
      integer iarg                                         ! Arguments

      real*8 angle,ekine                      ! Local variables

      if (iarg.eq.3.and.ir(np).eq.3) then
!  Angle w.r.t. z axis in degrees
        angle=acos(w(np))*180./3.14159
        if (iq(np).eq.0) then
          ekine=e(np)
        else
          ekine=e(np)-RM
!  Get kinetic energy
        end if
        write(6,100) ekine,iq(np),angle
100     format(T21,F10.3,T33,I10,T49,F10.1)
      end if
      return
      end


!------------------------last line of ausgab.f----------------------

!----------------------------howfar.f-------------------------------
!-------------------------------------------------------------------
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
! -------------------------------------------------------------------
! Required (geometry) subroutine for use with the EGS5 Code System
!*******************************************************************************
!
! 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
!
!                     |               |
!   Region 1          |    Region 2   |        Region 3
!                     |               |
!   e- ========>      |               | e- or photon ====>
!                     |               |
!   vacuum            |      Ta       |        vacuum
!
!*******************************************************************************
      subroutine howfar
```

```
      implicit none

      include 'include/egs5_h.f'                  ! Main EGS "header" file

      include 'include/egs5_epcont.f'    ! COMMONs required by EGS5 code
      include 'include/egs5_stack.f'

      common/geom/zbound
      real*8 zbound
!     geom passes info to our howfar routine

      real*8 tval                              ! Local variable

      if (ir(np).eq.3) then
        idisc=1
        return
! Terminate this history: it is past the plate
! We are in the Ta plate - check the geometry
      else if (ir(np).eq.2) then
        if (w(np).gt.0.0) then
! Going forward - consider first since  most frequent
! tval is dist to boundary in this direction
          tval=(zbound-z(np))/w(np)
          if (tval.gt.ustep) then
            return
! Can take currently requested step
          else
            ustep=tval
            irnew=3
            return
          end if
!    end of w(np)>0 case
!    Going back towards origin
        else if (w(np).lt.0.0) then
!    Distance to plane at origin
          tval=-z(np)/w(np)
          if (tval.gt.ustep) then
            return
!    Can take currently requested step
          else
            ustep=tval
            irnew=1
            return
          end if
!    End w(np)<0 case
!    Cannot hit boundary
        else if (w(np).eq.0.0) then
          return
        end if
! End of region 2 case
```

```
!  In regon with source
!  This must be a source particle on z=0 boundary
      else if (ir(np).eq.1) then
        if (w(np).gt.0.0) then
          ustep=0.0
          irnew=2
          return
        else
!  It must be a reflected particle-discard it
          idisc=1
          return
        end if
!  End region 1 case
      end if
      end
!-------------------------last line of howfar.f-----------------------


ELEM
 &INP IRAYL=0,IBOUND=0,INCOH=0,ICPROF=0,IMPACT=0 &END
TA                                TA
TA
ENER
 &INP AE=1.50,AP=0.10,UE=20.611,UP=20.0 &END
TEST
 &INP  &END
PWLF
 &INP  &END
DECK
 &INP  &END
```

This user code produces the following output file called **egs5job.out6** (a copy of this file is included with the EGS5 distribution, named **tutor1.out**).

```
 PEGS5-call comes next


 inseed=            1      (seed for generating unique sequences of Ranlux)
 ranlux luxury level set by rluxgo : 1     p=  48
 ranlux initialized by rluxgo from seed            1

 Start tutor1
 Call hatch to get cross-section data

 HATCH-call comes next

 EGS SUCCESSFULLY 'HATCHED' FOR ONE MEDIUM.

 Knock-on electrons can be created and any electron followed down to
```

```
                                   0.989 MeV kinetic energy
Brem photons can be created and any photon followed down to
                                   0.100 MeV


              Kinetic energy(MeV)  charge  angle w.r.t. z axis-degrees
Start history   1
                     1.867            0              2.3
                     0.680            0             10.8
                    15.455           -1             27.0
Start history   2
                     0.400            0             28.4
                     1.451            0             28.2
                    16.141           -1             21.6
Start history   3
                     0.301            0             12.4
                     1.402            0             33.3
                    16.128           -1             73.4
Start history   4
                     7.623            0             25.7
                    10.344           -1             31.4
Start history   5
                     0.678            0              3.2
                     1.131            0              5.8
                    16.286           -1             28.1
Start history   6
                    17.973           -1             35.0
Start history   7
                     2.874            0              2.9
                     1.654            0             16.5
                     2.588           -1             53.4
                    10.489           -1             12.0
Start history   8
                     0.180            0             83.7
                    17.685           -1             21.7
Start history   9
                    16.818           -1             12.2
Start history  10
                     1.697            0              4.0
                     0.440            0              7.5
                     1.905            0              9.7
                     2.210            0             41.9
                    11.533           -1             48.2
```

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 recognized that these are all bookkeeping extensions to
the problem – the physics itself already accomplished with EGS5 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 file 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 in detail in the EGS5 User Manual (Appendix B).

- The use of include statements to use values defined by parameter statements and to allow easy insertion of COMMONS.

- The technique required in order to define the array MEDIA.

- The definition of calling PEGS5 to produce material data used by user code.

- The definition of seeds for the RANLUX random number generator.

- The definition of calling parameters for the SHOWER routine.

- A very simple AUSGAB routine.

- A simple HOWFAR routine.

3.2 Tutorial 2 (Program tutor2.f)

In this example we use the same geometry as above, but we want the fraction of the incident energy that is reflected from, transmitted through, and deposited in the plate. The coding is essentially the same as tutor1 except that COMMON/SCORE/ and a new array ENCORE are defined at Step 1 in the sequence of steps required in the construction of a user code MAIN program, as described in the EGS5 User Manual of Appendix B. The latter is initialized to zero (Step 7) and subsequently output on the file (Step 9). The AUSGAB routine is considerably different as shown below.

```
!-------------------------------ausgab.f-------------------------------
!----------------------------------------------------------------------
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
! ---------------------------------------------------------------------
! Required subroutine for use with the EGS5 Code System
! ---------------------------------------------------------------------
!**********************************************************************
!
! 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
```

155

```
!  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.
!
!**********************************************************************
      subroutine ausgab(iarg)

      implicit none

      include 'include/egs5_h.f'                ! Main EGS "header" file

      include 'include/egs5_epcont.f'    ! epcont contains edep
      include 'include/egs5_stack.f'     ! stack contains x, y, z, u, v,
                                         ! w, ir and np

      common/score/escore(3)
      real*8 escore

      integer iarg                                    ! Arguments

      integer irl                              ! Local variables

      if (iarg.le.4) then
        irl=ir(np)
!   Pick up current region number
        escore(irl)=escore(irl)+edep
      end if
      return
      end
!------------------------last line of ausgab.f------------------------
```

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 above and further elaborated upon in Appendix B. In this example, but not always, we can sum EDEP for any value of IARG up to 4. The following is the output provided by **tutor2** (named tutor2.out in distribution file).

```
 PEGS5-call comes next


 inseed=            1      (seed for generating unique sequences of Ranlux)
 ranlux luxury level set by rluxgo : 1      p=  48
 ranlux initialized by rluxgo from seed            1

 Start tutor2
 Call hatch to get cross-section data
```

```
HATCH-call comes next

EGS SUCCESSFULLY 'HATCHED' FOR ONE MEDIUM.

Knock-on electrons can be created and any electron followed down to
                                   0.989 MeV kinetic energy
Brem photons can be created and any photon followed down to
                                   0.100 MeV

Fraction of energy reflected from plate=              0.655%
Fraction of energy deposited in plate=              12.958%
Fraction of energy transmitted through plate=        86.388%
                                                 -----------
Total fraction of energy accounted for=             100.000%
```

3.3   Tutorial 3 (Program tutor3.f)

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). The following shows the change required (at Step 8 and 9) and the new AUSGAB routine.

```
!-------------------------------------------------------------------------
! Step 8:  Shower-call
!-------------------------------------------------------------------------
! Initiate the shower ncase times
      ncase=10000
      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= min0 (int(ehist/bwidth + 0.999), 25)
        if (ibin.ne.0) then
          ebin(ibin)=ebin(ibin)+1
        end if
      end do


!-------------------------------------------------------------------------
! Step 9:  Output-of-results
!-------------------------------------------------------------------------
! Pick up maximum bin for normalization
      binmax=0.0
      do j=1,25
```

```
          binmax=max(binmax,ebin(j))
        end do
        write(6,160) ein,zbound
160     format(/' Response 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 j=1,48
          line(j)=' '
        end do
! Blank entire output array
        do j=1,25
          icol=int(ebin(j)/binmax*48.0+0.999)
          if (icol.eq.0) icol=1
          line(icol)='*'
!  Load output array at desired location
          write(6,170) bwidth*j,ebin(j)/float(ncase),line
170       format(F10.2,F10.4,48A1)
          line(icol)=' '
!  Reblank
        end do


        stop
        end
!------------------------last line of main code------------------------

!-----------------------------ausgab.f-------------------------------
!--------------------------------------------------------------------
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
! -------------------------------------------------------------------
! Required subroutine for use with the EGS5 Code System
! -------------------------------------------------------------------
!**********************************************************************
!
! 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.
!**********************************************************************
```

```
      subroutine ausgab(iarg)

      implicit none

      include 'include/egs5_h.f'                ! Main EGS "header" file

      include 'include/egs5_epcont.f'   ! epcont contains edep
      include 'include/egs5_stack.f'    ! stack contains x, y, z, u, v,
                                        ! w, ir and np

      common/score/ehist
      real*8 ehist

      integer iarg                                       ! Arguments

      if (iarg.le.2 .or. iarg.eq.4) then
        ehist=ehist + edep
      end if

      return
      end

!------------------------last line of ausgab.f-----------------------
```

The following is the output provided by tutor3 (named tutor3.out in distribution file).

```
 PEGS5-call comes next


 inseed=            1      (seed for generating unique sequences of Ranlux)
 ranlux luxury level set by rluxgo : 1     p=  48
 ranlux initialized by rluxgo from seed            1

 Start tutor3
 Call hatch to get cross-section data

 HATCH-call comes next

 EGS 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

 Response function
 for a    5.00 MeV pencil beam ofphotons on a   2.54 cm thick slab of NaI
     Energy   counts/incident photon
      0.20    0.0060        *
```

```
0.40    0.0058        *
0.60    0.0054       *
0.80    0.0050       *
1.00    0.0058        *
1.20    0.0065          *
1.40    0.0047       *
1.60    0.0043     *
1.80    0.0060        *
2.00    0.0047     *
2.20    0.0061         *
2.40    0.0053      *
2.60    0.0059        *
2.80    0.0059        *
3.00    0.0061        *
3.20    0.0053      *
3.40    0.0069          *
3.60    0.0079            *
3.80    0.0103               *
4.00    0.0273                       *
4.20    0.0226                    *
4.40    0.0155              *
4.60    0.0352                          *
4.80    0.0209                   *
5.00    0.0371                            *
```

3.4   Tutorial 4 (Program tutor4.f)

This program examines the dependence of EGS5 results on electron step-size. Recall that for electrons with low initial energies, the limitations inherent in the EGS4 transport mechanics mandated that the user specify quite small electron step-sizes (defined in terms of the fractional energy loss over a step, ESTEPE) in order to assure converged results. As noted in Chapter 2, the transport mechanics algorithm of EGS5 naturally mitigates these dependencies. In addition, EGS5 provides several prescriptions for user selection of step-sizes based on material and region geometries.

As in EGS4, the program **tutor4.f** is based on **tutor2.f**, but with a 2 mm slab of silicon as the medium and 2.0 MeV for the incident electron energy. In addition to scoring transmitted, deposited and reflected energy, the number of transmitted and reflected electrons are tallied in **tutor4.f**. The example problem actually consists of three distinct runs, two using step-size specification based on fractional energy loss (EFRACH and EFRACL) and one using a user specified "characteristic dimension" for the given medium. The main purpose of this tutorial is to demonstrate effectiveness of setting electron multiple scattering step sizes by specifying the characteristic dimension of the problem. This approach is much easier, even for expert users, than controling step sizes through EFRACH and EFRACL.

Recall that multiple-scattering step-sizes in EGS5 are defined in terms of the scattering strength

accumulated over the given distance, and can be specified by the parameters EFRACH and EFRACL, which correspond to the fractional energy loss over the step at the upper energy range of the problem UE and the lower energy limit AE, respectively. Thus EFRACH and EFRACL, which are set in the PEGS input file, **pgs5job.pegs5inp**, correspond to ESTEPE from EGS4. In the first **tutor4.f** example, the material in the slab is named "SI with long steps" and EFRACH and EFRACL are set to 0.30. It should be noted that these values were chosen to be artificially high (the defaults in EGS5 are 0.05 and 0.20, respectively) to help illustrate step-size dependence, since on this problem, EGS5 shows little dependence on step-size with the default values). In the second example, the material is switched to one named "SI with short steps," and EFRACH and EFRACL are set to be 0.01 and 0.02, respectively. (Thus this example problem also illustrates a method for creating different "media" which are actually the same material with different physics options invoked.)

In the third pass through **tutor4.f**, the first material is again used, but the step-size is selected by the specification of the characteristic dimension, CHARD. When CHARD is positive, EGS5 ignores step-sizes which correspond to EFRACH and EFRACL and instead automatically chooses values of the initial scattering strength which provide converged values for electron tracklength (to within 1% accuracy) for electrons impinging on a cylinders of diameter and thickness CHARD. For the current problem, CHARD can be taken to be equivalent to the slab thickness, as seen below.

Following is the source code (except for subroutine HOWFAR, which is not changed from **tutor2.f**) used by **tutor4.f**.

```
!***********************************************************************
!
!                      *************
!                      *           *
!                      *  tutor4.f  *
!                      *           *
!                      *************
!
!  An EGS5 user code. It lists the particles escaping from the back
!  of a 2 mm Si plate when a pencil beam of 2 MeV electrons
!  is incident on it normally.
!
!  For SLAC-R-730/KEK Report 2005-8: A simple example which scores
!  reflected, deposited, and transmitted particles and energy and
!  demonstrates step-size selection
!
!  The following units are used: unit 6 for output
!***********************************************************************
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
!----------------------------------------------------------------------
!--------------------------- main code ---------------------------
!----------------------------------------------------------------------


!----------------------------------------------------------------------
! Step 1: Initialization
```

```
!-------------------------------------------------------------------------

        implicit none

!       ------------
!       EGS5 COMMONs
!       ------------
        include 'include/egs5_h.f'                   ! Main EGS "header" file

        include 'include/egs5_bounds.f'
        include 'include/egs5_media.f'
        include 'include/egs5_misc.f'
        include 'include/egs5_thresh.f'
        include 'include/egs5_useful.f'
        include 'include/egs5_usersc.f'
        include 'include/randomm.f'

!       bounds contains ecut and pcut
!       media contains the array media
!       misc contains med
!       thresh contains ae and ap
!       useful contains RM
!       usersc contains emaxe

        common/geom/zbound
        real*8 zbound
!       geom passes info to our howfar routine

        common/score/escore(3),iscore(3)
        real*8 escore
        integer iscore

        real*8 ein,xin,yin,zin,                          ! Arguments
     *        uin,vin,win,wtin
        integer iqin,irin

        real*8 anorm,total                       ! Local variables
        real
     * tarray(2),tt,tt0,tt1,cputime
        integer loop,i,j,ncase
        character*24 medarr(2)

        real etime

!       ----------
!       Open files
!       ----------
        open(UNIT= 6,FILE='egs5job.out',STATUS='unknown')

        do loop = 1,3
```

```
!       ====================
        call counters_out(0)
!       ====================


!----------------------------------------------------------------------
! Step 2: pegs5-call
!----------------------------------------------------------------------
!       =============
        call block_set                  ! Initialize some general variables
!       =============

! nmed and dunit default to 1, i.e. one medium and we work in cm

        if(loop.eq.3) then
          chard(1) = 0.20d0       !  optional, but recommended to invoke
          chard(2) = 0.20d0       !  automatic step-size control
        else
          chard(1) = 0.00d0       !  optional, but recommended to invoke
          chard(2) = 0.00d0       !  automatic step-size control
        endif

        write(6,100) loop, chard(1)
100     FORMAT(72('*'),/,
       *'Initializing EGS5, loop = ',I1,': charD = ',f5.2,/,
       *72('*'),/)

         if(loop.eq.1) then

!       ---------------------------------
!       define media before calling PEGS5
!       ---------------------------------
        nmed=2
        medarr(1)='SI with long steps       '
        medarr(2)='SI with short steps      '

        do j=1,nmed
          do i=1,24
            media(i,j)=medarr(j)(i:i)
          end do
        end do
!       -----------------------------------------------
!       Run KEK version of PEGS5 before calling HATCH
!       (method was developed by Y. Namito - 010306)
!       -----------------------------------------------

        write(6,110)
110     FORMAT(' PEGS5-call comes next'/)

!       ==========
        call pegs5
!       ==========
```

```
      endif

      if(loop.lt.3) then
        write(6,120) loop,medarr(loop)
120   FORMAT(' Using media number ',i1,', ',a24,' for this run',/)
      endif

!------------------------------------------------------------------------
! Step 3: Pre-hatch-call-initialization
!------------------------------------------------------------------------
      nreg=3
!     nreg : number of region

      med(1)=0
      med(3)=0
      if(loop.eq.2) then
        med(2)=2
      else
        med(2)=1
      endif
! Vacuum in regions 1 and 3, Si in region 2
      ecut(2)=0.700
! Terminate electron histories at .700 MeV in the plate
      pcut(2)=0.010
! Terminate   photon histories at 0.01 MeV in the plate
!             Only needed for region 2 since no transport elsewhere
!             ecut is total energy = 0.189   MeV kinetic energy

!     ----------------------------------------------------
!     Random number seeds.  Must be defined before call hatch
!     or defaults will be used.  inseed (1- 2^31)
!     ----------------------------------------------------
      luxlev=1
      inseed=1
      kount=0
      mkount=0
      do i = 1, 25
        isdext(i) = 0
      end do
      write(6,150) inseed
150   FORMAT(/,' inseed=',I12,5X,
     *            ' (seed for generating unique sequences of Ranlux)')

!     =============
      call rluxinit  ! Initialize the Ranlux random-number generator
!     =============

!------------------------------------------------------------------------
! Step 4:  Determination-of-incident-particle-parameters
!------------------------------------------------------------------------
```

```fortran
! Define initial variables for 2 MeV beam of electrons incident
! perpendicular to the slab
      iqin=-1
!           Incident charge - electrons
!           2 MeV kinetic energy
      ein=2.d0 + RM
      xin=0.0
      yin=0.0
      zin=0.0
!     Incident at origin
      uin=0.0
      vin=0.0
      win=1.0
!           Moving along z axis
      irin=2
!           Starts in region 2, could be 1
!           weight = 1 since no variance reduction used
      wtin=1.0
!     Weight = 1 since no variance reduction used


!-------------------------------------------------------------------------
! Step 5:   hatch-call
!-------------------------------------------------------------------------
! Maximum total energy of an electron for this problem must be
! defined before hatch call
      emaxe = ein

      write(6,160)
160   FORMAT(/' Start tutor4'/' Call hatch to get cross-section data')

!     ----------------------------
!     Open files (before HATCH call)
!     ----------------------------
      open(UNIT=KMPI,FILE='pgs5job.pegs5dat',STATUS='old')
      open(UNIT=KMPO,FILE='egs5job.dummy',STATUS='unknown')

      write(6,170)
170   FORMAT(/,' HATCH-call comes next',/)

!     ==========
      call hatch
!     ==========

!     ----------------------------
!     Close files (after HATCH call)
!     ----------------------------
      close(UNIT=KMPI)
      close(UNIT=KMPO)

!    Pick up cross section data for ta
      write(6,180) ae(1)-RM, ap(1)
```

```
180   FORMAT(/' 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


!---------------------------------------------------------------------
! Step 6:  Initialization-for-howfar
!---------------------------------------------------------------------
      zbound=0.2
!     plate is 2 mm thick


!---------------------------------------------------------------------
! Step 7:  Initialization-for-ausgab
!---------------------------------------------------------------------
      do i=1,3
        iscore(i)=0
        escore(i)=0.d0
!  Zero scoring array before starting
      end do


!---------------------------------------------------------------------
! Step 8:  Shower-call
!---------------------------------------------------------------------
      tt=etime(tarray)
      tt0=tarray(1)

! Initiate the shower ncase times
      ncase=50000
      do i=1,ncase
        call shower(iqin,ein,xin,yin,zin,uin,vin,win,irin,wtin)
      end do

      tt=etime(tarray)
      tt1=tarray(1)
      cputime=tt1-tt0


!---------------------------------------------------------------------
! Step 9:  Output-of-results
!---------------------------------------------------------------------
      write(6,190) cputime,ncase
190   FORMAT('CPU time = ',1X,G15.5,' sec for ',I8,' cases')

      anorm = 100./float(ncase)
      write(6,200) iscore(1)*anorm,iscore(3)*anorm
200   FORMAT(/,
      *' Fraction of electrons reflected from plate=',T50,F10.1,'%',/,
      *' Fraction of electrons transmitted through plate=',T50,F10.1,'%')

! Normalize to % of total input energy
      anorm = 100./((ein-RM)*float(ncase))
```

```
      total=0.0
      do i=1,3
        total=total+escore(i)
      end do
      write(6,210) (escore(i)*anorm,i=1,3),total*anorm
210   FORMAT(/,/,
     *  ' Fraction of energy reflected from plate=',T50,F10.1,'%'
     */ ' Fraction of energy deposited in plate=',T50,F10.1,'%'/
     *' Fraction of energy transmitted through plate=',T50,F10.1,'%'/
     *T50,11('-')/' Total fraction of energy accounted for=', T50,
     *F10.1,'%'/)

      end do  ! do four times through


      stop
      end
!------------------------last line of main code------------------------

!------------------------------ausgab.f--------------------------------
!----------------------------------------------------------------------
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
! ----------------------------------------------------------------------
! Required subroutine for use with the EGS5 Code System
! ----------------------------------------------------------------------
!**********************************************************************
!
! In this AUSGAB routine for TUTOR4, we score the energy deposited
!  in the various regions and count transmitted and reflected
!  electrons.
!
!  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, and if it is an electron,
!  score it's region.
!
!**********************************************************************
      subroutine ausgab(iarg)

      implicit none

      include 'include/egs5_h.f'                ! Main EGS "header" file

      include 'include/egs5_epcont.f'    ! COMMONs required by EGS5 code
      include 'include/egs5_stack.f'
```

```
       common/score/escore(3), iscore(3)
       real*8 escore
       integer iscore

       integer iarg                                         ! Arguments

       integer irl                                 ! Local variables

       if (iarg.le.4) then
          irl=ir(np)
!   Pick up current region number
          escore(irl)=escore(irl)+edep
!   Pick up energy deposition/transmission/reflection
          if (iarg.eq.3 .and. iq(np).eq.-1) then
            iscore(irl)=iscore(irl)+1
!   Pick up electron transmission/reflection
          end if
       end if
       return
       end


!-------------------------last line of ausgab.f-----------------------
```

Following is the PEGS input file required by **tutor4.f**, which specifies two versions of the same material (silicon), one with long step-sizes and one with short step-sizes.

```
ELEM
 &INP EFRACH=0.30,EFRACL=0.30,
      IRAYL=0,IBOUND=0,INCOH=0,ICPROF=0,IMPACT=0 &END
SI with long steps                              SI
SI
ENER
 &INP AE=0.700,AP=0.010,UE=2.521,UP=2.1 &END
PWLF
 &INP  &END
DECK
 &INP  &END
ELEM
 &INP EFRACH=0.01,EFRACL=0.01,
      IRAYL=0,IBOUND=0,INCOH=0,ICPROF=0,IMPACT=0 &END
SI with short steps                             SI
SI
ENER
 &INP AE=0.700,AP=0.010,UE=2.521,UP=2.1 &END
PWLF
 &INP  &END
DECK
 &INP  &END
```

Following is the output produced by **tutor4.f**, showing that the severe step-size dependence exhibited by EGS4 on this problem is greatly diminished in EGS5. Recall that in the **tutor4** example of EGS4, runs using the default step-size algorithm predicted 1.3% reflection and 49.21.0%, EGS4 returned values of 6.4% reflection and 61.3% transmission. In contrast, an EGS5 run using very long multiple scattering steps (corresponding to 30% energy loss over the steps, and labeled "loop 1" in the output file) yields values for reflection (8.1%) and transmission (66.5%) which are fairly close to EGS5 results generated with 1% energy loss steps (7.3% and 64.4% for reflection and transmission, respectively, "loop 2" in the output). This level of agreement (roughly 3% error in the transmission fraction, even when using 30illustrates the power of the modified random hinge approach. In addition, the significant discrepancies between the 1% energy loss step results of EGS4 and EGS5 forcefully demonstrate the shortcomings of the EGS4 transport mechanics model, even for very small electron step sizes.

The results from the third part of this example ("loop 3") show that the expected 1% accuracy is obtained when the multiple scattering steps sizes are chosen automatically by EGS5 based on the given characteristic dimension of 2 mm for this problem. Note that for this dimension and at this energy, the step-sizes selected by EGS5 for silicon are roughly 10-15 times as large as those used in the 1% energy loss run (loop 2), thus providing significant speedup (a factor of three, as seen from the output) for calculations at this level of accuracy.

```
**************************************************************************
Initializing EGS5, loop = 1: charD =  0.00
**************************************************************************

 PEGS5-call comes next

 Using media number 1, SI with long steps        for this run


 inseed=            1       (seed for generating unique sequences of Ranlux)
 ranlux luxury level set by rluxgo : 1      p=  48
 ranlux initialized by rluxgo from seed            1

 Start tutor4
 Call hatch to get cross-section data

 HATCH-call comes next

 EGS SUCCESSFULLY 'HATCHED' FOR      2 MEDIA.
 WARNING in RMSFIT: no characteristic dimension input for media    1
 Using old data from gsdist.dat with:
 efrach, efrachl =      3.0000E-01     3.0000E-01
 WARNING in RMSFIT: no characteristic dimension input for media    2
 Using old data from gsdist.dat with:
 efrach, efrachl =      1.0000E-02     1.0000E-02

 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
CPU time =        13.368     sec for    50000 cases

 Fraction of electrons reflected from plate=         8.1%
 Fraction of electrons transmitted through plate=    66.5%


 Fraction of energy reflected from plate=            3.9%
 Fraction of energy deposited in plate=              59.2%
 Fraction of energy transmitted through plate=       36.9%
                                                 -----------
 Total fraction of energy accounted for=            100.0%


****************************************************************************
Initializing EGS5, loop = 2: charD =  0.00
****************************************************************************

 Using media number 2, SI with short steps       for this run


 inseed=             1      (seed for generating unique sequences of Ranlux)
 ranlux luxury level set by rluxgo : 1      p=  48
 ranlux initialized by rluxgo from seed            1

 Start tutor4
 Call hatch to get cross-section data

 HATCH-call comes next

 EGS SUCCESSFULLY 'HATCHED' FOR     2 MEDIA.
 WARNING in RMSFIT: no characteristic dimension input for media    1
 Using old data from gsdist.dat with:
 efrach, efrachl =      3.0000E-01     3.0000E-01
 WARNING in RMSFIT: no characteristic dimension input for media    2
 Using old data from gsdist.dat with:
 efrach, efrachl =      1.0000E-02     1.0000E-02

 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
CPU time =        111.67     sec for    50000 cases

 Fraction of electrons reflected from plate=         7.3%
 Fraction of electrons transmitted through plate=    64.4%


 Fraction of energy reflected from plate=            3.2%
 Fraction of energy deposited in plate=              62.2%
 Fraction of energy transmitted through plate=       34.6%
```

```
                                         -----------
 Total fraction of energy accounted for=            100.0%


*************************************************************************
Initializing EGS5, loop = 3: charD =  0.20
*************************************************************************


  inseed=            1       (seed for generating unique sequences of Ranlux)
  ranlux luxury level set by rluxgo : 1      p=  48
  ranlux initialized by rluxgo from seed            1

  Start tutor4
  Call hatch to get cross-section data

  HATCH-call comes next

  EGS SUCCESSFULLY 'HATCHED' FOR      2 MEDIA.

  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
CPU time =        36.373      sec for     50000 cases

 Fraction of electrons reflected from plate=           7.4%
 Fraction of electrons transmitted through plate=     64.8%


 Fraction of energy reflected from plate=              3.3%
 Fraction of energy deposited in plate=               62.0%
 Fraction of energy transmitted through plate=        34.7%
                                                -----------
 Total fraction of energy accounted for=             100.0%
```

3.5   Tutorial 5 (Program tutor5.f)

In this program we give an example that includes Rayleigh scattering and which makes use of the
variable called LATCH (contained in COMMON/STACK/). LATCH can be set for any particle on the
"stack" of particles being transported, and is passed on to all its progeny. This provides a simple
procedure for keeping track of the histories of particles. In this example 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.

    This user program also demonstrates the use of the IAUSFL array of flags (in COMMON/EPCONT/).
By setting the appropriate flags, the user can cause the EGS5 system to call the AUSGAB subroutine

in any combination of 31 well specified situations (see Appendix B). By default, EGS calls `AUSGAB`
only 5 out of the possible 31 situations. Here, by setting `IAUSFL(18)` and `IAUSFL(24)` from 0
(default) to 1 in the main program, we cause EGS 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. A complete listing
of **tutor5.f**, except for `HOWFAR` routine which is similar to the other examples, is given below.

```
!***********************************************************************
!
!                       *************
!                       *           *
!                       *  tutor5.f *
!                       *           *
!                       *************
!
!  An EGS5 user code which scores the number and average energy of the
!  primary, Rayleigh scattered and Compton scattered photons passing
!  through a  5 cm thick slab of water when a 50 keV pencil beam of
!  photons is incident normally
!
!
!  For SLAC-R-730/KEK Report 2005-8:  Example of including Rayleigh
!          scattering, and use of the LATCH feature
!
!  The following units are used: unit 6 for output
!***********************************************************************
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
!----------------------------------------------------------------------
!----------------------------- main code -----------------------------
!----------------------------------------------------------------------


!----------------------------------------------------------------------
! Step 1: Initialization
!----------------------------------------------------------------------

      implicit none

!     ------------
!     EGS5 COMMONs
!     ------------
      include 'include/egs5_h.f'                  ! Main EGS "header" file

      include 'include/egs5_bounds.f'
      include 'include/egs5_epcont.f'
      include 'include/egs5_media.f'
      include 'include/egs5_misc.f'
      include 'include/egs5_stack.f'
      include 'include/egs5_thresh.f'
      include 'include/egs5_useful.f'
      include 'include/egs5_usersc.f'
```

```fortran
      include 'include/randomm.f'

!     bounds contains ecut and pcut
!     epcont contains iausfl
!     media contains the array media
!     misc contains med
!     stack contains latchi
!     thresh contains ae and ap
!     useful contains RM
!     usersc contains emaxe

      common/geom/zbound
      real*8 zbound
!     geom passes info to our howfar routine

      common/score/count(3),entot(3)
      real*8 count,entot

      real*8 ein,xin,yin,zin,             ! Arguments
     *       uin,vin,win,wtin
      integer iqin,irin

      real*8 anorm                                ! Local variables
      integer i,j,ncase
      character*24 medarr(1)

!     ----------
!     Open files
!     ----------
      open(UNIT= 6,FILE='egs5job.out',STATUS='unknown')

!     ====================
      call counters_out(0)
!     ====================


!-----------------------------------------------------------------------
! Step 2: pegs5-call
!-----------------------------------------------------------------------
!     ==============
      call block_set                    ! Initialize some general variables
!     ==============

!     ----------------------------------
!     define media before calling PEGS5
!     ----------------------------------
      nmed=1
      medarr(1)='H2O                     '

      do j=1,nmed
        do i=1,24
          media(i,j)=medarr(j)(i:i)
```

173

```
          end do
       end do

! nmed and dunit default to 1, i.e. one medium and we work in cm

       chard(1) = 0.5d0          !  optional, but recommended to invoke
                                 !  automatic step-size control

!      ------------------------------------------------
!      Run KEK version of PEGS5 before calling HATCH
!      (method was developed by Y. Namito - 010306)
!      ------------------------------------------------
       write(6,100)
100    FORMAT(' PEGS5-call comes next'/)


!      =========
       call pegs5
!      =========


!-----------------------------------------------------------------------
! Step 3: Pre-hatch-call-initialization
!-----------------------------------------------------------------------
       nreg=3
!      nreg : number of region

       med(1)=0
       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

!      -----------------------------------------------------------
!      Random number seeds.  Must be defined before call hatch
!      or defaults will be used.  inseed (1- 2^31)
!      -----------------------------------------------------------
       luxlev=1
       inseed=1
       write(6,120) inseed
120    FORMAT(/,' inseed=',I12,5X,
      *           ' (seed for generating unique sequences of Ranlux)')


!      =============
       call rluxinit  ! Initialize the Ranlux random-number generator
!      =============
```

```
!-------------------------------------------------------------------------
! Step 4:  Determination-of-incident-particle-parameters
!-------------------------------------------------------------------------
! Define initial variables for 50 keV beam of photons normally incident
! on the slab
      iqin=0
!     Incident photons
!           50 keV
      ein=0.050
      xin=0.0
      yin=0.0
      zin=0.0
!     Incident at origin
      uin=0.0
      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 5:   hatch-call
!-------------------------------------------------------------------------
! Maximum total energy of an electron for this problem must be
! defined before hatch call
      emaxe = ein + RM

      write(6,130)
130   format(/' Start tutor5'/' Call hatch to get cross-section data')

!     ------------------------------
!     Open files (before HATCH call)
!     ------------------------------
      open(UNIT=KMPI,FILE='pgs5job.pegs5dat',STATUS='old')
      open(UNIT=KMPO,FILE='egs5job.dummy',STATUS='unknown')

      write(6,140)
140   format(/,' HATCH-call comes next',/)

!     ==========
      call hatch
!     ==========

!     ------------------------------
!     Close files (after HATCH call)
```

```
!         -----------------------------
      close(UNIT=KMPI)
      close(UNIT=KMPO)

!    Pick up cross section data for water
      write(6,150) ae(1)-RM, ap(1)
150   format(/' 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


!-----------------------------------------------------------------------
! Step 6:  Initialization-for-howfar
!-----------------------------------------------------------------------
      zbound=0.5
!     Plate is 0.5 cm thick


!-----------------------------------------------------------------------
! Step 7:  Initialization-for-ausgab
!-----------------------------------------------------------------------
      do i=1,3
        count(i)=0.0
        entot(i)=0.0
!  Zero scoring array at start
      end do


!  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)=1
      iausfl(24)=1


!-----------------------------------------------------------------------
! Step 8:  Shower-call
!-----------------------------------------------------------------------
! Initiate the shower ncase times
      ncase=10000
      do i=1,NCASE
        call shower(iqin,ein,xin,yin,zin,uin,vin,win,irin,wtin)
      end do


!-----------------------------------------------------------------------
! Step 9:  Output-of-results
!-----------------------------------------------------------------------
! Normalize to % of photon number
      anorm = 100./float(ncase)
      do i=1,3
        if (count(i).ne.0) then
          entot(i)=entot(i)/count(i)
!     Get average energies
```

```
       end if
      end do
      write(6,160) ein*1000.,zbound, pcut(2), (anorm*count(i),entot(i),
     *i=1,3)
160   format(/' 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'//)

      stop
      end
!------------------------last line of main code------------------------

!----------------------------ausgab.f----------------------------------
!----------------------------------------------------------------------
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
! ---------------------------------------------------------------------
! Required subroutine for use with the EGS5 Code System
! ---------------------------------------------------------------------
!**********************************************************************
!
!  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 occurred.
!**********************************************************************
      subroutine ausgab(iarg)

      implicit none

      include 'include/egs5_h.f'                    ! Main EGS "header" file

      include 'include/egs5_stack.f'      ! COMMONs required by EGS5 code

      common/score/count(3),entot(3)
      real*8 count,entot

      integer iarg                                        ! Arguments

      integer jj                                          ! Local variable
```

177

```
      if (iarg.eq.17) then
!  A Compton scatter is about to occur
          latch(np)=latch(np)+1
        else if (iarg.eq.23) then
!  A Rayleigh scatter is about to occur
          latch(np)=latch(np)+1000
!   If a history has terminated because leaving the slab, score it
!  Particle has left slab
        else if (iarg .eq. 3) then
          if (ir(np).eq.3 .or. ir(np) .eq. 1) then
!    It is transmitted or reflected
            jj=0
            if (latch(np) .eq. 0) then
!       No scattering - a primary
              jj=1
            else if (mod(latch(np),10000)-mod(latch(np),100) .ne. 0) then
!       at least one Rayleigh scatter
              jj=2
            else if (mod(latch(np),100) .ne. 0) then
!       at least one Compton scatter without Rayleigh
              jj=3
!       debug
            else
              write(6,1080) jj,latch(np)
1080          format(' jj,latch(np)=',2I10)
            end if
            if (jj .ne. 0) then
              count(jj)=count(jj) + 1.
              entot(jj) = entot(jj) + e(np)
            end if
!     End region 3 block
          end if
!  End iarg 3 block
        end if
        return
        end
!-------------------------last line of ausgab.f-----------------------
```

Following is the output provided by **tutor5.f**.

```
 PEGS5-call comes next

 inseed=           1      (seed for generating unique sequences of Ranlux)
 ranlux luxury level set by rluxgo : 1     p=  48
 ranlux initialized by rluxgo from seed            1

 Start tutor5
 Call hatch to get cross-section data
```

```
HATCH-call comes next

RAYLEIGH OPTION REQUESTED FOR MEDIUM NUMBER  1

EGS SUCCESSFULLY 'HATCHED' FOR ONE MEDIUM.

Knock-on electrons can be created and any electron followed down to
                                    0.010 MeV kinetic energy
Brem photons can be created and any photon followed down to
                                    0.010 MeV


For  50.0 keV photons incident on 0.5cm of H2O with PCUT=0.010 MeV

Transmitted primaries=                   88.89%  ave energy=     0.050 MeV

Fraction Rayleigh scattering=             0.95%  ave energy=     0.049 MeV

Fraction Compton scattering only=         8.60%  ave energy=     0.046 MeV
```

3.6 Tutorial 6 (Program tutor6.f)

One of the important features of the EGS5 Code System is that the user has direct control over the geometry in which the radiation transport takes place, and rather complex geometries can be described in a very simple manner with the aid of geometry subprograms. Subroutine HOWFAR, in the previous examples, was made more detailed than necessary for demonstration purposes. One can greatly simplify things, particularly for the simple slab geometry case, as shown in the following excerpts from **tutor6**.

```
!***********************************************************************
!
!                  *************
!                  *           *
!                  *  tutor6.f  *
!                  *           *
!                  *************
!
!  An EGS5 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.
!
!  NOTE: This program is the same as TUTOR1.f except that the
!        geometry subroutine (HOWFAR) is simplified by the use of
```

```
!        the general purpose geometry subroutines PLAN2P.
          .
          .
          .
!-----------------------------------------------------------------------
! Step 6:   Initialization-for-howfar
!-----------------------------------------------------------------------
! Define the coordinates and the normal vectors for the two planes.
! Information required by howfar (and auxiliary geometry subprograms)
! and passed through common/pladta/
!
! First plane (the x-y plane through the origin)
      pcoord(1,1)=0.0
      pcoord(2,1)=0.0
      pcoord(3,1)=0.0
! Coordinates
      pnorm(1,1) =0.0
      pnorm(2,1) =0.0
      pnorm(3,1)= 1.0
! Normal vectors
! Second plane (note: slab is 1 mm thick)
      pcoord(1,2)=0.0
      pcoord(2,2)=0.0
      pcoord(3,2)=0.1
! Coordinates
      pnorm(1,2) =0.0
      pnorm(2,2) =0.0
      pnorm(3,2)= 1.0
! Normal vectors
          .
          .
          .
!----------------------------howfar.f----------------------------
!-----------------------------------------------------------------------
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
! -----------------------------------------------------------------------
! Required (geometry) subroutine for use with the EGS5 Code System
!***********************************************************************
!
! The following is a general specification of howfar.  Essentially
! it is the same as that given in tutor1.f with the following
! exception: 1) Particles must be initially begin in region 2 and are
!               discarded when that enter region 1 or 3 (no check
!               is made on w(np)).
!            2) The coding is much simplified (i.e., modular)
!               As a result of using auxiliar geometry subprogram
!               plan2p (which calls plane1 and chgtr which require
!               commons epcont, pladta, and stack).
!
!   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
!
!                    |              |
!   Region 1         |   Region 2   |       Region 3
!                    |              |
!   e- ========>     |              | e- or photon ====>
!                    |              |
!   vacuum           |      Ta      |       vacuum
!
! DESCRIPTION - PLAN2P is generally called from subroutine HOWFAR
!   whenever a particle is in a region bounded by two planes that
!   ARE parallel.  Both subroutines PLANE1 and CHGTR are called
!   by PLAN2P (the second PLANE1 call is not made if the first
!   plane is hit, or if the trajectory is parallel).
!-----------------------------------------------------------------
!     NPL1   = ID number assigned to plane called first (input)
!     NRG1   = ID number assigned to region particle trajectory
!              will lead into
!     ISD1   =  1 normal points towards current region (input)
!            = -1 normal points away from current region (input)
!     NPL2   = Same (but for plane called second)
!     NRG2   = Same (but for plane called second)
!     ISD2   = Same (but for plane called second)
!*****************************************************************************
      subroutine howfar

      implicit none

      include 'include/egs5_h.f'                ! Main EGS "header" file

      include 'include/egs5_epcont.f'    ! epcont contains irnew, ustep
                                         ! and idisc
      include 'include/egs5_stack.f'     ! stack contains x, y, z, u, v,
                                         ! w, ir and np


!     ---------------------
!     Auxiliary-code COMMONs
!     ---------------------
      include 'auxcommons/aux_h.f'       ! Auxiliary-code "header" file

      include 'auxcommons/pladta.f'

      integer irl                                ! Local variable

      irl=ir(np)             ! Set local variable
      if (irl.ne.2) then
        idisc=1        ! Terminate this history if not in plate
      else             ! We are in the Ta plate - check the geometry
        call plan2p(irl,irl+1,1,irl-1,irl-1,-1)
      end if
```

```
      return
      end

!-------------------------last line of howfar.f----------------------
```

The actual `HOWFAR` code is now less than 10 lines long, the rest consisting of `COMMON`'s and comments. For a complete understanding of how `PLAN2P` and its related subroutines `PLANE1` and `CHGTR` are called and used, the reader should refer to comments in the appropriate subroutine (distributed with the EGS5 Code System). For a description of the concepts involved in modeling geometry for Monte Carlo programs in general (and EGS4 in particular), the reader may wish to refer to the document "How to Code Geometry: Writing Subroutine `HOWFAR`," which is provided with the EGS5 distribution.

3.7   Tutorial 7 (Program tutor7.f)

In this program we give an example that includes K- and L-fluorescence photons which can be possible in any material including a compound or a mixture. Here we investigate the reflected photon spectrum from 1 cm of lead when a 100 keV beam of photons is incident on it. If the **IEDGFL** flag is set to 1, fluorescence photons can be produced after K- or L-photoelectric effect interactions in that region. A complete listing of **tutor7** (except the `HOWFAR` routine, which is the same as **tutor6.f**) is given below.


```
!***********************************************************************
!
!                     *************
!                     *           *
!                     *  tutor7.f *
!                     *           *
!                     *************
!
! An EGS5 user code which scores the spectrum of reflection from
! 1.0 cm thick slab of lead when a 100 keV beam of photons is incident
! on it with or without fluorescence photons.
!
!  For SLAC-R-730/KEK Report 2005-8:  Example of including fluorescence
!
!  The following units are used: unit 6 for output
!***********************************************************************
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
!---------------------------------------------------------------------
!--------------------------- main code -------------------------------
!---------------------------------------------------------------------

!---------------------------------------------------------------------
```

```
! Step 1: Initialization
!----------------------------------------------------------------------

      implicit none

!     ------------
!     EGS5 COMMONs
!     ------------
      include 'include/egs5_h.f'                    ! Main EGS "header" file

      include 'include/egs5_bounds.f'
      include 'include/egs5_edge.f'
      include 'include/egs5_epcont.f'
      include 'include/egs5_media.f'
      include 'include/egs5_misc.f'
      include 'include/egs5_thresh.f'
      include 'include/egs5_useful.f'
      include 'include/egs5_usersc.f'
      include 'include/randomm.f'

!     bounds contains ecut and pcut
!     edge contains iedgfl
!     epcont contains iausfl
!     media contains the array media
!     misc contains med
!     thresh contains ae and ap
!     useful contains RM
!     usersc contains emaxe

!     ----------------------
!     Auxiliary-code COMMONs
!     ----------------------
      include 'auxcommons/aux_h.f'   ! Auxiliary-code "header" file

      include 'auxcommons/pladta.f'

      common/score/bwidth,ebin(50)
      real*8 bwidth,ebin

      real*8 ein,xin,yin,zin,                  ! Arguments
     *       uin,vin,win,wtin
      integer iqin,irin

      real*8 binmax                            ! Local variables
      integer i,icol,j,ncase
      character*24 medarr(1)
      character*4 line(48)

!     ----------
!     Open files
!     ----------
```

```
      open(UNIT= 6,FILE='egs5job.out',STATUS='unknown')


!     ===================
      call counters_out(0)
!     ===================


!-----------------------------------------------------------------------
! Step 2: pegs5-call
!-----------------------------------------------------------------------
!     =============
      call block_set                   ! Initialize some general variables
!     =============


!     --------------------------------
!     define media before calling PEGS5
!     --------------------------------
      nmed=1
      medarr(1)='PB                    '

      do j=1,nmed
        do i=1,24
          media(i,j)=medarr(j)(i:i)
        end do
      end do

! nmed and dunit default to 1, i.e. one medium and we work in cm

      chard(1) = 1.0d0        !  optional, but recommended to invoke
                             !  automatic step-size control


!     ------------------------------------------------
!     Run KEK version of PEGS5 before calling HATCH
!     (method was developed by Y. Namito - 010306)
!     ------------------------------------------------
      write(6,100)
100   FORMAT(' PEGS5-call comes next'/)


!     ==========
      call pegs5
!     ==========


!-----------------------------------------------------------------------
! Step 3: Pre-hatch-call-initialization
!-----------------------------------------------------------------------
      nreg=3
!     nreg : number of region

      med(1)=0
      med(3)=0
      med(2)=1
! Regions 1 and 3 are vacuum, region 2, lead
```

```
      iraylr(2)=1
!     Turn on rayleigh scattering in the slab
      iedgfl(2)=1
!     1: Turn on fluorescence production in the slab
!     0: Turn off fluorescence production 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


!     ---------------------------------------------------------
!     Random number seeds.  Must be defined before call hatch
!     or defaults will be used.  inseed (1- 2^31)
!     ---------------------------------------------------------
      luxlev=1
      inseed=1
      write(6,120) inseed
120   FORMAT(/,' inseed=',I12,5X,
     *            ' (seed for generating unique sequences of Ranlux)')


!     =============
      call rluxinit  ! Initialize the Ranlux random-number generator
!     =============


!-----------------------------------------------------------------------
! Step 4:  Determination-of-incident-particle-parameters
!-----------------------------------------------------------------------
! Define initial variables for 100 keV beam of photons normally incident
! on the slab
      iqin=0
!     Incident photons
!            100 keV
      ein=0.100
      xin=0.0
      yin=0.0
      zin=0.0
!     Incident at origin
      uin=0.0
      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 5:  hatch-call
!-----------------------------------------------------------------------
! Maximum total energy of an electron for this problem must be
! defined before hatch call
      emaxe = ein + RM
```

185

```
      write(6,130)
130   format(/' Start tutor7'/' Call hatch to get cross-section data')

!     ------------------------------
!     Open files (before HATCH call)
!     ------------------------------
      open(UNIT=KMPI,FILE='pgs5job.pegs5dat',STATUS='old')
      open(UNIT=KMPO,FILE='egs5job.dummy',STATUS='unknown')

      write(6,140)
140   FORMAT(/,' HATCH-call comes next',/)

!     ==========
      call hatch
!     ==========

!     ------------------------------
!     Close files (after HATCH call)
!     ------------------------------
      close(UNIT=KMPI)
      close(UNIT=KMPO)

!     Pick up cross section data for lead
      write(6,150) ae(1)-RM, ap(1)
150   format(/' 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

!-----------------------------------------------------------------------
! Step 6:  Initialization-for-howfar
!-----------------------------------------------------------------------
! Define the coordinates and the normal vectors for the two planes.
! Information required by howfar (and auxiliary geometry subprograms)
! and passed through common/pladta/
!
! First plane (the x-y plane through the origin)
      pcoord(1,1)=0.0
      pcoord(2,1)=0.0
      pcoord(3,1)=0.0
! Coordinates
      pnorm(1,1) =0.0
      pnorm(2,1) =0.0
      pnorm(3,1)= 1.0
! Normal vectors
! Second plane (note: slab is 1 cm thick)
      pcoord(1,2)=0.0
      pcoord(2,2)=0.0
      pcoord(3,2)=1.0
! Coordinates
```

```
      pnorm(1,2) =0.0
      pnorm(2,2) =0.0
      pnorm(3,2)= 1.0
! Normal vectors


!------------------------------------------------------------------------
! Step 7:  Initialization-for-ausgab
!------------------------------------------------------------------------
      do i=1,50
        ebin(i) = 0.0
!  Zero scoring array before starting
      end do
      bwidth = 0.002


!------------------------------------------------------------------------
! Step 8:  Shower-call
!------------------------------------------------------------------------
! Initiate the shower ncase times
      ncase=10000
      do i=1,NCASE
        call shower(iqin,ein,xin,yin,zin,uin,vin,win,irin,wtin)
      end do


!------------------------------------------------------------------------
! Step 8:  Output-of-results
!------------------------------------------------------------------------
! Use log10(10000.0) as maximum value
      binmax=dlog10(10000.d0)

      if (iedgfl(2).eq.1) then
        write(6,160) ein,pcoord(3,2)
160     format(/' Reflected photon spectrum'/' for a',F8.2,
     *  ' MeV pencil beam of photons on a',F7.2,
     *  ' cm thick slab of lead'/' with fluorescence photon'//T6,
     *  'Energy  counts/incident photon'/
     *  25X,' log(counts for 10^4 incident photons)')
       else
        write(6,170) ein,pcoord(3,2)
170     format(' Reflected photon spectrum'/' for a',F8.2,
     *  ' MeV pencil beam of photons on a',F7.2,
     *  ' cm thick slab of lead'/' without fluorescence photon'//T6,
     *  'Energy  counts/incident photon'/
     *  25X,' log(counts for 10^4 incident photons)')
       end if

      do j=1,48
        line(j)=' '
      end do
! Blank entire output array
      do j=1,50
        if(ebin(j).gt.0) then
```

```
          icol=
     *          int(dlog10(ebin(j)*10000.0/float(ncase))/binmax*48.0+0.999)
            if (icol.eq.0) icol=1
          else
            icol = 1
          endif
          line(icol)='*'
!  Load output array at desired location
          write(6,180) bwidth*j,ebin(j)/float(ncase),line
180       format(F10.4,F10.4,48A1)
          line(icol)=' '
!  Reblank
        end do

        stop
        end
!------------------------last line of main code------------------------

!-----------------------------ausgab.f---------------------------------
!----------------------------------------------------------------------
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
! ---------------------------------------------------------------------
! Required subroutine for use with the EGS5 Code System
! ---------------------------------------------------------------------
!**********************************************************************
!
!  In this AUSGAB routine for TUTOR7 we score photons reflected
!  from the slab (ir(np)=1 and iq(np)=0).
!**********************************************************************
        subroutine ausgab(iarg)

        implicit none

        include 'include/egs5_h.f'              ! Main EGS "header" file
        include 'include/egs5_stack.f'      ! COMMONs required by EGS5 code

        common/score/bwidth,ebin(50)
        real*8 bwidth,ebin

        integer iarg                                        ! Arguments

        integer ibin,irl                            ! Local variable

        irl=ir(np)                      ! Local variable
        if(irl.eq.1.and.iq(np).eq.0) then    ! Photon is reflected
!      Increment bin corresponding to photon energy
          ibin= min0 (int(e(np)/bwidth + 0.999), 50)
          if (ibin.ne.0) then
            ebin(ibin)=ebin(ibin)+1
          end if
        end if
```

188

```
      return
      end
!------------------------last line of ausgab.f-----------------------
```

The following is the output provided by **tutor7** with and without the fluorescence option (named tutor7_w.out and tutor7_wo.out in the distribution, respectively).

```
 PEGS5-call comes next


 inseed=              1       (seed for generating unique sequences of Ranlux)
 ranlux luxury level set by rluxgo : 1      p=  48
 ranlux initialized by rluxgo from seed            1

 Start tutor7
 Call hatch to get cross-section data

 HATCH-call comes next

 RAYLEIGH OPTION REQUESTED FOR MEDIUM NUMBER   1

 EGS SUCCESSFULLY 'HATCHED' FOR ONE MEDIUM.


 Knock-on electrons can be created and any electron followed down to
                                       0.010 MeV kinetic energy
 Brem photons can be created and any photon followed down to
                                       0.001 MeV


 Reflected photon spectrum
 for a    0.10 MeV pencil beam of photons on a   1.00 cm thick slab of lead
 with fluorescence photon

    Energy   counts/incident photon
                         log(counts for 10^4 incident photons)
    0.0020    0.0000*
    0.0040    0.0000*
    0.0060    0.0000*
    0.0080    0.0000*
    0.0100    0.0000*
    0.0120    0.0024                    *
    0.0140    0.0032                      *
    0.0160    0.0001*
    0.0180    0.0000*
    0.0200    0.0000*
    0.0220    0.0000*
    0.0240    0.0000*
    0.0260    0.0000*
    0.0280    0.0000*
```

189

```
0.0300    0.0000*
0.0320    0.0001*
0.0340    0.0000*
0.0360    0.0001*
0.0380    0.0001*
0.0400    0.0001*
0.0420    0.0000*
0.0440    0.0000*
0.0460    0.0000*
0.0480    0.0000*
0.0500    0.0000*
0.0520    0.0001*
0.0540    0.0000*
0.0560    0.0000*
0.0580    0.0001*
0.0600    0.0002   *
0.0620    0.0002   *
0.0640    0.0002   *
0.0660    0.0006           *
0.0680    0.0002   *
0.0700    0.0009             *
0.0720    0.0007           *
0.0740    0.0529                                *
0.0760    0.0815                                   *
0.0780    0.0008           *
0.0800    0.0004        *
0.0820    0.0003      *
0.0840    0.0003      *
0.0860    0.0340                              *
0.0880    0.0100                     *
0.0900    0.0000*
0.0920    0.0000*
0.0940    0.0000*
0.0960    0.0000*
0.0980    0.0000*
0.1000    0.0006           *
```

PEGS5-call comes next

inseed=   1   (seed for generating unique sequences of Ranlux)
ranlux luxury level set by rluxgo : 1   p= 48
ranlux initialized by rluxgo from seed   1

Start tutor7
Call hatch to get cross-section data

HATCH-call comes next

RAYLEIGH OPTION REQUESTED FOR MEDIUM NUMBER 1

```
EGS SUCCESSFULLY 'HATCHED' FOR ONE MEDIUM.

Knock-on electrons can be created and any electron followed down to
                                      0.010 MeV kinetic energy
Brem photons can be created and any photon followed down to
                                      0.001 MeV
Reflected photon spectrum
for a    0.10 MeV pencil beam of photons on a   1.00 cm thick slab of lead
without fluorescence photon

  Energy   counts/incident photon
                        log(counts for 10^4 incident photons)
  0.0020    0.0000*
  0.0040    0.0000*
  0.0060    0.0000*
  0.0080    0.0000*
  0.0100    0.0000*
  0.0120    0.0000*
  0.0140    0.0000*
  0.0160    0.0000*
  0.0180    0.0001*
  0.0200    0.0000*
  0.0220    0.0001*
  0.0240    0.0000*
  0.0260    0.0000*
  0.0280    0.0000*
  0.0300    0.0000*
  0.0320    0.0000*
  0.0340    0.0000*
  0.0360    0.0000*
  0.0380    0.0000*
  0.0400    0.0000*
  0.0420    0.0000*
  0.0440    0.0000*
  0.0460    0.0000*
  0.0480    0.0000*
  0.0500    0.0000*
  0.0520    0.0001*
  0.0540    0.0000*
  0.0560    0.0000*
  0.0580    0.0000*
  0.0600    0.0000*
  0.0620    0.0000*
  0.0640    0.0001*
  0.0660    0.0000*
  0.0680    0.0000*
  0.0700    0.0000*
  0.0720    0.0002   *
  0.0740    0.0016              *
  0.0760    0.0008         *
```

```
0.0780    0.0014                  *
0.0800    0.0012                *
0.0820    0.0002    *
0.0840    0.0002    *
0.0860    0.0001*
0.0880    0.0000*
0.0900    0.0000*
0.0920    0.0000*
0.0940    0.0000*
0.0960    0.0000*
0.0980    0.0000*
0.1000    0.0011                *
```

Chapter 4

ADVANCED EGS5 USER CODES

In this chapter we present user codes which demonstrate some of the more advanced features and capabilities of the EGS5 Code System. We will show how to transport charged particles in a magnetic field and how to produce and transport fluorescent photons following a photoelectric interaction. A generalized multi-cylinder, multi-slab geometryis presented in which both splitting and leading particle biasing are used in order to reduce the variance and speed up the calculation of energy deposition in accelerator targets and surrounding devices. We also show how to incorporate into EGS5 combinatorial geometry(CG) subprograms derived from MORSE-CG.

All of the user codes presented here are provided with the EGS5 distribution and the reader should study the appropriate program listings themselves in order to gain a more complete understanding of what they have been designed to do.

4.1 UCCYL - Cylinder-Slab Geometry and Importance Sampling

4.1.1 Generalized Multi-Cylinder, Multi-Slab Geometry

The user code UCCYL provides an example of how to set up a geometry consisting of multiple cylindrical shells that are orthogonal to multiple slabs. The groundwork for this generalized scheme was provided in Tutorials 6 and 7 of Chapter 3, where the geometry subprograms `PLANE1` and `PLAN2P` were presented. Several other geometry routines are required in the present case, including `CYLNDR`, which is the cylinder analog to `PLANE1`. For a description of the implementation and use of these and other auxiliary geometry routines provided with EGS5, the user is referred to the document "How to Code Geometry: Writing Subroutine `HOWFAR`," which is part of the EGS5 distribution.

The main purpose of this section, however, is to demonstrate how to incorporate importance sampling into an EGS5 problem, so we will not discuss the geometry techniques any further. Instead, those interested in the cylinder-slab generalization are referred to the commented FORTRAN listing of UCCYL.

4.1.2 Particle Splitting

A version of UCCYL was used to answer a number of questions about energy deposition in a positron production target, as well as regions surrounding such a target. Two of these questions are: what is the temperature rise in the target and what is the radiation damage to nearby vacuum O-rings, when 10^{13} electrons/sec with energies of 33 GeV strike the target?

One of the first things observed during initial EGS runs was that the statistics in the (backward) region designated as an "O-ring" were quite low because

1. the region of interest was too small,

2. the majority of the shower was forward-directed and only low energy radiation headed backwards towards the region, and

3. there was too much shielding material separating the the region of interest from the source.

However, by studying the energy deposition block diagrams (from auxiliary routine ECNSV1) and the associated event counters (from NTALLY), obtained during the initial computer runs, we were able to make the following observations and recommendations:

- Most of the charged particles depositing energy come from photons interacting in the region of interest–not from charged particles entering from surrounding regions (ı.e. the "cavity" is fairly big).

- The region ("cavity") is small enough, however, that only a *fraction* of the photons traversing the region interact (verified by simple calculation).

- Suggestion: Every time a photon first enters the region, split it into 10 identical photons, each carrying a weight of $\frac{1}{10}$ to the progeny; include the weight in the EDEP (and any other) scoring.

This was accomplished by placing the following statement in HOWFAR:

```
if (irsplt(irl).eq.1.and.irl.ne.irold.and.iq(np).eq.0) then
    ! Apply particle splitting
  if (lsplt.ne.0) then
```

```
        wt(np)=wt(np)/lsplt
        do isplt=1,nsplt
          x(np+isplt)=x(np)
          y(np+isplt)=y(np)
          z(np+isplt)=z(np)
          u(np+isplt)=u(np)
          v(np+isplt)=v(np)
          w(np+isplt)=w(np)
          ir(np+isplt)=ir(np)
          wt(np+isplt)=wt(np)
          dnear(np+isplt)=dnear(np)
          latch(np+isplt)=latch(np)
          e(np+isplt)=e(np)
          iq(np+isplt)=iq(np)
          k1step(np+isplt)=0.d0
          k1rsd(np+isplt)=0.d0
          k1init(np+isplt)=0.d0
        end do
        np=np+nsplt
      end if
    end if
```

where `IRL` is a local variable (`IR(NP)=` region of interest), `IRSPLT(IRL)` is a flag telling us that splitting is "turned on", and `IRL.NE.IRLOLD` provides us with a way of doing the splitting only when the photon (`IQ(NP)=0`) first enters the region.

Most of the variables above are associated with the current (i.e. `NP`) particle on the "stack" (i.e. `COMMON/STACK/`), and with the number of splits (`LSPLT=10` in UCCYL).

For convenience, the common

```
    common/passit/irsplt(66),lsplt,nsplt,nreg
    integer irsplt,lsplt,nsplt,nreg
```

was also included in the user code (`NSPLT=LSPLT-1`).

The result of this effort was an increase in the *number* of energy deposition events in the *important* region of interest and, as a result, a decrease in the variance (as judged by batch-run statistics.

4.1.3   Leading Particle Biasing

The second application of importance sampling that we will discuss primarily concerns the deposition of energy in an electromagnetic shower initiated by a high energy electron (or photon). The analog approach that is used throughout EGS5, in which each and every particle is generally followed to completion (ı.e. energy cutoff guarantees that high energy shower calculations will take lots of time. For the most energetic particle energies under consideration at present day accelerators (> 50 GeV), one can barely manage to simulate showers in this fashion because of computer time limitations (Recall that execution time per incident particle grows linearly with energy). Fortunately, a certain class of problems involving the calculation of energy deposition is well-suited for a non-analog treatment known as *leading particle biasing*[175]. Examples of such problems are radiobiological dose, heating effects, and radiation damage, although some care must be taken in not being too general with this statement (more on this later).

As a rule, variance reduction techniques of this type should only be used when there is some prior knowledge of the physical processes that are the most (or least) important to the answer one is looking for. Leading particle biasing is a classic example of this. The *most important* processes in the <u>development</u> of an EM shower, at least in terms of <u>total energy deposition</u>, are *bremsstrahlung* and *pair production*. Furthermore, after every one of these interactions the particle with the *higher* of the two energies is expected to contribute *most* to the total energy deposition.

Leading particle biasing is very easily implemented within the framework of EGS5 by means of the following statements:

```
    if (iarg.eq.7) then   ! Apply Leading Particle Biasing for brems.
      eks=e(np)+e(np-1)-RM   ! Kinetic energy before brems.
      ekenp=e(np)
      if(iq(np).ne.0) ekenp=e(np)-RM
      call randomset(rnnolp)
      if (rnnolp.lt.ekenp/eks) then  ! Follow np
        e(np-1)=e(np)
        iq(np-1)=iq(np)
        u(np-1)=u(np)
        v(np-1)=v(np)
        w(np-1)=w(np)
      end if
      ekenp=e(np-1)
      if (iq(np-1).ne.0) ekenp=e(np-1)-RM
      wt(np-1)=wt(np-1)*eks/ekenp
      np=np-1
    end if

    if (iarg.eq.16) then  ! Apply Leading Particle Biasing for pair.
      eks=e(np)+e(np-1)-2.0*RM
```

```
      ekenp=e(np)-RM
      call randomset(rnnolp)
      if (rnnolp.lt.ekenp/eks) then ! Follow np
        e(np-1)=e(np)
        iq(np-1)=iq(np)
        u(np-1)=u(np)
        v(np-1)=v(np)
        w(np-1)=w(np)
      end if
      ekenp=e(np-1)-RM
      wt(np-1)=wt(np-1)*eks/ekenp
      np=np-1
    end if
```

These statements to apply leading particle biasing for bremsstrahlung and pair are included in a version of UCCYL (called UC_LP) at the proper locations within subroutine AUSGAB.

If IAUSFL(8) or IAUSFL(17) is set as 1, AUSGAB is called for IARG of 7 and 16 respectively, then Leading Particle Biasing is applied after bremsstrahlung and pair production, respectively. The COMMON/EPCONT/ is now needed in MAIN in order to pass the IAUSFL(8) and IAUSFL(17) flags (1=on, 0=off). The subroutine RANDOMSET is explained in the EGS5 User Manual (see Appendix B). The other quantities are best understood with the help of the EGS5 Flow Diagrams for ELECTR and PHOTON (see Appendix A).

Either statement can be explained as follows. An interaction of the proper type occurs and the flag has been turned "on". A random number, uniformly distributed between 0 and 1, is drawn and compared with the *fraction* of the kinetic energy that was assigned to the current particle—*i.e.*, the lower energy particle of the two produced in the interaction. If the random number is less than this fraction, the lower energy (NP) particle is kept and the one below it on the stack (NP-1) is thrown away. Otherwise the higher energy particle (NP-1) is selected and the current particle (NP) is tossed out. Obviously the particles in the shower with the highest energy will preferentially be selected by such a scheme and, to assure that the Monte Carlo game is "played fairly", we have imposed two necessary requirements:

- The lowest energy particles must be selected *some* of the time (which we have done by *sampling*).

- The proper weight must be assigned to whichever particle chosen.

To complete the process, one should make use of the particle weight when scoring information in subroutine AUSGAB. Namely,

- Sum particle weights (WT(NP)) when "counting" particles.

- Sum the weighted energy deposition (`WT(NP)*EDEP`).

The easiest way to see that the weights have been properly assigned in the statements above is by example. Assume that the incident particle has a kinetic energy of 1000 MeV and that one of the progeny has energy 100 MeV and the other has 900 MeV. Then clearly the 100 MeV particle will be chosen 10% of the time and will have a weight of $\frac{1000}{100}$, whereas the 900 MeV particle will be chosen 90% of the time and will have a weight of $\frac{1000}{900}$. But the total particle count will *average* out to two (i.e. $0.1 \times \frac{1000}{100} + 0.9 \times \frac{1000}{900} = 2$).

This scheme has been found to increase the speed of shower calculations by a factor of 300 at 33 GeV. However, because the biasing can be somewhat severe at times during the calculation, the weights that are assigned tend to become rather large, and the net result is that the overall efficiency ($\frac{1}{\text{variance} \times \text{time}}$) is not usually 300 times better. Nevertheless, factors of 20 or more are generally obtained for many problems[75] and this technique can be invaluable.

As an example, at one time it took about one minute of CPU time on the IBM-3081 to completely generate *one* 50 GeV shower in a large absorber. When simulating "real" beams of particles having spatial and/or angular distributions, analog EGS calculations gave only 60 incident events in one hour runs, which was inadequate. The use of leading particle biasing with a factor of 300 increase in speed, on the other hand, produced 18,000 events/hour, which was sufficient.

As a final note, it should be understood that leading particle biasing, or any importance sampling scheme for that matter, should not be attempted in an arbitrary manner. For example, one should *not* use leading particle biasing for the "O-ring" problem above since, even though many more incident shower events will certainly be generated, the weights that are assigned to the low energy particles heading in the direction of the "O-ring" region will tend to be quite high, and no significant reduction in the variance will result. Quite the contrary, biasing of this sort can lead to very erroneous results, and one should really have a full grasp (i.e. pre-knowledge) of the important aspects of the radiation transport before attempting to apply any of these variance reduction methods with EGS5 (or any Monte Carlo program for that matter).

4.2   UCBEND - Charged Particle Transport in a Magnetic Field

Charged particle motion in a magnetic field is governed by the Lorentz force equation

$$\vec{F} = q\vec{v} \times \vec{B} = \frac{d\vec{p}}{dt} \tag{4.1}$$

where $\vec{B}$ is the magnetic field strength vector, $\vec{v}$ is the velocity vector, and q is the electric charge of the particle. This equation can be expanded into its Cartesian components to give

$$
\begin{aligned}
m\ddot{x} &= q(\dot{y}B_z - \dot{z}B_y) \\
m\ddot{y} &= q(\dot{z}B_x - \dot{x}B_z) \\
m\ddot{z} &= q(\dot{x}B_y - \dot{y}B_x)
\end{aligned}
\tag{4.2}
$$

where $\dot{x}$ and $\ddot{x}$ (for example) are the x-components of velocity and acceleration, respectively. No attempt is made to solve these coupled differential equations. Instead, in the example user code called UCBEND, we use a simple vector expression for the change in the direction of motion after a *very small* translation, Δl, in a constant magnetic field strength, B, in which the particle energy is assumed to remain constant. Namely,

$$\hat{v} = \hat{B}(\hat{v}_0 \cdot \hat{B}) + [\hat{v}_0 - \hat{B}(\hat{v}_0 \cdot \hat{B})] \cos\alpha - \hat{v}_0 \times \hat{B} \sin\alpha \qquad (4.3)$$

where

$$
\begin{aligned}
\alpha &= \Delta l/R, \\
R &= \text{radius of curvature} = qp/B, \\
p &= \text{particle momentum}, \\
\hat{B} &= \text{magnetic field unit vector}, \\
\hat{v}_0 &= \text{particle direction unit vector (before)} \\
&= U_0\hat{i} + V_0\hat{j} + W_0\hat{k}, \\
\hat{v} &= \text{particle direction unit vector (after)} \\
&= U\hat{i} + V\hat{j} + W\hat{k}.
\end{aligned}
$$

The problem that we attempt to solve with UCBEND relates to a series of spectral measurements performed at SLAC [100]. An RF cavity was tested and found to emit high levels of radiation. Because the cavity was subjected to very high power levels, surface electrons were found to be produced from two locations within the structure and were accelerated to kinetic energies of 8.5 and 3.5 MeV, respectively, depending on their point of origin. Upon exiting the cavity they passed through a 0.015 inch copper window, where they lost energy primarily by ionization and were multiply scattered. The experimentalists attempted to measure the spectrum of the electrons using a magnet, a lead slit, and a Faraday cup. A strong peak corresponding to the 8.5 MeV group was observed, but the 3.5 MeV peak was not observed. The object of the EGS5 simulation was to try to understand these observations, particularly with the help of graphics showing particle trajectories.

Subroutine `HOWFAR` of UCBEND was designed using the diagram shown in Figure 4.1. Electrons start from the origin (at plane 1), transport through the copper window (Region 2), emanate out into the air (Region 3), and pass through the magnetic field, which is constant and along the positive y-direction (in Region 4 *only*). They are then bent in the direction of the lead wall (Regions 6 and 8) where some of them pass through the slit (Region 7) and get scored (at Plane 10). By varying the field strength of the magnet one should be able to re-create the observations for both 3.5 MeV and 8.5 MeV.

In order to gain a more complete understanding of the geometry, it is suggested that the reader study the listing for UCBEND, which is included with the EGS5 distribution. What is of interest here is the method that we employ in order to transport the electrons through the magnetic field region. Using Equation 4.3 with $\Delta l/R << 1$, we obtain

$$U = (U_0 + \alpha W_0)F ,$$

199
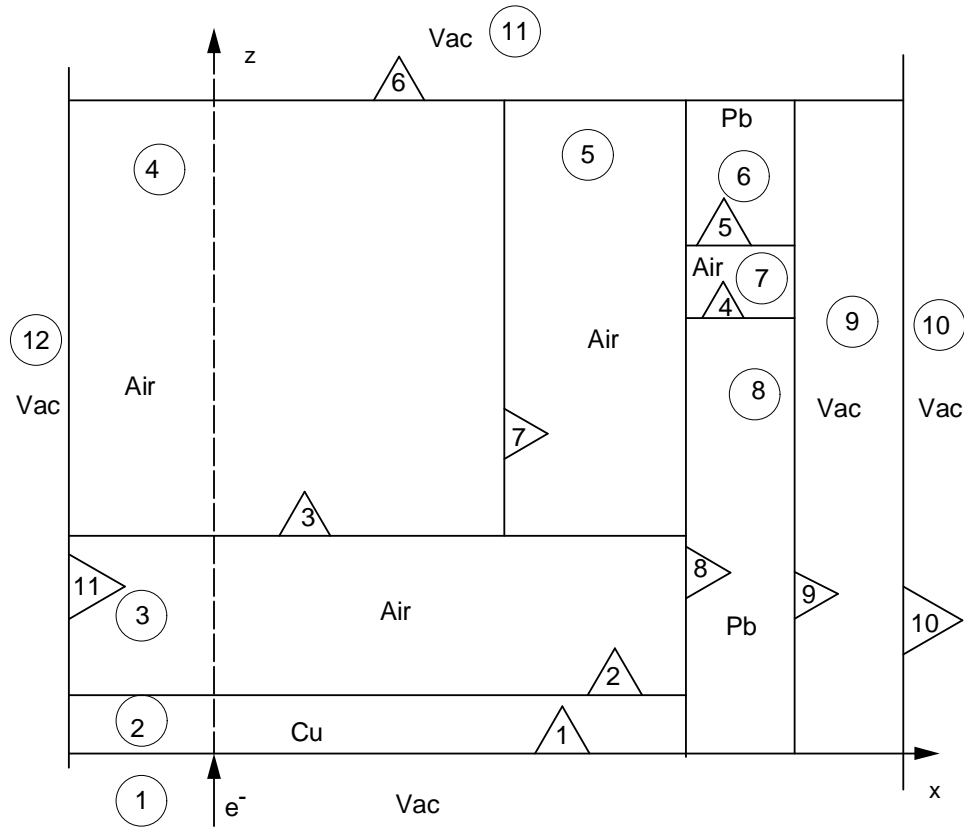
Figure 4.1: Diagram used with UCBEND (not to scale).

$$V = V_0 \;, \tag{4.4}$$
$$W = (W_0 - \alpha U_0)F \;,$$

where

$$F = \text{unit vector renormalization factor}$$
$$= (1 + \alpha^2)^{-1/2} \;. \tag{4.5}$$

The following code listing, which pertains only to that `HOWFAR` section relating to Region 4, should be fairly self-explanatory (Note that `USTEP` has been limited to 0.1 cm in this example, but this can be changed by the user).

```
else if (irl.eq.4) then
  if (ifield.ne.0.and.iq(np).ne.0) then
    psq=e(np)*e(np)-RMSQ
    if (psq.lt.0.0) psq=1.0E-9
    p=sqrt(psq)
```
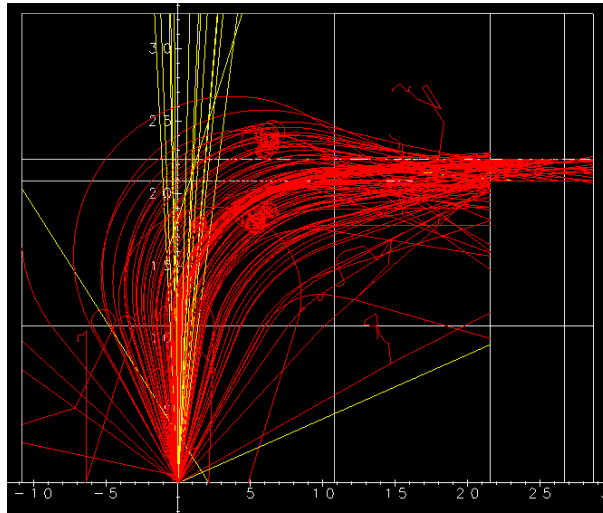
Figure 4.2: UCBEND simulation at 8.5 MeV (B=2.6 kG).

```
poverb=p/(bfield*0.3)
u0=u(np)
v0=v(np)
w0=w(np)
rcurv=-iq(np)*poverb
if (ustep.gt.0.1) then
  ustep=dmin1(ustep,0.1*abs(poverb))
end if
  ustep0=ustep
  irnew=irl
  alpha=ustep/rcurv
  renorf=1./sqrt(1.+alpha*alpha)
  w(np)=(w0-u0*alpha)*renorf
  u(np)=(u0+w0*alpha)*renorf
  v(np)=v0
  call plan2p(6,11,1,3,3,-1)
  call plan2p(7,5,1,11,12,-1)
else
  call plan2p(6,11,1,3,3,-1)
  call plan2p(7,5,1,11,12,-1)
end if
```

The result of running UCBEND at 8.5 and 3.5 MeV is shown in Figure 4.2 and Figure 4.3, respectively, for 100 incident electrons (the red lines are electrons and the yellow lines are photons).[1]

---

[1]These figures were created by using a freely distributed program called "cgview," which is briefly described in section 4.3.
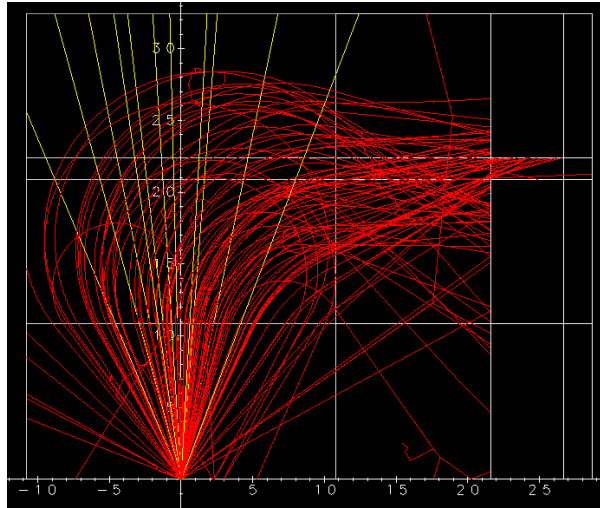
Figure 4.3: UCBEND simulation at 3.5 MeV (B=1.0 kG).

It becomes clear from pictures such as these that it is easier to focus the 8.5 MeV electrons through the slit than the 3.5 MeV ones. A number of EGS5 "experiments" can be performed rather easily now that the basic geometry and magnetic field transport is established. For example, one can replace the air regions and/or the copper window with vacuum in order to determine what effect they have. What was discovered by doing this is that the copper window has the greatest effect on the transmission through the slit. Note that Figure 4.4, illustrates what happens when the magnetic field is turned off. Users interested in charged particle transport in an *electric field* should see Rawlinson, Bielajew, Munro and Galbraith [133].

4.3   Using Combinatorial Geometry with EGS5

All of the user codes presented thus far have treated problems with simple geometries involving few regions and only basic shapes (parallel slabs and right circular cylinders). Construction of sub-routine HOWFAR for such geometries is rather straight-forward, as distances to boundaries are easily determined for planar and cylindrical surfaces, and identifying new region numbers subsequent to boundary crossings is almost trivial when there are few regions with limited intersections. However, for simulations involving complex devices such as radiotherapy accelerator heads or high energy physics detectors, subroutine HOWFAR can be quite cumbersome to construct. In such applications, regions can have complex shapes for which distance to boundary computations are more involved (*e.g.*, in concave regions there can be multiple intersection points on the same surface), and/or have surfaces which are bordered on one side by a large number of distinct volumes (*e.g.*, at a collimators interface), making region identification more difficult.

As the need to simulate complex geometry is a common problem in Monte Carlo modeling,
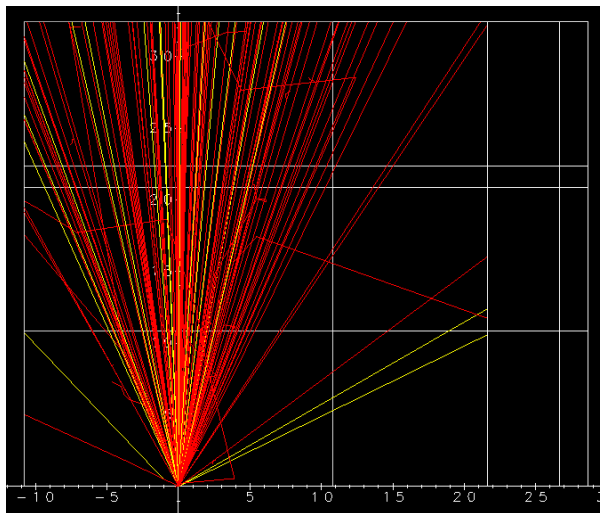
Figure 4.4: UCBEND simulation at 8.5 MeV (B=0 kG).

programs have been developed to assist in the construction of problem geometries by defining regions to be the unions and/or intersections of sets of basic volume elements of varying sizes. In Monte Carlo applications, this method for creating geometries through the combining of smaller volumes has come to be referred to as "combinatorial geometry,", or CG. Note that this usage is not exact, as the term combinatorial geometry actually denotes a much more extensive mathematical field (see, for example, the book by Pach and Agarwal[129]).

In this section we describe the advanced EGS5 user code UCSAMPCG, which exploits the features of combinatorial geometry. The CG package invoked by UCSAMPCG is included in the EGS5 distribution in the auxiliary source code file `cg_related.f`, and is based on the well-known MORSE-CG[168] combinatorial geometry Monte Carlo program. The MORSE-CG package was originally adapted to EGS4-PRESTA by Torii and Sugita [171], and Sugita and Torii have further modified it for EGS5 [169]. The subprograms of `cg_related.f` have retained the MORSE-CG methodology for constructing and evaluating geometry, and the EGS5 implementation is compatible with MORSE-CG input files. Five elemental volume shapes can be modeled: rectangular parallelepipeds (RPP), spheres (SPH), right circular cylinders (RCC), truncated right angle cones (TRC), and tori (TOR). As mentioned above, more complicated volumes can be specified by appropriately intersecting these five basic bodies. The reader is referred to the MORSE-CG manual [168] and the Torii and Sugita citations [171, 169] for detailed descriptions of the CG adaptation and implementation, as well as for instructions on creating input files.

User code UCSAMPCG uses combinatorial geometry to simulate the same problem which the sample user code UCSAMPL5, presented in great detail in the EGS5 User Manual (see Appendix B of this report), treats with a simple geometry model. The coupling of the CG package to EGS5 is accomplished entirely within subroutine `HOWFAR`, which performs the actual boundary and region checks. A once-only initialization is done by calling the CG routine `GEOMGT`, which reads in the

203

necessary geometry input. The following data, which is in the familiar MORSE-CG format [168] and which is included as file **ucsampcg.data** of the EGS5 distribution, is used by UCSAMPCG to model with CG the same geometry treated by UCSAMPL5 in a non-CG implementation.

```
  RPP    1           -1.5          1.5          -1.5          1.5           0.0
                      3.0
  RPP    2           -2.5          2.5          -2.5          2.5          -1.0
                      4.0
  END
   Z1         +1
   Z2         +2        -1
  END
   1    0
```

The `HOWFAR` routine supplied with UCSAMPCG is general and can be used for any CG geometry which can be specified by the five types of bodies listed above. All that is required of the user is the creation of an appropriately formatted input data file. It must be noted, though, that because of the computational overhead introduced in generalizing `HOWFAR`, for problems involving simple, regular geometries (such as cylindrical slabs or rectangular pixels), EGS5 user codes using CG will execute 1-2 times more slowly than user codes tailored to the particular problem geometries. (This is a substantial improvement relative to the CG implementation in EGS4, which was "significantly less efficient than the macro-geometry methods" [171].) For many problems, the greatly facilitated ease in setting up the simulation geometry with CG will more than offset the slight penalty in computational efficiency.

To facilitate the construction of CG geometry input files and to provide for the display of particle trajectories from EGS5 simulations, Namito and co-workers [123] developed a stand-alone visual interface program, cgview, which is freely distributed on the Internet at:

    http://rcwww.kek.jp/research/egs/kek/cgview/.

The geometry used in UCSAMPCG and some sample output particle trajectories drawn with cgview are shown in Fig. 4.5.

Cgview can also be used to assist in the construction of CG input data files. For geometries with simple shapes, users can select and position volume elements visually using tools provided with cgview, and later have the program create output data files in the correct MORSE-CG format. Additionally, the integrity of existing CG input data files can be checked through the use of the cgview pseudo particle feature. This function allows the user to specify locations and directions for pseudo source particles, which are then tracked by cgview through the input geometry, searching for regions of overlapping and undefined zones. Thus, errors in geometry specification may be corrected prior to being used in Monte Carlo simulations.
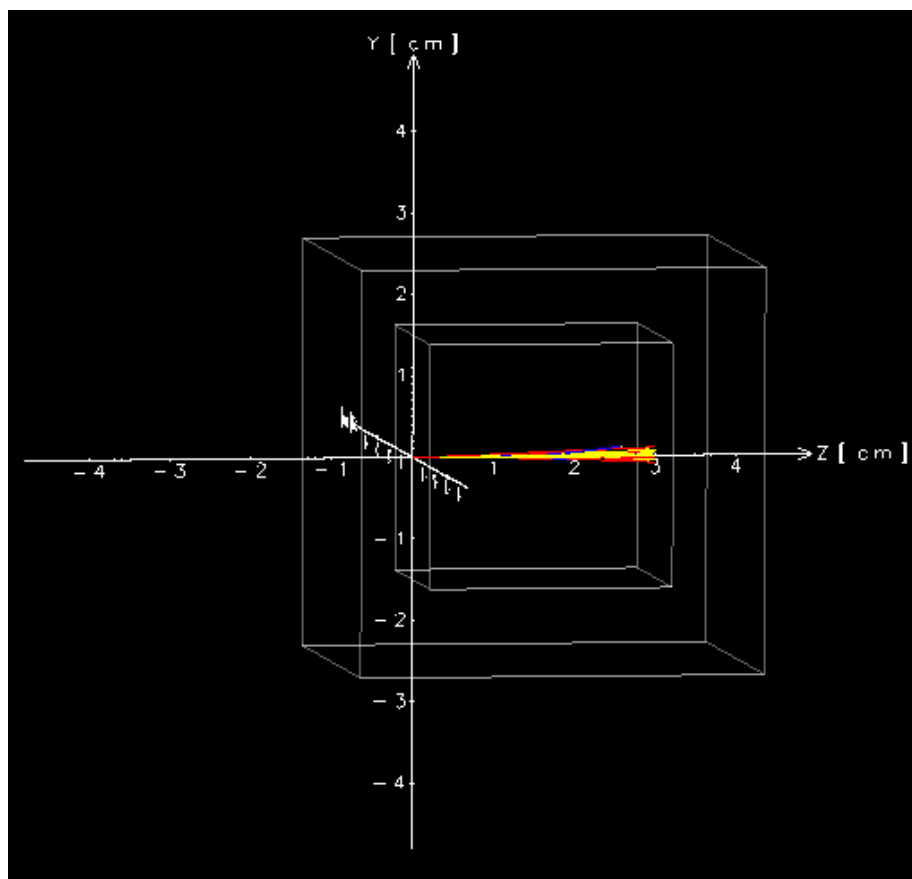
Figure 4.5: Geometry and particle trajectory of UCSAMPCG simulation. The stand alone program **cgview** was used to draw the picture.

Appendix A

EGS5 FLOW DIAGRAMS

Hideo Hirayama and Yoshihito Namito
Radiation Science Center
Advanced Research Laboratory
High Energy Accelerator Research Organization (KEK)
1-1 Oho Tsukuba-shi Ibaraki-ken 305-0801 JAPAN

KEY for EGS5 Flow Diagrams

subroutine
annih
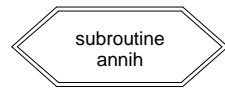
Beginning of subprogram.

iannih=iannih+1

Statement.

yes ← rnnow
.gt.
rejf
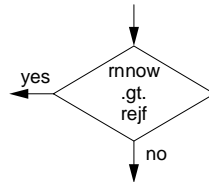
Conditional statement.

no

i=1
i=i+1   i>10 — yes

DO loop.

no

call
randomset(rnnow)

Call to external subprogram.

return
(to photon)

Return statement.

write(6,102) medium

Output (or Input) related statement.

stop

Stop statement.

Jump over flow chart line.

207

```
subroutine
annih
```

```
iannih=iannih+1
```

```
avip = e(np) + RM
a    = avip/RM
g    = a - 1.0
t    = g - 1.0
p    = sqrt(a*t)
pot  = p/t
ep0  = 1.0/(a + p)
```

"Sample 1/ep from ep = ep0 to 1 - ep0"

```
call
randomset(rnnow)
```

```
ep = ep0*exp(rnnow*log((1.0 - ep0)/ep0))
```

```
call
randomset(rnnow)
```

```
rejf = 1.0 - ep + (2.0*g - 1.0/ep)/a**2
```

"Rejection sampling loop"

```
rnnow
.gt.
rejf
```

yes

no

```
ep       = max(ep,1.D0 - ep)
esg1     = avip*ep
e(np)    = esg1
esg2     = avip - esg1
e(np + 1) = esg2
iq(np)    = 0
costhe = (esg1 - RM)*pot/esg1
costhe = min(1.D0,costhe)
sinthe = sqrt((1.0 - costhe)*(1.0 + costhe))
```

```
call
uphi(2,1)
```

```
np     = np + 1
iq(np) = 0
costhe = (esg2 - RM)*pot/esg2
costhe = min(1.D0,costhe)
sinthe = - sqrt((1.0 - costhe)*(1.0 + costhe))
```

```
call
uphi(3,2)
```

```
return
(to photon)
```

208

subroutine
aphi(br)

iaphi = iaphi +1

iarg=21

iausfl(iarg+1)
. ne.
0

yes → call
ausgab(iarg)

no

pnorm0 = sqrt(u(np)*u(np) + v(np)*v(np) + w(np)*w(np))
u(np) = u(np)/pnorm0
v(np) = v(np)/pnorm0
w(np) = w(np)/pnorm0
valmax = br + 1./br

call
randomset(rnnow)

ph0 = rnnow*twopi
sinph0 = sin(ph0)
cph0 = pi5d2 - ph0
cosph0 = sin(cph0)
valloc = sqrt(sinph0*sinph0 + cosph0*cosph0)
sinph0 = sinph0/valloc
cosph0 = cosph0/valloc
val = (valmax - 2.*sinthe*sinthe*cosph0*cosph0)/valmax

call
randomset(rnnow)

rnnow
. le.
val

no

"Rejection sampling loop"

yes

anorm2 =costhe*costhe*cosph0*cosph0 + sinph0*sinph0

call
randomset(rnnow)

(valmax - 2.)/(valmax - 2. + 2.*anorm2) .gt. rnnow
.or.
anorm2 .lt. 1.E-10

yes → ldpola = 1

no

ldpola = 0

1

209

1

```
ldpola
. eq.
1
```

no →
```
anormr = 1./sqrt(anorm2)
sineta = -anormr*sinph0
coseta = anormr*costhe*cosph0
```

yes →
```
call
randomset(rnnow)
```
→
```
eta = rnnow*twopi
sineta = sin(eta)
ceta = pi5d2 - eta
coseta = sin(ceta)
```

```
ufa = costhe*cosph0*coseta - sinph0*sineta
vfa = costhe*sinph0*coseta + cosph0*sineta
wfa = -sinthe*coseta
asav = u(np)
bsav = v(np)
csav = w(np)
sinps2 = asav*asav + bsav*bsav
```

```
sinps2
. lt.
1.E-20
```

no →
```
sinpsi = sqrt(sinps2)
sindel = bsav/sinpsi
cosdel = asav/sinpsi
cosomg = cosdel*csav*uf(np) +
         sindel*csav*vf(np) - sinpsi*wf(np)
sinomg = -sindel*uf(np) + cosdel*vf(np)
```

yes →
```
cosomg = uf(np)
sinomg = vf(np)
```

```
enorm = sqrt(uf(np)*uf(np) + vf(np)*vf(np) + wf(np)*wf(np))
```

```
enorm
. lt.
1.E-4
```

yes →
```
call
randomset(rnnow)
```
→
```
omg = rnnow*twopi
sinomg = sin(omg)
comg = pi5d2 - omg
cosomg = sin(comg)
```

no

```
cosphi = cosomg*cosph0 - sinomg*sinph0
sinphi = sinomg*cosph0 + cosomg*sinph0
ufb = cosomg*ufa - sinomg*vfa
vfb = sinomg*ufa + cosomg*vfa
wfb = wfa
```

```
sinps2
. lt.
1.E-20
```

yes →
```
uf(np) = ufb
vf(np) = vfb
wf(np) = wfb
```

no

```
uf(np) = cosdel*csav*ufb - sindel*vfb + asav*wfb
vf(np) = sindel*csav*ufb + cosdel*vfb + bsav*wfb
wf(np) = -sinpsi*ufb + csav*wfb
```

```
enorm = sqrt(uf(np)*uf(np) + vf(np)*vf(np) + wf(np)*wf(np))
uf(np) = uf(np)/enorm
vf(np) = vf(np)/enorm
wf(np) = wf(np)/enorm
```

```
return
(to photon)
```

210

```
                    ╱╲
                   ╱  ╲
                  ╱ subroutine ╲
                  ╲  bhabha    ╱
                   ╲  ╱
                    ╲╱
                     │
                     ▼
              ┌──────────────────┐
              │ ibhabha = ibhabha +1 │
              └──────────────────┘
                     │
                     ▼
  ┌────────────────────────────────────────────────────┐
  │ eip   = e(np)          ;ekin  = eip - RM            │
  │ t0    = ekin/RM        ;e0    = t0 + 1.0            │
  │ yy    = 1.0/(t0 + 2.0) ;e02   = e0*e0               │
  │ betai2= e02/(e02-1.0)  ;ep0   = te(medium)/ekin     │
  │ ep0c  = 1.0 - ep0      ;y2    = yy*yy               │
  │ yp    = 1.0 - 2.0*yy   ;yp2   = yp*yp               │
  │ b4    = yp2*yp         ;b3    = b4 + yp2            │
  │ b2    = yp*(3.0 + y2)  ;b1    = 2.0 - y2            │
  │ br    = ep0                                         │
  │ re1   = ep0c*(betai2-br*(b1-br*(b2-br*(b3-br*b4)))) │
  │ br    = 1.0                                         │
  │ re2   = ep0c*(betai2-br*(b1-br*(b2-br*(b3-br*b4)))) │
  │ remax = max(re1,re2)                                │
  └────────────────────────────────────────────────────┘
```

"Rejection sampling loop"

```
              ┌──────────────────┐
              │ call             │
              │ randomset(rnnow) │
              └──────────────────┘
                     │
                     ▼
          ┌──────────────────────────┐
          │ br = ep0/(1.0 - ep0c*rnnow) │
          └──────────────────────────┘
                     │
                     ▼
              ┌──────────────────┐
              │ call             │
              │ randomset(rnnow) │
              └──────────────────┘
                     │
                     ▼
    ┌──────────────────────────────────────────────────┐
    │ rejf = ep0c*(betai2-br*(b1-br*(b2-br*(b3-br*b4)))) │
    │ rejf = rejf/remax                                 │
    └──────────────────────────────────────────────────┘
                     │
                     ▼
                    ╱╲
        yes        ╱  ╲
   ◄──────────────╱ rnnow ╲
                  ╲ . gt.  ╱
                   ╲ rejf ╱
                    ╲  ╱
                     ╲╱
                     │ no
                     ▼
```

"Put e- on top
of stack"

```
                    ╱╲
                   ╱  ╲       yes    ┌──────────────────┐
                  ╱ br  ╲────────────►│ iq(np+1) = -1    │
                  ╲ . lt. ╱           │ k1step(np+1) = 0. │
                   ╲ 0.5 ╱            │ k1init(np+1) = 0. │
                    ╲  ╱              │ k1rsd(np+1) = 0.  │
                     ╲╱               └──────────────────┘
```

"Put e+ on top
of stack"

```
                     │ no
                     ▼
          ┌──────────────────────────┐
          │ iq(np) = -1              │
          │ iq(np+1) = 1             │
          │ br = 1.0 - br            │
          │ k1step(np+1) = k1step(np) │
          │ k1init(np+1) = k1init(np) │
          │ k1rsd(np+1) = k1rsd(np)  │
          │ k1step(np) = 0.          │
          │ k1init(np) = 0.          │
          │ k1rsd(np) = 0.           │
          └──────────────────────────┘
                     │
                     ▼
                     1
```

1

```
br = max(br,0.D0)
ekse2 = br*ekin
ese1 = eip - ekse2
ese2 = ekse2 + RM
e(np) = ese1
e(np+1) = ese2
h1 = (eip + RM)/ekin
dcosth = h1*(ese1 - RM)/(ese1 + RM)
ttt = 1.0 - dcosth
```

ttt
. le.
0.0

yes → sinthe = 0.0

no

sinthe = sqrt(1.0 - dcosth)

costhe = sqrt(dcosth)

call
uphi(2,1)

```
np = np + 1
dcosth = h1*(ese2 - RM)/(ese2 + RM)
ttt = 1.0 - dcosth
```

ttt
. le.
0.0

yes → sinthe = 0.0

no

sinthe = -sqrt(1.0 - dcosth)

costhe = sqrt(dcosth)

call
uphi(3,2)

return
(to electr)

212

subroutine
brems

ibrems = ibrems +1

eie = e(np)
np = np + 1

eie
.lt.
50.0

yes → "Bethe-Heitler distribution"

lvx = 1
lvl0 = 0

"Coulomb-corrected
Bethe Heitler distribution"

no

lvx = 2
lvl0 = 3

abrems = float(int(AILN2*log(eie/ap(medium))))

call
randomset(rnnow)

0.5
.lt.
(abrems*alphi(lvx,medium) + 0.5)*rnnow

no → "2br
subdistribution
sampling"

"(1-br)/br
subdistribution
sampling"

yes

call
randomset(rnnow)

idistr = abrems*rnnow
p = pwr2i(idistr+1)
lvl = lvl0 + 1

call
randomset(rnnow)

call
randomset(rnnow1)

call
randomset(rnnow2)

br = max(rnnow1,rnnow2)
lvl = lvl0 + 2

rnnow
.ge.
AI2LN2

no

"br <0.5"

call
randomset(rnnow)

br = rnnow*0.5

yes

call
randomset(rnnow)

call
randomset(rnnow1)

call
randomset(rnnow2)

h = max(rnnow1,rnnow2)
br = 1.0 - 0.5*h

rnnow
.gt.
0.5/br

yes

"Rejection sampling loop"

no

"Rejection sampling loop"

1

2

3

213

1   2   3

br = br*p

esg = eie*br

esg .lt. ap(medium) — yes → 1

no

ese = eie - esg

ese .lt. RM — yes →

no

del = br/ese

del .ge. delpos(lvx,medium) — yes →

no

delta = delcm(medium)*del

delta .lt. 1.0

yes → rejf = dl1(lvl,medium) + delta*(dl2(lvl,medium) + delta*dl3(lvl,medium))

no → rejf = dl4(lvl,medium) + dl5(lvl,medium)*log(delta + dl6(lvl,medium))

call randomset(rnnow)

rnnow .gt. rejf — yes → 1

no

"Photon energy accepted"

ibrdst .ne. 1 — yes → theta = RM/eie

no

"Rejection sampling loop"

```
ztarg = zbrang(medium)
tteie = eie/RM
ttese = ese/RM
esedei = ttese/tteie
y2max = (PI*tteie)**2
rjarg1 = 1.0 + esedei**2
rjarg2 = 3.0*rjarg1 - 2.0*esedei
rjarg3 = ((1.0 - esedei)/(2.0*tteie*esedei))**2
y2tst1 = (1.0 + 0.0e0)**2
rejmin = (4.0 + log(rjarg3 + ztarg/y2tst1))*
(4.0*esedei*0.0e0/y2tst1 - rjarg1) + rjarg2
y2tst1 = (1.0 + 1.0e0)**2
rejmid = (4.0 + log(rjarg3 + ztarg/y2tst1))*
(4.0*esedei*1.0e0/y2tst1 - rjarg1) + rjarg2
y2tst1 = (1.0 + y2max)**2
rejmax = (4.0 + log(rjarg3 + ztarg/y2tst1))*
(4.0*esedei*y2max/y2tst1 - rjarg1) + rjarg2
rejtop = max(rejmin,rejmid,rejmax)
```

4   5

"Rejection sampling loop"

call
randomset(rnnow)

y2tst =rnnow/(1.0 - rnnow + 1.0/y2max)
y2tst1 = (1.0 + y2tst)**2
rejtst = (4.0 + log(rjarg3 + ztarg/y2tst1))*
(4.0*esedei*y2tst/y2tst1 - rjarg1) + rjarg2

call
randomset(rnnow)

rnnow
.gt.
(rejtst/rejtop)

yes

no

theta = sqrt(y2tst)/tteie

call
uphi(1,3)

esg
.le.
ese

yes

no

iq(np) = iq(np-1)
iq(np-1) = 0
e(np) = ese
e(np-1) = esg
t = u(np)
u(np) = u(np-1)
u(np-1) = t
t = v(np)
v(np) = v(np-1)
v(np-1) = t
t = w(np)
w(np) = w(np-1)
w(np-1) = t
k1step(np) = k1step(np-1)
k1step(np-1) = 0.
k1init(np) = k1init(np-1)
k1init(np-1) = 0.
k1rsd(np) = k1rsd(np-1)
k1rsd(np-1) = 0.

iq(np) = 0
e(np) = esg
e(np-1) = ese
k1step(np) = 0.
k1init(np) = 0.
k1rsd(np) = 0.

return
(to electr)

```
         subroutine
         collis
         (lelec,irl,sig0,go1)
                │
                ▼
         icollis = icollis +1
                │
                ▼
         go1 = .false.
                │
                ▼
         eke = e(np) - RM
         elke = log(eke)
         lelke = eke1(medium)*elke + eke0(medium)
                │
                ▼
             lelec                        "electron"
             .lt.     ──yes──►  ebr1 = ebr11(lelke+iextp,medium)*elke +
              0                           ebr10(lelke+iextp,medium)
                │                              │
               no                              ▼
   "positron"                                eke
                                             .le.      ──yes──►  ebr1 = 0.
   pbr1 = pbr11(lelke,medium)*elke +       ap(med(irl))              │
         pbr10(lelke,medium)                  │  ◄───────────────────┘
                │                            no
                ▼                              ▼
             eke                            call
             .le.     ──yes──► pbr1 = 0.   randomset(rnnow)
          ap(med(irl))  │          │          │
                │  ◄────┘ ◄────────┘          ▼
               no                          rnnow
                ▼                          .lt.      ──yes──►  "Brems"
             call                          ebr1
          randomset(rnnow)                    │
                │                             no
                ▼                              ▼
            rnnow                           e(np)
            .lt.      ──yes──►              .le.      ──no──►
            pbr1                          thmoll(medium)
                │                              │
               no                            yes      "Not Moller"
                ▼                              ▼
   pbr2 = pbr21(lelke,medium)*elke +       ebr1
         pbr20(lelke,medium)               .le.       ──no──►
                │                            0.
                ▼                              │
            rnnow                            yes     "Not Brems either"
            .lt.      ──no──►                 ▼
            pbr2                           go1 = .true.  ──►
                │                              │
               yes                            ▼  "Moller"
                ▼                           iarg = 8
   "Bhabha"   iarg = 10                        │
                │                              ▼
                ▼                          iausfl(iarg+1)
            iausfl(iarg+1) ──yes──►        .ne.       ──yes──►
            .ne.            │               0                    call
             0              ▼                  │                 ausgab(iarg)
                │        call                 no  ◄──────────────┘
               no       ausgab(iarg)          ▼
                │  ◄───────┘               call
                ▼                          moller
             call                             │
             bhabha                           ▼
                │                          iarg =9
                ▼                              │
            iarg =11                          │
                │                              │
                ▼              ▼               ▼             ▼      ▼
                1              2               3             4      5
```

216

1　　　2　　　　　3　　　4　5

yes ← iausfl(iarg+1) .ne. 0 → yes

call ausgab(iarg)

no

call ausgab(iarg)

go1 = .true.

iq(np) .eq. 0 → yes

"Annihilation-in-flight"

no

iarg = 12

go1 = .true.

yes ← iausfl(iarg+1) .ne. 0

call ausgab(iarg)

no

"Brems"

call annih

iarg = 6

uf(np) = 0. ;uf(np+1) = 0.
vf(np) = 0. ;vf(np+1) = 0.
wf(np) = 0.;wf(np+1) = 0

iausfl(iarg+1) .ne. 0 → yes

iarg = 13

no

call ausgab(iarg)

yes ← iausfl(iarg+1) .ne. 0

call brems

call ausgab(iarg)

no

uf(np) = 0. ;uf(np+1) = 0.
vf(np) = 0. ;vf(np+1) = 0.
wf(np) = 0. ;wf(np+1) = 0

"Splitting has been requested"

ibrspl .eq. 1 → no

yes

nbrspl .gt. 1 .and. (np + nbrspl) .ge. MXSTACK

yes

nbrspl = (2*nbrspl + 1)/3
fbrspl = 1./float(nbrspl)

nbrspl .eq. 1 → yes → ibrspl=0

no

no ← (np+nbrspl) .lt. MXSTACK

yes

6　　　　　　　7　8

217

```
                    6
                    │
                    ▼
               ┌─────────┐                    ┌──────────────────────────────────────────┐   7   8
               │ iq(np)  │      yes            │ npstrt = np - 1      ;fdummy = u(np-1)    │   │   │
               │  .eq.   │─────────────────────│ u(np-1) = u(np)      ;u(np) = fdummy      │
               │   0     │                     │ fdummy = v(np-1) ;v(np-1) = v(np)         │
               └─────────┘                     │ v(np) = fdummy      ;fdummy = w(np-1)     │
                    │ no                        │ w(np-1) = w(np)     ;w(np) = fdummy       │
                    ▼                           │ fdummy = e(np-1) ;e(np-1) = e(np)         │
               ┌──────────┐                     │ e(np) = fdummy      ;fdummy = wt(np-1)    │
               │npstrt = np│                    │ wt(np-1) = wt(np)   ;wt(np) = fdummy      │
               └──────────┘                     │ idummy = iq(np-1);iq(np-1) = iq(np)       │
                    │                           │ iq(np) = idummy                           │
                    ▼                           │ idummy = latch(np-1)                      │
        ┌───────────────────────────┐          │ latch(np-1) = latch(np)                   │
        │ wt(np-1) = wt(np-1)*fbrspl │          │ latch(np) = idummy                        │
        │ frstbr = e(np-1)          │          │ fdummy = uf(np-1);uf(np-1) = uf(np)       │
        │ e(np) = e(np) + e(np-1)   │          │ uf(np) = fdummy    ;fdummy = vf(np-1)     │
        └───────────────────────────┘          │ vf(np-1) = vf(np)   ;vf(np) = fdummy      │
                    │                           │ fdummy = wf(np-1);wf(np-1) = wf(np)       │
                    ▼                           │ wf(np) = fdummy                           │
               ┌─────────┐                     │ k1step(np) = k1step(np-1)                 │
               │ iarg = 7 │                     │ k1step(np-1) = 0.                         │
               └─────────┘                     │ k1init(np) = k1init(np-1)                 │
                    │                           │ k1init(np-1) = 0.                         │
         yes        ▼                           │ k1rsd(np) = k1rsd(np-1)                   │
      ┌────────┌─────────────┐                 │ k1rsd(np-1) = 0.                          │
      │        │ iausfl(iarg+1)│                └──────────────────────────────────────────┘
      │        │   .ne.      │ no
      ▼        │    0        │
 ┌─────────┐   └─────────────┘
 │  call   │         │
 │ausgab(iarg)│      │
 └─────────┘         ▼
      └──────────►┌─────────┐
                  │ icsplt = 1│
                  └─────────┘
                       │
                       ▼
              ┌─────────────┐
         yes  │   icsplt    │
      ◄───────│   .ge.      │
              │   nbrspl    │
              └─────────────┘
                    │ no
                    ▼
              ┌──────────────┐
              │ icsplt = icsplt+1│
              └──────────────┘
                    │
                    ▼
               ┌─────────┐
               │ iarg = 6 │
               └─────────┘
                    │
         yes        ▼
      ┌────────┌─────────────┐
      │        │ iausfl(iarg+1)│
      │        │   .ne.      │ no
      ▼        │    0        │
 ┌─────────┐   └─────────────┘
 │  call   │         │
 │ausgab(iarg)│      │
 └─────────┘         ▼
      └──────────►┌─────────┐
                  │  call   │
                  │ brems   │
                  └─────────┘
                       │
                       ▼                        ┌──────────────────────────────────────────┐
               ┌─────────┐   yes                │ fdummy = u(np-1) ;u(np-1) = u(np)         │
               │ iq(np)  │──────────────────────│ u(np) = fdummy      ;fdummy = v(np-1)     │
               │  .eq.   │                       │ v(np-1) = v(np)     ;v(np) = fdummy       │
               │   0     │                       │ fdummy = w(np-1);w(np-1) = w(np)          │
               └─────────┘                       │ w(np) = fdummy      ;fdummy = e(np-1)     │
                    │ no                          │ e(np-1) = e(np)     ;e(np) = fdummy       │
                    ▼                             │ fdummy = wt(np-1);wt(np-1) = wt(np)       │
        ┌───────────────────────────┐            │ wt(np) = fdummy     ;idummy = iq(np-1)    │
        │ wt(np-1) = wt(np-1)*fbrspl │            │ iq(np-1) = iq(np)   ;iq(np) = idummy      │
        │ e(np) = e(np) + e(np-1)   │            │ idummy = latch(np-1)                      │
        └───────────────────────────┘            │ latch(np-1) = latch(np)                   │
                    │                             │ latch(np) = idummy                        │
                    ▼                             │ fdummy = uf(np-1);uf(np-1) = uf(np)       │
               ┌─────────┐                        │ uf(np) = fdummy    ;fdummy = vf(np-1)     │
               │ iarg = 7 │                        │ vf(np-1) = vf(np)   ;vf(np) = fdummy      │
               └─────────┘                        │ fdummy = wf(np-1);wf(np-1) = wf(np)       │
                    │                             │ wf(np) = fdummy                           │
 9      10          ▼                             │ k1step(np) = k1step(np-1)                 │
                    11                            │ k1step(np-1) = 0.                         │
                                                  │ k1init(np) = k1init(np-1)                 │
                                                  │ k1init(np-1) = 0.                         │
                                                  │ k1rsd(np) = k1rsd(np-1)                   │
                                                  │ k1rsd(np-1) = 0.                          │
                                                  └──────────────────────────────────────────┘
                                                                              12   13
```

218

```
e(np) = e(np) - frstbr;fdummy = u(np)
u(np) = u(npstrt)    ;u(npstrt) = fdummy
fdummy = v(np)       ;v(np) = v(npstrt)
v(npstrt) = fdummy   ;fdummy = w(np)
w(np) = w(npstrt)    ;w(npstrt) = fdummy
fdummy = e(np)       ;e(np) = e(npstrt)
e(npstrt) = fdummy   ;fdummy = wt(np)
wt(np) = wt(npstrt)  ;wt(npstrt) =
fdummy
idummy = iq(np)      ;iq(np) = iq(npstrt)
iq(npstrt) = idummy  ;idummy = latch(np)
latch(np) = latch(npstrt)
latch(npstrt) = idummy
fdummy = uf(np)      ;uf(np) = uf(npstrt)
uf(npstrt) = fdummy ;fdummy = vf(np)
vf(np) = vf(npstrt)  ;vf(npstrt) = fdummy
fdummy = wf(np)      ;wf(np) = wf(npstrt)
wf(npstrt) = fdummy
k1step(np) = k1step(npstrt)
k1step(npstrt) = 0.0
k1init(np) = k1init(npstrt)
k1init(npstrt) = 0.0
k1rsd(np) = k1rsd(npstrt)
k1rsd(npstrt) = 0.0
```

iausfl(iarg+1) . ne. 0

yes

call ausgab(iarg)

no

iarg = 7

iausfl(iarg+1) . ne. 0

yes

call ausgab(iarg)

no

iq(np) .eq. 0

yes

no

go1 = .true.

return (to electr)

```
        subroutine
          compt

       icompt = icompt +1

        irloc = ir(np)

    incohr(irloc) .ne. 1        yes
         .and.                    ─────►  stop
    iprofr(irloc) .eq. 1

              no

   eig = e(np)
   egp = eig/RM
   br0i =1. + 2.*egp
   alph1 = log(br0i)
   alph2 = egp*(br0i + 1.)/(br0i*br0i)
   sumalp = alph1 + alph2

        call
        randomset(rnnow)

        alph1
         .ge.           yes                    "1/br distribution"
       sumalp*rnnow     ─────►
                                               call
        no   "br distribution"                 randomset(rnnow)

        call
        randomset(rnnow1)                      br = exp(alph1*rnnow)/br0i

        call
        randomset(rnnow)

          egp
           .ge.
no       (egp + 1.)*rnnow

              yes

        call
        randomset(rnnow)

   rnnow1 = max(rnnow1,rnnow)

   br = ((br0i - 1.)*rnnow1 + 1.)/br0i

   esg = br*eig
   a1mibr = 1. - br                    "Determine seconday photon energy
   esedef = eig*a1mibr + RM             and polar angle from kinematics"
   temp = RM*a1mibr/esg
   sinthe = max(0.D0,temp*(2. - temp))

        call
        randomset(rnnow)

1                      2
```

"Rejection sampling loop"

1

2

"Rejection sampling loop"

```
rejf3 = 1. - br*sinthe/(1. + br*br)
irloc = ir(np)
```
"Calculate rejection function"

incohr(irloc)
.eq.
1
— no
— yes

```
medium = med(irloc)
alamb = 0.012398520
xval = sqrt(temp/2.)*eig/alamb
```

xval .ge. 5.E-3
.and.
xval .le. 80.
— no →
xval
.gt.
80.
— no

yes

xval .gt. 80. — yes

```
xlv = log(xval)
lxlv = sco1(medium)*xlv + sco0(medium)
sxz = sxz1(lxlv,medium)*xlv + sxz0(lxlv,medium)
```

sxz = 1.

sxz = 0.

rejf3 = rejf3*sxz

rnnow
.gt.
rejf3
— yes

no   "Photon energy accepted"

```
sinthe = sqrt(sinthe)
costhe = 1. - temp
irloc = ir(np)
```

iprofr(irloc)
.eq.
1
— no

yes   "Consider Compton profile"

```
esgmax = eig - cpimev
valloc = sqrt(eig*eig + esgmax*esgmax - 2.*eig*esgmax*costhe)
qvalmx = (eig - esgmax - eig*esgmax*(1. - costhe)/RM)*
137./valloc
```

qvalmx
.ge.
100.
— yes → esg = eig/(1. + eig/RM*(1. - costhe))

no

medium = med(irloc)

call
randomset(rnnow)

icprof(medium)
.eq.
1
— yes

no

```
lvallc = cco1(medium)*rnnow + cco0(medium)
cpr = cpr1(lvallc,medium)*rnnow + cpr0(lvallc,medium)
```

3   4   5   6

```
                                3    4              5                                    6


                                                icprof(medium)         no
                                                     .eq.      ──────────────────────────┐
                                                      3                                   │
                                                       │ yes                              │
                                                       ▼                                  │
                                                  call                                    │
                                                  randomset(rnnow1)                       │
                                                       │                                  │
                                                       ▼                                  │
                                          ishell=1                           yes          │
                                      ┌── ishell=ishell+1   ishell>mxshel(medium) ──┐     │
                                      │                          │ no               │     │
                                      │                          ▼                  │     │
                                      │                     rnnow1                  │     │
                                no    │                      .le.                   │     │
                                 └────┤              elecno(ishell,medium)          │     │
                                      │                          │ yes             │     │
                                      │                          ▼◄────────────────┘     │
                                 lvallc = ccos1(medium)*rnnow + ccos0(medium)             │
                                 cpr = cprs1(lvallc,ishell,medium)*rnnow + cprs0(lvallc,ishell,medium)  │
                                                       │◄──────────────────────────────────┘
                                                       ▼
                                 f1 = (cpr/137.)*(cpr/137.)
                                 f2 = (1. - costhe)/RM
                                 f3 = (1. + f2*eig)*(1. + f2*eig) - f1
                                 f4 = (f1*costhe - 1. - f2*eig)
                                 f5 = f4*f4 - f3 + f1*f3
                                 eps1 = 0.0
                                                       │
                             yes                       ▼
               ◄─────────────────────────────        f5
                                                     .lt.
                                                     eps1
                                                       │ no
                                                       ▼
                                 esg1 = (-f4 - sqrt(f5))/f3*eig
                                 esg2 = (-f4 + sqrt(f5))/f3*eig
                                                       │
                                                       ▼
                                                  call
                                                  randomset(rnnow2)
                                                       │
                                                       ▼
                                                  rnnow2        yes
                                                   .lt.    ──────────── esg = esg1
                                                   0.5                       │
                                                       │ no                  │
                                                       ▼                     │
                                                 esg = esg2 ◄────────────────┘
                                                       │
                                                       ▼
                                              icprof(medium)    yes
                                                   .eq.    ──────── esgmax = eig - capio(ishell,medium)
                                                    3                          │
                                                       │ no ◄─────────────────┘
                                                       ▼
                    yes                         esg .gt. esgmax      no        call              esg         no
            ◄─────────────────────             .or.             ──────────    randomset(rnnow3) ──── .lt.  ──────┐
                                                esg .lt. 0.                                      esgmax*rnnow3    │
                                                                                                     │ yes       │
                                                                                                     │◄──────────┤
                                   │◄────────────────────────────────────────────────────────────────┘          │
                                   ▼                   │                                                         ▼
                                   3                   7                                                         6
```

222

223

```
         ┌─────────────┐
         │ subroutine  │
         │ counters_out│
         └─────────────┘
                │
                ▼
            ╱ioflag╲
           ╱ .eq.   ╲
           ╲   0    ╱
            ╲      ╱
```

OPEN(UNIT=99,FILE='egs5job.out99',   yes
STATUS='UNKNOWN')

no

```
            ╱ioflag╲          no
           ╱ .eq.   ╲ ───────────────►
           ╲   1    ╱
            ╲      ╱
                │
               yes
```

write(6,*) ' *** Error using
subroutine counters_out'
write(6,*) '   ioflag=',ioflag
CLOSE(UNIT=99)
write(6,*) '   Program stopped'

```
       ( stop )
```

```
iannih   = 0
iaphi    = 0
ibhabha = 0
ibrems   = 0
icollis  = 0
icompt   = 0
iedgbin  = 0
ieii     = 0
ielectr  = 0
ihatch   = 0
ihardx   = 0
ikauger  = 0
ikshell  = 0
ikxray   = 0
ilauger  = 0
ilshell  = 0
ilxray   = 0
imoller  = 0
imscat   = 0
ipair    = 0
iphoto   = 0
iphoton  = 0
iraylei  = 0
ishower = 0
iuphi    = 0
itmxs    = 0
noscat  = 0
iblock  = 0
```

write(99,*) 'Values for subroutine-entry counters:'
write(99,*)
write(99,*) 'iannih =',iannih
write(99,*) 'iaphi =',iaphi
write(99,*) 'ibhabha=',ibhabha
write(99,*) 'ibrems=',ibrems
write(99,*) 'icollis =',icollis
write(99,*) 'icompt =',icompt
write(99,*) 'iedgbin=',iedgbin
write(99,*) 'ieii   =',ieii
write(99,*) 'ielectr=',ielectr
write(99,*) 'ihatch =',ihatch
write(99,*) 'ihardx =',ihardx
write(99,*) 'ikauger=',ikauger
write(99,*) 'ikshell=',ikshell
write(99,*) 'ikxray =',ikxray
write(99,*) 'ilauger=',ilauger
write(99,*) 'ilshell=',ilshell
write(99,*) 'ilxray =',ilxray
write(99,*) 'imoller=',imoller
write(99,*) 'imscat =',imscat
write(99,*) 'ipair =',ipair
write(99,*) 'iphoto =',iphoto
write(99,*) 'iphoton=',iphoton
write(99,*) 'iraylei=',iraylei
write(99,*) 'ishower=',ishower
write(99,*) 'iuphi =',iuphi
write(99,*) 'itmxs =',itmxs
write(99,*) 'noscat =',noscat
write(99,*) 'iblock =',iblock
CLOSE(UNIT=99)

```
       ┌──────────┐
       │  return  │
       └──────────┘
```

224

```
                    ⬡ subroutine
                      edgbin ⬡
                           │
                ┌──────────────────┐
                │ iedgbin = iedgbin +1 │
                └──────────────────┘
                           │
        ┌──────────────────┬──────────────┐      yes
        │ medium=1         │              │───────────────────────►
  ─────►│ medium=medium+1  │ medium>nmed  │
        └──────────────────┴──────────────┘
                           │ no
                           │
                ┌──────────────────┐
                │ ner = nepm(medium) │
                └──────────────────┘
                           │
                        ◇ ner ◇    yes
                        ◇ .gt. ◇──────────► ( stop )
                        ◇  20  ◇
                           │ no
                ┌──────────────────┐
                │ nedgb(medium) = 0 │
                └──────────────────┘
                           │
        ┌──────────────┬──────────────┐      yes
        │ izn=1        │              │───────────────────────►
  ─────►│ izn=izn+1    │  izn>ner     │
        └──────────────┴──────────────┘
                           │ no
                ┌───────────────────────┐
                │ iz1 = zelem(medium,izn) │
                └───────────────────────┘
                           │
        ┌──────────────┬──────────────┐      yes
        │ ikl=1        │              │───────────────────────►
  ─────►│ ikl=ikl+1    │   ikl>4      │
        └──────────────┴──────────────┘
                           │ no
                ┌───────────────────────────┐
                │ eee = eedge(ikl,iz1)/1000.0 │
                └───────────────────────────┘
                           │
          no            ◇ eee ◇
  ◄────────────────────◇ .gt. ◇
                        ◇ ap(medium) ◇
                           │ yes
        ┌─────────────────────────────────────────┐
        │ nedgb(medium) = nedgb(medium) + 1        │
        │ eig = log(eee)                            │
        │ ledgb(nedgb(medium),medium) =             │
        │    ge1(medium)*eig +ge0(medium)           │
        │ edgb(nedgb(medium),medium) = eee          │
        └─────────────────────────────────────────┘
                           │
                        ◇ nedgb(medium) ◇   no
                        ◇ .gt. ◇──────────────────►
                        ◇  0  ◇
                           │ yes
        ┌──────────────┬──────────────────┐   yes
        │ ii=1         │                  │──────────────►
  ─────►│ ii=ii+1      │ ii>nedgb(medium)  │
        └──────────────┴──────────────────┘
                           │ no
        ┌──────────────┬──────────────────┐   yes
        │ jj=1         │                  │──────────────►
  ─────►│ jj=jj+1      │ jj>nedgb(medium)  │
        └──────────────┴──────────────────┘
                           │ no
```

1   2   3   4   5   6   7

1   2   3   4   5   6   7

ii
.ne.
jj

no

yes

ledgb(ii,medium)
.eq.
ledgb(jj,medium)

no

yes

write(6,102) medium

return
(to hatch)

226

```
                    subroutine
                       eii
                         │
                         ▼
                  ieii = ieii +1
                         │
                         ▼
         ┌───────────────────────────────────┐
         │ nxray = 0                          │
         │ nauger = 0                         │
         │ capbind = eedge(1,iz)*0.001        │
         │ ese1 = e(np-1)                     │
         │ ese2 = e(np)                       │
         │ esesum = e(np) + e(np-1) - 2.*RM   │
         └───────────────────────────────────┘
                         │
                         ▼
┌──────────────────────────────┐  no    ╱ ese1 - RM .gt. capbind ╲
│ ratio = 1. - capbind/esesum  │◀───────   .or.
│ e(np-1) = (ese1 - rm)*ratio  │        ╲ ese2 - RM .gt. capbind ╱
│           + RM               │                   │ yes
│ e(np) = (ese2 - rm)*ratio    │                   ▼
│         + RM                 │              call
└──────────────────────────────┘           randomset(rnnow)
                         │                         │
                         │                         ▼
                         │            ╱ rnnow ╲  yes  ┌──────────────────────────┐
                         │            ╲ .gt.  ╱─────▶ │ e(np-1) = ese1 - capbind │
                         │              ╲0.5╱         │ e(np) = ese2             │
                         │                │ no        └──────────────────────────┘
                         │                ▼
                         │      ┌──────────────────────┐
                         │      │ e(np-1) = ese1        │
                         │      │ e(np) = ese2 - capbind│
                         │      └──────────────────────┘
                         │                │
                         │                ▼
                         │   no   ╱ e(np-1) .le. RM ╲
                         │ ◀──────   .or.
                         │        ╲ e(np) .le. RM  ╱
                         │                │ yes
                         │                ▼
                         │         ╱ rnnow ╲  yes  ┌──────────────────────────┐
                         │         ╲ .le.  ╱─────▶ │ e(np-1) = ese1 - capbind │
                         │           ╲0.5╱         │ e(np) = ese2             │
                         │             │ no        └──────────────────────────┘
                         │             ▼
                         │   ┌──────────────────────┐
                         │   │ e(np-1) = ese1        │
                         │   │ e(np) = ese2 - capbind│
                         │   └──────────────────────┘
                         │             │
                         └─────────────┤
                                       ▼
                              ╱ ieispl ╲   no
                              ╲  .eq.   ╱──────────────────┐
                                ╲  1  ╱                    │
                                  │ yes                    ▼
                                  ▼                ┌──────────────┐
                    ╱ neispl .gt.1      ╲   no      │ neispl = 1   │
                    ╲   .and.            ╱───┐      │ feispl = 1.  │
              ╱ (np + neispl) .ge. MXSTACK ╲  │     └──────────────┘
                                  │ yes     │            │
                                  ▼         ▼            ▼
                                  1         2            3
```

```
        1                    2           3
        │                    │           │
        │                    │           │
┌───────┴──────────────┐     │           │
│ neispl = (2*neispl + 1)/3  │     │           │
│ feispl = 1./float(neispl)  │     │           │
└───────┬──────────────┘     │           │
        │                    │           │
        ▼                    │           │
      ╱─────╲    yes  ┌──────────────┐   │           │
     ╱ neispl ╲──────▶│  ieispl = 0  │   │           │
     ╲ .eq.   ╱       └──────┬───────┘   │           │
      ╲  1  ╱                │           │           │
        │ no ◀───────────────┘           │           │
        ▼                                │           │
   no ╱──────────╲                       │           │
◀────╱ np + neispl ╲                     │           │
     ╲    .lt.     ╱                     │           │
      ╲ MXSTACK  ╱                       │           │
        │ yes ◀──────────────────────────┴───────────┘
        ▼
   ┌─────────┐
   │  call   │
   │  kshell │
   └────┬────┘
        ▼
  ╱──────────────────╲      yes   ┌──────────────────┐
 ╱  nxray .ge.1        ╲─────────▶│ enew = exray(1)  │
 ╲   .and.            ╱          │ ieie = 1         │
  ╲ exray(1) .gt. eedge(2,iz)*0.001          └────────┬─────────┘
        │ no                                          │
        ▼                                             │
   ┌────────────┐                                     │
   │ enew = 0.0 │                                     │
   │ ieie = 0   │                                     │
   └─────┬──────┘◀──────────────────────────────────┘
         ▼
  ┌─────────────────────┐
  │ edep = capbind - enew│
  │ etmp = e(np)        │
  │ e(np)= edep         │
  │ iqtmp = iq(np)      │
  │ iq(np) = -1         │
  └──────────┬──────────┘
             ▼
        ┌─────────┐
        │ iarg=4  │
        └────┬────┘
             ▼
    ╱──────────────╲    yes   ┌──────────────┐
   ╱ iausfl(iarg+1) ╲────────▶│ call         │
   ╲   . ne.        ╱        │ ausgab(iarg) │
    ╲    0        ╱          └──────┬───────┘
          │ no ◀────────────────────┘
          ▼
   ┌──────────────┐
   │ e(np) = etmp │
   │ iq(np) = iqtmp│
   └──────┬───────┘
          ▼
     ╱─────╲      no
    ╱ ieie  ╲──────────────────────────────▶
    ╲ .eq.  ╱
     ╲  1  ╱
       │ yes
       ▼
  ┌─────────────────────────┐
  │ np = np + 1             │
  │ e(np) = enew           │
  │ iq(np) = 0             │
  │ wt(np-1) = wt(np-1)*feispl│
  └───────────┬─────────────┘
              ▼
              4                              5
```

4

5

```
jnp=1          jnp>neispl    yes
jnp=jnp+1
                  no
```

```
iq(np-1+jnp) = iq(np)
e(np-1+jnp) = e(np)
```

call
randomset(rnnow)

```
costhe = 2.*rnnow - 1.
sinthe = sqrt(1. - costhe*costhe)
u(np) = 0.
v(np) = 0.
w(np) = 1.
```

call
uphi(2,1)

```
u(np-1+jnp) = u(np)
v(np-1+jnp) = v(np)
w(np-1+jnp) = w(np)
x(np-1+jnp) = x(np-1)
y(np-1+jnp) = y(np-1)
z(np-1+jnp) = z(np-1)
ir(np-1+jnp) = ir(np-1)
wt(np-1+jnp) = wt(np-1)
dnear(np-1+jnp) = dnear(np-1)
latch(np-1+jnp) = latch(np-1)
k1step(np-1+jnp) = 0.
k1init(np-1+jnp) = 0.
k1rsd(np-1+jnp) = 0.
```

```
wt(np-1) = wt(np-1)/feispl
np = np + neispl - 1
```

return
(to moller)

229

```
         ┌─────────────────┐
         │   subroutine    │
         │  electr(ircode) │
         └─────────────────┘
                 │
                 ▼
       ┌──────────────────┐      ┌──────────────────────┐        ◇ ircode          yes
       │ ielectr = ielectr+1│ ──→ │ deresid = 0.d0       │ ──→   ircode .eq. -1  ──────→
       └──────────────────┘      │ deinitial = 0.d0     │
                                 │ denstep = 0.d0       │        no
                                 │ hardstep = 0.d0      │
                                 │ dok1s0 = 0           │
                                 └──────────────────────┘
```

```
                                                    ek1s1(1,1).ne.0.d0     no
                                                         .or.
                                                    ek1s0(1,1).ne.0.d0
                                                         yes
                                                   ┌──────────────┐
                                                   │ dok1s0 = 1   │
                                                   │ nk1i = 0     │
                                                   └──────────────┘
```

```
        ┌──────────────┐
        │ ircode = 1   │
        │ irold = ir(np)│
        │ irl = ir(np) │
        └──────────────┘

"New electron loop"  ──────( 1 )

        ┌──────────────┐
        │ lelec = iq(np)│
        │ eie = e(np)  │
        │ medium = med(irl)│
        └──────────────┘
              │
              ▼
          ◇ medium .eq. 0      yes    ┌──────────────┐    "go to 6"
                                 ──→   │ tstep = vacdst│ ──────→
              no                       └──────────────┘

"Tstep loop"  ──────( 2 )

          e(np).le.ecut(irl)     yes              "go to 14"
              .and.          ──────────────────────→
          deresid.eq.0.d0

              no
              ▼
          ◇ hardstep .eq. 0.d0        no
                                ←──────────

              yes
              ▼
        ┌──────────────┐
        │ call         │
        │ randomset(rnnow)│
        └──────────────┘
              │
              ▼
        ┌────────────────────────────────┐
        │ hardstep = max(-log(rnnow),EPSEMFP)│
        └────────────────────────────────┘

"Ustep loop"  ──────( 3 )

        ┌────────────────────────┐
        │ rhof=rhor(irl)/rho(medium)│
        └────────────────────────┘
              │
              ▼
        ┌────────────────────────────────────┐
        │ eke = eie - RM                     │
        │ elke = log(eke)                    │
        │ lelke = eke1(medium)*elke + eke0(medium)│
        └────────────────────────────────────┘
              │
              ▼
              4
```

1 2 3   5 6

230

```
dedx0 = pdedx1(lelke,medium)*elke
    + pdedx0(lelke,medium)
scpow0 = pscpw1(lelke,medium)*elke
    + pscpw0(lelke,medium)
range0 = prang1(lelke,medium)*elke
    + prang0(lelke,medium)
range0 = range0 - pctrng(irl)
```

lelec
.eq.
-1

no

yes

```
dedx0 = ededx1(lelke,medium)*elke
    + ededx0(lelke,medium)
scpow0 = escpw1(lelke,medium)*elke
    + escpw0(lelke,medium)
range0 = erang1(lelke,medium)*elke
    + erang0(lelke,medium)
range0 = range0 - ectrng(irl)
```

```
dedx = rhof*dedx0
scpow = rhof*scpow0
range = rhof*range0 + ENEPS/dedx
```

useGSD(medium)
.eq.
0

no

"get Moliere
parameters"

yes

```
ems = e(np)
bms = (ems - RM)* (ems + RM)/ems**2
ams = blcc(medium)/bms
gms = rhof*(xcc(medium)/(ems*bms))**2
tmxs = 1.d0 / (log(ams/gms) * gms)
```

```
call hardx
(lelec,eke,lelke,elke,sig0)
```

sig = rhof*sig0

sig
.le.
0.

yes

thard = vacdst

no

thard = hardstep / sig

"set energy hinge distance"

denstep
.eq.
0.d0

no

yes

```
denstep = deresid
estepe = estep1(lelke,medium)*elke + estep0(lelke,medium)
detot = eke * estepe
```

estepr(irl)
.ne.
0

no

yes

detot = detot * estepr(irl)

```
                                          10                          11        12 13

                            yes    ┌─────────────┐
                        ┌──────────│ (e(np)-detot)│
                        │          │    .lt.      │
              ┌─────────────────┐  │  ecut(irl)   │
              │detot = e(np)-ecut(irl)│ └──────┬──────┘
              └─────────────────┘         │ no
                        │                 │
                        │          ┌──────────────┐
                        └──────────│    call      │
                                   │randomset(rnnow)│
                                   └──────┬───────┘
                                          │
                              ┌──────────────────────────┐
                              │deinitial = rnnow * detot  │
                              │deresid = detot - deinitial │
                              │denstep = denstep + deinitial│
                              └────────────┬─────────────┘
                                           │◄──────────────────┘
                              yes    ┌──────────────┐
         ┌──────────────┐◄──────────│    dedx      │
         │tinel = vacdst │           │    .le.      │
         │range = vacdst │           │     0.       │
         └───────┬──────┘           └──────┬───────┘
                 │                         │ no
                 │                  ┌──────────────────┐
                 │                  │tinel = denstep / dedx│
                 │                  └────────┬─────────┘
                 │            ( 4 )          │
                 └──────────────►│◄──────────┘
                            no    ┌──────────────┐
              ┌──────────────────│  k1step(np)  │
              │                   │    .eq.      │
              │                   │    0.d0      │
              │                   └──────┬───────┘
              │                          │ yes
              │                ┌──────────────────────┐
              │                │k1step(np) = k1rsd(np) │
              │                └───────────┬──────────┘
              │                            │
              │          no          ┌──────────────┐
              │     ┌───────────────│    lelec     │
              │     │                │    .eq.      │
              │     │                │     -1       │
              │  ┌─────────────────────────────┐  │ yes
              │  │kinit0 = pkini1(lelke,medium)*elke│
              │  │    + pkini0(lelke,medium)    │
              │  └──────────────┬──────────────┘
              │                 │        ┌──────────────────────────────┐
              │                 │        │kinit0 = ekini1(lelke,medium)*elke│
              │                 │        │    + ekini0(lelke,medium)    │
              │                 │        └───────────┬──────────────────┘
              │                 │                    │
              │                 └──────►│◄───────────┘
              │          no          ┌──────────────┐
              │     ┌───────────────│  k1Lscl(irl) │
              │     │                │    .ne.      │
              │     │                │    0.d0      │
              │     │                └──────┬───────┘
              │     │                       │ yes
              │     │  ┌──────────────────────────────────────────────┐
              │     │  │kinit0 = kinit0 * (k1Lscl(irl) + k1Hscl(irl) * elke)│
              │     │  └───────────────────┬──────────────────────────┘
              │     └──────────────────►│◄─┘
              │                           │
              │        ┌──────────┐  yes  ┌──────────┐   no
              │        │  dok1s0  │──────│  lelec   │─────────────────┐
              │        │   .eq.   │       │   .eq.   │                 │
              │        │    1     │       │    -1    │                 │
              │        └────┬─────┘       └────┬─────┘                 │
              │             │ no               │ yes                   │
              │             │        ┌──────────────────────────┐ ┌──────────────────────────┐
              │             │        │k1s0 = ek1s1(lelke,medium)*elke│ │k1s0 = pk1s1(lelke,medium)*elke│
              │             │        │   + ek1s0(lelke,medium)  │ │   + pk1s0(lelke,medium)  │
              │             │        └──────────────────────────┘ └──────────────────────────┘
```

```
                                    ┌──────────────────────┐
                                    │ k1s0 = k1s0*(2**nk1i) │
                                    │ nk1i = nk1i + 1       │
                                    └──────────────────────┘
                                               │
                      no           ╱ kinit0 ╲          yes
        ┌─────────┐  ◄────────────◄   .gt.   ►────────────►  ┌──────────────┐
        │ dok1s0=0 │                ╲  k1s0 ╱                │ kinit0 = k1s0 │
        └─────────┘                  ╲    ╱                  └──────────────┘
             │                                                      │
             └──────────────────────────┬──────────────────────────┘
                                        │
                               ┌─────────────────────┐
                               │ ktotal = rhof*kinit0 │
                               └─────────────────────┘
                                        │
                              ╱ useGSD(medium) ╲    no
                             ◄   .eq.           ►──────────────┐
  ┌───────────────┐          ╲    0           ╱                │
  │ itmxs = itmxs + 1 │◄──────   ╲    ╱      yes                │
  └───────────────┘  │            │                            │
         │           │            ▼                            │
         │      ╱ tmxset ╲  false   ╱ ktotal/scpow ╲  yes       │
         │     ◄          ►────────◄     .gt.        ►──────────┘
         │      ╲        ╱          ╲    tmxs       ╱
         │  true ╲      ╱            ╲    ╱                      │
         ▼        ▼                   no │                      │
  ┌──────────────────┐         ┌─────────────────────────┐     │
  │ ktotal = tmxs*scpow │      │ omega0 = ams * ktotal/scpow │  │
  └──────────────────┘         └─────────────────────────┘     │
         │                                  │                   │
         │         no          ╱ omega0 ╲                       │
         │     ◄──────────────◄   .lt.   ►                      │
         │                     ╲  2.80  ╱                       │
         │                      ╲    ╱                          │
         │                   yes │                              │
         │                ┌────────────────────────┐           │
         │                │ ktotal = scpow * 2.80 / ams │        │
         │                └────────────────────────┘           │
         └──────────────────────┬───────────────────┬──────────┘
                                │
                         ┌──────────────┐
                         │ call         │
                         │ randomset(rnnow) │
                         └──────────────┘
                                │
                  ┌───────────────────────────────────┐
                  │ k1init(np) = rnnow * ktotal        │
                  │ k1rsd(np) = ktotal - k1init(np)    │
                  │ k1step(np) = k1step(np) + k1init(np) │
                  └───────────────────────────────────┘
                                │
                    yes    ╱ scpow ╲
  ┌──────────────┐◄───────◄  .le.   ►
  │ tmscat = vacdst │      ╲   0.  ╱
  └──────────────┘         ╲    ╱
         │                  no │
         │          ┌─────────────────────────┐
         │          │ tmscat = k1step(np) / scpow │
         │          └─────────────────────────┘
         └────────────────┬──────┘
                         (5)
                          │
                ┌──────────────────────────┐
                │ tstep = MIN(tmscat,tinel,thard) │
                └──────────────────────────┘
                          │
                         (6)
                          │
                ┌──────────────────┐
                │ irnew = ir(np)    │
                │ idisc = 0         │
                │ ustep0 = tstep    │
                │ ustep=ustep0      │
                │ edep=0.d0         │
                │ medold = medium   │
                └──────────────────┘
```

23 24 25 26 27 28   29   30

ustep .gt. dnear(np) —yes→ call howfar

no

range .lt. dnear(np) .and. e(np) .le. esave(irl) .and. medium .ne. 0 —no→

yes   "user-range-discard"

lelec .eq. -1 —no→ idisc = 99

yes

idisc = 1

idisc .gt. 0 —yes→ "imedeate discard: go to 16"

"chck negative USTEP"   no

ustep .lt. 0. —yes→ ustep .lt. -1.D-9 —yes→ ierust=ierust+1

no   no

ustep=o.do

write(6,102) ierust,ustep,ir(np), irnew, irold,x(np), y(np),z(np), sqrt(x(np)**2 + y(np)**2)

ustep .ne. 0. —no→

yes

medium .ne. 0 —no→

yes

edep = ustep * dedx

ierust .gt. 1000 —no→

write(6,103)   yes

stop

iarg=0
e(np) = e(np) - deinitial + denstep

iausfl(iarg+1) . ne. 0 —yes→ call ausgab(iarg)

no

e(np) = e(np) + deinitial - denstep
x(np) = x(np) + u(np)*ustep
y(np) = y(np) + v(np)*ustep
z(np) = z(np) + w(np)*ustep

31 32 33 34 35 36   37   38   39 40

234

dnear(np) = dnear(np) - ustep
irold = ir(np)

medium
.ne.
0
— no

— yes

thard = thard - ustep
tmscat = tmscat - ustep
tinel = tinel - ustep
hardstep = thard * sig
k1step(np) = tmscat * scpow
denstep = tinel * dedx

ustep.ne.ustep0
.or.
ustep.eq.0.D0
— no

— yes

ir(np) = irnew
irl = irnew
medium = med(irl)

ustep
.ne.
0.d0
— no

— yes

iarg=5
e(np) = e(np) - deinitial + denstep

iausfl(iarg+1)
. ne.
0
— yes → call ausgab(iarg)

no

e(np) = e(np) + deinitial - denstep

medium
.eq.
0
— yes → tstep = vacdst

— no

irnew
.ne.
irold
— no

— yes

41 42 43 44 45 46 47 48 49

ecut(irnew)
.gt.
ecut(irold)
— no →

yes

ecsda = eie - deinitial + denstep

ecsda
.le.
ecut(irnew)
— no →

yes

e(np) = ecsda
deinitial = 0.d0
denstep = 0.d0
deresid = 0.d0

"go to 14" →

ecsda - denstep
.le.
ecut(irnew)
— no →

yes

eie = ecsda
e(np) = eie
detot = e(np) - ecut(irnew)

call
randomset(rnnow)

"go to 3" ←

deinitial = rnnow * detot
deresid = detot - deinitial
denstep = deinitial

eie - (deinitial + deresid)
.le.
ecut(irnew)
— no →

yes

"go to 3" ←

deresid = eie - deinitial - ecut(irnew)

"go to 5" ←
no

medium.ne.medold
.or.
rhor(irnew).ne.rhom(medium)

yes

"go to 2" ←
yes

denstep
.eq.
0.d0

"go to 3" ←

50 51 52 53 54 55 56 57

236

50 51 52 53 54

55

56 57

tinel
.eq.
0.

no

yes   "energy loss hinge"

de = deinitial + deresid

ekef = eke - de
eold = eie
enew = eold - de
edep = de

iarg=27

iausfl(iarg+1)
. ne.
0

yes

call
ausgab(iarg)

no

de = edep
eie = eie - de
e(np) = eie

iarg=28

iausfl(iarg+1)
. ne.
0

yes

call
ausgab(iarg)

no

"go to 3"

no

abs(eie - ecut(irl))
.lt.
1.d-14

yes

eie = ecut(irl)
e(np) = eie

denstep = deresid
deinitial = 0.d0
deresid = 0.d0

no

deresid
.eq.
0.d0

yes   "go to 14"

58 59   60 61

62

63 64

237

```
                                          62
                                           │
                                           ▼
                                    ╱╲
                                   ╱  ╲     no
                                  ╱tmscat╲──────────────────────────────────┐
                                  ╲ .eq.  ╱                                  │
                                   ╲ 0.0 ╱                                   │
                                    ╲╱                                       │
                                 yes │   "multiple scattering hinge"         │
                                     ▼                                       │
                        ┌──────────────────────────────────┐                │
                        │ tvstep = (k1init(np) + k1rsd(np)) / scpow │        │
                        └──────────────────────────────────┘                │
                                     │                                       │
                                     ▼                                       │
                                   ╱╲                                        │
                                  ╱  ╲   no                                  │
                                 ╱useGSD(medium)╲──────────────────────────┐ │
                                 ╲ .eq. ╱                                  │ │
                                  ╲ 0  ╱                                   │ │
                                   ╲╱                                      │ │
                                yes │                                      │ │
                                    ▼                                      │ │
                        ┌──────────────────────────┐                      │ │
                        │ omega0 = ams*tvstep*rhof │                      │ │
                        └──────────────────────────┘                      │ │
                                    │                                      │ │
                                    ▼                                      │ │
                                  ╱╲        yes    ┌──────────────┐        │ │
                                 ╱omega0╲─────────▶│ iskpms = 1   │────────┤ │
                                 ╲ .le. ╱          └──────────────┘        │ │
                                 ╲2.718282╱                                │ │
                                   ╲╱                                      │ │
                                 no │                                      │ │
                                    ▼                                      │ │
                        ┌──────────────────────┐                          │ │
                        │ iskpms = 0           │                          │ │
                        │ blc = log(omega0)    │                          │ │
                        │ blcold = blc         │                          │ │
                        └──────────────────────┘                          │ │
                                    │                                      │ │
      ┌──────────────────┐   yes    ▼                                     │ │
      │ b = -10.27666 +  │◀───────╱╲                                      │ │
      │ blc* (17.82596 - │        ╱blc ╲                                   │ │
      │ 6.468813*blc)    │        ╲.lt. ╱                                  │ │
      └──────────────────┘        ╲1.306853╱                              │ │
              │                     ╲╱                                     │ │
              │                   no │                                     │ │
              │                      ▼                                     │ │
              │          ┌──────────────────────────┐                     │ │
              │          │ ib = b0bgb + blc*b1bgb   │                     │ │
              │          └──────────────────────────┘                     │ │
              │                      │                                     │ │
              │                      ▼                                     │ │
              │   ┌──────────────┐ yes ╱╲                                 │ │
              │   │write(6,101)ib│◀───╱ib ╲                                │ │
              │   └──────────────┘    ╲.gt.╱                               │ │
              │         │             ╲nbgb╱                               │ │
              │         ▼              ╲╱                                  │ │
              │   ┌──────────┐       no │                                  │ │
              │   │ ib = nbgb│──────────┤                                  │ │
              │   └──────────┘          ▼                                  │ │
              │          ┌────────────────────────────────────────┐       │ │
              │          │ b = bgb0(ib) + blc*(bgb1(ib) + blc*bgb2(ib)) │  │ │
              │          └────────────────────────────────────────┘       │ │
              └────────────────────────┤◀─────────────────────────────────┘ │
                                       │◀───────────────────────────────────┘
                                       ▼
                                 ┌──────────┐
                                 │ iarg=29  │
                                 └──────────┘
                                       │
                                       ▼
                                    ╱╲          yes   ┌──────────────┐
                                   ╱iausfl(iarg+1)╲───▶│ call         │
                                   ╲ . ne. ╱           │ ausgab(iarg) │
                                    ╲ 0  ╱             └──────────────┘
                                     ╲╱                       │
                                  no │◀───────────────────────┘
                                     ▼
                                 ┌──────────┐
                                 │ call     │
                                 │ mscat    │
                                 └──────────┘
                                      │
                                      ▼
                                 ┌──────────┐
                                 │ iarg=30  │
                                 └──────────┘
                                      │
                                      ▼
```

65 66   67 68   69   70   71   72

```
                                  iausfl(iarg+1)      yes      call
                                  . ne.                        ausgab(iarg)
                                  0

                                       no

                    "go to 4"              call
                                           uphi(2,1)


     "go to 5"            no         thard
                                     .eq.
                                     0.

               "hard collision"     yes

                              e(np) = e(np) - deinitial + denstep
                              deresid = 0.d0
                              deinitial = 0.d0
                              denstep = 0.d0


                                   call collis
                                   (lelec,irl,sig0,go1)


          "go to 1"      no     iq(np)     yes      return
                                .eq.                (to shower)
                go1             0

            true

          "go to 2"                false

                              eie = e(np)


                                      (14)

                                 no      eie
                      idr = 2            .gt.
                                         ae(medium)

                                         yes

                                       idr = 1

                                  edep = e(np) - RM


                                   edep       yes      iarg = idr
                                   .ne.
                                   0

                                         no      iausfl(iarg+1)    yes     call
                                                 . ne.                     ausgab(iarg)
                                                 0

                                                       no

                                      (15)


                          no      lelec
                                  .eq.
                                  1

                                        yes

     73            74                   75                               76
```

73   74   75   76

```
              edep
      no      .lt.
   <------    e(np)
              yes      "positron anihiration
                        at rest"

              call
              randomset(rnnow)

              costhe = rnnow

              call
              randomset(rnnow)

              rnnow        yes
              .le.      --------->  costhe = -costhe
              0.5

              no

              sinthe = sqrt(1. - costhe**2)
              e(np) = RM
              iq(np) = 0
              u(np) = 0.
              v(np) = 0.
              w(np) = 1.

              call
              uphi(2,1)
                                    "first photon"

              uf(np) = 0.        ;uf(np+1) = 0.
              vf(np) = 0.        ;vf(np+1) = 0.
              wf(np) = 0.        ;wf(np+1) = 0.
              hardstep = 0.   ;denstep = 0.
              deinitial = 0.    ;deresid = 0.
              k1step(np) = 0.:k1init(np) = 0.
              k1rsd(np) = 0.
                                    "second photon"

              np = np + 1       ;e(np) = RM
              iq(np) = 0         ;x(np) = x(np-1)
              y(np) = y(np-1) ;z(np) = z(np-1)
              ir(np) = ir(np-1) ;wt(np) = wt(np-1)
              dnear(np) = dnear(np-1)
              latch(np) = latch(np-1)
              u(np) = -u(np-1);v(np) = -v(np-1)
              w(np) = -w(np-1)
              hardstep = 0.    ;denstep = 0.
              deinitial = 0.    ;k1step(np) = 0.
              k1init(np) = 0.   ;k1rsd(np) = 0.

              iarg = 14

   np = np - 1
   ircode = 2

              iausfl(iarg+1)  yes    call
              . ne.        -------->  ausgab(iarg)
              0

              no
```

77   78   79

240

return
(to shower)

16

"Ecut discard"

e(np) = e(np) - deinitial + denstep
deinitial = 0.d0
denstep = 0.d0
deresid = 0.d0

idisc = abs(idisc)

lelec .eq. -1
.or.
idisc .eq. 99

no → edep = eie + RM

yes

edep = eie - RM

iarg = 3

iausfl(iarg+1)
. ne.
0

yes → call
ausgab(iarg)

no

"go to 15"   yes   idisc
.eq.
99

no

np = np - 1
ircode = 2

return
(to shower)

241

```
         ┌─────────────────────────┐
         │      subroutine         │
         │        hardx            │
         │(charge,kEnergy,keIndex,keFraction,sig0)│
         └─────────────────────────┘
                     │
         ┌─────────────────────────┐
         │    ihardx = ihardx +1   │
         └─────────────────────────┘
                     │
```

        no   charge
  ◄───────────   .lt.
                    0
                     │ yes

┌──────────────────────────────────────┐
│ sig0 = psig1(keIndex,medium)*keFraction + │
│   psig0(keIndex,medium)   │
└──────────────────────────────────────┘

        iextp = 0

                  kEnergy   no
                   .lt.   ─────────►
              (thmoll(medium)-RM)*1.5
                     │ yes

┌──┐
│ mollerThresh = thmoll(medium) - RM   │
│ logMollerThresh = log(mollerThresh)   │
│ mollerIndex = eke1(medium)*logMollerThresh + eke0(medium) │
└──┘

                mollerIndex   no
                   .eq.   ─────────►
                  keIndex
                     │ yes

              thmoll(medium)-RM
                   .le.   no   ┌──────────┐
                  kEnergy   ─────► │ iextp = -1│
                     │ yes   └──────────┘

              iextp = 1

┌──┐
│ sig0 = esig1(keIndex+iextp,medium)*keFraction +   │
│   esig0(keIndex+iextp,medium)   │
└──┘

         ┌─────────────────────────┐
         │   return   │
         │   (to electr)   │
         └─────────────────────────┘

```
        subroutine
          hatch
```

ihatch = ihatch +1

iblock .ne. 1 — yes → write(6,5000) → stop

no

i1st .ne. 0 — no →

yes

i1st=0

.not. rluxset — true → write(6,*) 'RNG ranlux not initialized: doing so in HATCH' → call rluxgo

false

latchi = 0.0

```
nisub = mxsinc_loc - 2
fnsss = nsinss
wid = PI5D2/float(nisub)
wss = wid/(fnsss - 1.0)
zeros(1) = 0.
zeros(2) = PI
zeros(3) = TWOPI
```

isub=1
isub=isub+1   isub >mxsinc_loc — yes →

no

```
sx = 0.
sy = 0.
sxx = 0.
sxy = 0.
xs0 = wid*float(isub - 2)
xs1 = xs0 + wid
iz = 0
```

izz=1
izz=izz+1   izz >3 — yes →

no

xs0 .le. zeros(izz) .and. zeros(izz) .le. xs1 — no →

yes

1   2   3   4   5

```
                    1    2                     3              4                          5

                                             ┌──────────┐
                                             │ iz = izz │
                                             └──────────┘
                                                   │◄────────────────┘
                                                   ▼
                                              ╱ iz  ╲      yes    ┌──────────┐
                                            ╱  .eq.   ╲─────────►│ xsi = xs0 │
                                             ╲   0    ╱            └──────────┘
                                               ╲   ╱
                                                 │ no
                                                 ▼
                                        ┌──────────────────┐
                                        │ xsi = zeros(iz)  │
                                        └──────────────────┘
                                                 │◄─────────────────────┘
                                                 ▼
                              ┌──────────┬────────────────┐   yes
                        ┌────►│ iss=1    │ iss >nsinss     │─────────────────────┐
                        │     │ iss=iss+1│                 │                     │
                        │     └──────────┴────────────────┘                     │
                        │                 │ no                                  │
                        │                 ▼                                     │
                        │   ┌───────────────────────────────────────────┐      │
                        │   │ xs = wid*float(isub - 2) + wss*float(iss - 1) -xsi │
                        │   │ ys = sin(xs + xsi)                          │      │
                        │   │ sx = sx + xs                                │      │
                        │   │ sy = sy + ys                                │      │
                        │   │ sxx = sxx + xs*xs                           │      │
                        │   │ sxy = sxy + xs*ys                           │      │
                        │   └───────────────────────────────────────────┘      │
                        └─────────────────┘              │                      │
                                                         ▼                      │
                                                    ╱ iz  ╲      yes    ┌────────────────────────────┐
                                                  ╱  .ne.   ╲──────────►│ sin1(isub) = sxy/sxx       │
                                                   ╲   0    ╱            │ sin0(isub) = -sin1(isub)*xsi│
                                                     ╲   ╱               └────────────────────────────┘
                                                       │ no
                                                       ▼
                                  ┌────────────────────────────────────────────────┐
                                  │ del = fnsss*sxx-sx*sx                           │
                                  │ sin1(isub) = (fnsss*sxy - sy*sx)/del            │
                                  │ sin0(isub) = (sy*sxx - sx*sxy)/del - sin1(isub)*xsi│
                                  └────────────────────────────────────────────────┘
                    ◄────────────────────────────────────┘          │              │
                                                                                    ▼
                                        ┌──────────────────┐
                                        │ sinc0 = 2.       │
                                        │ sinc1 = 1./wid   │
                                        └──────────────────┘
                                                 │
                                        no       ▼
                              ◄──────────────╱ istest ╲
                                            ╱  .ne.    ╲
                                             ╲   0     ╱
                                               ╲    ╱
                                                 │ yes
                                                 ▼
                                        ┌──────────────────┐
                                        │ adev = 0.        │
                                        │ rdev = 0.        │
                                        │ s2c2mn = 10.     │
                                        │ s2c2mx = 0.      │
                                        └──────────────────┘
                                                 │
                                                 ▼
                              ┌──────────┬────────────────┐   yes
                        ┌────►│ isub=1   │ isub >nisub     │─────────────────────────────►
                        │     │ isub=isub+1│               │
                        │     └──────────┴────────────────┘
                        │                 │ no
                        │                 ▼
                        │     ┌──────────┬────────────────┐   yes
                        │ ┌──►│ iss=1    │ iss >nsinss     │──────────────────►
                        │ │   │ iss=iss+1│                 │
                        │ │   └──────────┴────────────────┘
                        │ │                │ no
                                           ▼

                    6    7    8         9              10                 11   12
```

244

```
theta = wid*float(isub - 1) + wss*float(iss - 1)
cthet = PI5D2 - theta
sinthe = sin(theta)
costhe = sin(cthet)
sint = sin(theta)
cost = cos(theta)
asd = abs(sinthe - sint)
acd = abs(costhe - cost)
adev = max(adev,asd,acd)
```

```
sint
.ne.
0.
```
yes → `rdev = max(rdev,asd/abs(sint))`

no

```
cost
.ne.
0.
```
yes → `rdev = max(rdev,acd/abs(cost))`

no

```
s2c2 = sinthe**2 + costhe**2
s2c2mn = min(s2c2mn,s2c2)
s2c2mx = max(s2c2mx,s2c2)
```

```
isub
.lt.
11
```
yes → `write(6,1340) theta,sinthe,sint, costhe,cost`

no

```
write(6,1350) mxsinc_loc,nsinss
write(6,1360) adev,rdev,s2c2mn,s2c2mx
```

```
adev = 0.
rdev = 0.
s2c2mn = 10.
s2c2mx = 0.
```

```
irn=1
irn=irn+1
```
`irn >nrna` yes →

no

```
call
randomset(rnnow)
```

```
theta = rnnow*PI5D2
cthet = PI5D2 - theta
sinthe = sin(theta)
costhe = sin(cthet)
sint = sin(theta)
cost = cos(theta)
asd = abs(sinthe - sint)
acd = abs(costhe - cost)
adev = max(adev,asd,acd)
```

13 14 15 16 17

sint .ne. 0. — yes → rdev = max(rdev,asd/abs(sint))

no

cost .ne. 0. — yes → rdev = max(rdev,acd/abs(cost))

no

s2c2 = sinthe**2 + costhe**2
s2c2mn = min(s2c2mn,s2c2)
s2c2mx = max(s2c2mx,s2c2)

write(6,1380) nrna
write(6,1390) adev,rdev,s2c2mn,s2c2mx

p = 1.

i=1
i=i+1 | i >50 — yes

no

pwr2i(i) = p
p = p/2.

j=1
j=j+1 | j >nmed — yes

no

i=1
i=i+1 | i >nreg — yes

no

iraylr(i) .eq. 1
.and.
med(i) .eq. j — yes → iraylm(j) = 1

no

j=1
j=j+1 | j >nmed — yes

no

i=1
i=i+1 | i >nreg — yes

no

incohr(i) .eq. 1
.and.
med(i).eq.j — yes → incohm(j) = 1

no

18

```
          ┌────┬────────┐   yes
          │j=1 │        ├─────────────────────────────────────▶
          │j=j+1│j >nmed │
          └────┴────────┘
                 │ no
          ┌────┬────────┐   yes
          │i=1 │        ├──────────────────────────▶
          │i=i+1│i >nreg │
          └────┴────────┘
                 │ no
     no   ╱ iprofr(i) .eq. 1 ╲   yes   ┌──────────────┐
    ◀─────╲    .and.         ╱────────▶│ iprofm(j) = 1│
          ╲ med(i) .eq. j   ╱          └──────────────┘
```

```
          ┌────┬────────┐   yes
          │j=1 │        ├─────────────────────────────────────▶
          │j=j+1│j >nmed │
          └────┴────────┘
                 │ no
          ┌────┬────────┐   yes
          │i=1 │        ├──────────────────────────▶
          │i=i+1│i >nreg │
          └────┴────────┘
                 │ no
     no   ╱ impacr(i).eq.1 ╲   yes   ┌──────────────┐
    ◀─────╲    .and.       ╱────────▶│ impacm(j) = 1│
          ╲ med(i) .eq. j ╱          └──────────────┘
```

```
              ╱ rewind kmpi ╱
              ┌──────────┐
              │  nm = 0  │
              └──────────┘
          ┌────┬────────┐   yes
          │im=1 │        ├─────────────────────────────────▶
          │im=im+1│im >nmed │
          └────┴────────┘
                 │ no
              ┌──────────┐
              │ lok(im) = 0│
              └──────────┘
     no   ╱ iraylm(im) ╲   yes   ╱ write(6,1440) im ╱
    ◀─────╲    .eq.    ╱────────▶
          ╲     1     ╱
```

```
          ┌────┬────────┐   yes
          │im=1 │        ├─────────────────────────────────▶
          │im=im+1│im >nmed │
          └────┴────────┘
                 │ no
     no   ╱ incohm(im) ╲   yes   ╱ write(6,4090) im ╱
    ◀─────╲    .eq.    ╱────────▶
          ╲     1     ╱
```

```
im=1
im=im+1    im >nmed      yes

              no

      iprofm(im)
no      .eq.        yes    write(6,4110) im
         1

im=1
im=im+1    im >nmed      yes

              no

      impacm(im)
no      .eq.        yes    write(6,4130) im
         1
```

(5)

read(kmpi,1270,end=1470) mbuf

```
ib=1
ib=ib+1    ib >lmdl      yes

              no

       mbuf(ib)
no      .ne.        yes
      mdlabl(ib)
```

```
im=1
im=im+1    im >nmed      yes

              no

ib=1
ib=ib+1    i b>lmdn      yes

              no

      il = lmdl + ib

       mbuf(il)              ib
yes     .ne.      no        .eq.      yes
      media(ib,im)          lmdn

                             no
```

```
      lok(im)
yes     .ne.
         0

         no
```

```
                                    no

                          lok(im) = 1
                          nm = nm + 1

                     write(kmpo,1510) im,mbuf

              read(kmpi,2000,err=1000) (mbuf(i),i=1,5),        "1000"
               rhom(im),nne(im),iunrst(im),epstfl(im),iaprim(im)

                                             backspace(kmpi)

                                     read(kmpi,2001) (mbuf(i),i=1,5),rhom(im),
                                      nne(im),iunrst(im),epstfl(im),iaprim(im)

               write(kmpo,1540) (mbuf(i),i=1,5),rhom(im),nne(im)


                     ie=1        ie >nne(im)     yes

                     ie=ie+1

                                     no

              read(kmpi,1560) (mbuf(i),i=1,6),(asym(im,ie,i),i=1,2),
               zelem(im,ie),wa(im,ie),pz(im,ie),rhoz(im,ie)

              write(kmpo,1570) (mbuf(i),i=1,6),(asym(im,ie,i),i=1,2),
               zelem(im,ie),wa(im,ie),pz(im,ie),rhoz(im,ie)


                          write(kmpo,1580)

              read(kmpi,1260) rlcm(im),ae(im),ap(im),ue(im),up(im)
              write(kmpo,1260)rlcm(im),ae(im),ap(im),ue(im),up(im)


                          te(im) = ae(im) - RM
                          thmoll(im) = te(im)*2. + RM

                          write(kmpo,4280)

     read(kmpi,1250) msge(im),mge(im),mseke(im),meke(im),mleke(im),
      mcmfp(im),mrange(im),irayl,ibound,incoh,icprof(im),impact
     write(kmpo,1250) msge(im),mge(im),mseke(im),meke(im),mleke(im),
      mcmfp(im),mrange(im),irayl,ibound,incoh,icprof(im),impact


                          nsge = msge(im)
                          nge = mge(im)
                          nseke = mseke(im)
                          neke = meke(im)
                          nleke = mleke(im)
                          ncmfp = mcmfp(im)
                          nrange = mrange(im)


              write(kmpo,1600)
              read(kmpi,1260) (dl1(i,im),dl2(i,im),dl3(i,im),dl4(i,im),
                              dl5(i,im),dl6(i,im),i=1,6)
              write(kmpo,1260) (dl1(i,im),dl2(i,im),dl3(i,im),dl4(i,im),
                              dl5(i,im),dl6(i,im),i=1,6)
```

```
write(kmpo,1610)
read(kmpi,1260) delcm(im),(alphi(i,im),bpar(i,im),
    delpos(i,im),i=1,2)
write(kmpo,1260) delcm(im),(alphi(i,im),bpar(i,im),
    delpos(i,im),i=1,2)
write(kmpo,1620)
read(kmpi,1260) xr0(im),teff0(im),blcc(im),xcc(im)
write(kmpo,1260) xr0(im),teff0(im),blcc(im),xcc(im)
write(kmpo,1630)
read(kmpi,1260) eke0(im),eke1(im)
write(kmpo,1260) eke0(im),eke1(im)
write(kmpo,1640)
read(kmpi,1260) (esig0(i,im),esig1(i,im),psig0(i,im),psig1(i,im),
    ededx0(i,im),ededx1(i,im),pdedx0(i,im),pdedx1(i,im),ebr10(i,im),
    ebr11(i,im),pbr10(i,im),pbr11(i,im),pbr20(i,im),pbr21(i,im),
    tmxs0(i,im),tmxs1(i,im),escpw0(i,im),escpw1(i,im),pscpw0(i,im),
    pscpw1(i,im),ekini0(i,im),ekini1(i,im),pkini0(i,im),pkini1(i,im),
    erang0(i,im),erang1(i,im),prang0(i,im),prang1(i,im),estep0(i,im),
    estep1(i,im),i=1,neke)
write(kmpo,1260) (esig0(i,im),esig1(i,im),psig0(i,im),psig1(i,im),
    ededx0(i,im),ededx1(i,im),pdedx0(i,im),pdedx1(i,im),ebr10(i,im),
    ebr11(i,im),pbr10(i,im),pbr11(i,im),pbr20(i,im),pbr21(i,im),
    tmxs0(i,im),tmxs1(i,im),escpw0(i,im),escpw1(i,im),pscpw0(i,im),
    pscpw1(i,im),ekini0(i,im),ekini1(i,im),pkini0(i,im),pkini1(i,im),
    erang0(i,im),erang1(i,im),prang0(i,im),prang1(i,im),estep0(i,im),
    estep1(i,im), i=1,neke)
```

```
write(kmpo,4340)
read(kmpi,1260) tebinda,ge0(im),ge1(im)
write(kmpo,1260) tebinda,ge0(im),ge1(im)
write(kmpo,1660)
read(kmpi,1260) (gmfp0(i,im),gmfp1(i,im),gbr10(i,im),gbr11(i,im),
    gbr20(i,im),gbr21(i,im),i=1,nge)
write(kmpo,1260) (gmfp0(i,im),gmfp1(i,im),gbr10(i,im),gbr11(i,im),
    gbr20(i,im),gbr21(i,im),i=1,nge)
```

stop

iraylm(im).eq.1 .and. (irayl.lt.1.or.irayl .gt.2) — yes → write(6,4360) im → stop

write(6,4375) im

yes

incohm(im).eq.1 .and. ibound.ne.1 — no ← incohm(im).eq.1 .and. incoh.ne.1 — yes → write(6,4370) im → stop

no (downward)

write(6,4380) im

yes

iprofm(im) .eq. 1 — yes → icprof(im) .ne. 3 — no

stop

write(6,4400) im

yes

incohm(im) .eq. 0 — yes → write(6,4390) im → stop

no

impacm(im).eq.1 .and. impact.eq.0

no

ibound .eq. 0 — yes → write(6,4400) im → stop

250

25

irayl.eq.1.or.irayl.eq.2 .or. irayl.eq.3 — no →

yes

write(kmpo,1680)
read(kmpi,1250) ngr(im)
write(kmpo,1250) ngr(im)

ngrim = ngr(im)

write(kmpo,1690)
read(kmpi,1260) rco0(im),rco1(im)
write(kmpo,1260) rco0(im),rco1(im)
write(kmpo,1700)
read(kmpi,1260) (rsct0(i,im),rsct1(i,im),i=1,ngrim)
write(kmpo,1260) (rsct0(i,im),rsct1(i,im),i=1,ngrim)
write(kmpo,1710)
read(kmpi,1260) (cohe0(i,im),cohe1(i,im),i=1,nge)
write(kmpo,1260) (cohe0(i,im),cohe1(i,im),i=1,nge)

iraylm(im) .ne. 1 — yes → write(6,1720) im

no

incoh .eq. 1 — no →

yes

write(kmpo,4450)
read(kmpi,1250) ngs(im)
write(kmpo,1250) ngs(im)
ngsim = ngs(im)
write(kmpo,4460)
read(kmpi,1260) sco0(im),sco1(im)
write(kmpo,1260) sco0(im),sco1(im)
write(kmpo,4470)
read(kmpi,1260) (sxz0(i,im),sxz1(i,im),i=1,ngsim)
write(kmpo,1260) (sxz0(i,im),sxz1(i,im),i=1,ngsim)

incohm(im) .ne. 1 — yes → write(6,4480) im

no

icprof(im).eq.1 .or. icprof(im).eq.2 — no →

yes

write(kmpo,4490)
read(kmpi,1250) ngc(im)
write(kmpo,1250) ngc(im)
ngcim = ngc(im)
write(kmpo,4500)
read(kmpi,1260) cco0(im),cco1(im),cpimev
write(kmpo,1260) cco0(im),cco1(im),cpimev
write(kmpo,4510)
read(kmpi,1260) (cpr0(i,im),cpr1(i,im),i=1,ngcim)
write(kmpo,1260) (cpr0(i,im),cpr1(i,im),i=1,ngcim)

27

28

iprofm(im) .ne. 1 — yes → write(6,4520) im

no

icprof(im).eq.3 .or. icprof(im) .eq. 4 — no →

yes

```
write(kmpo,4530)
read(kmpi,1250) mxshel(im),ngc(im)
write(kmpo,1250) mxshel(im),ngc(im)
ngcim = ngc(im)
mxsim = mxshel(im)
write(kmpo,4540)
read(kmpi,1260) (elecno(i,im),i=1,mxsim)
write(kmpo,1260) (elecno(i,im),i=1,mxsim)
write(kmpo,4550)
read(kmpi,1260) (capio(i,im),i=1,mxsim)
write(kmpo,1260) (capio(i,im),i=1,mxsim)
write(kmpo,4560)
read(kmpi,1260) ccos0(im),ccos1(im)
write(kmpo,1260) ccos0(im),ccos1(im)
write(kmpo,4570)
read(kmpi,1260) ((cprs0(i,is,im),cprs1(i,is,im),is=1,mxsim),
    i=1,ngcim)
write(kmpo,1260)
((cprs0(i,is,im),cprs1(i,is,im),is=1,mxsim),
    i=1,ngcim)
```

iprofm(im) .ne. 1 — yes → write(6,4580) im

no

impact .ge. 1 — no →

yes

```
write(kmpo,4590)
read(kmpi,1250) nepm(im)
write(kmpo,1250) nepm(im)
ner = nepm(im)
write(kmpo,4610)
read(kmpi,1250) neii(im)
write(kmpo,1250) neii(im)
write(kmpo,4620)
read(kmpi,1260) eico0(im),eico1(im)
write(kmpo,1260) eico0(im),eico1(im)
write(kmpo,4630)
read(kmpi,1260) ((eii0(i,ifun,im),eii1(i,ifun,im),
    ifun=1,ner),i=1,neii(im))
write(kmpo,1260) ((eii0(i,ifun,im),eii1(i,ifun,im),
    ifun=1,ner),i=1,neii(im))
```

impacm(im) .ne. 1 — yes → write(6,4640) im

no

nm .ge. nmed — no ←   yes →

29

emaxe .eq. 0.d0 — yes → emaxe=1.d50 → j=1 / j=j+1 → j >nmed — yes →

no ↓

no → emaxe=min(ue(j),up(j)+RM,emaxe)

emaxe .lt. 1.d-4 — yes → write(6,5007) emaxe → stop

write(6,5006) emaxe

no ↓

j=1 / j=j+1 → j >nmed — yes →

emaxe .gt. ue(j) — yes → write(6,5008) emaxe, ue(j), j → stop

no ↓

emaxe-RM .gt. up(j) — yes → write(6,5009) emaxe-RM, up(j), j → stop

no ↓

dunitr = dunit

dunit .lt. 0. — no →

yes ↓

id = max0(1,min0(MXMED,int(-dunit)))
dunit = rlc(id)

dunit .ne. 1. — yes → write(6,1730) dunitr,dunit

no ↓

im=1 / im=im+1 → im >nmed — yes →

no ↓

dfact = rlc(im)/dunit
dfacti = 1.0/dfact
i = 1

i = i + 1

i - meke(im) .gt. 0 — yes →

no ↓

30   31   32   33   34

```
esig0(i,im) = esig0(i,im)*dfacti
esig1(i,im) = esig1(i,im)*dfacti
psig0(i,im) = psig0(i,im)*dfacti
psig1(i,im) = psig1(i,im)*dfacti
ededx0(i,im) = ededx0(i,im)*dfacti
ededx1(i,im) = ededx1(i,im)*dfacti
pdedx0(i,im) = pdedx0(i,im)*dfacti
pdedx1(i,im) = pdedx1(i,im)*dfacti
tmxs0(i,im) = tmxs0(i,im)*dfact
tmxs1(i,im) = tmxs1(i,im)*dfact
```

i = 1

i = i + 1

i - meke(im) .gt. 0 — yes

no

dunitr .eq. 1 — yes — dfactr = 1.d0

no

dfactr = 1.d0 / dunit

```
escpw0(i,im) = escpw0(i,im)*dfactr
escpw1(i,im) = escpw1(i,im)*dfactr
pscpw0(i,im) = pscpw0(i,im)*dfactr
pscpw1(i,im) = pscpw1(i,im)*dfactr
ekini0(i,im) = ekini0(i,im)*dfactr
ekini1(i,im) = ekini1(i,im)*dfactr
pkini0(i,im) = pkini0(i,im)*dfactr
pkini1(i,im) = pkini1(i,im)*dfactr
erang0(i,im) = erang0(i,im)*dfact
erang1(i,im) = erang1(i,im)*dfact
prang0(i,im) = prang0(i,im)*dfact
prang1(i,im) = prang1(i,im)*dfact
```

```
teff0(im) = teff0(im)*dfact
blcc(im) = blcc(im)*dfacti
xcc(im) = xcc(im)*sqrt(dfacti)
rldu(im) = rlc(im)/dunit
```

i = 1

i = i + 1

i - mge(im) .gt. 0 — yes

no

```
gmfp0(i,im) = gmfp0(i,im)*dfact
gmfp1(i,im) = gmfp1(i,im)*dfact
```

35

```
                              35
                               │
                               ▼
          ┌──────────────────────────────────┐
          │ ecutmn = 1.d20                    │
          │ vacdst = vacdst*dunito/dunit      │
          │ dunito = dunit                    │
          └──────────────────────────────────┘
                               │
                               ▼
          ┌──────┬──────────────┐              yes
          │ jr=1 │              ├──────────────────────────────────────►
          ├──────┤ jr >nreg     │
       ┌─►│jr=jr+1│             │
       │  └──────┴──────────────┘
       │              │ no
       │              ▼
       │      ┌────────────────┐
       │      │ md = med(jr)   │
       │      └────────────────┘
       │              │
       │              ▼
       │           ╱ md.ge.1 ╲      no
       │          ╱   .and.    ╲──────────────────────────────┐
       │          ╲ md.le.nmed ╱                              │
       │           ╲         ╱                                │
       │              │ yes                                   │
       │              ▼                                       │
       │  ┌──────────────────────────────────────────────┐   │
       │  │ ecut(jr) = max(ecut(jr),ae(md))               │   │
       │  │ pcut(jr) = max(pcut(jr),ap(md))               │   │
       │  │ ecutmn = min(ecutmn,ecut(jr))                 │   │
       │  │ eke = ecut(jr) - RM                           │   │
       │  │ elke = log(eke)                               │   │
       │  │ lelke = eke1(md)*elke + eke0(md)              │   │
       │  │ ectrng(jr) = erang1(lelke,md)*elke + erang0(lelke,md) │
       │  │ pctrng(jr) = prang1(lelke,md)*elke + prang0(lelke,md) │
       │  └──────────────────────────────────────────────┘   │
       │              │                                       │
       │              ▼                                       │
       │           ╱ rhor(jr) ╲    yes   ┌────────────────────┐
       │          ╱   .eq.     ╲────────►│ rhor(jr) = rho(md) │
       │          ╲   0.0      ╱         └────────────────────┘
       │           ╲         ╱                    │            │
       │              │ no                        │            │
       └──────────────┴◄──────────────────────────┴────────────┘
                               │
                               ▼
                        ╱ ibrdst.eq.1 ╲    no
                       ╱    .or.        ╲──────────────────────────┐
                       ╲ iprdst .gt. 0  ╱                          │
                        ╲             ╱                            │
                               │ yes                               │
                               ▼                                   │
          ┌──────┬──────────────┐              yes                 │
          │ im=1 │              ├──────────────────────────────────┤
          ├──────┤ im >nmed     │                                  │
       ┌─►│im=im+1│             │                                  │
       │  └──────┴──────────────┘                                  │
       │              │ no                                         │
       │              ▼                                            │
       │      ┌────────────────┐                                   │
       │      │ zbrang(im) = 0. │                                  │
       │      │ pznorm = 0.    │                                   │
       │      └────────────────┘                                   │
       │              │                                            │
       │              ▼                                            │
       │  ┌──────┬──────────────┐              yes                 │
       │  │ ie=1 │              ├─────────────────┐                │
       │  ├──────┤ ie >nne(im)  │                 │                │
       │ ┌►│ie=ie+1│            │                 │                │
       │ │└──────┴──────────────┘                 │                │
       │ │            │ no                         │                │
       │ │            ▼                            │                │
       │ │┌────────────────────────────────────┐  │                │
       │ ││ zbrang(im) = zbrang(im) + pz(im,ie)*│  │                │
       │ ││    zelem(im,ie)*(zelem(im,ie) + 1.0)│  │                │
       │ ││ pznorm = pznorm + pz(im,ie)         │  │                │
       │ │└────────────────────────────────────┘  │                │
       │ └────────────────────────────────────────┘                │
       │              ▼                                            │
       │  ┌──────────────────────────────────────────────────────┐ │
       └──┤ zbrang(im) = (8.116224E-05)*(zbrang(im)/pznorm)**(1./3.)│ │
          └──────────────────────────────────────────────────────┘ │
                               │                                    │
                               ▼◄───────────────────────────────────┘
          ┌──────┬──────────────┐              yes
          │ ii=1 │              ├──────────────────────────────────►
          ├──────┤ ii >nreg     │
       ┌─►│ii=ii+1│             │
       │  └──────┴──────────────┘
       │              │ no
       │              ▼
       36             37             38
```

255

36   37   38

iedgfl(ii).ne.0 — yes → call edgbin
.or.
iauger(ii).ne.0
no

nmed — yes → write(6,1830)
.eq.
1
no
write(6,1840) nmed

call
rk1

call
rmsfit

i=1   i >nmed — yes →
i=i+1
no

emaxe/ue(i).lt.0.1d0 — no →
.and.
charD(i).eq.0.d0

write(6,5005) (media(j,i),j=1,24)

return
(to shower)

1470

write(6,1850) kmpi

im=1   im >nmed — yes →
im=im+1
no

lok(im) — no →
.ne.
1
yes

write(6,1870)
(media(i,im),i=1,lmdn)

stop

256

```
                 ╔═══════════════╗
                 ║  subroutine   ║
                 ║    kauger     ║
                 ╚═══════════════╝
                         │
                         ▼
              ┌────────────────────┐
              │ ikauger = ikauger + 1 │
              └────────────────────┘
                         │
                         ▼
                    ╱─────────╲              yes
                   ╱ dfkaug(13,iz) ╲──────────────────────────────►
                   ╲    .eq.     ╱
                    ╲    0.   ╱
                      ╲─────╱
                         │ no
                         ▼
              ┌────────────────────┐
              │ nauger = nauger + 1 │
              └────────────────────┘
                         │
                         ▼
              ┌────────────────────┐
              │  call              │
              │  randomset (rnnow) │
              └────────────────────┘
```

kaug=1
kaug=kaug+1   kaug>13   yes

eauger(nauger) = ekaug(14,iz)*1.E-3

no

rnnow
.le.
dfkaug(kaug,iz)   yes

eauger(nauger) = ekaug(kaug,iz)*1.E-3

no

kaug
.eq.
1   "K-L1L1"   call lshell(1)   call lshell(1)

yes

no

kaug
.eq.
2   "K-L1L2"   call lshell(1)   call lshell(2)

yes

no

kaug
.eq.
3   "K-L1L3"   call lshell(1)   call lshell(3)

yes

no

kaug
.eq.
4   "K-L2L2"   call lshell(2)   call lshell(2)

yes

no

kaug
.eq.
5   "K-L2L3"   call lshell(2)   call lshaell(3)

yes

no

1   2

1   2

"K-L3L3"
kaug .eq. 6 — yes → call lshell(3) → call lshell(3) →

no

kaug. eq. 7 — "K-L1M" yes

no

kaug .eq. 10 — "KL1N" yes → call lshell(1) →

no

kaug .eq. 8 — "K-L2M" yes

no

kaug .eq. 11 — "KL2N" yes → call lshell(2) →

no

kaug .eq. 9 — "K-L3M" yes

no

kaug .eq. 12 — "KL3N" yes → call lshell(3) →

no

return
(to kshell)

258

```
                                          Version
         subroutine                       051219-1435
           kshell


      ikshell = ikshell + 1


         call
         randomset (rnnow)


                           "K-auger"
            rnnow                          call
             .gt.            yes           kauger
          omegak(iz)

         no    "K-xray"


          call
          kxray


         return
         (to photo)
```

```
       ┌─────────────────┐
      《  subroutine      》
      《    kxray         》
       └────────┬────────┘
                │
       ┌─────────────────┐
       │ ikxray = ikxray + 1 │
       └────────┬────────┘
                │
          ╱─────────╲
  yes    ╱  dfkx(9,iz) ╲
 ◄──────╱    .eq.       ╲
         ╲     0.       ╱
          ╲───────────╱
                │ no
       ┌─────────────────┐
       │ nxray = nxray + 1 │
       └────────┬────────┘
                │
       ┌─────────────────┐
       │ call            │
       │ randomset (rnnow)│
       └────────┬────────┘
                │
    ┌──────┬─────────┐  yes
    │ ik=1 │         ├──────►  exray(nxray) = ekx(10,iz)*1.E-3
    ├──────┤  ik>9   │
    │ik=ik+1│        │
    └──────┴────┬────┘
                │ no
          ╱─────────╲
   no    ╱  rnnow    ╲   yes
 ◄──────╱   .le.      ╲──────►  exray(nxray) = ekx(ik,iz)*1.E-3
         ╲ dfkx(ik,iz)╱
          ╲─────────╱
                │
          ╱─────────╲
         ╱   ik       ╲  "K-L3"   ┌──────┐
         ╲   .eq.     ╱──────────►│ call │
          ╲   1      ╱    yes     │lshell(3)│
           ╲────────╱             └──────┘
                │ no
          ╱─────────╲
         ╱   ik       ╲  "K-L2"   ┌──────┐
         ╲   .eq.     ╱──────────►│ call │
          ╲   2      ╱    yes     │lshell(2)│
           ╲────────╱             └──────┘
                │ no
          ╱─────────╲
         ╱   ik       ╲  "K-L1"   ┌──────┐
         ╲   .eq.     ╱──────────►│ call │
          ╲   3      ╱    yes     │lshell(1)│
           ╲────────╱             └──────┘
                │ no
       ┌─────────────────┐
      《   return         》
      《  (to kshell)     》
       └─────────────────┘
```

```
          subroutine
            lauger
```

ilauger = ilauger + 1

call
randomset (rnnow)

ii .eq. 1 — yes

ii .eq. 2 — no (from ii .eq. 1 "no")

ii .eq. 3 — no (from ii .eq. 2 "no")

no (from ii .eq. 3) → 1

yes (ii .eq. 2)

yes (ii .eq. 3)

dfl1aug(5,iz) .eq. 0. — yes → (to 3)

no

nauger = nauger + 1

laug=1 / laug=laug+1 | laug>5 — yes

eauger(nauger) = el1aug(6,iz)*1.E-3 → (to 3)

no

rnnow .le. dfl1aug(laug,iz) — no / yes

eauger(nauger) = el1aug(laug,iz)*1.E-3 → (to 3)

df21aug(5,iz) .eq. 0. — yes → (to 3)

no

nauger = nauger + 1

laug=1 / laug=laug+1 | laug>5 — yes

eauger(nauger) = el2aug(6,iz)*1.E-3 → (to 3)

no

rnnow .le. dfl2aug(laug,iz) — no / yes

eauger(nauger) = el2aug(laug,iz)*1.E-3 → (to 3)

1   2   3

1

2

3

dfl3aug(5,iz) .eq. 0.

yes

no

nauger = nauger + 1

laug=1
laug=laug+1

laug>5

yes

eauger(nauger) = el3aug(6,iz)*1.E-3

no

rnnow .le. dfl3aug(laug,iz)

no

yes

eauger(nauger) = el3aug(laug,iz)*1.E-3

return
(to kshell)

subroutine
lshell(ll)

ilshell = ilshell + 1

ickflg=0

ll
.eq.
2

"L2-hole is created"
yes

no

ll
.eq.
3

"L3-hole is created"
yes

no   "L1-hole is created"

call
randomset (rnnow)

ebind = eedge(2,iz)*1.E-3

rnnow
.gt.
omegal1(iz)

"L1-X-ray"
no

call
lxray(1)

yes

rnnow
.le.
omegal1(iz) + f12(iz)

"Coster-Kronig
f12"
yes

ickflg=1

no

rnnow.
le.
omegal1(iz) + f12(iz) + f13(iz)

"Coster-Kronig
f13"
yes

ickflg=1

no   "L1-Auger"

call
lauger(1)

call
randomset (rnnow)

ickflg
.eq.
0

yes

ebind = eedge(3,iz)*1.E-3

no

rnnow
.gt.
omegal2(iz)

"L2-X-ray"
no

call
lxray(2)

yes

1   2   3   4

263

1　　　　　　　2　　　　　　　　　　　　　　　　　　　　3　　4

```
            rnnow
             .le.              "Coster-Kronig
        omegal2(iz) + f23(iz)      f23"              ickflg=1
                                   yes

              no

           call
           lauger(2)
```

```
           ickflg
            .eq.              yes         ebind = eedge(4,iz)*1.E-3
             0

             no

           call
           randomset (rnnow)
```

```
            rnnow
             .gt.             "L3-X-ray"          call
          omegal3(iz)             no              lxray(3)

         yes    "L3-Auger"

           call
           lauger(3)
```

```
           return
           (to photo)
```

```
                    ⬡ subroutine
                      lxray(ll) ⬡
                         │
                         ▼
                 ┌──────────────────┐
                 │ ilxray = ilxray + 1 │
                 └──────────────────┘
                         │
                         ▼
                 ┌──────────────────┐
                 │ call             │
                 │ randomset (rnnow)│
                 └──────────────────┘
                         │
                         ▼
                      ◇ ll ◇              "L2-Xray"
                      ◇.eq.◇ ─────────────────────────────► 2
                      ◇ 2 ◇                  yes
                         │ no
                         ▼
      "L3-Xray"       ◇ ll ◇
   ◄──────────────────◇.eq.◇
        yes           ◇ 3 ◇
                         │ no
     "L1-Xray"           ▼
                    ◇ dflx1(7,iz) ◇           yes
                    ◇   .eq.      ◇ ──────────────────────►
                    ◇    0.       ◇
                         │ no
                         ▼
                 ┌──────────────────┐
                 │ nxray=nxray+1    │
                 └──────────────────┘
                         │
                         ▼
           ┌──────┬──────────┐
           │ lx=1 │          │       ┌──────────────────────────────┐
           ├──────┤  lx>7    │─yes──►│ exray(nxray) = elx1(8,iz)*1.E-3 │──►
        ┌─►│lx=lx+1│          │       └──────────────────────────────┘
        │  └──────┴──────────┘
        │             │ no
        │             ▼
        │        ◇ rnnow  ◇              ┌──────────────────────────────┐
        └──no────◇  .le.  ◇──yes────────►│ exray(nxray) = elx1(lx,iz)*1.E-3 │──►
                 ◇dflx1(lx,iz)◇          └──────────────────────────────┘
                         │
                         ▼
                    ◇ dflx2(4,iz) ◇           yes
                    ◇   .eq.      ◇ ──────────────────────────────►
                    ◇    0.       ◇
                         │ no
                         ▼
                 ┌──────────────────┐
                 │ nxray=nxray+1    │
                 └──────────────────┘
                         │
                         ▼
           ┌──────┬──────────┐
           │ lx=1 │          │       ┌──────────────────────────────┐
           ├──────┤  lx>4    │─yes──►│ exray(nxray) = elx2(5,iz)*1.E-3 │──►
        ┌─►│lx=lx+1│          │       └──────────────────────────────┘
        │  └──────┴──────────┘
        │             │ no
        │             ▼
        │        ◇ rnnow  ◇              ┌──────────────────────────────┐
        └──no────◇  .le.  ◇──yes────────►│ exray(nxray) = elx2(lx,iz)*1.E-3 │──►
                 ◇dflx2(lx,iz)◇          └──────────────────────────────┘
                         │
                         ▼
                         1                                          2
```

1   2

```
        dflx3(6,iz)          yes
           .eq.          ─────────────────────────────────────►
            0.

            │ no
            ▼
    ┌──────────────────┐
    │ nxray=nxray+1    │
    └──────────────────┘
            │
            ▼
    ┌─────────┬─────────┐   yes   ┌──────────────────────────────────┐
    │  lx=1   │         ├────────►│ exray(nxray) = elx3(7,iz)*1.E-3  │────►
    ├─────────┤  lx>6   │         └──────────────────────────────────┘
    │ lx=lx+1 │         │
    └─────────┴─────────┘
            │ no
            ▼
         rnnow            yes   ┌──────────────────────────────────┐
          .le.      ──────────►│ exray(nxray) = elx3(lx,iz)*1.E-3  │────►
       dflx3(lx,iz)            └──────────────────────────────────┘

     no
            ▼
        ╱──────────╲
       │  return    │
       │ (to lshell)│
        ╲──────────╱
```

```
        ┌─────────────┐
        │ subroutine  │
        │   moller    │
        └─────────────┘
               │
     ┌──────────────────────┐
     │  imoller = imoller +1 │
     └──────────────────────┘
               │
     ┌──────────────────────────────┐
     │ eie   = e(np)                │
     │ ekin  = eie - RM             │
     │ t0    = ekin/RM              │
     │ e0    = t0 + 1.0             │
     │ extrae= eie - thmoll(medium) │
     │ e02   = e0*e0                │
     │ ep0   = te(medium)/ekin      │
     │ g2    = t0*t0/e02            │
     │ g3    = (2.0*t0 + 1.0)/e02   │
     │ gmax  = (1.0 + 1.25*g2)      │
     └──────────────────────────────┘
```

"Rejection sampling loop"

```
     ┌──────────────────────┐
     │ call                 │
     │ randomset (rnnow)    │
     └──────────────────────┘
               │
     ┌────────────────────────────────────────┐
     │ br = te(medium)/(ekin - extrae*rnnow)   │
     │ r = br/(1.0 - br)                       │
     └────────────────────────────────────────┘
               │
     ┌──────────────────────┐
     │ call                 │
     │ randomset (rnnow)    │
     └──────────────────────┘
               │
     ┌────────────────────────────────────────┐
     │ rejf = 1.0 + g2*br*br + r*(r - g3)      │
     │ rnnow = gmax*rnnow                      │
     └────────────────────────────────────────┘
               │
            ◇ rnnow
     yes    .gt.        no
            rejf
```

```
     ┌──────────────────────┐
     │ ekse2 = br*ekin      │
     │ ese1 = eie - ekse2   │
     │ ese2 = ekse2 + RM    │
     │ e(np) = ese1         │
     │ e(np+1) = ese2       │
     └──────────────────────┘
               │
     ┌─────────────────────────────────────┐
     │ h1 = (eie + RM)/ekin                │
     │ dcosth = h1*(ese1 - RM)/(ese1 + RM) │
     │ sinthe = sqrt(1.0 - dcosth)         │
     │ costhe = sqrt(dcosth)               │
     └─────────────────────────────────────┘
               │
     ┌──────────────────────┐
     │ call                 │
     │ uphi(2,1)            │
     └──────────────────────┘
               │
     ┌─────────────────────────────────────┐
     │ np = np + 1                         │
     │ iq(np)=-1                           │
     │ dcosth = h1*(ese2 - RM)/(ese2 + RM) │
     │ sinthe = - sqrt(1.0 - dcosth)       │
     │ costhe = sqrt(dcosth)               │
     └─────────────────────────────────────┘
               │
     ┌──────────────────────┐
     │ call                 │
     │ uphi(3,2)            │
     └──────────────────────┘
               │
     ┌──────────────────────┐
     │ k1step(np) = 0.      │
     │ k1init(np) = 0.      │
     │ k1rsd(np) = 0.       │
     └──────────────────────┘
               │
               1
```

267

1

impacr(ir(np)).eq.1 .and. iedgfl(ir(np)).ne.0 — no

yes

call randomset (rnnow)

eke = eie - RM
elke2 = log(eke)
lelke2 = eico1(medium)*elke2 + eico0(medium)

ifun=1
ifun=ifun+1
ifun>nepm(medium) — yes

no

eiir = eii1(lelke2,ifun,medium)*elke2 + eii0(lelke2,ifun,medium)

rnnow .lt. eiir — no

yes

iarg=25

iausfl(iarg+1) . ne. 0 — yes → call ausgab(iarg)

no

iz = zelem(medium,ifun)

call eii

iarg=26

iausfl(iarg+1) . ne. 0 — yes → call ausgab(iarg)

no

return (to electrl)

268

```
                    ╔═══════════════╗              Version
                    ║   subroutine  ║              060313-1005
                    ║     mscat     ║
                    ╚═══════════════╝
                            │
                    ┌───────────────────┐
                    │ imscat = imscat +1 │
                    │ im = medium        │
                    └───────────────────┘
                            │
              no           ╱ useGSD(im).ne.0 ╲
   ◄────────────────────── ◄    .and.         ►
                           ╲ eke.lt.msgrid(nmsgrd(im),im) ╱
                            │ yes
                           ╱ iq(np)  ╲    no     ┌──────────┐
                          ◄  .eq.     ►────────► │ iprt = 2 │
                           ╲  -1     ╱           └──────────┘
                            │ yes                     │
                    ┌──────────┐                      │
                    │ iport = 1 │                     │
                    └──────────┘                      │
"find the correct energy interval                     │
        ┌────────────────────────────────────────────────────┐
        │ delog = DLOG10(eke*1.d6)                            │
        │ demod = MOD(delog,1.d0)                             │
        │ decade = delog - demod                             │
        │ iegrid = nmsdec(im) * (decade - initde(im) + demod) - jskip(im) │
        └────────────────────────────────────────────────────┘
                            │
                           ╱ iegrid  ╲    no
                          ◄  .ge.      ►──────────────────────┐
                           ╲ nmsgrd(im)╱                      │
                            │ yes                             │
                    ┌─────────────────────┐                  │
                    │ iegrid = nmsgrd(im) - 1 │              │
                    └─────────────────────┘                  │
                            │ ◄──────────────────────────────┘
        ┌────────────────────────────────────────────────┐
        │ fject = (eke - msgrid(iegrid,im)) /             │
        │   (msgrid(iegrid+1,im) - msgrid(iegrid,im))     │
        └────────────────────────────────────────────────┘
                            │
                    ┌──────────────┐
                    │ call         │
                    │ randomset (xi)│
                    └──────────────┘
                            │
                           ╱  xi   ╲    yes    ┌──────────────────┐
                          ◄  .lt.   ►────────► │ iegrid = iegrid +1│
                           ╲ fject ╱           └──────────────────┘
                            │ no                      │
                            │ ◄───────────────────────┘
        ┌─────────────────────────────────────────────────┐
        │ ktot = k1rsd(np) + k1init(np)                    │
        │ ik1 = DLOG(ktot/k1grd(iprt,1)) / dk1log(iprt) + 1 │
        └─────────────────────────────────────────────────┘
                            │
                           ╱  ik1  ╲    yes    ┌────────────┐
                          ◄  .gt.   ►────────► │ ik1 = NK1-1 │──┐
                           ╲  NK1  ╱           └────────────┘   │
                            │ no                                │
                           ╱  ik1  ╲    yes    ┌─────────┐      │
                          ◄  .le.   ►────────► │ ik1 = 1 │──────┤
                           ╲   0   ╱           └─────────┘      │
                            │ no  ◄────────────────────────────┘
1                           2
```

```
                    1                          2

                         ┌──────────────────────────────────┐
                         │ fject = (ktot - k1grd(iprt,ik1)) /│
                         │   (k1grd(iprt,ik1+1) - k1grd(iprt,ik1))│
                         └──────────────────────────────────┘
                                       │
                         ┌──────────────────────────┐
                         │ call                     │
                         │ randomset (xi)           │
                         └──────────────────────────┘
                                       │
                              ╱╲
                            ╱      ╲        yes      ┌──────────────┐
                          ╱   xi     ╲─────────────→ │ ik1 = ik1 +1 │
                          ╲   .lt.   ╱               └──────────────┘
                            ╲ fject╱                        │
                              ╲╱                            │
                          no   │                            │
                               │←───────────────────────────┘
                         ┌──────────────────────────┐
                         │ call                     │
                         │ randomset (xi)           │
                         └──────────────────────────┘
                                       │
                         ┌──────────────────────────┐
                         │ iamu = xi * neqp(im) + 1  │
                         └──────────────────────────┘
                                       │
                              ╱╲
                            ╱      ╲        no
                          ╱  iamu    ╲────────────────────────────────────┐
                          ╲   .eq.   ╱                                     │
                            ╲neqp(im)╱                                     │
                              ╲╱                                           │
                          yes  │                                          │
                         ┌──────────────────────────┐                     │
                         │ call                     │                     │
                         │ randomset (xi)           │                     │
                         └──────────────────────────┘                     │
                                       │                                  │
                              ╱╲                                          │
                            ╱      ╲        yes      ┌──────────┐          │
                          ╱   xi     ╲─────────────→ │ iamu = 1 │          │
                          ╲  .lt.    ╱               └──────────┘          │
                      ╲ecdf(2,iprt,iegrid,ik1,im)╱        │                │
                              ╲╱                          │                │
                          no   │                          │                │
                         ┌──────────────────────────────┐ │                │
                         │ call findi(ecdf(1,iprt,iegrid,ik1,im),│         │
                         │   xi,neqa(im)+1,iamu)         │ │                │
                         └──────────────────────────────┘ │                │
                                       │←─────────────────┘                │
                         ┌──────────────────────────────┐                  │
                         │ iamu = iamu + neqp(im) - 1    │                  │
                         └──────────────────────────────┘                  │
                                       │←────────────────────────────────  ┘
                         ┌──────────────────────────────────────┐
                         │ b1 = ebms(iamu,iprt,iegrid,ik1,im)    │
                         │ eta = eetams(iamu,iprt,iegrid,ik1,im) │
                         │ x1 = eamu(iamu,iprt,iegrid,ik1,im)    │
                         │ x2 = eamu(iamu+1,iprt,iegrid,ik1,im)  │
                         │ c1 = (x2 + eta) / (x2 - x1)           │
                         │ fmax = 1.d0 + b1 * (x2 - x1)**2       │
                         └──────────────────────────────────────┘
                                       │
         ┌────────────────→     "Sample Wentzel shape part of fit"
         │                                │
         │               ┌──────────────────────────┐
         │               │ call                     │
  "Rejection            │ randomset (xi)           │
  sampling loop"         └──────────────────────────┘
         │                                │
         │      ┌──────────────────────────────────────────────┐
         │      │ xmu = ((eta * xi) + (x1 * c1)) / (c1 - xi)    │
         │      │ fject = 1.d0 + b1 * (xmu - x1) * (x2 - xmu)   │
         │      └──────────────────────────────────────────────┘
         │                                │
         │               ┌──────────────────────────┐
         │               │ call                     │
         │               │ randomset (xi)           │
         │               └──────────────────────────┘
         │                                │
         │                       ╱╲
         │          yes        ╱      ╲
         └────────────────────╱  xi * fmax ╲
                              ╲   .gt.     ╱
                                ╲ fject  ╱
                                  ╲╱
                                   │ no

                    3                          4
```

270

3   4

```
costhe = 1.d0 - 2.d0 * xmu
sinthe =  DSQRT(1.d0 - costhe * costhe)
iskpms = 0
```

nomsct(ir(np)).eq.1
.or.
iskpms.ne.0 — yes →

```
sinthe = 0.
costhe = 1.
theta = 0.
noscat = noscat + 1
iskpms = 0
```

no

`xr = sqrt(gms*tvstep*b)`

"Set bi (B-inverse) that will be
used in sampling (bi must not
be larger than 1/lambda=1/2)"

b
.gt.
2. — yes → `bi = 1./b`

no

`bi = 0.5`

```
bmd = 1. + 1.75*bi
bm1 = (1. - 2./b)/bmd
bm2 = (1. + 0.025*bi)/bmd
```

" Loop for Bethe correction
factor (or other) rejection"

```
call
randomset (rms1)
```

"Gaussian, F1"

rms1
.le.
bm1 — yes →

```
call
randomset (rms2)
```

no

rms2
.eq.
0. — no →

yes

`rms2 = 1.E-30`

`thr = sqrt(max(0.D0,-log(rms2)))`

rms1
.le.
bm2 — no →

"Tail, F3"

yes

```
call
randomset (rms3)
```

```
call
randomset (rms4)
```

```
eta = max(rms3,rms4)
i31 = b0g31 + eta*b1g31
g31 = g310(i31) + eta*(g311(i31) + eta*g312(i31))
i32 = b0g32 + eta*b1g32
g32 = g320(i32) + eta*(g321(i32) + eta*g322(i32))
g3 = g31 + g32*bi
```

```
call
randomset (rms5)
```

"Rejection sampling loop"

rms5
.gt.
g3 — yes →

no

5   6   7   8   9

271

thr = 1./eta

"Central correction, F2"

call
randomset (rms6)

```
thr = rms6
i21 = b0g21 + thr*b1g21
g21 = g210(i21) + thr*(g211(i21) + thr*g212(i21))
i22 = b0g22 + thr*b1g22
g22 = g220(i22) + thr*(g221(i22) + thr*g222(i22))
g2  = g21 + g22*bi
```

call
randomset (rms7)

rms7
.gt.
g2

yes

no

theta = thr*xr   "Real angle (thr is the reduced angle)"

theta
.ge.
PI

yes

no

sinthe = sin(theta)

call
randomset (rms8)

rms8**2*theta
.le.
sinthe

no

yes

cthet = PI5D2 - theta
costhe = sin(cthet)

return
(to electr)

"Rejection sampling loop"

272

```
         ╔═══════════════════╗
         ║   subroutine      ║
         ║     pair          ║
         ╚═══════════════════╝
                  │
        ┌──────────────────┐
        │  ipair = ipair +1│
        └──────────────────┘
                  │
        ┌──────────────────┐
        │   eig = e(np)    │
        └──────────────────┘
                  │
                ◇ eig                      "Approximate:smoothing
                  .le.    ──yes──►          connection at boundary"
                  2.1                      ┌──────────────────┐
                  │                        │  call            │
                 no                        │  randomset(rnnow1)│
                  │                        └──────────────────┘
                  │                                 │
                  │                        ┌─────────────────────────────┐
                  │                        │ ese2 = RM + rnnow*(eig/2. - RM)│
                  │                        └─────────────────────────────┘
   ┌────────┐     ◇ eig
   │ lvx = 1│◄─yes── .lt.
   │ lvl0 = 0│        50.
   └────────┘         │
        │            no
        │             │
        │      ┌────────┐
        │      │ lvx = 2│
        │      │ lvl0 = 3│
        │      └────────┘
        └──────────┤
                   │
        ┌──────────────────┐
        │  call            │
        │  randomset(rnnow1)│
        └──────────────────┘
                   │
        ┌──────────────────┐
        │  call            │
        │  randomset(rnnow)│
        └──────────────────┘
                   │
                 ◇ rnnow                "Uniform distribution"
                   .ge.    ──no──►     ┌────────────────┐
                   bpar(lvx,medium)    │ lvl = lvl0 + 3 │
                   │                   │ br = rnnow1*0.5 │
                  yes                  └────────────────┘
                   │
   "12(br-0.5)**2  ┌──────────────┐
    subdistribution"│ lvl = lvl0 + 1│
                   └──────────────┘
                   │
        ┌──────────────────┐
        │  call            │
        │  randomset(rnnow2)│
        └──────────────────┘
                   │
        ┌──────────────────┐
        │  call            │
        │  randomset(rnnow3)│
        └──────────────────┘
                   │
   ┌──────────────────────────────────────────┐
   │ br = 0.5*(1.0 - max(rnnow1,rnnow2,rnnow3))│
   └──────────────────────────────────────────┘
                   │
              ◇ eig*br
      ◄──yes──  .lt.
                RM
                   │
                  no
                   │
   ┌──────────────────────────────┐
   │ del = 1.0/(eig*br*(1.0 - br)) │
   └──────────────────────────────┘
                   │
              ◇ del
      ◄──yes──  .ge.
                delpos(lvx,medium)
                   │
                  no
```

"Rejection sampling loop"

1   2   3

273

1

2

3

"Rejection sampling loop"

delta = delcm(medium)*del

```
       delta
        .lt.          no        rejf = dl4(lvl,medium) +
        1.0          ----->         dl5(lvl,medium)*log(delta +
                                       dl6(lvl,medium))
```

yes

rejf = dl1(lvl,medium) +
   delta*(dl2(lvl,medium) +
   delta*dl3(lvl,medium))

call
randomset(rnnow)

```
yes        rnnow
<------      .gt.
            rejf
```

no

ese2 = br*eig

"Set up secondary electron #1
(electron #2 has lower energy)"

ese1 = eig - ese2
e(np) = ese1
e(np+1) = ese2
k1step(np) = 0.
k1init(np) = 0.
k1rsd(np) = 0.
k1step(np+1) = 0.
k1init(np+1) = 0.
k1rsd(np+1) = 0.

"Sample to get polar angles
of secondary electrons"

```
(iprdst.eq.1).or.
((iprdst.eq.2).and.
 (eig.lt.4.14))
```

"Sample lowest-order
angular distribution"

```
  ichrg=1
  ------------  ichrg>2    yes
  ichrg=ichrg+1           ------->
```

no

```
       ichrg         no
        .eq.        ------>    ese = ese2
        1
```

yes

ese = ese1

pse = sqrt(max(0.D0,(ese - RM)*(ese + RM)))

call
randomset(rnnow)

costhe = 1.0 - 2.0*rnnow
sinthe = RM*sqrt((1.0 - costhe)*(1.0 + costhe))/(pse*costhe + ese)
costhe = (ese*costhe + pse)/(pse*costhe + ese)

4   5   6   7

274

```
              6                                    7
         ichrg          no    np = np + 1
         .eq.      ────────►   sinthe = -sinthe
          1
          │ yes                      │
          ▼                          ▼
        call                       call
        uphi(2,1)                  uphi(3,2)
```

```
        call
        randomset(rnnow)

         rnnow         no    iq(np) = -1
         .le.     ────────►  iq(np-1) = 1
         0.5
          │ yes
          ▼
        iq(np) = 1
        iq(np-1) = -1
```

```
                 no      (iprdst.eq.2)
           ◄──────────    .and.
                          (eig.ge.4.14)
 "Sample from                  │ yes
 Motz-Olsen-Koch               ▼
 (1969) distribution"    ztarg = zbrang(medium)
                         tteig = eig/RM
```

```
        ichrg=1
                    ichrg>2    yes
        ichrg=ichrg+1      ────────►
                      │ no
                      ▼
         ichrg          no    ese = ese2
         .eq.      ────────►
          1
          │ yes
          ▼
        ese = ese1
```

```
ttese = ese/RM
ttpse = sqrt((ttese - 1.0)*(ttese + 1.0))
esedei = ttese/(tteig - ttese)
eseder = 1.0/esedei
ximin = 1.0/(1.0 + (PI*ttese)**2)
rejmin = 2.0 + 3.0*(esedei + eseder) - 4.00*(esedei +
        eseder + 1.0 - 4.0*(ximin - 0.5)**2)*(1.0 +
        0.25*log(((1.0 + eseder)*(1.0 + esedei)/
        (2.0*tteig))**2 + ztarg*ximin**2))
ya = (2.0/tteig)**2
xitry = max(0.01D0,max(ximin,min(0.5D0,sqrt(ya/ztarg))))
galpha = 1.0 + 0.25*log(ya + ztarg*xitry**2)
gbeta = 0.5*ztarg*xitry/(ya + ztarg*xitry**2)
galpha = galpha - gbeta*(xitry - 0.5)
ximid = galpha/(3.0*gbeta)
```

```
                          ┌─────────┐
                          │ galpha  │  no
                          │  .ge.   ├──────────┐
                          │  0.0    │          │
                          └────┬────┘          ▼
                            yes │      ┌──────────────────────────────────┐
                                │      │ ximid = 0.5 - ximid - sqrt(ximid**2+0.25) │
                                │      └──────────────────────────────────┘
                                ▼                │
                    ┌──────────────────────────────────┐
                    │ ximid = 0.5 - ximid + sqrt(ximid**2 + 0.25) │
                    └──────────────────────────────────┘
                                │
                                ▼
        ┌────────────────────────────────────────────────────┐
        │ ximid = max(0.01D0,max(ximin,min(0.5D0,ximid)))     │
        │ rejmid = 2.0 + 3.0*(esedei + eseder) - 4.0*(esedei +│
        │     eseder + 1.0 - 4.0*(ximid - 0.5)**2)*(1.0 +     │
        │     0.25*log(((1.0 + eseder)*(1.0 + esedei)/        │
        │     (2.0*tteig))**2 + ztarg*ximid**2))              │
        │ rejtop = 1.02*max(rejmin,rejmid)                    │
        └────────────────────────────────────────────────────┘
                                │
                                ▼
                        ┌──────────────┐
                        │ call         │
                        │ randomset(xitest) │
                        └──────────────┘
                                │
                                ▼
        ┌────────────────────────────────────────────────────┐
        │ rejtst = 2.0 + 3.0*(esedei + eseder) - 4.0*(esedei +│
        │     eseder + 1.0 - 4.0*(xitst - 0.5)**2)*(1.0 +     │
        │     0.25*log(((1.0 + eseder)*(1.0 + esedei)/        │
        │     (2.0*tteig))**2 + ztarg*xitst**2))              │
        └────────────────────────────────────────────────────┘
                                │
                                ▼
                        ┌──────────────┐
                        │ call         │
                        │ randomset(rtest) │
                        └──────────────┘
                                │
                                ▼
                ┌──────────────────────────────────┐
                │ theta = sqrt(1.0/xitst - 1.0)/ttese │
                └──────────────────────────────────┘
                                │
                                ▼
                        ┌──────────────────────┐
              yes       │ rtest.gt.(rejtst/rejtop) │
          ┌─────────────┤      .or.             │
          │             │  (theta .ge. PI)      │
          │             └──────────┬───────────┘
          │                     no │
          │                        ▼
          │             ┌──────────────────┐
          │             │ sinthe=sin(theta) │
          │             │ costhe=cos(theta) │
          │             └──────────────────┘
          │                        │
          │                        ▼
          │                ┌──────────┐   no   ┌──────────────────┐
          │                │ ichrg    ├───────▶│ np = np + 1      │
          │                │ .eq.     │        │ sinthe = -sinthe │
          │                │  1       │        └──────────────────┘
          │                └────┬─────┘                │
          │                 yes │                       ▼
          │                     ▼               ┌──────────────┐
          │             ┌──────────────┐        │ call         │
          │             │ call         │        │ uphi(3,2)    │
          │             │ uphi(2,1)    │        └──────────────┘
          │             └──────────────┘
```

"Rejection sampling loop"

```
                                ▼
                        ┌──────────────┐
                        │ call         │
                        │ randomset(rnnow) │
                        └──────────────┘
                                │
                                ▼
                        ┌──────────┐   no   ┌──────────────┐
                        │ rnnow    ├───────▶│ iq(np) = -1  │
                        │ .le.     │        │ iq(np-1) = 1 │
                        │  0.5     │        └──────────────┘
                        └────┬─────┘
                         yes │
                             ▼
                    ┌──────────────┐
                    │ iq(np) = 1   │
                    │ iq(np-1) = -1 │
                    └──────────────┘
```

276

theta=RM/eig

"Polar angle is
m/E (default)"

call
uphi(1,1)

np = np + 1
sinthe = -sinthe

call
uphi(3,2)

call
randomset(rnnow)

rnnow
.le.
0.5

no → iq(np) = -1
iq(np-1) = 1

yes

iq(np) = 1
iq(np-1) = -1

return
(to photon)

277

```
                    ┌─────────────┐
                    │ subroutine  │
                    │   photo     │
                    └─────────────┘
                           │
                           ▼
                  ┌──────────────────┐
                  │ iphoto = iphoto +1│
                  └──────────────────┘
                           │
                           ▼
```

"Calculate energy
dependent sub-shell ratio"

```
              ┌──────────────────────────┐
              │ nxray = 0                 │
              │ nauger = 0                │
              │ irl = ir(np)              │
              │ eig = e(np)               │
              │ phol = log(eig)           │
              │ medium = med(irl)         │
              │ eigk = eig*1000.D0        │
              │ pholk = log(eigk)         │
              │ pholk2 = pholk*pholk      │
              │ pholk3 = pholk2*pholk     │
              │ total = 0.                │
              └──────────────────────────┘
```

"Shell-wise photoelectric calculation"

```
        ┌─────┐ ┌────────────┐     yes
        │ i=1 │ │i>nne(medium)│──────────────►
        │i=i+1│ │            │
        └─────┘ └────────────┘
                      │ no
                      ▼
              ┌──────────────────┐
              │ iz = zelem(medium,i)│
              └──────────────────┘
                      │
                      ▼
                  ╱ eigk ╲        yes
                 ╱  .le.  ╲──────────► ┌──────────┐
                 ╲eedge(1,iz)╱         │crosk(i)=0.0│
                  ╲       ╱            └──────────┘
                      │ no
                      ▼
      ┌────────────────────────────────────────┐
      │ crosk(i) = exp(pm0(1,iz) + pm1(1,iz)*pholk +│
      │    pm2(1,iz)*pholk2 + pm3(1,iz)*pholk3) │
      └────────────────────────────────────────┘
                      │
                      ▼
                  ╱pm0(2,iz).eq.0╲   yes
                 ╱    .or.       ╲──────► ┌──────────┐
                 ╲eigk .le. eedge(2,iz)╱  │crosl1(i) = 0.│
                  ╲             ╱         └──────────┘
                      │ no
                      ▼
      ┌────────────────────────────────────────┐
      │ crosl1(i) = exp(pm0(2,iz) + pm1(2,iz)*pholk +│
      │    pm2(2,iz)*pholk2 + pm3(2,iz)*pholk3) │
      └────────────────────────────────────────┘
                      │
                      ▼
                  ╱pm0(3,iz).eq.0╲   yes
                 ╱    .or.       ╲──────► ┌──────────┐
                 ╲eigk .le. eedge(3,iz)╱  │crosl2(i) = 0.│
                  ╲             ╱         └──────────┘
                      │ no
                      ▼
      ┌────────────────────────────────────────┐
      │ crosl2(i) = exp(pm0(3,iz) + pm1(3,iz)*pholk +│
      │    pm2(3,iz)*pholk2 + pm3(3,iz)*pholk3) │
      └────────────────────────────────────────┘
                      │
                      ▼
                  ╱pm0(4,iz).eq.0╲   yes
                 ╱    .or.       ╲──────► ┌──────────┐
                 ╲eigk .le. eedge(4,iz)╱  │crosl3(i) = 0.│
                  ╲             ╱         └──────────┘
                      │ no
                      ▼
      ┌────────────────────────────────────────┐
      │ crosl3(i) = exp(pm0(4,iz) + pm1(4,iz)*pholk +│
      │    pm2(4,iz)*pholk2 + pm3(4,iz)*pholk3) │
      └────────────────────────────────────────┘
                      │
                      ▼

   1                  2                              3
```

278

1   2   3

eigk
.le.
embind(iz)

yes → tcros(i) = 0.

no

pm0(5,iz)
.eq.
0.

yes → crosm(i) = 0.

no

crosm(i) = exp(pm0(5,iz) + pm1(5,iz)*pholk +
pm2(5,iz)*pholk2 + pm3(5,iz)*pholk3)

tcros(i) = crosk(i) + crosl1(i) + crosl2(i) +crosl3(i) + crosm(i)
bshk(i) = crosk(i)/tcros(i)
bshl1(i) = (crosk(i) + crosl1(i))/tcros(i)
bshl2(i) = (crosk(i) + crosl1(i) + crosl2(i))/tcros(i)
bshl3(i) = (tcros(i) - crosm(i))/tcros(i)

tcros(i) = tcros(i)*pz(medium,i)
total = total + tcros(i)

total
.eq.
0.0

"below M-edge"
yes → edep = eig

iarg=4

iausfl(iarg+1)
. ne.
0

yes → call
ausgab(iarg)

no

e(np) = 0.

no

nne(medium)
.eq.
1

"element"
yes → iz = zelem(medium,1)
noel = 1

no   "compound
or
mixture"

i=1
i=i+1   i>nne(medium)-1   yes

no

i
.eq.
1

yes → pbran(i) = tcros(i)/total

no

pbran(i) = pbran(i-1) + tcros(i)/total

call
randomset (rnnow)

i=1
i=i+1   i>nne(medium)-1   yes

no

4   5   6   7   8

279

```
                                                        10      11
                          ┌─────────┐
                          │  nxray  │    no
                          │  .ge.   │ ──────────────────────────►
                          │   1     │
                          └─────────┘
                               │ yes
                               ▼
              ┌──────────┬──────────────┐  yes
              │ iphot=1  │              │ ─────────►
      ┌──────►│iphot=iphot+1│ iphot>nxray │
      │       └──────────┴──────────────┘
      │                    │ no
      │                    ▼
      │       ┌────────────────────────┐
      └───────│ edep = edep - exray(iphot) │
              └────────────────────────┘
                               │
                               ▼
                          ┌─────────┐
                          │ nauger  │    no
                          │  .ge.   │ ──────────────────────────►
                          │   1     │
                          └─────────┘
                               │ yes
                               ▼
              ┌──────────┬──────────────┐  yes
              │ ielec=1  │              │ ─────────►
      ┌──────►│ielec=ielec+1│ ielec>nauger │
      │       └──────────┴──────────────┘
      │                    │ no
      │                    ▼
      │       ┌────────────────────────┐
      └───────│ edep = edep - eauger(ielec) │
              └────────────────────────┘
                               │
                               ▼
                          ┌─────────┐  yes   ┌─────────┐
                          │  edep   │ ─────► │ edep=0.0 │
                          │  .lt.   │        └─────────┘
                          │   0.    │             │
                          └─────────┘             │
                               │ no               │
                               ▼◄─────────────────┘◄────────────
                          ┌─────────┐
                          │ e(np)=edep │
                          └─────────┘
                               │
                               ▼
                          ┌─────────┐
                          │ iarg=4  │
                          └─────────┘
                               │
                               ▼
                          ┌──────────┐  yes   ┌─────────┐
                          │iausfl(iarg+1)│ ─────► │  call   │
                          │   . ne.   │        │ausgab(iarg)│
                          │    0     │        └─────────┘
                          └──────────┘             │
                               │ no ◄──────────────┘
                               ▼
              ┌────────────────────────┐
              │ iq(np) = -1            │  "Photoelectron (always set up)
              │ e(np) = eig - ebind + RM │
              └────────────────────────┘
                               │
         no                    ▼
  ◄──────────────────────┌──────────┐
                         │iphter(ir(np))│
                         │   .eq.    │
                         │    1     │
                         └──────────┘
                               │ yes      "Select photoelectron direction"
                               ▼
                          ┌─────────┐
                          │ eelec = e(np) │
                          └─────────┘
                               │
         no                    ▼
  ◄──────────────────────┌──────────┐
                         │  eelec   │
                         │  .gt.    │
                         │ ecut(ir(np)) │
                         └──────────┘
                               │ yes
                               ▼
```

Version
051219-1435

```
beta = sqrt((eelec - RM)*(eelec + RM))/eelec
gamma = eelec/RM
alpha = 0.5D0*gamma - 0.5D0 + 1.D0/gamma
ratio = beta/alpha
```

```
call
randomset (rnnow)
```

```
rnpht = 2.D0*rnnow - 1.D0
```

ratio
.le.
0.2D0

no

"Rejection sampling loop"

yes

```
xi = gamma*gamma*(1.D0 + alpha*(sqrt(1.D0 +
      ratio*(2.D0*rnpht + ratio))- 1.D0))
costhe = (1.D0 - 1.D0/xi)/beta
```

```
fkappa = rnpht + 0.5D0*ratio*(1.D0 - rnpht)*(1.D0 + rnpht)
costhe = (beta + fkappa)/(1.D0 + beta*fkappa)
xi = 1.D0/(1.D0 - beta*costhe)
```

```
sinth2 = max(0.D0,(1.D0 - costhe)*(1.D0 + costhe))
```

```
call
randomset (rnnow)
```

no

rnnow
.le.
0.5D0*(1.D0 + gamma)*sinth2*xi/gamma

yes

```
sinthe = sqrt(sinth2)
```

```
call
uphi(2,1)
```

nauger
.ne.
0

no

yes

"Set up Auger electrons"

| ielec=1 | ielec>nauger | yes |
| ielec=ielec+1 | | |

no

```
np = np + 1
e(np) = eauger(ielec) + RM
iq(np) = -1
```

```
call
randomset (rnnow)
```

```
costhe = 2.D0*rnnow - 1.D0
sinthe = sqrt(1.D0 -costhe*costhe)
u(np) = 0.
v(np) = 0.
w(np) = 1.D0
```

```
call
uphi(2,1)
```

```
x(np) = x(np-1)
y(np) = y(np-1)
z(np) = z(np-1)
ir(np) = ir(np-1)
wt(np) = wt(np-1)
dnear(np) = dnear(np-1)
latch(np) = latch(np-1)
k1step(np) = 0.
k1init(np) = 0.
k1rsd(np) = 0.
```

nxray .ne. 0   no

"Set up fluorescent photons"   yes

```
iphot=1
iphoto=iphot+1
```
iphot>nxray   yes

no

```
np = np + 1
e(np) = exray(iphot)
iq(np) = 0
```

```
call
randomset (rnnow)
```

```
costhe = 2.D0*rnnow - 1.D0
sinthe = sqrt(1.D0 - costhe*costhe)
u(np) = 0.
v(np) = 0.
w(np) = 1.D0
```

```
call
uphi(2,1)
```

```
x(np) = x(np-1)
y(np) = y(np-1)
z(np) = z(np-1)
ir(np) = ir(np-1)
wt(np) = wt(np-1)
dnear(np) = dnear(np-1)
latch(np) = latch(np-1)
k1step(np) = 0.
k1init(np) = 0.
k1rsd(np) = 0.
```

```
return
(to photon)
```

283

```
          ┌──────────────┐
         ╱  subroutine    ╲
        ╱    photon        ╲
        ╲                  ╱
         ╲────────────────╱
                │
                ▼
      ┌────────────────────┐
      │ iphoton = iphoton +1│
      └────────────────────┘
                │
                ▼
      ┌────────────────────┐
      │ ircode = 1          │
      │ eig = e(np)         │
      │ irl = ir(np)        │
      │ medium = med(irl)   │
      └────────────────────┘
                │
                ▼
             ◇ eig
             .le.        yes ──────────────────────►
             pcut(irl) ◇
                │ no
                ▼
      ┌────────────────────┐
      │ gle = log(eig)      │
      └────────────────────┘
                │
                ▼
      ┌────────────────────┐
      │ call                │
      │ randomset (rnnow)   │
      └────────────────────┘
                │
                ▼
             ◇ rnnow
             .eq.    yes ──►  ┌──────────────┐
             0. ◇             │ rnnow = 1.E-30│
                │ no          └──────────────┘
                ▼
      ┌────────────────────┐
      │ dpmfp = -log(rnnow) │
      └────────────────────┘
                │
                ▼
             ◇ cexptr
             .ne.    no ──────────────►
             0. ◇
                │ yes
    "exponential
    transform"
                ▼
             ◇ w(np)
             .gt.    no ──────────────►
             0. ◇
                │ yes
                ▼
  ┌──────────────────────────────────┐
  │ temp = cexptr*w(np)              │
  │ bexptr = 1./(1. - temp)          │
  │ dpmfp = dpmfp*bexptr             │
  │ wt(np) = wt(np)*bexptr*exp(-dpmfp*temp)│
  └──────────────────────────────────┘
                │
                ▼
      ┌────────────────────┐
      │ irold = ir(np)      │
      └────────────────────┘
                │
                ▼
             ◇ medium
             .ne.    no ──────────────►
             0 ◇
                │ yes
                ▼
  ┌──────────────────────────────────┐
  │ lgle = ge1(medium)*gle + ge0(medium)│
  │ iextp=0                           │
  └──────────────────────────────────┘
                │
                ▼
```

"NEW-ENERGY loop"

"NEW-MEDIUM loop"

1   2   3   4   5

284

3 4 5

```
                                    ┌──────────┐
                                    │   eig    │  no
                                    │  .lt.    ├──────────────────────────┐
                                    │  0.15    │                          │
                                    └────┬─────┘                          │
                                         │ yes                            │
                              ┌──────────┬──────────────┐   yes           │
                              │ iij=1    │ iij>nedgb     ├───────────────┐ │
                              │ iij=iij+1│ (medium)      │               │ │
                              └──────────┴──────┬────────┘               │ │
                                                │ no                     │ │
                                    ┌──────────┐                         │ │
                                    │ledgb(iij,│  no                     │ │
                                    │medium)   ├────────────┐            │ │
                                    │.eq. lgle │            │            │ │
                                    └────┬─────┘            │            │ │
                                         │ yes              │            │ │
                                    ┌──────────┐            │            │ │
                                    │edgb(iij, │  no   ┌──────────┐      │ │
                                    │medium)   ├──────►│iextp = -1│      │ │
                                    │.le. eig  │       └────┬─────┘      │ │
                                    └────┬─────┘            │            │ │
                                         │ yes              │            │ │
                                    ┌──────────┐            │            │ │
                                    │iextp = 1 │            │            │ │
                                    └────┬─────┘            │            │ │
```

gmfpr0 = gmfp1(lgle+iextp,medium)*gle +
         gmfp0(lgle+iextp,medium)

```
                                    ┌──────────┐
                                    │ medium   │  yes   ┌──────────────┐
                                    │  .eq.    ├───────►│tstep = vacdst│
                                    │   0      │        └──────────────┘
                                    └────┬─────┘
                                         │ no
                              ┌──────────────────────────┐
                              │ rhof = rhor(irl)/rho(medium)│
                              │ gmfp = gmfpr0/rhof          │
                              └──────────────┬──────────────┘
                                    ┌──────────┐
                                    │iraylr(irl)│  no
                                    │  .eq.    ├──────────────┐
                                    │   1      │              │
                                    └────┬─────┘              │
      "Apply Rayleigh correction"       │ yes                │
                    ┌──────────────────────────────────┐     │
                    │ cohfac = cohe1(lgle+iextp,medium)*gle +│ │
                    │          cohe0(lgle+iextp,medium) │     │
                    │ gmfp = gmfp*cohfac                │     │
                    └──────────────────┬────────────────┘     │
                              ┌──────────────────┐
                              │ tstep = gmfp*dpmfp│
                              └──────────┬────────┘
                              ┌──────────────────┐
                              │ irnew = ir(np)   │
                              │ idisc = 0        │
                              │ ustep = tstep    │
                              └──────────┬────────┘
                                    ┌──────────┐
                                    │  ustep   │  yes   ┌──────────┐
                                    │  .gt.    ├───────►│  call    │
                                    │ dnear(np)│        │  howfar  │
                                    └────┬─────┘        └────┬─────┘
                                         │ no                │
```

6 7 8 9 10

"NEW-ENERGY loop"

"NEW-MEDIUM loop"

"PHOTON-TRANSPORT loop"

9

idisc
.gt.
0 →yes→ 10

no

edep = 0.

iarg = 0

iausfl(iarg+1)
.ne.
0 →yes→ call ausgab(iarg)

no

x(np) = x(np) + u(np)*ustep
y(np) = y(np) + v(np)*ustep
z(np) = z(np) + w(np)*ustep
dnear(np) = dnear(np) - ustep

medium
.ne.
0 →no→

yes

dpmfp = max(0.D0,dpmfp - ustep/gmfp)

irold = ir(np)
medold = medium

irnew
.ne.
irold →no→

yes

ir(np) = irnew
irl = irnew
medium = med(irl)

iarg=5

iausfl(iarg+1)
. ne.
0 →yes→ call ausgab(iarg)

no

eig
.le.
pcut(irl) →yes→

no

idisc
.lt.
0 →yes→

no

edep = eig

iarg=3

"NEW-ENERGY loop"

iausfl(iarg+1) .ne. 0 → yes → call ausgab(iarg)

no

call pair

iarg=16

iausfl(iarg+1) . ne. 0 → yes → call ausgab(iarg)

no

gbr2 = gbr21(lgle+iextp,medium)*gle + gbr20(lgle+iextp,medium)

rnnow .lt. gbr2 → no

yes "Compton scattering"

iarg=17

iausfl(iarg+1) . ne. 0 → yes → call ausgab(iarg)

no

call compt

iq(np).eq.0 .and. e(np-1).lt.ecut(ir(np-1)) → no

yes

```
iq(np) = iq(np-1)
iq(np-1) = 0
t = e(np); e(np) = e(np-1)
e(np-1) = t;
t = u(np);u(np) = u(np-1)
u(np-1) = t
t = v(np);v(np) = v(np-1)
v(np-1) = t
t = w(np);w(np) = w(np-1)
w(np-1) = t
t = uf(np);uf(np) = uf(np-1)
uf(np-1) = t
t = vf(np);vf(np) = vf(np-1)
vf(np-1) = t
t = wf(np);wf(np) = wf(np-1)
wf(np-1) = t
```

iarg=18

289

27

28

```
iausfl(iarg+1)
. ne.
0
```

yes →

```
call
ausgab(iarg)
```

no

```
ircode = 2
np = np - 1
```

return
(to shower)

290

```
              subroutine
                raylei

                  │
                  ▼
          iraylei = iraylei +1

                  │
                  ▼
               call
            randomset (rnnow)

                  │
                  ▼
  lxxx = rco1(medium)*rnnow + rco0(medium)
  x2 = rsct1(lxxx,medium)*rnnow + rsct0(lxxx,medium)
  q2 = x2*RMSQ/(20.60744*20.60744)
  costhe = 1.-q2/(2.*e(np)*e(np))

                  │
                  ▼
      yes      abs(costhe)
   ◄──────        .gt.
                   1.
                  │ no
                  ▼
       csqthe = costhe*costhe
       rejf = (1. + csqthe)/2.

                  │
                  ▼
               call
            randomset (rnnow)

                  │
                  ▼
      no         rnnow
   ◄──────        .le.
                  rejf
                  │ yes
                  ▼
      sinthe = sqrt(1. - csqthe)

                  │
                  ▼
             lpolar(ir(np))      yes        call
                .eq.        ──────────►   uphi(2,1)
                 0
                  │ no
                  ▼
               br = 1.

                  │
                  ▼
               call
              aphi(br)

                  │
                  ▼
               call
              uphi(3,1)

                  │
                  ▼
              return
            (to photon)
```

```
                    subroutine
                       rk1

            open(17,file='data/k1.dat',',STATUS='old')

                  read(17,*) nmatk1

         i=1
                  i>nmatk1        ─── yes ───→
         i=i+1

                    │ no

            read(17,*) z, z2k1w(i), watot, rhok1(i)

              z2k1w(i) = z2k1w(i) / watot

                 read(17,*) nek1(i)

         j=nek1(i)
                    j<1          ─── yes ───→
         j=j-1

                    │ no

         read(17,*) ek0k1(j,i), slopek1(j,i), bk1(j,i),
                    dlowk1(j,i),k1low(j,i),dhighk1(j,i),k1high(j,i)

         bk1(j,i) = bk1(j,i) - slopek1(j,i) * DLOG(rhok1(i))
         dhighk1(j,i) = dhighk1(j,i) * rhok1(i)
         dlowk1(j,i) = dlowk1(j,i) * rhok1(i)


                     close(17)

         im=1
                  im>nmed        ─── yes ───→
         im=im+1

                    │ no

                   charD(im)
      ─── yes ───    .eq.
                     0.d0

                    │ no

         lcharD = dlog(charD(im)*rhom(im))
         watot = 0.d0
         z2w = 0.d0


    1                    2                    3
```

292

```
            ┌──────────┐
            │   ie=1   │         yes
            ├──────────┤ ie>nne(im) ──────────────────►
            │  ie=ie+1 │
            └──────────┘
                 │ no
                 ▼
  ┌─────────────────────────────────────────────────────┐
  │ z2w = z2w + pz(im,ie) * zelem(im,ie) * (zelem(im,ie) + 1.d0) │
  │ watot = watot + pz(im,ie) * wa(im,ie)               │
  └─────────────────────────────────────────────────────┘
                 │
                 ▼
         ┌─────────────────┐
         │  z2w = z2w / watot │
         └─────────────────┘
                 │
                 ▼
             ╱╲
            ╱  ╲       yes    ┌──────────────┐
           ╱ z2w╲───────────► │ nz2 = 1      │
           ╲ .gt.╱            │ iz2 = nmatk1 │──────►
           ╱z2k1w(nmatk1)     │ zfrac = 0.d0 │
            ╲  ╱              └──────────────┘
             ╲╱
              │ no
              ▼
             ╱╲
            ╱  ╲       yes    ┌──────────────┐
           ╱ z2w╲───────────► │ nz2 = 1      │
           ╲ .lt.╱            │ iz2 = 1      │──────►
           ╱z2k1w(1)╲         │ zfrac = 0.d0 │
            ╲  ╱              └──────────────┘
             ╲╱
              │ no
              ▼
         ┌─────────┐
         │ nz2 = 2 │
         └─────────┘
              │
              ▼
   ┌──────────────────────────┐
   │ call findi               │
   │ (z2k1w,z2w,nmatk1,iz2)   │
   └──────────────────────────┘
              │
              ▼
  ┌──────────────────────────────────────────────────────┐
  │ zfrac = (z2w - z2k1w(iz2)) / (z2k1w(iz2+1) - z2k1w(iz2)) │
  └──────────────────────────────────────────────────────┘
              │
              ▼
            ┌──────────┐
            │   n=1    │        yes
            ├──────────┤ n>nz2 ──────────────────►
            │  n=n+1   │
            └──────────┘
                 │ no
                 ▼
      ┌────────────────────────────┐
      │ idx = iz2 + n - 1          │
      │ eke = ek0k1(nek1(idx),idx) │
      └────────────────────────────┘
                 │
                 ▼
             ╱╲
            ╱  ╲       no    ┌────────────────┐
           ╱ue(im)-RM───────►│ scpeMax(n) = 0.0 │
           ╲ .gt.╱           └────────────────┘
           ╱ eke ╲
            ╲  ╱
             ╲╱
              │ yes
              ▼
 ┌───────────────────────────────────────────────┐
 │ elke = log(eke)                               │
 │ j = eke0(im) + elke * eke1(im)                │
 │ scpeMax(n) = escpw1(j,im)*elke + escpw0(j,im) │
 └───────────────────────────────────────────────┘
              │
              ▼
        ┌──────────────────┐
        │ neke = meke(im)  │
        └──────────────────┘
              │
              ▼
```

293

294

k1ez(m) = k1high(iedx,idx)

k1ez(m) = k1low(iedx,idx)

k1sez(m) = k1low(iedx,idx)

k1ez(m) = DEXP(slopek1(iedx,idx)*
lcharD + bk1(iedx,idx))

k1z(n) = k1ez(1) + frace * (k1ez(2) - k1ez(1))
k1z(n) = extrape * k1z(n)
k1sz(n) = k1sez(1) + frace * (k1sez(2) - k1sez(1))
k1sz(n) = extrape * k1sz(n)

k1new = k1z(1) + zfrac * (k1z(2) - k1z(1))
k1s = k1sz(1) + zfrac * (k1sz(2) - k1sz(1))
k1s = k1s

k1new
.gt.
k1maxe

k1new
.lt.
k1mine

no

no

yes

k1maxe = k1new

yes

k1mine = k1new

k1new*scprat
.gt.
k1maxp

k1new*scprat
.lt.
k1minp

no

no

yes

k1maxp = k1new*scprat

yes

k1minp = k1new*scprat

k1s
.lt.
k1mine

no

yes

k1mine = k1s

k1s*scprat
.lt.
k1minp

no

yes

k1minp = k1s*scprat

j
.gt.
1

no

yes

delke = elke - elkeold
ekini1(j,im) = (k1new - k1old) / delke
ekini0(j,im) = (k1old * elke - k1new * elkeold) / delke
pkini1(j,im) = ekini1(j,im) * scprat
pkini0(j,im) = ekini0(j,im) * scprat
ek1s1(j,im) = (k1s - k1sold) / delke
ek1s0(j,im) = (k1sold * elke - k1s * elkeold) / delke
pk1s1(j,im) = ek1s1(j,im) * scprat
pk1s0(j,im) = ek1s0(j,im) * scprat

```
elkeold = elke
k1old = k1new
k1sold = k1s
```

```
ekini1(1,im) = ekini1(2,im)
ekini0(1,im) = ekini0(2,im)
pkini1(1,im) = pkini1(2,im)
pkini0(1,im) = pkini0(2,im)
ekini1(neke,im) = ekini1(neke-1,im)
ekini0(neke,im) = ekini0(neke-1,im)
pkini1(neke,im) = pkini1(neke-1,im)
pkini0(neke,im) = pkini0(neke-1,im)
ek1s1(1,im) = ek1s1(2,im)
ek1s0(1,im) = ek1s0(2,im)
pk1s1(1,im) = pk1s1(2,im)
pk1s0(1,im) = pk1s0(2,im)
ek1s1(neke,im) = ek1s1(neke-1,im)
ek1s0(neke,im) = ek1s0(neke-1,im)
pk1s1(neke,im) = pk1s1(neke-1,im)
pk1s0(neke,im) = pk1s0(neke-1,im)
```

(10)

```
j=1
j=j+1      j>nreg
```

yes

no

k1Lscl(j).gt.0.d0
.and.
k1Hscl(j).gt.0.d0

no

k1Lscl(j)
.ne.
0.d0

yes

```
im = med(j)
c2 = (k1Lscl(j) - k1Hscl(j)) /
    dlog( (ae(im) - RM)/(ue(im) - RM) )
c1 = k1Hscl(j) - dlog(ue(im) - RM) * c2
k1Lscl(j) = c1
k1Hscl(j) = c2
```

```
k1Lscl(j) = 0.d0
```

return
(to hatch)

subroutine
rmsfit

openMS = .false.
useGS = .false.

k=1
k=k+1 | k>nmed — yes

no

charD(k)
.eq.
0.d0 — yes → openMS = .true.

no

useGSD(k)
.eq.
0 — yes → nmsgrd(k) = 1
msgrid(1,k) = 0.d0

no

useGS = .true.

.not.
(useGS.or.
openMS) — true → return

false

useGS — false

true

k=1
k=k+1 | k>nmed — yes

no

useGSD(k)
.eq.
0.d0 — no

yes

write(6,1001) k

useGSD(k) = 1

GSFile = .false.
doGS=.false.

open(17,file='gsdist.dat',STATUS='old,ERR=14')

GSFile = .true.

13 → close(17)

open(17,file='pgs5job.msfit',STATUS='old')

1

2

297

"end of file; go to 13"

read(17,*,end=13) nfmeds

j=1
j=j+1 | j>nfmeds — yes

no

read(17,'(72a1)',end=1000) buffer
read(17,*) hasGS, charD0, efrch0, efrcl0, ue0, ae0

k=1
k=k+1 | k>nmed — yes

no

ib=1
ib=ib+1 | ib>lmdn

no

il = lmdl + ib

yes ← buffer(il) .ne. media(ib,k) → no

no — ib .eq. lmdn

yes

useGS .and. hasGS.eq.0 — yes → write(6,1002) k

doGS = .true.

no

charD(k) .eq. 0.d0 — no

yes

charD0 .ne. 0.d0 — no → write(6,1005) k, efrch0, efrcl0

yes

useGS — false → write(6,1004) k

(stop)

true

write(6,1003) k, charD0

useGS .and. (ue0.lt.ue(j) .or. ae0.gt.ae(j)) — true → write(6,1006) k, ae0, ue0, ae(j), ue(j)

doGS = .true.

false

3

298

3

.not.useGS —true→ close(17) → return

false

| j=1 | j>nreg |
| j=j+1 | |
→ yes

no

k = med(j)

k
.ne.
0 —no→

yes

rhom(k)
.ne.
rhor(j) —no→

yes

write(6,1007) j,k → stop

| j=1 | j>nreg |
| j=j+1 | |
→ yes

no

(k1Hscl(j) + k1Lscl(j))
.gt.
0.d0 —no→

yes

k1Hscl(j) = 0.d0
k1Lscl(j) = 0.d0

write(6,1008) j

GSFile
.and.
.not.doGS —false→

true

read(17,'(72a1)',end=1000) buffer
read(17,'(72a1)',end=1000) buffer

| i=1 | i>NK1 |
| i=i+1 | |
→ yes

no

read(17,*) idummy, k1grd(1,i), k1grd(2,1)

4   5

299

```
                            4                           5
                            │                           │
                            ▼                           │
                  abs(k1mine-k1grd(1,1))/k1mine.gt.1.d-6 │
                           .or.                          │
                  abs(k1minp-k1grd(2,1))/k1minp.gt.1.d-6 │
  "go to 30"              .or.                           │
  ◄─────────  abs(k1maxe-k1grd(1,NK1))/k1maxe.gt.1.d-6   │
     no                   .or.                           │
                  abs(k1maxp-k1grd(2,NK1))/k1maxp.gt.1.d-6│
                            │                            │
                            │ yes                        │
                            ▼                            │
                      write(6,1009)                      │
                            │                            │
                      doGS = .true.                      │
                            │◄───────────────────────────┘
                            ▼
                         doGS ──── false ──────────────────►
                            │
                            │ true
                            ▼
                        close(17)
                            │
                            ▼
         open(UNIT=17,FILE='gsdist.dat',STATUS='unknown')
                            │
                            ▼
                          call
                         elastino
                            │
                            ▼
                        close(17)
                            │
                            ▼
            open(UNIT=17,FILE='gsdist.dat',STATUS='old')
                            │
                            ▼
               j=1
               j=j+1   j>nfmeds*2+1  ──── yes ──┐
                            │ no                │
                            ▼                   │
          read(17,'(72a1)',end=1000) buffer     │
                            │◄──────────────────┘
                            ▼
                  read(17,'(72a1)') buffer
                  read(17,'(72a1)') buffer
                            │
                            ▼
               i=1
               i=i+1      i>NK1  ──── yes ──┐
                            │ no            │
                            ▼               │
          read(17,*) idummy, k1grd(1,i), k1grd(2,i)
                            │◄──────────────┘
                            ▼
                          (30)
                            │
                            ▼
            dk1log(1) = dlog(k1grd(1,2)/k1grd(1,1))
            dk1log(2) = dlog(k1grd(2,2)/k1grd(2,1))
                            │
                            ▼
                         nm = 0
                            │
                            ▼
                            6
```

6

k=1	
k=k+1	k>nmed

yes

no

lok(k) = 0

5

read(17,'(72a1)',end=1000) buffer

ib=1	
ib=ib+1	ib>lmdl

yes

no

buffer(ib)
.ne.
mdlabl(ib)

no

yes

k=1	
k=k+1	k>nmed

yes

no

ib=1	
ib=ib+1	ib>lmdn

yes

no

il = lmdl + ib

buffer(il)
.ne.
media(ib,k)

yes

no

ib
.eq.
lmdn

no

yes

lok(k)
.ne.
0

yes

no

| lok(k) = 1 |
| nm = nm + 1 |

7

8

301

```
7                            8

            read(17,'(72a1)') buffer
            read(17,'(72a1)') buffer
            read(17,*) nmsgrd
            read(17,'(72a1)') buffer
            read(17,*) initde
            read(17,'(72a1)') buffer
            read(17,*) nmsdec
            read(17,'(72a1)') buffer
            read(17,*) jskip
            read(17,'(72a1)') buffer
            read(17,*) neqp
            read(17,'(72a1)') buffer
            read(17,*) neqa

            jskip = jskip - 1

            iprt=1           ┌─────────┐  yes
            ─────────        │ iprt>2  │ ──────────────►
            iprt=iprt+1      └─────────┘

                              │ no

            read(17,'(72a1)') buffer

            i=1       ┌──────────────┐  yes
            ───────   │ i>nmsgrd(k)  │ ──────────────►
            i=i+1     └──────────────┘

                       │ no

            read(17,*) msgrid(i,k)

            ik1=1       ┌──────────┐  yes
            ─────────   │ ik1>NK1  │ ──────────────►
            ik1=ik1+1   └──────────┘

                         │ no

            read(17,'(72a1)') buffer
            read(17,*) pnoscat(iprt,i,ik1,k)
            read(17,'(72a1)') buffer

            eamu(1,iprt,i,ik1,k) = 0.0

            j=1       ┌──────────────────────┐  yes
            ───────   │ j>neqp(k)+neqa(k)-1   │ ──────►
            j=j+1     └──────────────────────┘

                       │ no

            read(17,*) dummy, eamu(j+1,iprt,i,ik1,k),
                       ebms(j,iprt,i,ik1,k),
                       eetams(j,iprt,i,ik1,k)

            ebms(iprt,i,ik1,j,k) = ebms(iprt,i,ik1,j,k) * .25d0

            read(17,'(72a1)') buffer

            j=1       ┌──────────────────┐  yes
            ───────   │ j>neqa(k) + 1    │ ──────────►
            j=j+1     └──────────────────┘

                       │ no

9     10    11    12         13         14       15    16    17    19
```

14

read(17,*) ecdf(j,iprt,i,ik1,k)

13

15

ecdf(neqa(k)+1,iprt,i,ik1,k) = 1.d0

12

16

11

17

10

19

9

nm .ge. nmed

no

yes

nmed .eq. 1

yes → write(6,5001)

no → write(6,5002) nmed

close(17)

return

1000

write(6,5003)
write(6,5004)

k=1
k=k+1

k>nmed

yes

no

lok(k) .ne. 1

no

yes

write(6,'(24a1)') (media(i,k),i=1,lmdn)

close(17)

stop

303

```
subroutine shower
(iqi,ei,xi,yi,zi,ui,vi,wi,iri,wti)
```

ishower = ishowe +1

```
np = 1          ;dneari = 0.0
iq(1) = iqi     ;e(1) = ei
u(1) = ui       ;v(1) = vi
w(1) = wi       ;x(1) = xi
y(1) = yi       ;z(1) = zi
ir(1) = iri     ;wt(1) = wti
dnear(1) = dneari
latch(1) = latchi
deresid = 0.d0;deinitial = 0.d0
denstep = 0.d0;k1step(1) = 0.d0
k1init(1) = 0.d0;k1rsd(1) = 0.d0
```

no ← iqi .eq. 2

yes

"π0-decay"

ei**2 .le. PI0MSQ → yes → write(6,5002) nmed → stop

no

call randomset (rnnow)

```
dcsth = rnnow
dei = ei
dpi = dsqrt(dei*dei - PI0MSQ)
deg = dei + dpi*dcsth
dpgl = dpi + dei*dcsth
dcosth = dpgl/deg
costhe = dcosth
sinthe = dsqrt(1.d0 - dcosth*dcosth)
iq(1) = 0
e(1) = deg/2.
```

call uphi(2,1)

```
np = 2
deg = dei - dpi*dcsth
dpgl = dpi - dei*dcsth
dcosth = dpgl/deg
costhe = dcosth
sinthe = -dsqrt(1.d0 - dcosth*dcosth)
iq(2) = 0
e(2) = deg/2.
```

1

2

1   2

call
uphi(3,2)

ircode = -1

"e+,e- or photon"

iq(np)
.eq.
0

yes

no

call
electr(ircode)

ircode
.eq.
2

yes

no

call
photon(ircode)

ircode
.eq.
2

no

yes

np
.gt.
0

yes

return

```
subroutine
uphi(ientry,lvl)
```

iuphi = iuphi + 1

iarg=21

iausfl(iarg+1) .ne. 0 — yes → call ausgab(iarg)

no

ientry .eq. 1 — yes →
```
sinthe = sin(theta)
cthet = PI5D2 - theta
costhe = sin(cthet)
```

no

ientry .eq. 2 — yes →
```
call
randomset (rnnow)
```

no

```
phi = rnnow*TWOPI
sinphi = sin(phi)
cphi = PI5D2 - phi
cosphi = sin(cphi)
```

ientry .eq. 3 — no → write(6,100) ientry,lvl → stop

yes

lvl .eq. 4 — yes →
```
usav = u(np)
vsav = v(np)
wsav = w(np)
```

no

lvl .eq. 6 — yes →
```
usav = u(np-1)
vsav = v(np-1)
wsav = w(np-1)
```

no

lvl .eq. 5 — yes →
```
x(np) = x(np-1)
y(np) = y(np-1)
z(np) = z(np-1)
ir(np) = ir(np-1)
wt(np) = wt(np-1)
dnear(np) =
dnear(np-1)
latch(np) =
latch(np-1)
k1step(np) = 0.d0
k1init(np) = 0.d0
k1rsd(np) = 0.d0
```

no

sinps2 = usav*usav + vsav*vsav

sinps2 .lt. 1.e-20 — yes →
```
u(np) = sinthe*cosphi
v(np) = sinthe*sinphi
w(np) = wsav*costhe
```

no →
```
sinpsi = sqrt(sinps2)
us = sinthe*cosphi
vs = sinthe*sinphi
sindel = vsav/sinpsi
cosdel = usav/sinpsi
u(np) = wsav*cosdel*us - sindel*vs + usav*costhe
v(np) = wsav*sindel*us + cosdel*vs + vsav*costhe
w(np) = -sinpsi*us + wsav*costhe
```

1

```
        1
        │
        ▼
   ┌─────────┐
   │ iarg=22 │
   └─────────┘
        │
        ▼
       ╱─╲              yes   ┌──────────────┐
      ╱   ╲ iausfl(iarg+1) ─────▶│ call         │
      ╲   ╱   . ne.              │ ausgab(iarg) │
       ╲ ╱      0                └──────────────┘
        │                               │
     no │◀─────────────────────────────┘
        ▼
       ╱───────╲
      ╱         ╲
     │  return   │
      ╲         ╱
       ╲───────╱
```

```
                    ┌──────────────────┐
                    │    subroutine    │         Version
                    │ randomset(rndum) │         051219-1435
                    └──────────────────┘
                             │
                             ▼
              ┌────────────────────────────────┐
              │ uni = seeds(j24) - seeds(i24) - carry │
              └────────────────────────────────┘
                             │
                             ▼
 ┌──────────────┐   yes    ╱ uni ╲    no    ┌───────────┐
 │ uni = uni + 1.0 │◄──────│  .lt.  │──────►│ carry = 0. │
 │ carry = twom24  │        ╲  0.  ╱        └───────────┘
 └──────────────┘           ╲    ╱               │
         │                    ▼                   │
         └────────────────────┼───────────────────┘
                              ▼
                  ┌────────────────────┐
                  │ seeds(i24) = uni   │
                  │ i24 = next(i24)    │
                  │ j24 = next(j24)    │
                  │ rndum = uni        │
                  └────────────────────┘
                              │
                              ▼
        no              ╱   uni    ╲
 ┌──────────────────────│   .lt.    │
 │                       ╲ twom12  ╱
 │                        ╲        ╱
 │                         │ yes
 │                         ▼
 │        ┌───────────────────────────────────┐
 │        │ rndum = rndum + twom24*seeds(j24)  │
 │        └───────────────────────────────────┘
 │                         │
 │                         ▼
 │                   ╱ rndum ╲   yes    ┌─────────────────┐
 │                  │  .eq.   │────────►│ rndum = twom48   │
 │                  ╲   0.   ╱          └─────────────────┘
 │                   ╲     ╱                    │
 │                    │ no                      │
 └────────────────────┼────────────────────────┘
                      ▼
            ┌──────────────────┐
            │ in24 = in24 + 1  │
            └──────────────────┘
                      │
                      ▼
                ╱  in24  ╲    no
               │  .eq.    │─────────────────────────────┐
                ╲   24   ╱                               │
                 ╲      ╱                                │
                  │ yes                                  │
                  ▼                                      │
       ┌──────────────────────────┐                     │
       │ in24 = 0                 │                     │
       │ kount = kount + nskip    │                     │
       └──────────────────────────┘                     │
                  │                                      │
                  ▼                                      │
       ┌─────────┬─────────────┐   yes                  │
       │ isk=1   │ isk>nskip   │──────────────────────┐ │
    ┌─►│ isk=isk+1│            │                      │ │
    │  └─────────┴─────────────┘                      │ │
    │            │ no                                  │ │
    │            ▼                                     │ │
    │  ┌────────────────────────────────┐             │ │
    │  │ uni = seeds(j24) - seeds(i24) - carry │       │ │
    │  └────────────────────────────────┘             │ │
    │            │                                     │ │
    │            ▼                                     │ │
    │       ╱  uni  ╲   yes   ┌──────────────┐         │ │
    │      │  .lt.   │───────►│ uni = uni + 1.0 │       │ │
    │      ╲   0.   ╱         │ carry = twom24  │       │ │
    │       ╲     ╱           └──────────────┘         │ │
    │        │ no                   │                  │ │
    │        ▼                      │                  │ │
    │  ┌───────────┐                │                  │ │
    │  │ carry = 0. │               │                  │ │
    │  └───────────┘                │                  │ │
    │        │                      │                  │ │
    │        ▼                      │                  │ │
    │  ┌────────────────────┐       │                  │ │
    │  │ seeds(i24) = uni   │◄──────┘                  │ │
    └──│ i24 = next(i24)    │                          │ │
       │ j24 = next(j24)    │                          │ │
       └────────────────────┘                          │ │
                  │                                     │ │
                  ▼                                     │ │
                  ◄──────────────────────────────────────┘
                  │
                  ▼
                  1
```

1

```
kount = kount + 1
```

kount
.ge.
igiga

no

yes

```
mkount = mkount + 1
kount = kount - igiga
```

return

subroutine
rluxinit

tisdext = 0

i=1	i>25
i=i+1	

yes

no

tisdext = tisdext + isdext(i)

tisdext
.ne.
0

yes

call
rluxin

no

call
rluxgo

return

```
                    ╔═══════════════╗                    Version
                    ║  subroutine   ║                    051219-1435
                    ║    rluxgo     ║
                    ╚═══════════════╝
                            │
                            ▼
                      luxlev.le 0        yes      ┌──────────────────────────────────┐
                         .or.        ──────────▶  │ write (6,'(a,i7)')               │
                   luxlev.gt.maxlev                │  ' illegal ranlux level in       │
                            │                      │    rluxgo: ',luxlev              │
                         no │                      └──────────────────────────────────┘
                            ▼                                        │
                 ┌────────────────────┐                             ▼
                 │ nskip = ndskip(luxlev) │                     ┌────────┐
                 └────────────────────┘                         │  stop  │
                            │                                   └────────┘
                            ▼
        ┌──────────────────────────────────────────────────┐
        │ write(6,'(a,i2,a,i4)') ' ranlux luxury level set  │
        │   by rluxgo :', luxlev,'    p=', nskip+24          │
        └──────────────────────────────────────────────────┘
                            │
                            ▼
                    ┌──────────────┐
                    │   in24 = 0    │
                    └──────────────┘
                            │
                            ▼
                       inseed           yes      ┌──────────────────────────────────┐
                        .lt.        ──────────▶  │ write (6,'(a)') ' Illegal          │
                         0                        │  initialization in rluxgo,         │
                            │                     │  negative input seed'              │
                         no │                     └──────────────────────────────────┘
                            ▼                                        │
                                                                     ▼
                       inseed       yes   ┌──────────────────┐   ┌────────┐
┌──────────────────┐    .gt.    ◀────────│ jseed = inseed   │   │  stop  │
│ jseed = inseed   │     0                └──────────────────┘   └────────┘
└──────────────────┘      │
         ◀────────────────┘
         │             no │
         ▼                ▼
┌──────────────────────────┐   ┌──────────────────┐
│ write(6,'(a,i12)')       │   │  jseed = jsdflt   │
│  ' ranlux initialized by │   └──────────────────┘
│  rluxgo from seed', jseed │            │
└──────────────────────────┘            ▼
         │            ┌──────────────────────────────────┐
         │            │ write(6,'(a)')' ranlux initialized │
         │            │  by rluxgo from default seed'      │
         │            └──────────────────────────────────┘
         │                          │
         └──────────────┐   ┌───────┘
                        ▼   ▼
                 ┌──────────────────┐
                 │ inseed = jseed   │
                 │ twom24 = 1.      │
                 └──────────────────┘
                        │
                        ▼
              ┌─────┬───────┐          yes
              │ i=1 │  i>24 │  ──────────────────────────┐
              │i=i+1│       │                            │
              └─────┴───────┘                            │
                        │ no                             │
                        ▼                                │
       ┌──────────────────────────────────────────┐     │
       │ twom24 = twom24 * 0.5                      │     │
       │ k = jseed/53668                            │     │
       │ jseed = 40014*(jseed-k*53668) -k*12211     │     │
       └──────────────────────────────────────────┘     │
                        │                                │
                        ▼                                │
                      jseed      yes  ┌──────────────────┐│
  ┌──────────────────┐ .lt.  ◀───────│ jseed = jseed+icons││
  │ jseed = jseed+icons│  0           └──────────────────┘│
  └──────────────────┘      │                             │
         │               no │                             │
         ▼                  ▼                             │
  ┌──────────────────────────────────────┐               │
  │ iseeds(i) = mod(jseed,itwo24)         │ ◀─────────────┘
  └──────────────────────────────────────┘
                        │
                        ▼
         ┌──────────────────────────────┐
         │ itwom12 = twom24 * 4096.      │
         └──────────────────────────────┘
                        │
                        ▼
                        1
```

```
                    ┌──1
                    ▼
              ┌──────┬───────┐
          ┌──►│ i=1  │       │          yes
          │   ├──────┤ i>24  ├───────────────────►
          │   │i=i+1 │       │
          │   └──────┴───────┘
          │            │ no
          │            ▼
          │   ┌──────────────────────────────┐
          └───┤ seeds(i) = real(iseeds(i))*twom24 │──►
              │ next(i) = i-1                  │
              └──────────────────────────────┘
                             │
                             ▼
                    ┌──────────────────┐
                    │ next(1) = 24     │
                    │ i24 = 24         │
                    │ j24 = 10         │
                    │ carry = 0.       │
                    └──────────────────┘
                             │
            yes              ▼
  ┌───────────────┐    ╱ seeds(24) ╲
  │jscarry = twom24│◄──╲   .eq.    ╱
  └───────────────┘    ╲    0.    ╱
          │             ╲────╱
          │                │ no
          ▼                ▼
          └──────────────►╱ kount+mkount ╲   no
                          ╲    .ne.      ╱───►
                          ╲     0       ╱
                           ╲────╱
                              │ yes
                              ▼
              ╱──────────────────────────────────────────╲
              │ write(6,'(a,i,a,i)') ' Restarting ranlux with kount = ',│
              │    kount,' and mkount = ',mkount          │
              ╲──────────────────────────────────────────╱
                              │
                              ▼
              ┌──────────┬──────────────────┐
          ┌──►│iouter=1  │                  │   yes
          │   ├──────────┤ iouter>mkount+1  ├────────►
          │   │iouter=   │                  │
          │   │iouter+1  │                  │
          │   └──────────┴──────────────────┘
          │                   │ no
          │                   ▼
          │            ┌──────────────┐
          │            │ jinner = igiga│
          │            └──────────────┘
          │                   │
          │        yes        ▼
          │   ┌──────────┐ ╱ iouter ╲
          │   │inner=kount│◄╲  .eq.  ╱
          │   └──────────┘ ╲mkount+1╱
          │        │        ╲────╱
          │        │           │ no
          │        ▼           ▼
          │        └─────────►┌──────┬──────────┐
          │            ┌─────►│isk=1 │          │  yes
          │            │      ├──────┤ isk>inner├──────►
          │            │      │isk=  │          │
          │            │      │isk+1 │          │
          │            │      └──────┴──────────┘
          │            │           │ no
          │            │           ▼
          │            │   ┌──────────────────────────────┐
          │            │   │ uni = seeds(j24) - seeds(i24) - carry│
          │            │   └──────────────────────────────┘
          │            │           │
          │    yes     │           ▼
          │ ┌──────────┐     ╱  uni  ╲
          │ │uni=uni+1.0│◄───╲  .lt.  ╱
          │ │carry=twom24│    ╲  0.   ╱
          │ └──────────┘     ╲────╱
          │      │              │ no
          │      │              ▼
          │      │      ┌──────────┐
          │      │      │ carry = 0.│
          │      │      └──────────┘
          │      │           │
          │      ▼           ▼
          │      └─────────►┌──────────────┐
          │            ┌────┤ seeds(i24) = uni│
          │            │    │ i24 = next(i24) │
          └────────────┘    │ j24 = next(j24) │
                            └──────────────┘
                                   │
                                   ▼
                                   2
```

312

2

in24 = mod(kount, nskip+24)
in24 = mod(kount, nskip+24)

mkount .gt. 0 —— no ——→

yes

izip = mod(igiga, nskip+24)
izip2 = mkount*izip + in24
in24 = mod(izip2, nskip+24)

in24 .gt. 23 —— yes ——→

no

write (6,'(a/a,3i11,a,i5)') ' Error in RESTARTING with RLUXGO:',' The values', inseed, kount, mkount, ' cannot occur at luxury level', luxlev

stop

rluxset = .true.

return

---

subroutine rluxout

i=1
i=i+1   i>24 —— yes ——→

no

isdext(i) = int(seeds(i)*twop12*twop12)

isdext(25) = i24 + 100*j24 + 10000*in24 + 1000000*luxlev

carry .gt. 0. —— yes ——→ cisdext(25) = -isdext(25)arry = twom24

no

return

313

```
                    ┌──────────────────┐
                    ║   subroutine     ║          Version
                    ║    rluxin        ║          051219-1435
                    └──────────────────┘
                             │
                             ▼
   ┌─────────────────────────────────────────────────────┐
  /  write(6,'(a)') ' full initialization of ranlux with 25 integers:' /
 /   write(6,'(5x,5i12)') isdext                          /
   └─────────────────────────────────────────────────────┘
                             │
                             ▼
                    ┌──────────────────┐
                    │   twom24 = 1.    │
                    └──────────────────┘
                             │
                             ▼
              ┌──────┬──────────┐           yes
              │ i=1  │          │ ─────────────────────►
              │ i=i+1│  i>24    │
              └──────┴──────────┘
                        │ no
                        ▼
              ┌─────────────────────────┐
              │  next(i) = i-1          │
              │  twom24 = twom24 * 0.5  │
              └─────────────────────────┘
                        │
                        ▼
              ┌─────────────────────────┐
              │  next(1) = 24           │
              │  twom12 = twom24 * 4096.│
              └─────────────────────────┘
                        │
                        ▼
              ┌──────┬──────────┐           yes
              │ i=1  │          │ ─────────────────────►
              │ i=i+1│  i>24    │
              └──────┴──────────┘
                        │ no
                        ▼
              ┌─────────────────────────────────┐
              │ seeds(i) = real(isdext(i))*twom24│
              └─────────────────────────────────┘
                        │
                        ▼
                 ┌──────────────┐
                 │  carry = 0.  │
                 └──────────────┘
                        │
                        ▼
                   ◇ isdext(25) ◇    yes    ┌─────────────────┐
                   ◇   .lt.     ◇ ────────► │ carry = twom24  │
                   ◇    0       ◇           └─────────────────┘
                        │ no
                        ▼
              ┌─────────────────────────┐
              │ isd = iabs(isdext(25))  │
              │ i24 = mod(isd,100)      │
              │ isd = isd/100           │
              │ j24 = mod(isd,100)      │
              │ isd = isd/100           │
              │ in24 = mod(isd,100)     │
              │ isd = isd/100           │
              │ luxlev = isd            │
              └─────────────────────────┘
                        │
                        ▼
                   ◇  luxlev  ◇    no    ┌─────────────────────────────────┐
                   ◇   .le.   ◇ ───────► │ write (6,'(a,i5)') ' ranlux illegal luxury │
                   ◇ maxlev   ◇          │        rluxin: ',luxlev          │
                        │ yes            └─────────────────────────────────┘
                        ▼                              │
              ┌─────────────────────────┐              ▼
              │ nskip = ndskip(luxlev)  │           ( stop )
              └─────────────────────────┘
                        │
                        ▼
   ┌─────────────────────────────────────────────────────┐
  /  write (6,'(a,i2)')  ' ranlux luxury level set by    /
 /   rluxin to: ', luxlev                                /
   └─────────────────────────────────────────────────────┘
                        │
                        ▼
              ┌─────────────────────────┐
              │  inseed = -1            │
              │  rluxset = .true.       │
              └─────────────────────────┘
                        │
                        ▼
                 ┌──────────────┐
                 │    return    │
                 └──────────────┘
```

314

```
          ┌─────────────┐
         ╱ subroutine    ╲
        ⟨   rluxout       ⟩
         ╲               ╱
          └──────┬──────┘
                 │
         ┌───────┴───────┬────────┐      yes
         │    i=1         │        │─────────────►
         │    i=i+1       │  i>24  │
         └───────────────┴────┬───┘
                              │ no
                              ▼
  ┌───────────────────────────────────────┐
  │ isdext(i) = int(seeds(i)*twop12*twop12) │
  └───────────────────────────────────────┘

  ┌──────────────────────────────────────────────────────┐
  │ isdext(25) = i24 + 100*j24 + 10000*in24 + 1000000*luxlev │
  └──────────────────────────────────────────────────────┘
                 │
                 ▼
            ╱╲
           ╱   ╲      no
          ╱carry ╲──────────►
          ╲ .gt.  ╱
           ╲ 0.  ╱
            ╲  ╱
             ││ yes
             ▼
  ┌──────────────────────┐
  │ isdext(25) = -isdext(25) │
  └──────────────────────┘
                 │
                 ▼
          ┌─────────────┐
         ╱   return      ╲
        ⟨                 ⟩
         ╲               ╱
          └─────────────┘
```

Appendix B

EGS5 USER MANUAL

Hideo Hirayama and Yoshihito Namito
Radiation Science Center
Advanced Research Laboratory
High Energy Accelerator Research Organization (KEK)
1-1 Oho Tsukuba-shi Ibaraki-ken 305-0801 JAPAN

Alex F. Bielajew and Scott J. Wilderman
Department of Nuclear Engineering and Radiological Sciences
The University of Michigan
2355 Bonisteel Boulevard
Ann Arbor, MI 48109, USA

Walter R. Nelson
Department Associate in the Radiation Physics Group (retired)
Radiation Protection Department
Stanford Linear Accelerator Center
2575 Sand Hill Road Menlo Park, CA 94025, USA

B.1   Introduction

Version 5 of the EGS code system is written exclusively in FORTRAN, marking a departure from the use of the MORTRAN programming language, first introduced with Version 2. To retain some of the functionality and flexibility that MORTRAN provided, EGS5 employs some common extensions of FORTRAN-77, most notably "include" statements, which are used to import identical versions of all the EGS5 `COMMON` blocks into all appropriate subroutines. Users can thus alter the values of parameters set in the `COMMON` block files which are "included," in the various source codes, thus emulating the use of MORTRAN macros in specifying array dimensions. Each of the `COMMON` block files contains the declarations for just one `COMMON` block, with all files containing EGS-related `COMMON` blocks located in a directory named **include** and all PEGS-related files in a directory called **pegscommons**.

Additionally, many of the features and options in EGS4 which were invoked through MORTRAN macro substitutions have been retained in the base shower code in EGS5 and can be"turned on" by user specification of the appropriate flags and parameters.

B.2   General Description of Implementation

As described in Chapter 2 of SLAC-R-730/KEK-2005-8 ("The EGS5 Code System"), to use EGS the user must write a "user code" consisting of a `MAIN` program and subroutines `HOWFAR` and `AUSGAB`. The user defines and controls an EGS5 shower simulation by initializing, tallying, and in some cases altering variables found in `COMMON` blocks shared by user code `MAIN` and a set of four EGS5 subroutines which `MAIN` must call (`BLOCK_SET`, `PEGS5`, `HATCH`, and `SHOWER`). The user can access and manipulate variables located in many additional `COMMON` blocks which are shared by EGS5 subroutines which call the user subroutines `HOWFAR` and `AUSGAB` at points in the simulation specified by the user.

The user's `MAIN` program first calls the EGS5 `BLOCK_SET` subroutine to set default values for variables in EGS5 `COMMON` blocks which are too large to be defined in `BLOCK DATA`. `MAIN` also initializes variables needed by `HOWFAR`, and defines the values of EGS5 `COMMON` block variables corresponding to 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` next calls EGS5 subroutine `PEGS5` (to create basic material data) and then calls EGS5 subroutine `HATCH`, which "hatches" EGS by reading the material data created by PEGS for the media in the given problem. Once the initialization is complete, `MAIN` then calls the EGS5 subroutine `SHOWER`, with each call to `SHOWER` resulting in the simulation of one history (often referred to as a "case"). The arguments to `SHOWER` specify the parameters of the incident particle initiating the cascade. The user subroutine `HOWFAR` is required for modeling the problem geometry (which it does primarily by keeping track of and reporting to EGS) the regions in which the particles lie), while user subroutine `AUSGAB` is typically used to score the results of the simulation.

Table B.1: Variable descriptions for `COMMON` block `BOUNDS`, **include** file **egs5_bounds.f** of the EGS5 distribution.

ECUT	Array of region-dependent charged particle cutoff energies in MeV.
PCUT	Array of region-dependent photon cutoff energies in MeV.
VACDST	Distance to transport in vacuum (default=1.E8).

In addition to `MAIN`, `HOWFAR`, and `AUSGAB`, additional subprograms may be included in the user code to facilitate the geometry computations of `HOWFAR`, among other things. (Sample "auxiliary" subroutines useful in performing distance-to-boundary computations and in moving particles across regions in a variety of common geometries are provided with the EGS5 distribution.)

The interaction between the user code and the EGS5 modules is best illustrated in Figure B.1.

In summary, the user controls an EGS5 simulation by means of:

<div align="center">

Calls to subroutines:

</div>

PEGS5	to create media data
HATCH	to establish media data
SHOWER	to initiate the cascade

<div align="center">

Calls from EGS to user subroutines:

</div>

HOWFAR	to specify the geometry
AUSGAB	to score and output the results

<div align="center">

Altering elements of `COMMON` *blocks:*

</div>

parameters	inside EGS5 source code, to set array dimensions
variables	inside user code, to specify problem data

The following sections discuss the above mechanisms in greater detail.

B.3   Variables in EGS5 `COMMON` Blocks

Listed in Tables B.1 through B.17 are the variables in EGS5 `COMMON` blocks which may be relevant to the user, along with a brief description of their functions. Methods for manipulating these variables to either control EGS5 shower simulations or to retrieve results will be discussed in subsequent sections. Note that an asterisk (*) after a variable name in any of the tables indicates a change from the original EGS4 default.

```
    +---------+                         +-------------+
    |  User   |                         | Information |
    | Control |                         |  Extracted  |
    |  Data   |                         | From Shower |                   U
    +---------+                         +-------------+              S
         |                                    /|\                    E
         |                                     |                     R
        \|/                                    |
+----------------+        +--------+      +--------+
|     MAIN       |        | HOWFAR |      | AUSGAB |<----- +  C
+----------------+        +--------+      +--------+       |  O
     |          |            /|\ /|\       /|\   /|\       |  D
     |          |             |   |         |     |        |  E
     |          |             |  +-----------+    |        |
==== | ======= | =========== | ========= |  |  | ======= |==
     |          |             |           |  |  |         |
     |          |             |  +--------+ |  |           |
    \|/        \|/            |  |         |  |           |
+---------+ +--------+   +--------+      +--------+        |
|BLOCK DATA| | SHOWER |--->| ELECTR |    +->| PHOTON |--> +  |
|BLOCK_SET | +--------+   +--------+   |  +--------+      |  |
|(Defaults)|     |           |         |                 |  | E
+---------+    +------->  | -------->  +                 |  | G
     |            +--+                                   |  | S
     |            |  +-------+           +-------+  |  |
    \|/           +-->| MSCAT |      +--| COMPT  |<-+ |  | C
+----------+      |   +-------+      |  +-------+ |  |  | O
|  PEGS5   |      |                  +--| PAIR   |<-+ |  | D
| (Create  |      |   +-------+      |  +-------+ |  |  | E
|  Media   |      +-->| ANNIH |--+   |            |  |
|  Data)   |      |   +-------+  |   |  +-------+ |  |
+----------+      +-->| BHABHA|--+   |  | PHOTO  |<-+ |
     |            |   +-------+  |   |  +-------+     |
     |            +-->| MOLLER|--+   |       |        |
    \|/           |   +-------+  |   |       |        |
+----------+      +-->| BREMS |--+   |   +------> +   |
|  HATCH   |      |   +-------+  |   |                |
| (Access  |      |            \|/\|/                 |
|  PEGS5   |      |           +------+                 |
|  Data)   |      +---------> | UPHI |------------> +
+----------+                  +------+
```
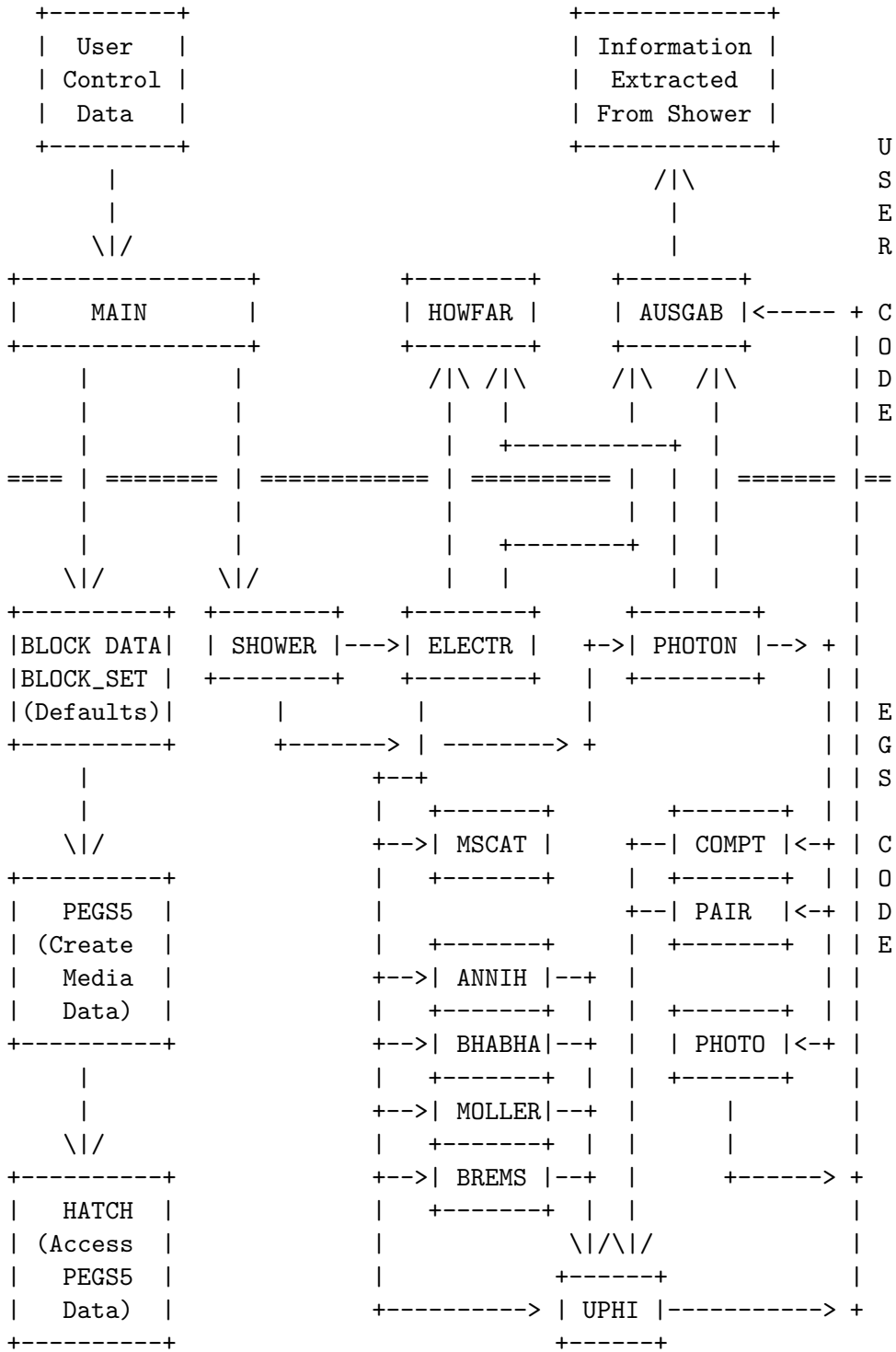
Figure B.1: EGS5 user code control and data flow diagram.

Table B.2: Variable descriptions for `COMMON` block `BREMPR`, **include** file **egs5_brempr.f** of the EGS5 distribution.

IBRDST*	Flag for turning on (=1) sampling of bremsstrahlung polar angle from (default=0 implies angle given by m/E).
IPRDST*	Flag for specifying order of sampling of polar angles of pair electrons (default=0, implies angles given by m/k).
IBRSPL*	Flag for turning on (=1) splitting of bremsstrahlung photons (default=0 implies no splitting).
NBRSPL*	Number of bremsstrahlung photons for splitting when IBRSPL=1

B.4 Sequence of Actions Required of User Code `MAIN`

The exact sequence of procedures required of user code `MAIN` for the specification and control of an EGS5 simulation is listed below. Details for implementing the necessary steps are provided in subsequent subsections.

Step 1	Pre-`PEGS5` initializations
Step 2	`PEGS5` call
Step 3	Pre-`HATCH` initializations
Step 4	Specification of incident particle parameters
Step 5	`HATCH` call
Step 6	Initializations for `HOWFAR`
Step 7	Initializations for `AUSGAB`
Step 8	`SHOWER` call
Step 9	Output of results

Steps 4, 6, and 7 may actually fall anywhere after step 1 and before step 8, and step 8 must be executed at least once prior to step 9. Step 2 may be skipped if an existing `PEGS5` data file has been prepared and properly linked.

B.4.1 Pre-`PEGS5` Initializations (Step 1)

Prior to calling `PEGS5` , users *must* define certain variables and may, at their discretion, override some of the EGS5 parameter defaults. As noted earlier, all EGS5 variables are readily accessed through `COMMON` blocks which are imported into user code through "include" statements, as in:

```
include 'include/egs5_h.f'        ! Main EGS "header" file

include 'include/egs5_bounds.f'  ! bounds contains ecut and pcut
include 'include/egs5_edge.f'     ! edge contains iedgfl
```

Table B.3: Variable descriptions for COMMON block COUNTERS, **include** file **counters.f** of the EGS5 distribution. All variables in COMMON block COUNTERS are initialized to 0 by a call to subroutine COUNTERS_OUT(0) with argument of 0.

IANNIH*	Number of times calling subprogram ANNIH.
IAPHI*	Number of times calling subprogram APHI.
IBHABHA*	Number of times calling subprogram BHABHA.
IBREMS*	Number of times calling subprogram BREMS.
ICOLLIS*	Number of times calling subprogram COLLIS.
ICOMPT*	Number of times calling subprogram COMPT.
IEDGBIN*	Number of times calling subprogram EDGBIN.
IEII*	Number of times calling subprogram EII.
IELECTR*	Number of times calling subprogram ELECTR.
IHARDX*	Number of times calling subprogram HARDX.
IHATCH*	Number of times calling subprogram HATCH.
IKAUGER*	Number of times calling subprogram KAUGER.
IKSHELL*	Number of times calling subprogram KSHELL.
IKXRAY*	Number of times calling subprogram KXRAY.
ILAUGER*	Number of times calling subprogram LAUGER.
ILSHELL*	Number of times calling subprogram LSHELL.
ILXRAY*	Number of times calling subprogram LXRAY.
IMOLLER*	Number of times calling subprogram MOLLER.
IMSCAT*	Number of times calling subprogram MSCAT.
IPAIR*	Number of times calling subprogram PAIR.
IPHOTO*	Number of times calling subprogram PHOTO.
IPHOTON*	Number of times calling subprogram PHOTON.
IRAYLEI*	Number of times calling subprogram RAYLEI.
ISHOWER*	Number of times calling subprogram SHOWER.
IUPHI*	Number of times calling subprogram UPHI.
ITMXS*	Number of times requested multiple scattering step was truncated in ELECTR because pathlength was too long.
NOSCAT	Number of times multiple scattering was aborted in MSCAT because the pathlength was too small (Note change in that NOSCAT has been moved here from EGS4 COMMON block MISC).
IBLOCK*	Number of times calling subprogram BLOCK_SET.

Table B.4: Variable descriptions for COMMON block EDGE2, **include** file **egs5_edge.f** of the EGS5 distribution.

IEDGFL*	Array of flags for turning on (=1) explicit treatment of K and L-edge fluorescent photons (default=0).
IAUGER*	Array of flags for turning on (=1) explicit treatment of K and L Auger electrons (default=0).

Table B.5: Variable descriptions for COMMON block EIICOM, **include** file **egs5_eiicom.f** of the EGS5 distribution.

IEISPL*	Flag for turning on (=1) splitting of x-rays generated by electron-impact ionization (default=0 implies no splitting).
NEISPL*	Number of electron impact ionization x-rays for splitting when IEISPL=1.

Table B.6: Variable descriptions for COMMON block EPCONT, **include** file **egs5_epcont.f** of the EGS5 distribution.

EDEP	Energy deposited in MeV.
TSTEP	Distance to next interaction (cm).
USTEP	User (straight line) step length requested and granted.
TVSTEP	Actual total Multiple scattering step to be transported. (Note that because of the use of the random hinge transport mechanics in EGS5, the EGS4 variables TUSTEP and VSTEP are redundant and so have been removed, as has the variable TSCAT.)
RHOF	Value of density scaling correction (default=1).
EOLD	Charged particle (total) energy at beginning of step in MeV.
ENEW	Charged particle (total) energy at end of step in MeV.
EKE	Kinetic energy of charged particle in MeV.
ELKE	Natural logarithm of EKE.
BETA2	β^2 for present particle. (Note that EGS4 variable BETA is no longer included.)
GLE	Natural logarithm of photon energy.
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.
IROLD	Index of previous region.
IRNEW	Index of new region.
IAUSFL	Array of flags for turning on various calls to AUSGAB.

Table B.7: Variable descriptions for `COMMON` block `MEDIA`, **include** file **egs5_media.f** of the EGS5 distribution.

`RLCM`	Array containing radiation lengths of the media in cm. (Note the name change necessitated by combining EGS and PEGS.)
`RLDU`	Array containing radiation lengths of the media in distance units established by `DUNIT`.
`RHOM`	Array containing density of the media in g/cm^3. (Note the name change necessitated by combining EGS and PEGS.)
`NMED`	Number of media being used (default=1).
`MEDIA`	Array containing names of media (default is NaI).
`IRAYLM`	Array of flags for turning on (=1) coherent (Rayleigh) scattering in various media. Set in `HATCH` based on values of `IRAYLR`.
`INCOHM*`	Array of flags for turning on (=1) use of incoherent scattering function for Compton scattering angles in various media. Set in `HATCH` based on values of `INCOHR`.
`IPROFM*`	Array of flags for turning on (=1) Doppler broadening of Compton scattering energies in various media. Set in `HATCH` based on values of `IPROFR`.
`IMPACM*`	Array of flags for turning on (=1) electron impact ionization in various media. Set in `HATCH` based on values of `IMPACR`.
`CHARD*`	Array of "characteristic dimensions," or representative size (in cm) of scoring regions in various media. Set by user code `MAIN` prior to `PEGS5` call to invoke automated electron step-size selection.
`USEGSD*`	Array of flags indicating (on =1) whether given media uses Goudsmit-Saunderson multiple scattering distribution. Set by user code `MAIN` prior to `HATCH` call (default=0). Note that in the current implementation, it is a requirement that if one elects to use this option in one media, one must use it in all media.

Table B.8: Variable descriptions for `COMMON` block `MISC`, **include** file **egs5_misc.f** of the EGS5 distribution.

NREG*	Number of regions for the problem, set by user code `MAIN` prior to `HATCH` call.
MED	Array containing medium index for each region, set by user code `MAIN` prior to `HATCH` call.
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 containing the density for each region (g/cm^3). If this is different than the default density of the material in that region, the cross sections and stopping powers (with the exception of the density effect) are scaled appropriately.
NOMSCT*	Array of flags forcing multiple scattering to be bypassed (on =1) in subroutine `MSCAT` for various regions (default=0, off).
IRAYLR	Array of flags for turning on (=1) coherent (Rayleigh) scattering in various regions (default=0).
LPOLAR*	Array of flags for turning on (=1) linearly polarized photon scattering in various regions (default=0).
INCOHR*	Array of flags for turning on (=1) use of incoherent scattering function for Compton scattering angles in various regions (default=0).
IPROFR*	Array of flags for turning on (=1) Doppler broadening of Compton scattering energies in various regions (default=0).
IMPACR*	Array of flags for turning on (=1) electron impact ionization in various regions (default=0).
K1HSCL*	Array of parameters for scaling region scattering strength at highest problem energy, set in user code `MAIN` prior to `HATCH` call.
K1LSCL*	Array of parameters for scaling region scattering strength at lowest problem energy, set in user code `MAIN` prior to `HATCH` call.

Table B.9: Variable descriptions for `COMMON` block `MS`, **include** file **egs5_ms.f** of the EGS5 distribution.

TMXSET*	Flag to force truncation of requested multiple scattering steps which violate Bethe criteria (default=`.true.`, enforce limit).

Table B.10: Variable descriptions for `COMMON` block `RLUXDAT`, **include** file **randomm.f** of the EGS5 distribution.

LUXLEV*	Luxury level of random number generator `RANLUX` (called `RANDOMSET` in EGS5) (default=1).
INSEED*	Initial seed used with `RANLUX` random number generator (default = 314159265).
KOUNT*	Number of random numbers delivered plus number skipped at any point in simulation (up to 10^9).
MKOUNT*	Number of sets of 10^9 random numbers delivered at any point in simulation.
ISDEXT*	Array of integer representations of the current `RANLUX` seeds at any point in simulation.

Table B.11: Variable descriptions for `COMMON` block `STACK`, **include** file **egs5_stack.f** of the EGS5 distribution. This `COMMON` contains information about particles currently in the shower. All variables are arrays except for `NP, LATCHI, DEINITIAL, DERESID` and `DENSTEP`.

E	Total energy in MeV.
X,Y,Z	Position of particle in units established by `DUNIT`.
U,V,W	Direction cosines of particle.
UF,VF,WF*	Electric field vectors of polarized photon.
DNEAR	A lower bound on the distance from the coordinates (`X,Y,Z`) to nearest surface of current region.
WT	Statistical weight of current particle (default=1.0). Used in conjunction with variance reduction techniques as determined by user.
K1STEP*	Scattering strength remaining before the next multiple scattering hinge.
K1RSD*	Scattering strength remaining after the current multiple scattering hinge to the end of the full, current multiple scattering step.
K1INIT*	Scattering strength from the end of the previous multiple scattering step to the current multiple scattering hinge.
DENSTEP*	Energy loss remaining before the next energy loss hinge.
DERESID*	Energy loss remaining after the current energy loss hinge to the end of the full, current energy loss step.
DEINITIAL*	Energy loss from the end of the previous energy loss step to the current energy loss hinge.
IQ	Integer charge of particle, +1,0,-1, for positrons, photons, and electrons, respectively.
IR	Index of particle's current region.
LATCH*	Latching variable
LATCHI*	Initialization for latch
NP	The stack pointer (i.e., the particle currently being pointed to). Also, the number of particles on the stack.

Table B.12: Variable descriptions for `COMMON` block `THRESH`, **include** file **egs5_thresh.f** of the EGS5 distribution.

RMT2	Twice the electron rest mass energy in MeV.
RMSQ	Electron rest mass energy squared in MeV^2.
AP	Array containing PEGS lower photon cutoff energy for each medium in MeV.
UP	Array containing PEGS upper photon cutoff energy for each medium in MeV.
AE	Array containing PEGS lower charged particle cutoff energy for each medium in MeV.
UE	Array 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 containing the Møller threshold energy (THMOLL=AE+TE) for each medium in MeV.

Table B.13: Variable descriptions for `COMMON` block `UPHIOT`, **include** file **egs5_uphiot.f** of the EGS5 distribution.

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	π
TWOPI	2π
PI5D2*	$5\pi/2$

Table B.14: Variable descriptions for `COMMON` block `USEFUL`, **include** file **egs5_useful.f** of the EGS5 distribution.

MEDIUM	Index of current medium. If vacuum, then MEDIUM=0.
MEDOLD	Index of previous medium.
RM	Electron rest mass energy in MeV.
IBLOBE	Flag indicating if photon is below binding energy (EBINDA) after a photoelectric interaction (yes=1).

Table B.15: Variable descriptions for `COMMON` block `USERSC`, **include** file **egs5_usersc.f** of the EGS5 distribution.

`ESTEPR*`	Array of factors by which to scale the energy hinge steps in various regions (default=0, implying no scaling).
`ESAVE*`	Array of energies below which to discard electrons which have ranges less than the perpendicular distances to their current region boundaries (default=0., implying no range-based discard).
`EMAXE*`	Maximum total energy (in MeV) of any electron in the simulation.

Table B.16: Variable descriptions for `COMMON` block `USERVR`, **include** file **egs5_uservr.f** of the EGS5 distribution.

`CEXPTR*`	Constant used in exponential transform of photon collision distance (default=0, no transformation).

```
      include 'include/egs5_epcont.f'  ! epcont contains iausfl
      include 'include/egs5_media.f'   ! media contains the array media
      include 'include/egs5_misc.f'    ! misc contains med
      include 'include/egs5_thresh.f'  ! thresh contains ae and ap
      include 'include/egs5_uphiot.f'  ! uphiot contains PI
      include 'include/egs5_useful.f'  ! useful contains RM
      include 'include/egs5_usersc.f'  ! usersc contains emaxe
      include 'include/randomm.f'
```

Note that most of the variables accessed in a typical user code `MAIN` program can be found in the `COMMON` files referenced by the `include` statements in the above example. Other variables which a user code might wish to access and the EGS5 **include** files which contain them were given in Tables B.1 through B.17 of the previous section.

Note that all EGS5 variables are explicitly declared (all EGS5 subroutines and functions begin with the statement `IMPLICT NONE`), and that all floating-point variables (except some of those used in the random number generator and in sample user codes which call intrinsic functions to compute CPU time) are declared as `REAL*8`.

Table B.17: Variable descriptions for `COMMON` block `USERXT`, **include** file **egs5_userxt.f** of the EGS5 distribution.

`IPHTER*`	Array of flags for turning on (=1) sampling of angular distributions of photoelectrons in various regions (default=0, implying emission in direction of incident photon).

Optional parameter modifications

The EGS5 file **include/egs5_h.f** is different from the other files in the **include** directory in that it contains not COMMON blocks, but rather declarations and specifications of the FORTRAN parameters used by the other EGS5 **include** files to define array dimensions. This is done so that users may trivially update the dimensions of all arrays throughout the EGS5 code system simply by changing the values of the appropriate variables in the PARAMETER statements of **include/egs5_h.f**. The principal parameters defined in **include/egs5_h.f** which users may wish to adjust are MXMED (the maximum number of media for the problem), MXREG (the maximum number of regions), and MXSTACK (the maximum stack size). Most of the other parameters defined in **include/egs5_h.f** should be altered only under exceptional circumstances. Some examples of parameter modifications are given in the comments in **include/egs5_h.f**, as seen below:

```
! Maximum number of different media (excluding vacuum)
      integer MXMED
      parameter (MXMED = 4)
!      parameter (MXMED = 10)

! Maximum number of regions allocated
      integer MXREG
      parameter (MXREG = 2000)
!      parameter (MXREG = 2097153)
```

Required initializations

Two sets of initializations *must* be performed in the user's MAIN program. First, MAIN must call the EGS5 subroutine BLOCK_SET to initialize common block variables not defined in BLOCK DATA. This is done simply by including the statement:

```
!      =============
      call block_set                  ! Initialize some general variables
!      =============
```

Also, if the user is interested in tracking the number of calls to the various subroutines of EGS5, the counters in common block COUNTERS may also be initialized at this point by calling subroutine COUNTERS_OUT with argument 0, as in:

```
!      ===================
      call counters_out(0)
!      ===================
```

Second, because of the way PEGS and EGS are linked in EGS5, the specification of the names of the problem media prior to calling `PEGS5` is now a requirement of EGS5 user codes. The `COMMON MEDIA` variables `NMED` (the number of media for the current problem) and `MEDIA` (a character array of the names of the media) *must* be set prior to a call to `PEGS5`. Note that the media names must be exactly 24 characters long. An example of a typical method for filling the `MEDIA` array (using lead, steel, and air at NTP as the media), is shown below. First, a local array is declared and initialized in `MAIN` , and then copied into `MEDIA` as in:

```
      character*24 medarr(3)

      medarr(1)='PB                      '
      medarr(2)='STEEL                   '
      medarr(3)='AIR AT NTP              '

      nmed=3      !Number of media used
      do j=1,nmed
        do i=1,24
          media(i,j)=medarr(j)(i:i)
        end do
      end do
```

One final variable, which is optional but recommended, must be set prior to `PEGS5` being called if it is to be used. As described in chapter 2 of SLAC-R-730/KEK-2005-8, EGS5 provides a method for selecting nearly optimal electron multiple-scattering step-sizes in most applications. The method requires the input specification of a material-dependent parameter `CHARD`, dimensioned `CHARD(MXMED)` and related to the size (in cm) of the smallest scoring region for a given material. Values (in cm) of `CHARD`, which is part of `COMMON MEDIA`, can be passed to `PEGS5` simply by assigning values, as in:

```
      chard(1) = .60d0        !  optional, but recommended to invoke
      chard(2) = .10d0        !  automatic step-size control
      chard(3) = .85
```

If `CHARD` is not specified or is set to 0 (the default) for a given material, `PEGS5` will use a method for determining scattering strengths (and hence step-sizes) for electron multiple scattering based on fractional energy losses, also described in chapter 2 of the EGS5 Code System report.

B.4.2   `PEGS5` Call (Step 2)

`MAIN` may now call `PEGS5` to create material data files for the problem. Specifications for the PEGS input is found in the "PEGS User Manual," Appendix C of SLAC-R-730/KEK-2005-8. Note that

the call to PEGS5 may be skipped if the working user code directory contains an existing PEGS data file generated with parameters compatible with the current EGS5 simulation specifications. Checks for compatibility are performed in HATCH.

B.4.3 Pre-HATCH Initializations (Step 3)

Users are strictly required to define the following variables prior to HATCH being called: NREG, the number of regions in the geometry; MED, an array containing the material numbers (as set prior to the call to PEGS5) of each region, and EMAXE, the maximum total energy of any electron in the problem. No other variables used by HATCH (and then by the EGS5 system in simulating showers), need be explicitly specified. However, if the user wishes to use any of the non-default options or features of EGS5, the appropriate flags for invoking such requests must be specified prior to the call to HATCH , even if the data needed to execute such options has been generated by PEGS. All of the variables processed by HATCH in setting up an EGS5 simulation are described below.

Variables required by HATCH

EMAXE   This variable, the maximum energy of an electron in the problem, is located in COMMON USERSC found in **include/egs5_usersc.f**, and is used by HATCH to perform checks on the compatibility of the EGS5 problem specification and the PEGS data file being used.

NREG   HATCH uses the variable NREG when verifying and loading region-dependent options for the problem materials.

MED   The array MED, 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 vacuum. Indices are defined by the order specified by the user, and are independent of the order in which the materials are defined in the PEGS data file being used. Consider the three media example above (from the pre-PEGS5 initialization section with vacuum defined as a fourth regions. The EGS5 user code to accomplish this might look like:

```
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
```

Optional variables and flags processed by HATCH

ECUT and PCUT   The ECUT and PCUT arrays contain the cutoff energies (in MeV) for the termination of the tracking of charged particles and photons, respectively, for each region. They are dimensioned ECUT(MXREG) and PCUT(MXREG) and are initialized to 0.0 in BLOCK_SET. Note that HATCH will override any user defined values of ECUT and PCUT if these values are lower than the threshold energies set in PEGS for the generation of secondary electrons and photons (the parameters AE and AP). Thus, by assigning values of ECUT and PCUT prior to the HATCH call, the user can raise (but not lower) the cutoff energies. This can be illustrated by considering the four region example from above. The statements

```
do i=1,3
  ecut(i)=10.0
  pcut(i)=100.0
end do
```

when put in Step 3 of the user code result 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 will be determined by the values of AE and AP as established by PEGS. ECUT and PCUT are elements of COMMON BOUNDS .

IRAYLR   The elements of this array (dimensioned IRAYLR(MXREG) and contained in COMMON/MISC/), are set to 1 prior to calling HATCH when coherent (Rayleigh) scattering is to be modeled in particular regions. Execution of EGS is terminated if Rayleigh scattering data is not included in the PEGS data file, however.

INCOHR   The elements of this array (dimensioned INCOHR(MXREG) and found in COMMON/MISC/), are set to 1 prior to calling HATCH when incoherent scattering functions are to be used in sampling Compton scattering angles in particular regions. Execution of EGS5 is terminated if the appropriate incoherent scattering function data is not found in the PEGS data file being used, however. Note that when INCOHR(I)=1, it is necessary to have used IBOUND=1 for the corresponding materials when PEGS was run.

IPROFR   The elements of this array (dimensioned IPROFR(MXREG) and accessed via COMMON/MISC/), are set to 1 prior to calling HATCH if Doppler broadening of the energies of Compton scattered photons is to be modeling in particular regions. EGS5 execution is terminated if the Doppler broadening data is not found by HATCH in the PEGS data file being used, however. Note that when IPROFR(I)=1, it is necessary to have set IBOUND=1 and INCOH=1 in the corresponding materials when PEGS was run, and that MAIN must set INCOHR(I)=1 for the corresponding regions as well.

IMPACR   The elements of this array (dimensioned IMPACR(MXREG) and found in COMMON/MISC/), are set to 1 prior to calling HATCH when electron impact ionization is to be simulated in particular

331

regions. Execution of EGS5 is terminated if the electron impact ionization data is not found in the PEGS data, however.

IPHTER   The elements of this array (dimensioned `IPHTER(MXREG)` and located in `COMMON/USERXT/`), are set to 1 if photoelectron angles are to be sampled in particular regions. The default (`IPHTER=0`) assumes emission in the direction of the incident photon.

IEDGFL   The elements of this array (dimensioned `IEDGFL(MXREG)` and passed in `COMMON/EDGE2/`), are set to 1 if K and L-edge fluorescence is be explicitly modeled in specific regions.

IAUGER   The elements of this array (dimensioned `IAUGER(MXREG)` and found in `COMMON/EDGE2/`), are set to 1 if K and L-edge Auger electrons are to be generated in given regions.

LPOLAR   The elements of this array (dimensioned `LPOLAR(MXREG)` and contained in `COMMON/MISC/`), are set to 1 if linearly polarized photon scattering is to be modeled in specified regions.

DUNIT   The parameter `DUNIT` defines the unit of distance to be used in the shower simulation (the default is cm if `DUNIT=1`). On input to `HATCH`, `DUNIT` is interpreted as follows:

1. `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.

2. `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 the 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`, which results in the shower being expressed entirely in radiation lengths of the first medium.

Note that the unit of distance used in PEGS is the radiation length. After `HATCH` interprets `DUNIT`, it scales all PEGS data by units of distance as specified by the user, so that all subsequent operations in EGS will be performed with distances in units of `DUNIT` (default value: 1.0 cm).

K1HSCL **and** K1LSCL   The parameters `K1HSCL` and `K1LSCL` permit the user to apply energy-dependent scaling of the material-dependent scattering strength (which is roughly proportional to the multiple-scattering step-size distance) on a region-by-region basis. When `K1HSCL` and `K1LSCL` are non-zero for a region, the scattering strength at `EMAXE` is scaled by the factor `K1HSCL` and

the scattering strength at `ECUT` for the region is scaled by `K1LSCL`. Scaling at other electron energies is determined by logarithm interpolation. `K1HSCL(MXREG)` and `K1LSCL(MXREG)` are found in `COMMON/MISC/` and are initialized to 0.0 in `BLOCK_SET`, which implies no scaling.

`USEGSD`   If the user wishes to use the Goudsmit-Saunderson multiple-scattering distribution function instead of the Molière distribution function for a material, `USEGSD(MXMED)` must be set to be non-zero prior to the call to `HATCH`. In the current version of EGS5, all regions must use the Goudsmit-Saunderson distribution if any of them do. `USEGSD` is a part of `COMMON` block `MEDIA`.

`RHOR`   Media of similar materials but with varying density in different regions can be defined by setting non-zero values of the region density in the variable `RHOR(MXREG)` of `COMMON/MISC/` prior to calls to `HATCH` . This feature eliminates the need for the user to create a distinct new media for each region which has a a given material but with a different density. Values of `RHOR` should be specified in terms of the actual density in each region, not the density relative to the reference density. `RHOR` is initialized to 0 in `BLOCK_SET` and assigned the default density of the medium by `HATCH` unless specified by the user prior to `HATCH` being called.

Flags and variables which may be set either before or after `HATCH` is called

The following variables can be set either before or after the call to `HATCH`.

`TMXSET`   When `TMXSET` is `.true.`, any multiple-scattering step, whether selected by the user or determined by EGS5 using `CHARD`, which violate the Bethe criteria for the maximum allowed step length (see chapter 2 of SLAC-R-730/KEK-2005-8) will be truncated in `ELECTR` to the maximum. If the user wishes to over-ride this limit, `TMXSET` (which is material dependent and part of `COMMON MS` and defaults to `.true.`) can be set to `.false.` at any point in an EGS5 user code.

`ESTEPR`   Electron energy hinge steps are scaled on a region-dependent basis when users set non-zero values of `ESTEPR(MXREG)` prior to a call to `SHOWER`. Since energy hinge step sizes are determined in PEGS, `ESTEPR` provides the user the capability to take smaller or larger steps in certain materials or regions for increased accuracy or efficiency, respectively. `ESTEPR`, which is part of EGS5 `COMMON USERSC`, is initialized to 0.0 in `BLOCK_SET` and ignored in `ELECTR` unless set by the user.

`ESAVE`   The variable `ESAVE`, dimensioned `ESAVE(MXREG)` and part of `COMMON USERSC`, can be employed by users to speed computations for applications which involve the transport of electrons across boundaries between scoring and non-scoring regions. For example, if a user is interested in energy deposition in a gas detector, only those electrons which are energetic enough to escape the solid walls surrounding the gas of the detector have a chance to be scored. Thus the simulation

of the transport in the walls of electrons with ranges less than the closest normal distances to the outer walls adds nothing but CPU time to the simulation. If, however, the user specifies a non-zero value of `ESAVE` for a given region, `ELECTR` will discard the electron if its energy is less than `ESAVE` and its range is less than `DNEAR` (see below), thus speeding the computation. This technique is commonly called "range rejection," and is most effective when `ESAVE` is much larger than `ECUT`. Note that the "range" of the electron is defined very crudely here, as simply `E(NP)` divided by the stopping power of the medium. This assures that the decision to discard a particle based on range rejection will be conservative as long as the stopping power of the medium increases at energies below `ESAVE`.

`IBRDST`   The parameter `IBRDST`, which has a default value of 0 and is part of `COMMON BREMPR`, determines the procedure for determining the angle of bremsstrahlung photons (relative to the incident electrons), as described below:

`IBRDST`	Method for determining θ
0	fixed at $m/\breve{E}_0$
1	sampled from Koch and Motz formula 2BS

Values of `IBRDST` set by the user apply to all media and regions in a simulation.

`IPRDST`   The value of the parameter `IPRDST` determines the method used for determining the angles of electron and positron pairs resulting from photon pair-production in the same way that `IBRDST` is used to select the sampling method for bremsstrahlung photon angles. `IPRDST`, which is part of `COMMON BREMPR`, has a default value of 0 and controls pair electron angles (relative to the incident photon direction) as follows:

`IPRDST`	Method for determining θ
0	fixed at m/k
1	sampled from Motz, Olsen and Koch formula 3D-2000
2	sampled from Motz, Olsen and Koch formula 3D-2003

`IEISPL` **and** `NEISPL`   In order to speed up EGS5 simulations for applications involving x-rays generated from electron-impact ionizations, a method for creating additional x-rays using the familiar Monte Carlo technique of splitting is provided (see chapter 4 of SLAC-R-730/KEK-2005-8 for the description of an EGS5 application involving splitting of particles). If the flag `IEISPL`, which is part of `COMMON EIICOM` and defaulted to 0, is set to be 1, each electron-impact ionization event which leads to the production of a characteristic x-ray will result in `NEISPL` appropriately weighted x-rays being produced.

IBRSPL and NBRSPL   The parameters `IBRSPL` and `NBRSPL` allow the user to improve the efficiencies of simulations in which low-probability bremsstrahlung photon production is important by splitting the secondary particles. The method is similar to that described above for splitting in x-ray production following electron-impact ionization. When the parameter `IBRSPL`, which has a default value of 0 and is found in `COMMON BREMPR`, is set to 1, each electron bremsstrahlung event will result in the generation of `NBRSPL` appropriately weighted photons.

CEXPTR   The parameter `CEXPTR`, found in `COMMON USERVR`, is a scaling factor which can be used to either force or inhibit photon collisions in regions with cross section that are very small or very large. If λ is the photon mean free path and we use C to represent the scaling factor `CEXPTR`, we have for the interaction probability distribution:

$$\tilde{p}(\lambda)d\lambda = (1 - C\mu)e^{-\lambda(1-C\mu)}d\lambda,$$

where the overall multiplier $1 - C\mu$ is introduced to ensure that the probability is correctly normalized, *i.e.*$\int_0^\infty \tilde{p}(\lambda)d\lambda = 1$. For $C = 0$, we have the unbiased probability distribution $e^{-\lambda}d\lambda$. One sees that for $0 < C < 1$, the average distance to an interaction is stretched and for $C < 0$, the average distance to the next interaction is shortened. Note that the average number of mean free paths to an interaction, $\langle\lambda\rangle$, is given by $\langle\lambda\rangle = \int_0^\infty \lambda\tilde{p}(\lambda)d\lambda = \frac{1}{1-C\mu}$.

NOMSCT   The user may override all treatment of electron multiple-scattering in a given region by setting the switch `NOMSCT(MXREG)` to be 1 for that region. `NOMSCT` which is a part of `COMMON MISC` and is initialized 0, is used primarily as a debugging and code development tool, and is included in this description for completeness only.

Random number generator initialization

Whenever EGS (including any part of the user code) requires a floating point random number taken uniformly from the interval (0,1) to be returned to a variable (all EGS5 routines use the variable name `RNNOW`), the following statement is required:

```
call randomset(rnnow)
```

EGS5 employs the random number generator `RANLUX`, implemented by James. Depending on the input specification, called the "luxury level," `RANLUX` provides random sequences which pass different levels of tests for randomness and execute at different speeds. Independent random sequences for the same luxury level can be generated with `RANLUX` by simply specifying a different input "seed," any integer in the range from 1 to 2^{31}. The default luxury level, as defined in the variable `LUXLEV` of `COMMON RLUXDAT` in file **include/randomm.f**, is 1, and the default seed, `INSEED`, is 314159265. `RANLUX` is initialized by `HATCH` using the defaults for `LUXLEV` and `INSEED` unless the user specifies different values prior to the `HATCH` call. In addition, users may initialize the generator themselves at any time by invoking

```
    call rluxinit
```

after specifying `LUXLEV` and `INSEED`.

The user may also restart `RANLUX` at any desired point in a previously used sequence of random numbers using either of two ways. `RANLUX` keeps a tally of the number of random numbers delivered through the variables `MKOUNT` and `KOUNT` as `MKOUNT*100000000 + KOUNT`, and `MKOUNT` and `KOUNT` are accessible at all times through `COMMON RLUXDAT` . If the user calls `RLUXINIT` and supplies values of `MKOUNT` and `KOUNT` in addition to `LUXLEV` and `INSEED`, the `RANLUX` will be restarted at exactly that point in the sequence defined by `MKOUNT` and `KOUNT`.

Alternatively, the user may execute the following statement

```
    call rluxout
```

at any time, at which point integer representations of the current values of the seeds in `RANLUX` will be returned via the array `ISDEXT` of `COMMON RLUXDAT` . A call to `RLUXINIT` at any time when the values of `ISDEXT` are non-zero will result in a restart of `RANLUX` based on the seeds in `ISDEXT`.

Thus the restart options can be summarized as follows:

1. A brute-force method involves calling `RLUXINIT` with the values of the luxury level, initial seed, and number of delivered randoms up to the time of the desired restart. The values of `MKOUNT` and `KOUNT` can be obtained at any time directly from `COMMON RLUXDAT`.

2. A more elegant restart using the actual seeds can be done by passing the integer seeds at the time of the restart to `RLUXINIT` via `ISDEXT` in `COMMON RLUXDAT`. The seeds in `ISDEXT` can be obtained for later restart at any convenient time (such as the end of a shower, or the end of a batch) by a call to `RLUXOUT`.

B.4.4 Specification of Incident Particle Parameters (Step 4)

This step required in constructing a `MAIN` user code is self-explanatory. An example of suitable coding is given as follows:

```
  iqi=-1          !Incident particle is an electron
  xi=0.0          !Particle coordinates
  yi=0.0
  zi=0.0
  ui=0.0          !Direction cosines
```

```
      vi=0.0
      wi=1.0

      iri=2           !Region number 2 is the incident region
      wti=1.0         !Weight factor in importance sampling
      ncases=10       !Number of histories to run
      idinc=-1

      ei=1000.d0      !Total energy (MeV)
      ekin=ei+iqi*RM !Incident kinetic energy
```

Note that the variables initialized above are the ones passed to EGS5 subroutine `SHOWER`, as described below in step 8.

B.4.5 HATCH Call (Step 5)

When the user code `MAIN` calls the EGS `HATCH` subroutine, EGS is "hatched" by executing some necessary once-only initializations and reading material data for the media from a data set that created by PEGS. The required call is, trivially:


```
!     ==========
      call hatch
!     ==========
```


Some examples of reports from `HATCH` are shown below. 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.

   EGS SUCCESSFULLY 'HATCHED' FOR ONE MEDIUM.
```


For a non-default specification of `DUNIT` (`DUNIT=2.54`, for example), the output report from `HATCH` would look like the following (for two media and no Rayleigh data):


```
   DUNIT REQUESTED&USED ARE:   2.54000E+00   2.54000E+00(CM.)
   EGS SUCCESSFULLY 'HATCHED' FOR    2 MEDIA.
```


Failure to successfully "hatch" a medium because it could not be found in the PEGS data file results in message below, and execution is terminated by `HATCH` .

```
END OF FILE ON UNIT 12

PROGRAM STOPPED IN HATCH BECAUSE THE
FOLLOWING NAMES WERE NOT RECOGNIZED:
                (list of names)
```

Note that one cannot ask for the same medium twice, though one can define two media which are physically identical to be distinct for the purposes of EGS by using different names for them in PEGS input files.

B.4.6   Initializations for `HOWFAR` (Step 6)

As stated previously, `HOWFAR` is the routine that describes the translation of particles through the geometry of the various regions in the problem. Note that initialization of data required by `HOWFAR` may be done at any step prior to calling `SHOWER` in Step 8, and that in fact, for some trivial versions of `HOWFAR`, no initializations are required at all. For versions of `HOWFAR` which model realistic geometries, however, it is likely that some initialization will be required in `MAIN` or auxiliary user subprograms called by `MAIN`. In such cases it will also be necessary that local auxiliary `COMMON` blocks be defined to pass geometry data to `HOWFAR`.

B.4.7   Initializations for `AUSGAB` (Step 7)

This step is similar to initialization for `HOWFAR` above in that it could actually be done anywhere in `MAIN` prior to `SHOWER` being called. An example initialization based on a three region geometry is given here. Suppose that we wish to know the total energy deposited in each of the three regions. We could declare a scoring array, `ESUM` in a `COMMON` block `TOTALS` in both `MAIN` and in `AUSGAB` as:

```
common/totals/esum(3)
```

This array would be initialized in `MAIN` by the statements:

```
do i=1,3
  esum(i)=0.0
end do
```

Then the statement

```
esum(ir(np))=esum(ir(np)) + edep
```

338

in `AUSGAB` would keep a running total of the energy deposited in each region under consideration. Note that global auxiliary subroutines `ECNSV1` and `NTALLY` are provided with the EGS5 distribution to facilitate scoring of energy deposition and the numbers of various types of events, respectively.

B.4.8 `SHOWER` Call (Step 8)

The calling sequence for `SHOWER` is:

```
call shower(iqi,ei,xi,yi,zi,ui,vi,wi,iri,wti)
```

All of the arguments in this call are declared `real*8` in `SHOWER` , except for iqi and iri which are integer. These variables, which can have any names the user wishes in `MAIN` , 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 the listing in Table B.11). In a typical problem `SHOWER` is called repeatedly in a loop over a number of "histories" or "cases" as in

```
do i=1,NCASES
  call shower(iqi,ei,xi,....,etc.)
 end do
```

The statistical weight `WTI` of the incident particle is generally taken as unity unless variance reduction techniques are employed by the user. Note 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.

Specification of electric vector of photon for `SHOWER`

This is necessary only if the incident particle is a photon and the scattering of linearly polarized photons is being modeled. The following 3 examples illustrate the specification of the photon electric field vector and the passing of that data to `SHOWER`.

Example 1. Completely linearly polarized photon source with electric vector along +y-direction:

```
ufi=0.0
vfi=1.0
wfi=0.0
```

```
      do i=1,ncases
        uf(1)=ufi
        vf(1)=vfi
        wf(1)=wfi
        call shower(iqi,e,xi,yi,zi,ui,vi,wi,iri,wti)
      end do
```

Example 2. Partially linearly polarized photon source with source propagation vector along the z-direction and polarization vector along the y-axis with P=0.85 (P is the degree of linear polarization):

```
      ui=0.0
      vi=0.0
      wi=1.0

      pval=0.85              ! Degree of linear polarization
      pratio=0.5+pval*0.5  ! Ratio of y-polarization

      do i=1,ncases
        call randomset(value)
        if(value.lt.pratio) then
          ufi=0.0
          vfi=1.0
          wfi=0.0
        else
          ufi=1.0
          vfi=0.0
          wfi=0.0
        end if
        uf(1)=ufi
        vf(1)=vfi
        wf(1)=wfi
        call shower(iqi,e,xi,yi,zi,ui,vi,wi,iri,wti)
      end do
```

Example 3. Unpolarized photon source. In a photon transport simulation modeling linear polarization, an unpolarized photon source is automatically generated by setting:

```
      uf(1)=0.0
      vf(1)=0.0
      wf(1)=0.0
```

inside the shower call loop.

B.4.9 Output of Results (Step 9)

This step is self-explanatory, and is included only for the sake of completeness.

B.5 Specifications for `HOWFAR`

EGS calls user code `HOWFAR` when it reaches the point at which it has determined, because of step-size specifications and/or interaction probabilities, that it would like to transport the top particle on the stack a straight line distance `USTEP` in the current media. All of the parameters of the particle are available to the user via `COMMON/STACK/` as described earlier. The user controls the transport upon return to EGS by altering one or more of the following variables: `USTEP`, `IDISC`, `IRNEW`, and `DNEAR(NP)`. Except for `DNEAR` (which is in `COMMON/STACK/`), these are available to the user via `COMMON/EPCONT/`. The ways in which these variables may be changed and the way EGS will interpret these changes is discussed in detail below. (Note, flow diagrams for subroutines `ELECTR` and `PHOTON` have been included in Appendix A of SLAC-R-730/KEK-2005-8 for the user who requires a more complete understanding of what actually takes place during particle transport.)

`IDISC`   If the user decides that the current particle should be discarded, then `IDISC` must be set nonzero (the usual convention is to set `IDISC=1`).

A positive value for `IDISC` will cause the particle to be discarded immediately. A negative value for `IDISC` will cause EGS to discard the particle when it completes the transport. 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, *e.g.*outside the problem geometry.

`USTEP` **and** `IRNEW`   If immediate discard has not been requested, then the `HOWFAR` should check to see whether transport by distance `USTEP` will cause a region boundary to be crossed. 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.

`DNEAR(NP)`   The setting of `DNEAR(NP)` by the user is optional. However, in many situations a significant gain in efficiency will result by defining `DNEAR(NP)` in `HOWFAR`. It is obvious that distance to boundary calculations are computationally expensive and should be avoided whenever possible. For electrons traveling in regions in which their step sizes are much smaller than the region dimensions, interrogation of the problem geometry at each electron step can greatly slow the simulation. In order to avoid this inefficiency, each particle has stored on the stack a variable called `DNEAR(NP)`, which is used by EGS to hold a lower bound on the distance from the particle's current position to the nearest region boundary. This variable is used by EGS in the following ways:

1. `DNEAR` for the incident particle is initialized to zero.

2. Whenever a particle is actually moved (by a straight line distance `TVSTEP`) the path length transported is deducted from the `DNEAR` for the particle.

3. Whenever a particle interacts, the `DNEAR` values for the product particles are set from the `DNEAR` value of the parent particle.

4. When EGS 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`. It is this feature which permits EGS to avoid potentially cumbersome geometry computations whenever possible.

In summary, to take advantage of these efficiency features, the user should set `DNEAR(NP)` equal to the perpendicular distance to the nearest region boundary from the particle's current position. If it is easier for the user to compute some quick lower bound on the actual nearest distance, this could be used to set `DNEAR` with time savings depending on how close the lower bound is to the actual nearest distance on the average. It should be understood, however, that if the boundary separations are smaller than the mean step size, subroutine `HOWFAR` will still be called and the overall efficiency will decrease as a result of having to perform the `DNEAR` calculation so many times. Finally, 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.

B.5.1   Sample `HOWFAR` User Code

Consider, as an example of how to write a `HOWFAR` subprogram, the three region geometry in B.2. 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 correctly models this geometry:
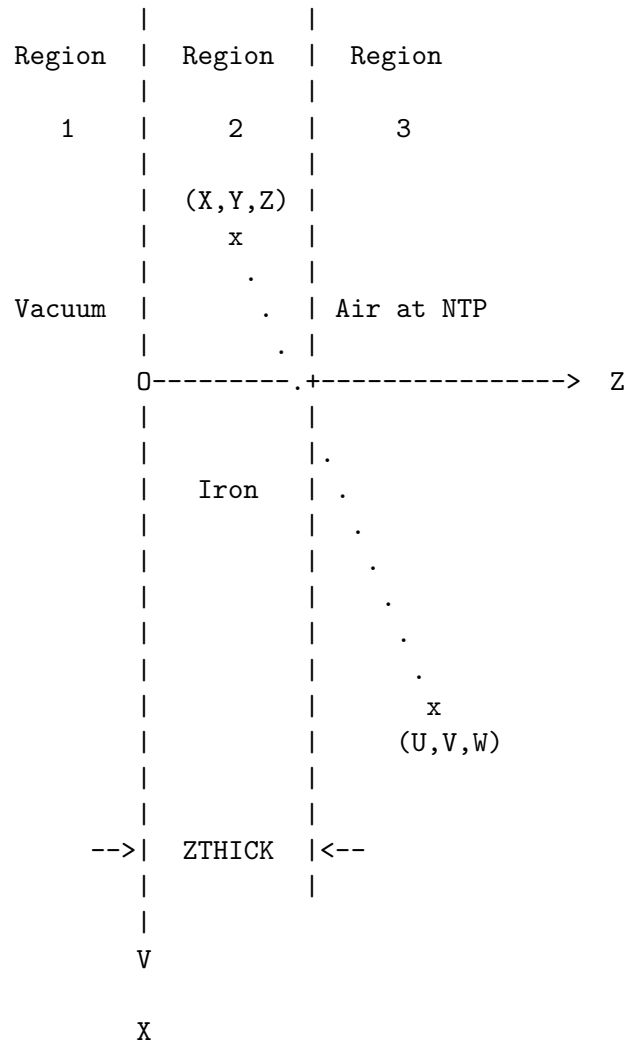
```
        |         |
Region  | Region  |  Region
        |         |
   1    |    2    |     3
        |         |
        | (X,Y,Z) |
        |    x    |
        |      .  |
Vacuum  |       . | Air at NTP
        |        .|
        O---------.+---------------->  Z
        |         |
        |         |.
        |   Iron  | .
        |         |  .
        |         |   .
        |         |    .
        |         |     .
        |         |      .
        |         |        x
        |         |      (U,V,W)
        |         |
        |         |
    -->|   ZTHICK  |<--
        |         |
        |
        V

        X

   (Y into paper)
```

Figure B.2: A three-region geometry for a `HOWFAR` example code.

343

```fortran
      subroutine howfar

      implicit none

      include 'include/egs5_h.f'                  ! Main EGS "header" file

      include 'include/egs5_epcont.f'    ! COMMONs required by EGS5 code
      include 'include/egs5_stack.f'

      common/passit/zthick
      real*8 zthick

      real*8 deltaz                      ! Local variables
      integer irnxt

      if (ir(np).ne.2) then
        idisc = 1
        return
      end if

      dnear(np) = dmin1(z(np),zthick-z(np))

!---------------------------------
! Particle going parallel to planes
!---------------------------------
      if(w(np).eq.0) return

!----------------------------------------------------------
! Check forward plane first since shower heading that way
! most of the time
!----------------------------------------------------------
      if (w(np).gt.0.0) then
        deltaz=(zthick-z(np))/w(np)
        irnxt=3
!----------------------------------------------------------
! Otherwise, particle must be heading in backward direction.
!----------------------------------------------------------
      else
        deltaz=-z(np)/w(np)
        irnxt=1
      end if

      if (deltaz.le.ustep) then
        ustep=deltaz
        irnew=irnxt
```

```
        end if

        return
        end
```

Note that a number of auxiliary geometry subprograms are distributed with the EGS5 Code
System in order to make it easier to write `HOWFAR`. For example, subroutine `PLAN2P` could have been
used in place of several lines above and the program would have been easier to read. Example user
codes which employ several of the auxiliary geometry subprograms are described in Chapter 4 of
SLAC-R-730/KEK-2005-8.

B.6   Specifications for `AUSGAB`

The user subroutine `AUSGAB` is called at more than 40 places inside various EGS5 subroutines with
the statement:


```
        call ausgab(iarg)
```


The argument `IARG` indicates the situation under which `AUSGAB` is being called. `IARG` can take
on 31 values starting from zero (i.e., `IARG=0` through `IARG=30`), although only the first five are
called in the default version of EGS. The remaining 26 `IARG` situations must be "switched on" via
specification of the array `IAUSFL`. The 5 values of `IARG` which are turned on by default and the
corresponding situations in which they initiate calls to user code `AUSGAB` are given in Table B.18.

The above `IARG` values are the ones required in the majority of situations in which EGS5 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 being scored. The
large number of situations which initiate calls to `AUSGAB` for various `IARG` values allows the user
to extract information about EGS5 simulations without making changes to the EGS code. The
user controls when `AUSGAB` is to be called by specifying in the user code values of the integer flag
array, `IAUSFL(J)`, for J=1 through 31. `IAUSFL(J)` 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` of 0 through 4,
`IAUSFL(J)` is set to 1 by default, and `AUSGAB` is always called for the situations listed in Table B.18
For the remaining values of J, corresponding to `IARG` =5 through 31, `IAUSFL(J)` is set to 0 by
default, and the user must modify `IAUSFL(J)` in order to initiate any desired `AUSGAB` calls. The
value for `IARG` and the corresponding situations for this upper set of `IARG` values are shown in
Table B.19.

Note that the code statuses for `IARG` values from 0 to 3 and from 5 to 24 are the identical
to those found in EGS4. A slight modification has been made in EGS5 for `IARG` of 4, and the
situations in which `IARG` values of 25 through 30 initiate a call to `AUSGAB` are newly added in EGS5.

345

Table B.18: `IARG` values program status for default `AUSGAB` calls.

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).
4	Part of particle energy is deposited due to the binding energy. This situation occurs in one of the following 3 cases: 1. A photoelectric interaction has occurred and the difference in the electron binding energy and the secondary particle (X-ray or Auger electron) energy is deposited. 2. Compton interaction has occurred and the electron binding energy is deposited locally. This is enabled only when the Doppler-broadening option is turned on. 3. The K-shell EII has occurred and the difference between the electron binding energy and the secondary particle (K-X ray) energy is deposited. This is enabled only when the EII option is turned on.

Table B.19: `IARG` values, `IAUSFL` indices, and program status for `AUSGAB` calls.

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 Møller 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.
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	An EII interaction is about to occur.
26	27	Returned to `MOLLER` after a call to `EII` was made.
27	28	An energy hinge is about to occur in `ELECTR`.
28	29	An energy hinge has just occurred in `ELECTR`.
29	30	A multiple-scattering hinge is about to occur in `ELECTR`.
30	31	A multiple-scattering hinge has just occurred in `ELECTR`.

As an example of how to write an `AUSGAB` subprogram, consider the previous three region geometry of Figure B.2. Suppose that we wish to score only photons that emanate 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)
      implicit none
      include 'include/egs5_h.f'              ! Main EGS "header" file
      include 'include/egs5_stack.f'
      integer iarg                                        ! Arguments
      if(iarg.eq.3.and.iq(np).eq.0.and.ir(np).eq.3) then
        write(6,1000)e(np),x(np),y(np),z(np),u(np),v(np),w(np),
     1    iq(np),ir(np),iarg
        end if
1000  format(7g15.7,3i5)
      return
      end
```

B.7 UCSAMPL5 — An Example of a "Complete" EGS5 User Code

The following user code, called UCSAMPL5, simulates electro-magnetic cascade showers initiated by 1 GeV electrons that are incident (normally) on a 3 cm, semi-infinite slab of iron. The upstream region of the slab is vacuum and the downstream region is air at NTP. A particle is discarded whenever it leaves the slab (on either side), or whenever its total energy falls below a preset cutoff energy of 100 MeV. (Note that the medium assigned to Region 3 is really not important in this example, and was included solely for purposes of illustration.) Some of the stack variable information E(NP), Z(NP), W(NP), IQ(NP), IR(NP), plus the IARG value, is printed out on the printer (first 15 lines only) for photons reaching Region 3.

A total of 10 cases of incident electrons are run and the total energy fraction for each region is summed and printed out at the end of the run for an energy balance check.

The UCSAMPL5 user code is given below.

```
!***********************************************************************
!
!                     *************
!                     *           *
!                     * ucsampl5.f *
!                     *           *
!                     *************
!
! A complete example of a EGS5 user code, using a simple plane
! geometry.  For SLAC-R-730/KEK Report 2005-8.
!
! This user code corresponds to ucsampl4.mor for egs4.
! The following shows the geometry
!***********************************************************************
!                                                                     *
!          ------------------------------------------------           *
!          1-Dimensional Plane Z Geometry (ucsampl5 example)          *
!          ------------------------------------------------           *
!                                                                     *
!              Y (X into page)                                        *
!              ^                                                      *
!              |                                                      *
!              |    |                                                 *
!              | Fe |    Air                                          *
!              |    |                                                 *
!              |    |                                                 *
!     1 GeV    |    |                                                 *
!      =========>+----+----------------------> Z                      *
!     electron  0    3.0                                              *
!                                                                     *
```

```fortran
!*********************************************************************
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
!---------------------------------------------------------------------
!---------------------------- main code ------------------------------
!---------------------------------------------------------------------


!---------------------------------------------------------------------
!     Step 1. Initialization
!---------------------------------------------------------------------

      implicit none

!     ------------
!     EGS5 COMMONs
!     ------------
      include 'include/egs5_h.f'                   ! Main EGS "header" file

      include 'include/egs5_bounds.f'
      include 'include/egs5_edge.f'
      include 'include/egs5_media.f'
      include 'include/egs5_misc.f'
      include 'include/egs5_thresh.f'
      include 'include/egs5_useful.f'
      include 'include/egs5_usersc.f'
      include 'include/egs5_userxt.f'
      include 'include/randomm.f'

!     ---------------------
!     Auxiliary-code COMMONs
!     ---------------------
      include 'auxcommons/lines.f'

      common/passit/zthick
      real*8 zthick

      common/totals/esum(3)
      real*8 esum

      real*8 ei,ekin,etot,totke,xi,yi,zi,   ! Arguments
     *       ui,vi,wi,wti
      real tarray(2)

      real t0,t1,timecpu,tt                 ! Local variables
      real etime
      integer i,idinc,iqi,iri,j,ncases
      character*24 medarr(2)

!     ----------
!     Open files
!     ----------
      open(UNIT= 6,FILE='egs5job.out',STATUS='unknown')
```

350

```
!       ==================
        call counters_out(0)
!       ==================


!----------------------------------------------------------------------
! Step 2: pegs5-call
!----------------------------------------------------------------------
!       =============
        call block_set                  ! Initialize some general variables
!       =============


!       --------------------------------
!       define media before calling PEGS5
!       --------------------------------

        nmed=2
        medarr(1)='FE-RAYLEIGH             '
        medarr(2)='AIR AT NTP              '

        do j=1,nmed
          do i=1,24
            media(i,j)=medarr(j)(i:i)
          end do
        end do

        chard(1) = 3.0
        chard(2) = 3.0
!       ----------------------------
!       Run PEGS5 before calling HATCH
!       ----------------------------
        write(6,100)
100     FORMAT(' PEGS5-call comes next')

!       =============
        call pegs5
!       =============



!----------------------------------------------------------------------
! Step 3: Pre-hatch-call-initialization
!----------------------------------------------------------------------

        med(1)=0
        med(2)=1
        med(3)=2


!       --------------------------------
!       Set of option flag for region 2-3
!       1: on, 0: off
!       --------------------------------
```

351

```
        nreg=3

        do i=2,nreg
          ecut(i)=100.0        ! egs cut off energy for electrons
          pcut(i)=100.0        ! egs cut off energy for photons
          iphter(i) = 0        ! Switches for PE-angle sampling
          iedgfl(i) = 0        ! K & L-edge fluorescence
          iauger(i) = 0        ! K & L-Auger
          iraylr(i) = 0        ! Rayleigh scattering
          lpolar(i) = 0        ! Linearly-polarized photon scattering
          incohr(i) = 0        ! S/Z rejection
          iprofr(i) = 0        ! Doppler broadening
          impacr(i) = 0        ! Electron impact ionization
        end do

!       ----------------------------------------------------------
!       Random number seeds.  Must be defined before call hatch.
!       ins (1- 2^31)
!       ----------------------------------------------------------
        inseed=1
        luxlev=1


!       ============
        call rluxinit   ! Initialize the Ranlux random-number generator
!       ============


!-----------------------------------------------------------------------
! Step 4:  Determination-of-incident-particle-parameters
!-----------------------------------------------------------------------
        iqi=-1
        xi=0.0
        yi=0.0
        zi=0.0
        ui=0.0
        vi=0.0
        wi=1.0
        iri=2
        wti=1.0
        ncases=1000
        idinc=-1
        ei=1000.D0
        ekin=ei+iqi*RM


!-----------------------------------------------------------------------
! Step 5:   hatch-call
!-----------------------------------------------------------------------
! Total energy of incident source particle must be defined before hatch
! Define posible maximum total energy of electron before hatch
        if (iqi.ne.0) then
          emaxe = ei                  ! charged particle
        else
```

```
            emaxe = ei + RM           ! photon
        end if

!       ----------------------------
!       Open files (before HATCH call)
!       ----------------------------
        open(UNIT=KMPI,FILE='pgs5job.pegs5dat',STATUS='old')
        open(UNIT=KMPO,FILE='egs5job.dummy',STATUS='unknown')

        write(6,130)
130     FORMAT(/,' HATCH-call comes next',/)

!       =========
        call hatch
!       =========

!       ----------------------------
!       Close files (after HATCH call)
!       ----------------------------
        close(UNIT=KMPI)
        close(UNIT=KMPO)

! -----------------------------------------------------------
! Print various data associated with each media (not region)
! -----------------------------------------------------------
        write(6,140)
140     FORMAT(/,' Quantities associated with each MEDIA:')
        do j=1,nmed
          write(6,150) (media(i,j),i=1,24)
150       FORMAT(/,1X,24A1)
          write(6,160) rhom(j),rlcm(j)
160       FORMAT(5X,' rho=',G15.7,' g/cu.cm     rlc=',G15.7,' cm')
          write(6,170) ae(j),ue(j)
170       FORMAT(5X,' ae=',G15.7,' MeV    ue=',G15.7,' MeV')
          write(6,180) ap(j),up(j)
180       FORMAT(5X,' ap=',G15.7,' MeV     up=',G15.7,' MeV',/)
        end do

!-------------------------------------------------------------------------
! Step 6:  Initialization-for-howfar
!-------------------------------------------------------------------------
        zthick=3.0
!       plate is 3 cm thick

!-------------------------------------------------------------------------
! Step 7:  Initialization-for-ausgab
!-------------------------------------------------------------------------
        do i=1,nreg
          esum(i)=0.D0
        end do
```

```
      nlines=0
      nwrite=15


!-----------------------------------------------------------------------
! Step 8:  Shower-call
!-----------------------------------------------------------------------
      tt=etime(tarray)
      t0=tarray(1)

      write(6,190)
190   format(/,' Shower Results:',///,7X,'e',14X,'z',14X,'w',10X,
     1   'iq',3X,'ir',2X,'iarg',/)

      do i=1,ncases

        if (nlines.lt.nwrite) then
        write(6,200) i,ei,zi,wi,iqi,iri,idinc
200     format(i2,3G15.7,3I5)
        nlines=nlines+1
        end if

        call shower(iqi,ei,xi,yi,zi,ui,vi,wi,iri,wti)

      end do

      tt=etime(tarray)
      t1=tarray(1)

      timecpu=t1-t0
      write(6,210) timecpu
210   format(/,' Elapsed Time (sec)=',1PE12.5)


!-----------------------------------------------------------------------
! Step 9:  Output-of-results
!-----------------------------------------------------------------------
      totke=ncases*ekin
      write(6,220) ei,zthick,ncases
220   format(//,' Incident total energy of electron=',F12.1,' MeV',/, '
     *Iron slab thickness=',F6.3,' cm',/, ' Number of cases in run=',I7,
     *//,' Energy deposition summary:',/)

      etot=0.D0
      do i=1,nreg
        etot=etot+esum(i)
        esum(i)=esum(i)/totke
        write(6,230) i, esum(i)
230     format(' Fraction in region ',I3,'=',F10.7)
      end do

      etot=etot/totke
      write(6,240) etot
```

```
240   FORMAT(//,' Total energy fraction in run=',G15.7,/,
     *'    Which should be close to unity')
!     -----------
!     Close files
!     -----------
      close(UNIT=6)

      stop
      end
!-----------------------last line of main code-----------------------
!------------------------------ausgab.f------------------------------
! Version:    050701-1615
! Reference: SLAC-R-730, KEK-2005-8 (Appendix 2)
!-------------------------------------------------------------------
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
! ------------------------------------------------------------------
! Required subroutine for use with the EGS5 Code System
! ------------------------------------------------------------------
! A simple AUSGAB to:
!
!   1) Score energy deposition
!   2) Print out stack information
!   3) Print out particle transport information (if switch is turned on)
!
! ------------------------------------------------------------------

      subroutine ausgab(iarg)

      implicit none

      include 'include/egs5_h.f'                  ! Main EGS "header" file

      include 'include/egs5_epcont.f'   ! COMMONs required by EGS5 code
      include 'include/egs5_stack.f'
      include 'auxcommons/lines.f'

      common/totals/esum(3)
      real*8 esum

      integer iarg                                          ! Arguments

!     --------------------
!     Add deposition energy
!     --------------------

      esum(ir(np))=esum(ir(np)) + edep


!     ------------------------------------------------------------------
!     Print out stack information (for limited number cases and lines)
!     ------------------------------------------------------------------
```

```
      if (nlines.lt.nwrite) then
        write(6,1240) e(np),z(np),w(np),iq(np),ir(np),iarg
1240    FORMAT(3G15.7,3I5)
        nlines=nlines+1
      end if

      return
      end
!-------------------------last line of ausgab.f-----------------------
!--------------------------howfar.f------------------------------
! Version:   050701-1615
! Reference: SLAC-R-730, KEK-2005-8 (Appendix 2)
!----------------------------------------------------------------------
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
! ---------------------------------------------------------------------
! Required (geometry) subroutine for use with the EGS5 Code System
! ---------------------------------------------------------------------
! This is a 1-dimensional plane geometry.
! ---------------------------------------------------------------------

      subroutine howfar

      implicit none

      include 'include/egs5_h.f'              ! Main EGS "header" file

      include 'include/egs5_epcont.f'    ! COMMONs required by EGS5 code
      include 'include/egs5_stack.f'

      common/passit/zthick
      real*8 zthick

      real*8 deltaz                     ! Local variables
      integer irnxt

      if (ir(np).ne.2) then
        idisc = 1
        return
      end if

      dnear(np) = dmin1(z(np),zthick-z(np))

!---------------------------------
! Particle going parallel to planes
!---------------------------------
      if(w(np).eq.0) return


!--------------------------------------------------------
! Check forward plane first since shower heading that way
! most of the time
!--------------------------------------------------------
```

```fortran
      if (w(np).gt.0.0) then
        deltaz=(zthick-z(np))/w(np)
        irnxt=3
!-------------------------------------------------------------
! Otherwise, particle must be heading in backward direction.
!-------------------------------------------------------------
      else
        deltaz=-z(np)/w(np)
        irnxt=1
      end if

      if (deltaz.le.ustep) then
        ustep=deltaz
        irnew=irnxt
      end if

      return
      end
!-------------------------last line of howfar.f------------------------
```

Appendix C

PEGS USER MANUAL

Hideo Hirayama and Yoshihito Namito
Radiation Science Center
Advanced Research Laboratory
High Energy Accelerator Research Organization (KEK)
1-1 Oho Tsukuba-shi Ibaraki-ken 305-0801 JAPAN

Alex F. Bielajew and Scott J. Wilderman
Department of Nuclear Engineering and Radiological Sciences
The University of Michigan
2355 Bonisteel Boulevard
Ann Arbor, MI 48109, USA

Walter R. Nelson
Department Associate in the Radiation Physics Group (retired)
Radiation Protection Department
Stanford Linear Accelerator Center
2575 Sand Hill Road Menlo Park, CA 94025, USA

C.1 Introduction

PEGS (Preprocessor for EGS) is a set of FORTRAN subprograms which generate material data for use with the EGS5 Monte Carlo shower code and which also provide utilities for researchers studying 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 printed plots of selected functions.

- Evaluation of functions at selected points.

- Comparison of functions with sampled spectra.

Associated with these active functionals are other operations:

- Selection of material to which the functions refer.

- Selection of energy cutoffs for fits.

- Printing of fitted data.

This manual describes the full general functionality of PEGS. Users interested only in preparing data sets for EGS5 can proceed directly to section C.3.

C.2 Structural Organization of PEGS

Beginning with version 5 of the EGS code system, MORTRAN is no longer supported, and PEGS is no longer a stand-alone program, but rather a subroutine to be called by EGS user code. The main PEGS subroutine (called `PEGS5`), however is essentially the same as the `MAIN` program in previous versions of PEGS, and EGS and PEGS are still quite distinct in EGS5.

The full PEGS5 system consists of almost 7000 of FORTRAN source code that make up over 100 subprograms, plus an additional 610 lines of FORTRAN in 40 files containing `COMMON` blocks accessed by the various PEGS subprograms through `include` statements. New to version 5 of PEGS, all floating point variables are declared as `double precision`.

Despite its size, PEGS has a simple structure. Figures C.1 and C.2 below contain a schematic flowchart of main subroutine (`PEGS5`) of PEGS. After the once-only initializations, an option loop

is entered. On each pass through this loop, an option is read (option names are four characters and are read as 4A1), numeric control parameters are read (using the FORTRAN `NAMELIST` I/O extension), and then the option name is looked up in the option table. If an option name is not found, the job is aborted. When an option name is found, the appropriate code is executed and the program returns 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 input file. Detailed descriptions of the use of the options are contained in section C.3 of this manual.

Figures C.3 and C.4 show some of the subprogram relationships of PEGS[1]. Boxed items in the figures are subprograms, and option names (identified by ":" delimiters, as in ":CALL:") are used to show which subprograms correspond to which options. The general structure of PEGS is clearly evident. Subprograms which compute physical quantities are accessed directly by the PWLF option, but when the utility options TEST, PLTN, PLTI, HPLT, or CALL are invoked, physics routines are referenced using the function `FI`, the so-called "function multiplexer." The function multiplexer `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. Calls to `FI` return values returned to `FI` by the called function specified by `I`. The full list of physical functions which can be called by `FI` is given in Figure C.4.

It should also be noted that there are relationships between the PEGS functions shown in Figure C.3 which are not indicated there. We show the most complicated of these in Figures C.5 and C.6 (Bremsstrahlung Related Functions) and in Figures C.7 and C.8 (Pair Production Related Functions). One reason for the complexity of these particular subprograms is that higher level forms of the relevant cross sections must be obtained by numerical integration of the more differential forms.

Tables C.1 and C.2 list the subroutines used in PEGS and provide brief descriptions of their functions. All of the subroutines in Table C.2 are new to PEGS5 and deal with either the calculation of the Goudsmit-Saunderson multiple scattering distribution or the determination of electron energy loss steps. Note that in the current implementation, when the Goudsmit-Saunderson distribution is requested, the routines which prepared the data are actually called by EGS during the set up operations performed in `HATCH`. We nevertheless refer to these new subprograms as part of PEGS, as they do involve pre-processing of data.

Tables C.3 through C.6 list the FORTRAN `FUNCTIONS` used in PEGS along with their mathematical symbols and definitions if appropriate. The names of most of the functions have been chosen in a 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

---

[1]Subprograms involving the low energy physics and the multiple scattering models new to PEGS with the release of EGS5 are not included in the diagrams.

```
                          +------+
                          | PEGS |
                          +------+
                            |  |          +-------------+
                initialize  |  +------>| OPTION LOOP |<---+
              +-------------+          +-------------+    A
              |                             |            |
              V                             V            |
+-----------------------+        +-----------------+     |
|    Compute Physical & |        |    Read Option  |     |
| Mathematical Constants|        |    Name (4A1)   |     |
+-----------------------+        |        &        |     |
              |                  |    Read Control |     |
              |                  |    Parameters   |     |
              V                  | (NAMELIST/INP/) |     |
+-----------------------+        +-----------------+     |
| Read Pair Production, |                |               |
|   Photoelectric, and  |                |               |
|   Rayleigh Data from  |                |               |
! PGS5PEPR and PGS5FORM  |                |               |
+-----------------------+                |               |
                                         V               |
          +------+      Yes      +--------------+         |
          | Stop |<------------ | End of File? |         |
          +------+               +--------------+        |
                                         |               |
                                         | No            |
                                         V               |
    +----------------------+      +--------------+        |
    | Illegal Option---Stop |<----- | Select Option |     |
    +----------------------+      +--------------+        |
                                         |               |
                                         V               |
      +<------------------------------------- +          |
      |                                                  |
      V                                                  |
   (to next figure)                   (from next figure)
```
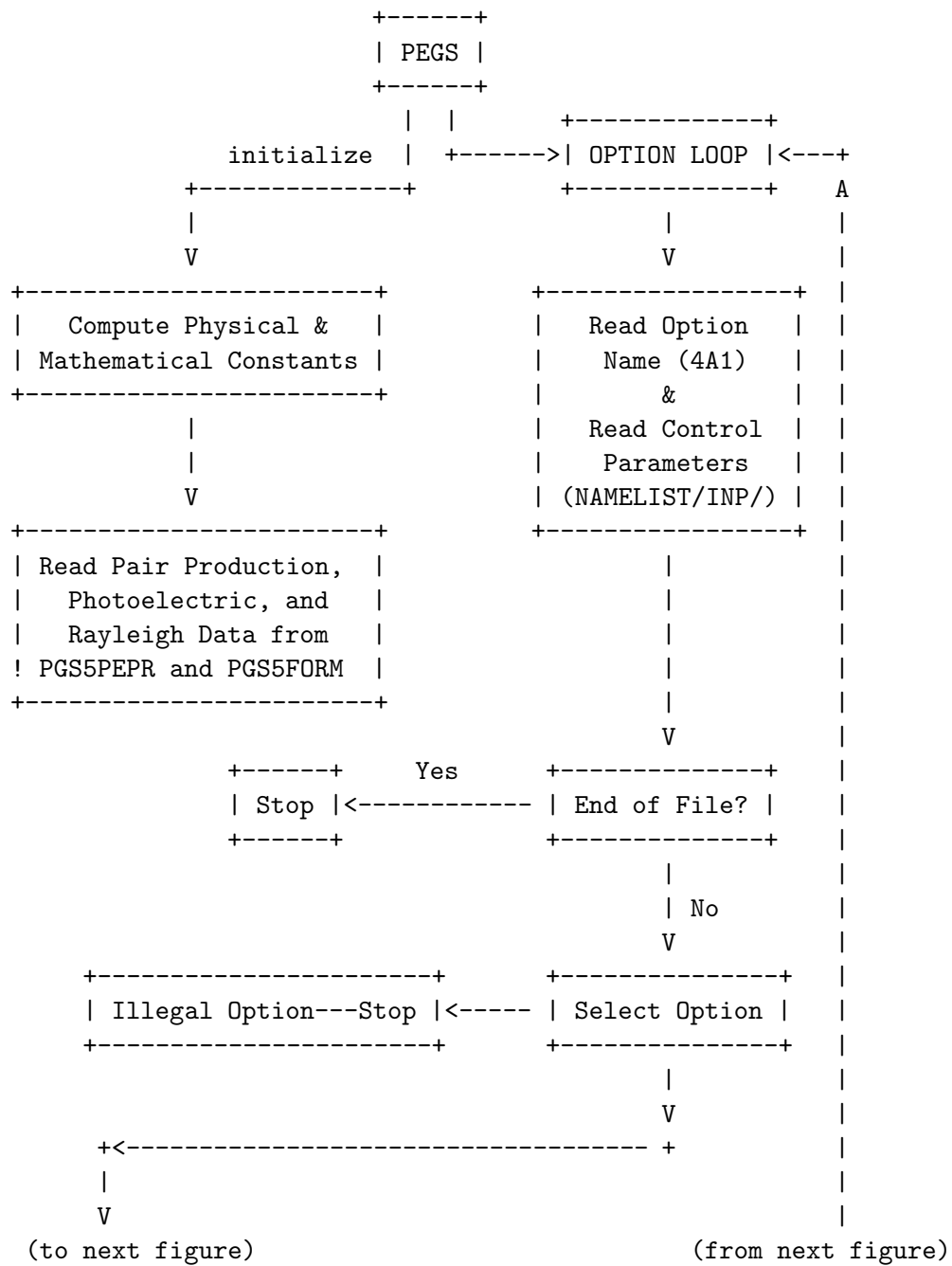
Figure C.1: Flowchart of the PEGS5 subprogram of PEGS, part 1.

```
(from previous figure)                                    (to previous figure)
    |                                                              A
    |              +----------------------+                        |
   +-->: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: | Print File of Material | ----------------->  +
    |              | Dependent Data          |
    |              +-----------------------+
   +-->:STOP:
```

Figure C.2: Flowchart of the `PEGS5` subprogram of PEGS, part 2.

```
+------------+          +------+
| BLOCK DATA |          | MAIN |
+------------+          +------+
                           |
    + --- + - + ----- + ----- + ----- + ----- + ---- + --- +
    |     |   |       |       |       |       |      |     |
    |     |        :PWLF:  :DECK:  :TEST:  :HPLT: :CALL: :ENER:
    |     |           |       |     :PLTN:    |      |
 +------+ |           |     +---+   :PLTI:  +-----+   |
 |PMDCON| |           |     |LAY|     |     |HPLT1|   |
 +------+ |           |     +---+     |     +-----+   |
    |     |           |             +----+    |       |
    :ELEM:     +---+----+           |PLOT|    |       |
    :MIXT:     |        |           +----+    |       |
    :COMP:  +-----+  +-----+          |       |       |
       |    |EBIND|  |PWLF1|        +----->  | <--- +
       |    +-----+  +-----+                  |
   + ----- + ----- +    |                     |
   |       |       |  +----+                   |
+---+ +------+ +------+ |QFIT|      (to next figure)
|MIX| |SPINIT| |DIFFER| +----+
+---+ +------+ +------+    |
                 + - + --- + -------------- +
                 |         |                |
              +-----+   +-----+          +-----+
              |EFUNS|   |GFUNS|          |RFUNS|
              +-----+   +-----+          +-----+
                 |         |                |
    +-----+----+-----+--+---+---+ |   +------+    +-----+
    |     |    |     |  |   |   | +---|PHOTTE|    |AINTP|
 +------+ | +------+ | +------+ | |   +------+    +-----+
 |SPTOTP| | |ANIHTM| | |BHABTM| | |   +------+
 +------+ | +------+ | +------+ | +---|COMPTM|
    |     |    |     |  |   |   | |   +------+
 +------+ | +------+ | +------+ | |   +------+
 |SPTOTE|-+ |AMOLTM|-+ |BREMTM|-+ +---|PAIRTU|
 +------+   +------+   +------+  |   +------+
                                |   +------+
                            +---|COHETM|
                                +------+
```
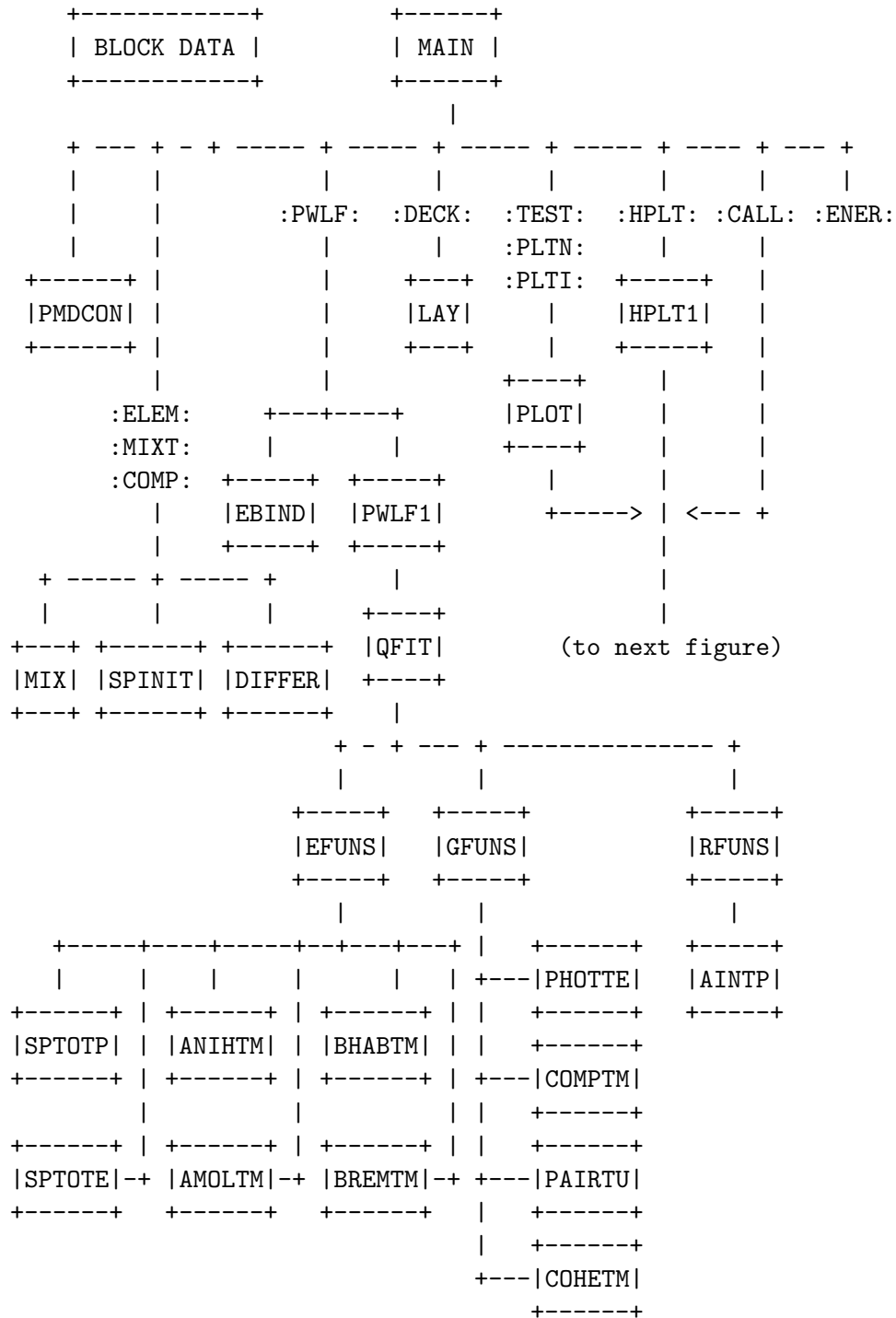
Figure C.3: Subprogram relationships in PEGS, part 1.

```
              (from previous figure)
                       |
                       |
          + --------------------- +
          |          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    |
  |   DLOG      BREMFR    EDEDX     PBR2    |
  |   DEXP      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    EEITM             |
  |             COHETZ                      |
  |             COMPDM                      |
  + ------------------------------------- +
```

Figure C.4: Subprogram relationships in PEGS, part 2.

```
   +------+                +------+
   |BREMTM|---------->|BREMRM|
   +------+                +------+
                              |
                              V
 +------+                +------+  initialize  +------+
 |  QD  |<------------- |BREMRZ|-------------> |BREMDZ|
 +------+                +------+    BREMFZ     +------+
    |                                              |
    V                                              V
 +------+   +------+   +------+                +------+
 |DCADRE|-->|BREMFZ|--->|BRMSFZ|<------------- |BRMSDZ|
 +------+   +------+   +------+                +------+
                          A                      A |
                          |                      | |   +------+
             +---------------+                   | +-->|APRIM |
             |                                   | |   +------+
       +------+        +------+  initialize      | |
       |DCADRE|   +----|BRMSRZ|--------------> + |   +------+
       +------+   |    +------+     BRMSFZ        +-->|  XSIF |
          A       |       A                       |   +------+
          |       |       |                       |
       +------+   |    +------+                    |   +------+
       |  QD  |<--+    |BRMSRM|                    +-->|FCOULC|
       +------+        +------+                        +------+
                          A
                          |
   +------+            +------+              +------+
   |SPTOTE|----------> |BRMSTM|<----------- |SPTOTP|
   +------+            +------+              +------+
      |                                         |
      V                                         V
   +------+            +------+              +------+
   |SPIONE|----------> |SPIONB|<----------- |SPIONP|
   +------+            +------+              +------+
```
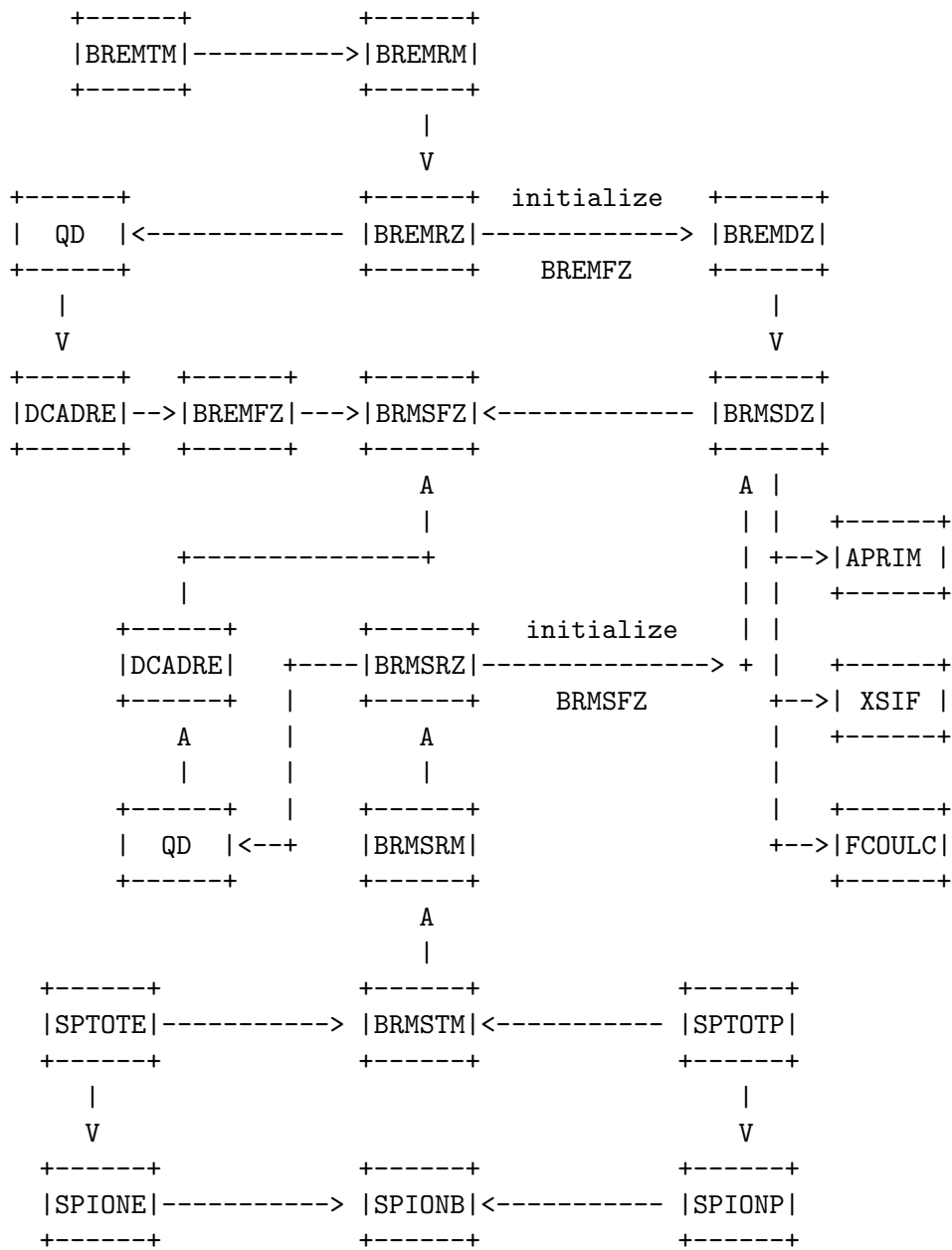
Figure C.5: Bremsstrahlung related functions—most accurate form (used to produce the total cross sections and stopping power).

```
                +------+
                |BREMTR|
                +------+
                   |
                   V
                +------+
                |BREMRR|
                +------+
                   |
          initialize V
        + <--------+--------> +
        |   BREMFR            |
        V                     V
    +------+              +------+
    |BREMDR|              |  QD  |
    +------+              +------+
        |                     |
        |                     V
        |                 +------+
        |                 |DCADRE|
        |                 +------+
        |     +------+        |
        + ---> |BREMFR| <---- +
              +------+
```
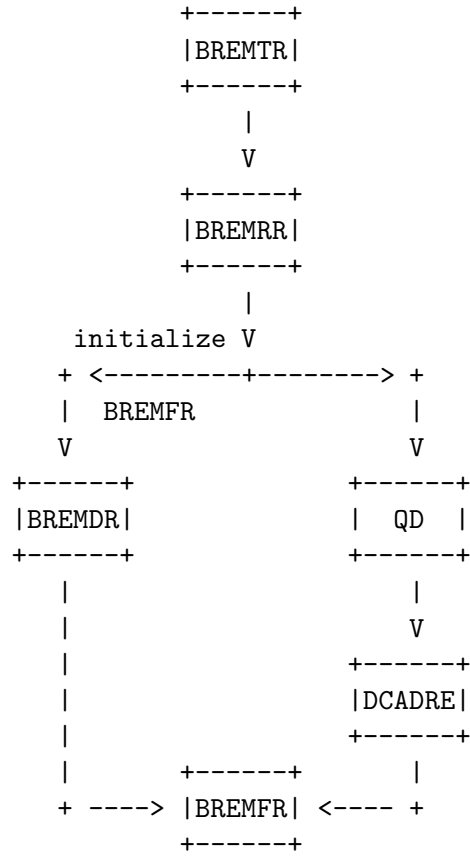
Figure C.6: Bremsstrahlung related functions—with run-time approximations (for comparison with sampled spectra).

```
                            +------+
                            |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|
     +------+           +------+
```
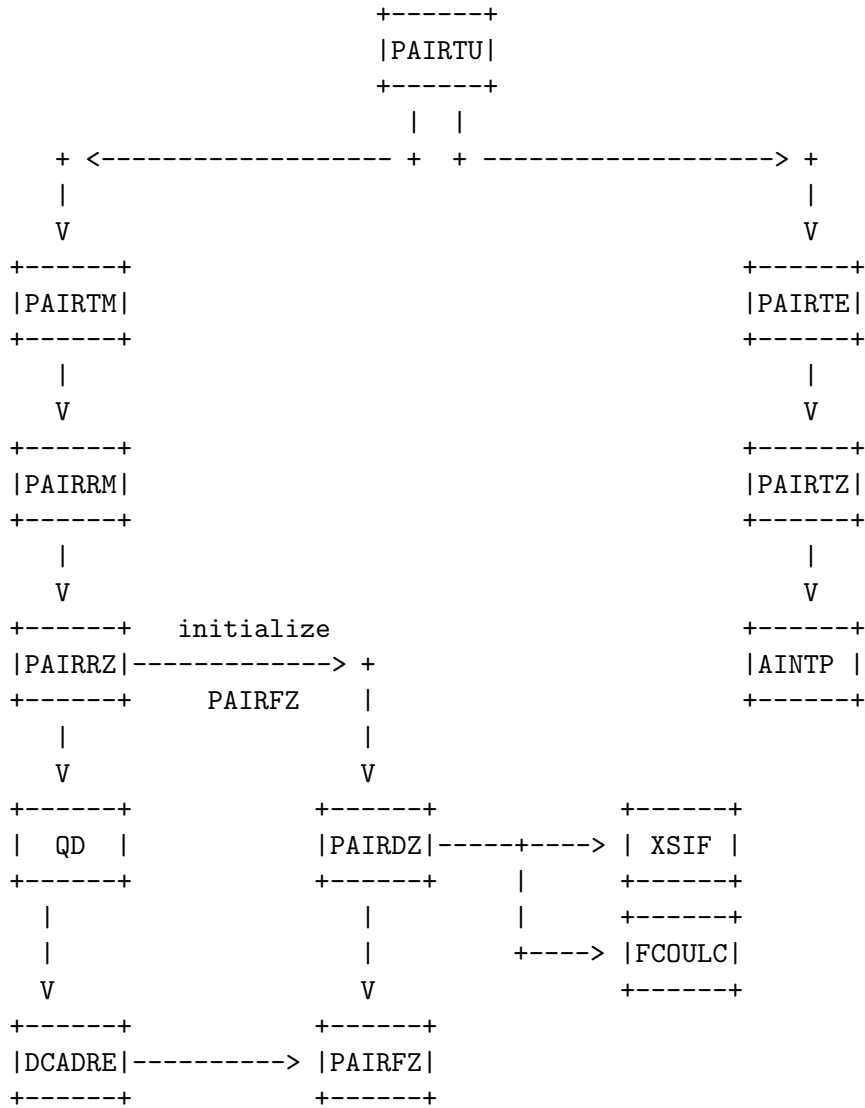
Figure C.7: Pair production related functions—most accurate form (used to produce the total cross sections and stopping power).

```
                    +------+
                    |PAIRTR|
                    +------+
                       |
                       V
                    +------+
                    |PAIRRR|
                    +------+
                       |
          initialize V
          + <-------- + ------> +
          |   PAIRFR            |
          V                     V
       +------+              +------+
       |PAIRDR|              |  QD  |
       +------+              +------+
          |                     |
          |                     V
          |                  +------+
          |                  |DCADRE|
          |                  +------+
          |     +------+        |
          + ----> |PAIRFR| <---- +
                +------+
```
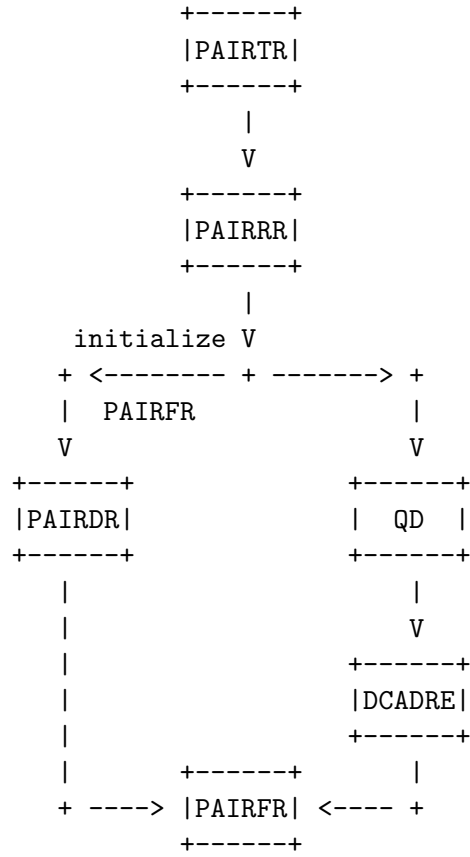
Figure C.8: Pair production related functions—with run-time approximations (for comparison With sampled spectra).

Table C.1: Subroutines in PEGS.

Name	Description
DIFFER	Determines the various parameters needed for bremsstrahlung and pair production energy sampling.
EFUNS	Subprogram to compute electron functions to be fit in a way that avoids repetition.
GFUNS	Subprogram to compute photon functions to be fit in a way that avoids repetition.
HPLT1	Creates line printer plot comparisons of EGS-sampled data (UCTESTSR User Code) and theoretical functions of PEGS.
LAY	Subprogram to produce a file of material dependent data (for subsequent use by EGS).
MIX	Computes Z-dependent parameters that reside in COMMON/MOLVAR/.
PLOT	Subprogram to plot a given function (referenced by number).
PMDCON	Determines the physical, mathematical, and derived constants in a very mnemonic way.
PWLF1	Subprogram to piecewise linearly fit up to 10 functions simultaneously on an interval (XL,XU).
RFUNS	Subprogram to compute Rayleigh scattering functions to be fit in a way that avoids repetition.
SPINIT	Initializes stopping power functions for a particular medium.
SFUNS	Subroutine to compute incoherent scattering functions to be fit.
CFUNS3	Subroutine to compute c.d.f. of shell-wise Compton profile to be fit.
CFUNS4	Subroutine to compute shell-wise Compton profile to be fit.
EIIFUNS	Subroutine to compute EII/Møller ratio to be fit.
MAKEK1	Prepares tables of initial scattering strength to be fit.

Table C.2: Goudsmit-Saunderson-related subroutines in PEGS.

Name	Description
DCSLOAD	Loads molecular elastic differential cross sections tables for multi-material PEGS runs.
DCSSTOR	Stores molecular elastic differential cross sections tables for multi-material PEGS runs.
DCSTAB	Computes tables of the molecular elastic differential cross sections for electrons and positrons.
ELASTINO	Prepares cumulative Goudsmit-Saunderson multiple scattering distributions on problem dependent energy and scattering strength grids.
ELINIT	Reads differential elastic scattering cross section data for electrons and positrons.
ESTEPLIM	Computes the maximum electron energy hinge step (in units of fractional energy loss) which assures a given error tolerance (currently fixed at 0.1%) in the total energy loss and total scattering power accumulated over the step.
FINDI	Does table look-up.
FITMS	Prepares special fit to Goudsmit-Saunderson multiple scattering data for fast sampling in EGS.
GAULEG	Provides abscissas and weights for Gauss-Legendre quadrature.
GSCOEF	Computes Goudsmit-Saunderson transport coefficients.
GSDIST	Computes Goudsmit-Saunderson multiple scattering probability density functions.
INIGRD	Initializes energy and scattering strength grid tables.
INTEG	Integrates cubic spline data.
LEGENP	Computes Legendre polynomials.
PRELASTINO	Writes multiple scattering grid parameters for later consistency checks.
SPLINE	Computes cubic splines.
WMSFIT	Writes fitted Goudsmit-Saunderson multiple scattering data parameters.

convention) the word is prefixed with the letter A. A few examples are given below:

AMOLDM	is the differential Møller 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.

Note that 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.

C.3 PEGS Options and Input Specifications

C.3.1 Interrelations Between Options

Figure C.9 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 option is seen to be independent of the others.

In the following sections we give for each option its function, parameters which control it, the format of the data lines in a PEGS input file needed to invoke it, and an explanation of the routines (if any) that are used to implement it. The data lines for a given option are named with the first part of their name being the option name, and the last part the line number. For example, "MIXT2" is the name of the second line needed for the MIXT option. The information is summarized in Tables C.7 through C.18. The single line referred to as being read by NAMELIST may in fact be several lines, provided that the proper convention for continuing NAMELIST lines is followed. Once the first line (indicating the option) has been read in, however, the second line (i.e., NAMELIST/INP/) must follow (see examples at the end of the next section). It should also be noted that different compilers may require different formats for NAMELIST data.

Table C.3: Functions in PEGS, part 1.

Name	Description
AFFACT	Atomic form factor (squared) for an element or mixture of elements.
AINTP	Linear or log interpolation function.
ALKE	Log of kinetic energy (`LOG(E-RM)`), used as a cumulative distribution function for fits and plots.
ALKEI	Inverse of ALKE (`=EXP(X)+RM`).
ALIN	Linear cumulative distribution function for plots (`ALIN(X)=X`).
ALINI	Inverse of ALIN (=same as ALIN). Used as inverse cumulative distribution function in plots.
ADFMOL	Approximate cumulative distribution function for Møller and Bhabha cross sections (`ADFMOL(E)=-1/(E-RM)`).
ADIMOL	Inverse of ADFMOL.
ADDMOL	Derivative of ADFMOL.
AMOLDM	Møller differential cross section for a mixture of elements.
AMOLFM	One argument form of AMOLDM.
AMOLRM	Møller cross section, integrated over some energy range, for a mixture of elements.
AMOLTM	Møller total cross section for a mixture of elements.
ANIHDM	Annihilation differential cross section for a mixture of elements.
ANIHFM	One argument form of ANIHDM.
ANIHRM	Annihilation cross section, integrated over some energy range, for a mixture of elements.
ANIHTM	Annihilation total cross section for a mixture of elements.
APRIM	Empirical correction factor in bremsstrahlung cross section.
AREC	Reciprocal function (=derivative of `ALOG(X)`). Used as probability density function in log plots (`AREC(X)=1/X`).
BHABDM	Bhabha differential cross section for a mixture of elements.
BHABFM	One argument form of BHABDM.
BHABRM	Bhabha cross section, integrated over some energy range, for a mixture of elements.
BHABTM	Bhabha total cross section for a mixture of elements.

Table C.4: Functions in PEGS, part 2.

Name	Description
BREMDR	Bremsstrahlung differential cross section for a run-time mixture of elements.
BREMFR	One argument form of BREMDR.
BREMDZ	Bremsstrahlung differential cross section for an element.
BREMFZ	One argument form of BREMDZ.
BREMRM	Bremsstrahlung cross section, integrated over some energy range, for a mixture of elements.
BREMRR	Bremsstrahlung cross section, integrated over some energy range, for a run-time mixture of elements.
BREMRZ	Bremsstrahlung cross section, integrated over some energy range, for an element.
BREMTM	Bremsstrahlung total cross section for a mixture of elements.
BREMTR	Bremsstrahlung total cross section for a run-time mixture of elements.
BRMSDZ	Soft bremsstrahlung differential cross section for an element.
BRMSFZ	One argument form of BRMSDZ.
BRMSRM	Soft bremsstrahlung cross section, integrated over some energy range, for a mixture of elements.
BRMSRZ	Soft bremsstrahlung cross section integrated over some energy range, for an element.
BRMSTM	Soft bremsstrahlung total cross section for a mixture of elements.
COHETM	Coherent (Rayleigh) total cross section for a mixture of elements.
COHETZ	Coherent (Rayleigh) total cross section for an element.
COMPDM	Compton differential cross section for a mixture of elements.
COMPFM	One argument form for COMPDM.
COMPRM	Compton cross section, integrated over some energy range, for a mixture of elements.
COMPTM	Compton total cross section for a mixture of elements.
CPRFIL	Shell-wise Compton profiles.
CRATIO	Coherent (Rayleigh) cross section ratio.
CSDAR	Returns previously computed and stored values of CSDA range for electrons and positrons.
ESIG	Determines the total electron(-) interaction cross section (probability per radiation length).
DCADRE	Quadrature routine to integrate f(x) between a and b using cautious Romberg extrapolation.
DCSEL	Computes differential elastic scattering cross sections from data which has been fitted by cubic splines.
DCSN	Computes arbitrary Goudsmit-Saunderson elastic scattering transport coefficients from cross section data which has been fitted by cubic splines.
EBIND	Function to get an average photo-electric binding energy.
EBR1	Function to determine the electron(-) branching ratio (brem/total).
EDEDX	Evaluates SPTOTE with cutoff energies of AE and AP.

Table C.5: Functions in PEGS, part 3.

Name	Description
EIITM	K shell EII cross section
ESTEPMAX	Returns previously computed and stored values of maximum electron energy hinge steps.
FCOULC	Coulomb correction term in pair production and bremsstrahlung cross sections.
FI	Function multiplexer.
G1E	Computes scatter powering for positrons and electrons from either splines of partial wave cross sections or analytical form of Molière cross section.
G1EDEDX	Computes product of first elastic transport cross section and inverse stopping power for electrons.
G1PDEDX	Computes product of first elastic transport cross section and inverse stopping power for positrons.
GBR1	Function to determine the gamma-ray branching ratio (pair/total).
GBR2	Function to determine the gamma-ray branching ratio ((pair+Compton)/total).
GMFP	Function to determine the gamma-ray mean free path.
K1E	Computes scattering strength for electrons and positrons from spline data.
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.
PAIRDZ	Pair production differential cross section for an element.
PAIRFR	One argument form of PAIRDR.
PAIRFZ	One argument form of PAIRDZ.
PAIRRM	Pair production cross section, integrated over some energy range, for a mixture of elements.
PAIRRR	Pair production cross section, integrated over some energy range, for a run-time mixture of elements.
PAIRRZ	Pair production cross section, integrated over some energy range, for an element.
PAIRTE	Empirical total pair production production cross section for a mixture (=SUM(PZ(I)*PAIRTZ(Z(I))).
PAIRTM	Pair production total cross section for a mixture of elements, obtained by numerical integration of differential cross section.
PAIRTR	Pair production total cross section for a run-time mixture of elements.
PAIRTU	Pair production total cross section actually used. Same as PAIRTE for primary energy less than 50 MeV; otherwise, same as PAIRTM.
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 Storm-Israel data.

Table C.6: Functions in PEGS, part 4.

Name	Description
PBR1	Function to determine the positron branching ratio (brem/total).
PBR2	Function to determine the positron branching ratio ((brem+Bhabha)/total).
PDEDX	Evaluates SPTOTP with cutoff energies of `AE` and `AP`.
PHOTTE	Determines the proper mix of PHOTTZ's for a mixture.
PHOTTZ	Determines the interpolated total photoelectric cross section from tabulated data.
PSIG	Determines the total positron interaction cross section (probability per radiation length).
QD	Driver function for DCADRE, the numerical integration routine.
QFIT	Utility logical function for the piecewise linear fit subroutine, PWLF1. It returns `.TRUE.` if a given partition gives a good fit.
SUMGA	Integrates a function using a 20-point Gauss quadrature method with an adaptive bipartition scheme.
SPIONB	Does the work for SPIONE and SPIONP. One argument tells whether to compute stopping power for electron or positron.
SPIONE	Calculates the stopping power due to ionization for electrons(-).
SPIONP	Calculates the stopping power due to ionization for positrons.
SPTOTE	Calculates the total stopping power (ionization plus soft bremsstrahlung) for electrons(-) for specified cutoffs.
SPTOTP	Calculates the total stopping power (ionization plus soft bremsstrahlung) for positrons for specified cutoffs.
TMXB	Determines the maximum total step length consistent with Bethe's criterion.
TMXS	Determines the minimum of TMXB and 10 radiation lengths.
TMXDE2	Included for possible future modification purposes (`TMXDE2 = TMXB / (E**2 * BETA**4)`). It might be easier to fit this quantity than to fit TMXB and then apply the denominator in EGS at run-time.
XSIF	Function to account for bremsstrahlung and pair production in the field of the atomic electrons.
ZTBL	Given the atomic symbol for an element, it returns the atomic number.

```
+------+     +------+      +------+           +------+
|:ELEM:|     |:MIXT:|      |:COMP:|           |:ENER:|
+------+     +------+      +------+           +------+
   |            |            |                   |
   |            V            |                   |
   + ------> OR <-------- +                   |
                |                                 V
       +------------------------> AND ----+
                |                     |    |
                V                     |    |
     + -------+----+----+------- +    |    |
     |        |         |        |    |    |
     V        V         V        V    |    |
  +------+ +------+ +------+ +------+ +------+  |    |
  |:PLTN:| |:PLTI:| |:HPLT:| |:CALL:| |:TEST:|<---+  |
  +------+ +------+ +------+ +------+ +------+       |
                                                    V
                                              +------+
                                              |:PWLF:|
                                              +------+
                                                 |
                                                 V
  +------+                                    +------+
  |:STOP:|                                    |:DECK:|
  +------+                                    +------+
```
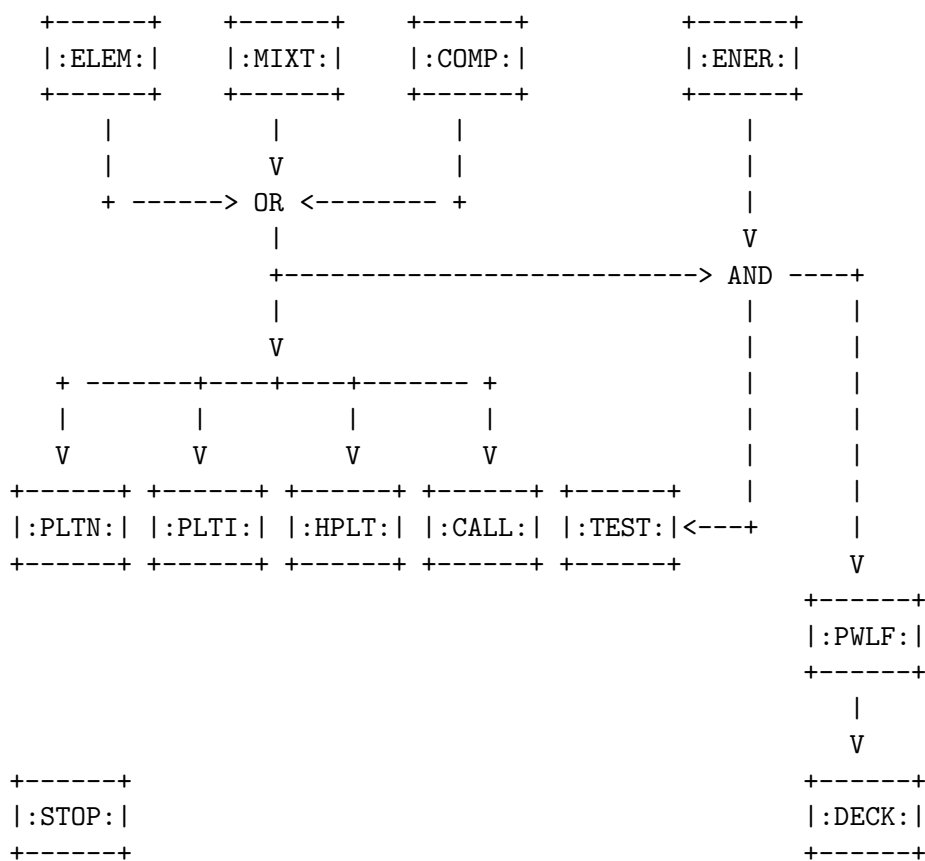
Figure C.9: Logical relationship between the options of PEGS.

C.3.2   The ELEM, MIXT, COMP Options

The purposes of the ELEMent, MIXTure, and COMPound options are to specify the materials 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)).

The ELEMent option is used if the material being used has only one type of atom. In this case PEGS defines that NE=1, takes the density from its internal data 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. If either RHO or WA(I) are specified by the user, they override 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.

Table C.7: ELEM option input data lines in PEGS, part 1.

Line and format	Variables	Comments
ELEM1 (4A1)	OPT(1:4)	'ELEM'. Means "select material that is an element."
ELEM2 NAMELIST	RHO	Optional. If given, this overrides the PEGS default density (g/cm^3) for the element.
	WA(1)	Optional. Atomic weight of element. If given, this overrides the PEGS default.
	IRAYL	Optional. Set to 1 (or 2, which requires additional data, see below) to model Rayleigh scattering. (default=0, ignore).
	IUNRST	Optional. For printing stopping powers (default = 0, restricted stopping powers, for EGS runs). 1 unrestricted collision only 2 unrestricted collision and radiative 3 unrestricted collision, restricted radiative . 4 restricted collision, unrestricted radiative 5 unrestricted radiative only 6 restricted radiative only 7 restricted collision only
	IAPRIM	Optional. For correction to bremsstrahlung cross section. Default (=1), normalizes integrated cross section to ICRU-37 radiative stopping power; set to 0 uses Motz *et al.*empirical; set to 2 uses no correction.
	ISSB	Optional. Set to 1 to use user-supplied density effect parameters (see below).
	EPSTFL	Optional. Set to 1 to use user-supplied density effect corrections (see below).
	FUDGEMS	Optional. Constant in $Z(Z + \xi_{MS})$ which describes the contribution of soft electron collisions to multiple scattering (default = 1.0).
	IBOUND	Optional. Set to 1 to use bound total Compton cross section (default = 0, use free Compton cross section).
	INCOH	Optional. Set to 1 to use incoherent scattering function to model electron binding effect on Compton scattering angles (default = 0, ignore).
	ICPROF	Optional. Set to 3,4,-3 or -4 (see below) to use shell-wise Compton profiles to get Compton scattering energies. (default = 0, ignore).

Table C.8: ELEM option input data lines in PEGS, part 2.

Line and format	Variables	Comments
ELEM2 (cont) `NAMELIST`	`IMPACT`	Optional. For selecting electron impact ionization cross section (default = 0, ignore): 1 Casnati's formula 2 Kolbenstvedt's rev. formula 3 Kolbenstvedt's formula 4 Jakoby's formula 5 Gryziński's formula 6 Gryziński's relativistic. formula
	`NLEG0`	Optional. Number of Legendre coefficients in computing Goudsmit-Saunderson distribution (default = 1000).
	`EFRACH`	Optional and not recommended. Fractional energy loss over a multiple scattering step at $E = $ `UE` (default = 0.05).
	`EFRACL`	Optional and not recommended. Fractional energy loss over a multiple scattering step at $E = $ `AE` (default = 0.20).
ELEM3 `(24A1,6X,24A1)`	`MEDIUM`	Identifier assigned to data set to be produced.
	`IDSTRN`	Optional. Identifier of media name of desired Sternheimer-Seltzer-Berger density effect coefficients in PEGS (see Table 2.3 of SLAC-R-730/KEK-2005-8 for a list of default media supplied with the EGS5 distribution.) If not specified, MEDIUM(1:24) is used.
ELEM4 `(24(A2,1X))`	`ASYM(1)`	Atomic symbol for element.

Table C.9: COMP option input data lines in PEGS.

Line and Format	Variables	Comments
COMP1 (4A1)	OPT(1:4)	'COMP'. Means "select material that is a compound."
COMP2 NAMELIST	NE	Number of elements in compound.
	RHO	Density (g/cm^3) of compound (at NTP for gases).
	(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 override default atomic weights (e.g., to allow for special isotopes).
	IRAYL	Same as for ELEM, line 2.
	IUNRST	Same as for ELEM, line 2.
	IAPRIM	Same as for ELEM, line 2.
	ISSB	Same as for ELEM, line 2.
	EPSTFL	Same as for ELEM, line 2.
	FUDGEMS	Same as for ELEM, line 2.
	IBOUND	Same as for ELEM, line 2.
	INCOH	Same as for ELEM, line 2.
	ICPROF	Same as for ELEM, line 2.
	IMPACT	Same as for ELEM, line 2.
	NELGO	Same as for ELEM, line 2.
	EFRACH	Same as for ELEM, line 2.
	EFRACL	Same as for ELEM, line 2.
COMP3 (24A1,6X,24A1)	MEDIUM	Same as for ELEM, line 3.
	IDSTRN	Same as for ELEM, line 3.
COMP4 (24(A2,IX))	(ASYM(I),I=1,NE)	Atomic symbols for the atoms in the compound. Duplicates are allowed if several isotopes of the same element are present, or may be required for diatomic molecules (e.g. nitrogen gas).

Table C.10: MIXT option input data lines in PEGS.

Line and Format	Variables	Comments
MIXT1 (4A1)	OPT(1:4)	'MIXT'. Means "select material that is a mixture."
MIXT2 NAMELIST	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).
	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 override default atomic weights.
	IRAYL	Same as for ELEM, line 2.
	IUNRST	Same as for ELEM, line 2.
	IAPRIM	Same as for ELEM, line 2.
	ISSB	Same as for ELEM, line 2.
	EPSTFL	Same as for ELEM, line 2.
	FUDGEMS	Same as for ELEM, line 2.
	IBOUND	Same as for ELEM, line 2.
	INCOH	Same as for ELEM, line 2.
	ICPROF	Same as for ELEM, line 2.
	IMPACT	Same as for ELEM, line 2.
	NELGO	Same as for ELEM, line 2.
	EFRACH	Same as for ELEM, line 2.
	EFRACL	Same as for ELEM, line 2.
MIXT3 (24A1,6X,24A1)	MEDIUM	Same as for ELEM, line 3.
	IDSTRN	Same as for ELEM, line 3.
MIXT4 (24(A2,1X))	(ASYM(I),I=1,NE)	Same as for COMP, line 4.

Table C.11: ENER option input data lines in PEGS. Note that if the user supplies negative values for the energy limits above, the absolute values given will be interpreted as being given in units of the electron rest mass energy. Thus, `AE=-1` is equivalent to `AE=0.511` MeV.

Line and Format	Variables	Comments
ENER1 (4A1)	OPT(1:4)	'ENER'. Means "select energy limits."
ENER2 NAMELIST	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).

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 values of `WA(I)` taken from the default PEGS tables can be overridden if specified by the user.

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 values of `RHOZ(I)` using `RHOZ(I)=PZ(I)*WA(I)`. Otherwise, `PZ(I)` values 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.

Users may supply values for the various parameters used in calculating the density effect correction by setting the flag `ISSB` to 1 and specifying the six parameters, `AFACT, SK, X0, X1, CBAR,` and `IEV`) in the `NAMELIST/INP` input in ELEM, MIXT, or COMP. Note that if one only wants to override `IEV`, one must still input all six parameters.

Alternatively, users may request values of the density effect correction taken from the NBS (now NIST) database EPSTAR, in which case collision stopping powers calculated by PEGS will correspond with values from the ICRU-37 standard. To invoke this option, the flag `EPSTFL` in `NAMELIST/INP/` should be set to 1, and the appropriate data file from either the **elements** or the **compounds** subdirectory of the **density_corrections** directory of the EGS5 distribution data files should be copied or linked to a file called "epstar.dat" in the working user code directory. Note that when this option is employed and multiple materials are being processed, all of the appropriate data files from **density_corrections** must be concatenated onto **epstar.dat** in the order in which the materials are referenced in the PEGS input file.

Rayleigh (coherent) scattering data will be appended to the normal PEGS output data if the

Table C.12: PWLF option input data lines in PEGS. Note: The PWLF parameters above (see section C.3.4 for longer descriptions) are optional and may be overridden by the user. The default values (assigned in `BLOCK DATA`) are indicated above, and all of the parameters are elements of both `NAMELIST/PWLFNM` and `NAMELIST/INP`.

Line and Format	Variables	Comments
PWLF1 (4A1)	OPT(1:4)	'PWLF'. Means "perform piecewise linear fit."
PWLF2 NAMELIST	EPE/0.01/	Electron EP parameter.
	EPG/0.01/	Gamma EP parameter.
	EPR/0.01/	Rayleigh scattering EP parameter.
	EPSF/0.01/	Incoherent scattering function EP parameter.
	EPCP/0.03/	Compton profile EP parameter.
	ZTHRE(1:8)/8*0./	Electron ZTHR parameter.
	ZTHRG(1:3)/0.,.1,0./	Gamma ZTHR parameter.
	ZTHRS(1:40)/40*0./	Incoherent scattering function ZTHR parameter.
	ZTHRR(1:40)/40*0./	Rayleigh scattering ZTHR parameter.
	ZTHRC(1:200)/200*0./	Compton profile ZTHR parameter.
	ZEPE(1:8)/8*0./	Electron ZEP parameter.
	ZEPG(1:3)/0.,.01,0./	Gamma ZEP parameter.
	ZEPS(1:40)/40*0./	Incoherent scattering function ZEP parameter.
	ZEPR(1:40)/40*0./	Rayleigh scattering ZEP parameter.
	ZEPC(1:200)/200*0./	Compton profile ZEP parameter.
	NIPE/20/	Electron NIP parameter.
	NIPG/20/	Gamma NIP parameter.
	NIPS/20/	Incoherent scattering function NIP parameter.
	NIPR/20/	Rayleigh scattering NIP parameter.
	NIPC/20/	Compton profile NIP parameter.
	NALE/150/	Electron NIMX parameter.
	NALG/1000/	Gamma NIMX parameter.
	NALR/100/	Rayleigh scattering NIMX parameter.
	NALS/100/	Incoherent scattering function NIMX parameter.
	NALC/2000/	Compton profile NIMX parameter.

Table C.13: DECK option input data lines in PEGS.

Line and Format	Variables	Comments
DECK1 (4A1)	OPT(1:4)	'DECK'. Means "print fitted data and other useful parameters."
DECK2	NAMELIST	No parameters, but &INP and &END tags must be present.

Table C.14: TEST option input data lines in PEGS.

Line and Format	Variables	Comments
TEST1 (4A1)	OPT(1:4)	'TEST'. Means "Prepare data for plotting the fitted functions."
TEST2 NAMELIST	NPTS	Optional. Number of points to plot per function (Default=50).

Table C.15: CALL option input data lines in PEGS.

Line and Format	Variables	Comments
CALL1 (4A1)	OPT(1:4)	'CALL'. Means "Call the designated function and print value."
CALL2 NAMELIST	XP(1:4)	Values for up to four arguments of the function.
CALL3 (6A1)	NAME(1:6)	Name of function to be evaluated.

Table C.16: PLTI option input data lines in PEGS.

Line and Format	Variables	Comments
PLTI1 (4A1)	OPT(1:4)	'PLTI'. Means "Plot function given its index and the index of the distribution function."
PLTI2 NAMELIST	IFUN	The index of the function to be plotted.
	XP(1:4)	Values for the static arguments (parameters).
	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.

Table C.17: PLTN option input data lines in PEGS.

Line and Format	Variables	Comments
PLTN1 (4A1)	OPT(1:4)	'PLTN'. Means "Plot the named function."
PLTN2 NAMELIST	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.

Table C.18: HPLT option input data lines in PEGS.

Line and Format	Variables	Comments
HPLT1 (4A1)	OPT(1:4)	'HPLT'. Means "Plot histogram to compare the sampled spectrum with the range-integrated and the differential theoretical values."
HPLT2 NAMELIST	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 7=MSCAT (not implemented).
HPLT3: FORMAT(' 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: FORMAT(' 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).

384

IRAYL flag is set to 1 or 2 in NAMELIST/INP/. An IRAYL value of 2 indicates that interference effects in coherent scattering are to be modeled, the user must supply pre-computed values of the coherent scattering cross section and coherent scattering form factors for the material in question. This is done by copying or linking the appropriate data files to files named "ics.dat" and 'iff.dat' in the working user code directory. Sample data files in the appropriate format for several materials are included in the **int_coherent_cs** and **int_form_factor** directories of the EGS5 distribution data set. As with the use of the EPSTFL flag, when multiple materials are being used all of the relevant material data files must be concatenated together in the proper order. Note that an option IRAYL=3 exists in which PEGS will print the Form Factors themselves, rather than cumulative distributions for sampling in EGS runs, when the user wishes to study this data. Note that EGS will abort if it encounters a PEGS data file prepared with IRAYL=3.

Altering the value of the NAMELIST/INP/ flag IUNRST causes different versions of the stopping power to be computed and printed in place of the default required by EGS (*i.e.*, the sum of the restricted collision and radiative stopping powers). This functionality is most useful in extracting data sets from PEGS-only user codes for comparison with publish results, and should *not* be invoked when preparing data files for EGS runs. The available options and their effects are given in Table C.7.

Users may specify the way in which PEGS normalizes bremsstrahlung cross sections by altering the value of the NAMELIST/INP/ flag IAPRIM, which has a default value of 1 (see Chapter 2 of SLAC-R-730/KEK-2005-8 for a complete discussion). Since radiative stopping powers calculated by default in PEGS comply with ICRU standards, changing IAPRIM is not recommended except during the development of and testing of new bremsstrahlung cross sections.

Users may input values of the empirical constant which is to be used to represent the contribution of soft electron collisions to multiple nuclear elastic scattering distributions by specifying the value of the parameter FUDGEMS in NAMELIST/INP/. The default of FUDGEMS is 1, which corresponds to approximating angular deflection due to electron scattering by replacing the Z^2 term in the nuclear elastic cross section with $Z(Z+1)$. See Chapter 2 of SLAC-R-730/KEK-2005-8 for a full discussion.

When the parameter IBOUND is set to 1 in NAMELIST/INP/, the effect of electron binding on the total Compton scattering cross section is taken into account, as described in Chapter 2 of SLAC-R-730/KEK-2005-8. In the default case (IBOUND = 0), atomic electrons are considered to be free and at rest for the purpose of modeling Compton scattering.

The effects of electron binding on the angular distribution of Compton scattered photons are modeled when, in conjunction with setting IBOUND to 1, the user also sets the parameter INCOH to be 1 in NAMELIST/INP/.

The effects of atomic electron motion on the energy distribution of photons Compton scattered through given angles (this is often referred to as "Doppler broadening") can be explicitly treated by specifying the value of the parameter ICPROF. Currently, ICPROF may be set to only 3 (in which case the user must supply values of shell-by-shell Compton profiles) or -3, in which case shell-wise Compton profile data is read from data files provided with the EGS5 distribution. When

`ICPROF` is set to 3, the user must copy or link the input profile data to a file named "scp.dat" in the working user code directory. If multiple materials are being processed, data for all materials must concatenated to **scp.dat** in the order in which the materials are specified in the PEGS input file. In PEGS-only user codes designed to generate physics data for examination and study, users may request that PEGS output raw Compton profile data by specifying `ICPROF` to be 4 (for user supplied profile data) or -4 (for PEGS generated profile data). In almost all instances, users wishing to model Compton Doppler broadening should use `ICPROF=-3`. Note that EGS requires that `IBOUND` and `INCOH` be set to 1 when modeling of Doppler broadening is requested, and that EGS will not accept PEGS data sets prepared with `ICPROF` values of 4 or -4.

If electron impact ionization is to be explicitly treated, the user must specify which cross section is to be used by defining a non-zero value for the parameter `IMPACT` in `NAMELIST/INP`. The list of available cross sections and the values of `IMPACT` needed to invoke them is given in Table C.8. The user is referred to Chapter 2 of SLAC-R-730/KEK-2005-8 for a discussion of the derivations and limitations of the various cross sections.

Electron and positron multiple scattering step sizes to be taken in EGS run may be specified by the user when the parameters `EFRACH` and `EFRACL` are defined in `NAMELIST/INP` in a PEGS input file. Including these EGS-related quantities in PEGS arises from the need to compute electron multiple scattering step sizes in terms of scattering strength for the new EGS5 transport mechanics. The parameters `EFRACH` and `EFRACL` refer to the fractional energy loss which occurs over the step at energies corresponding to `UE` and `AE` respectively. See Chapter 2 of SLAC-R-730/KEK-2005-8 for a detailed description of multiple scattering step-size selection. Note that it is strongly recommended that users *not* use `EFRACH` and `EFRACL` to control electron step sizes, but instead specify problem "characteristic dimensions," as described in the EGS User Manual and Chapter 2 of SLAC-R-730/KEK-2005-8.

In addition to physically specifying the material being used, a unique name must be supplied (`MEDIUM(1:24)`) for identification purposes. This name is included in the output file when the `DECK` option is selected. One useful feature of this implementation is that different names can be used for data sets created with the same materials but different parameters. 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.).

The quantity `IDSTRN(1:24)` is used to identify the Sternheimer-Seltzer-Berger density effect parameters that are tabulated in `BLOCK DATA` (see Chapter 2 of SLAC-R-730/KEK-2005-8 for a complete list of identifier names and a general discussion of the density effect). 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. Note that the options for computing stopping powers and the density effect invoked by either `ISSB` or `EPSTFL` take precedence over the specification of `IDSTRN`.

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

386

parameters for the pair production and bremsstrahlung sampling routines.

The following are examples of sets of input data lines that can be used with the ELEM, MIXT, and COMP options:

1. Material - Element is Iron with defaults taken.

```
                        Column
Line    1234567891123456789212345678931234567894123456 78..etc.
ELEM1   ELEM
ELEM2   &INP &END
ELEM3   IRON                            FE
ELEM4   FE
```

2. Material - Element is Helium-3 with the density and atomic weight overridden by user.

```
                        Column
Line    1234567891123456789212345678931234567894123456 78..etc.
ELEM1   ELEM
ELEM2   &INP RHO=1.E-2,WA(1)=3 &END
ELEM3   HELIUM-3                    HE
ELEM4   HE
```

3. Material - Compound is sodium iodide, and IDSTRN(1:24) defaults to MEDIUM(1:24).

```
                        Column
Line    1234567891123456789212345678931234567894123456 78..etc.
COMP1   COMP
COMP2   &INP NE=2,RHO=3.667,PZ(1)=1,PZ(2)=1 &END
COMP3   NAI
COMP4   NA I
```

4. 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
Line    1234567891123456789212345678931234567894123456 78..etc.
COMP1   COMP
COMP2   &INP NE=2,RHO=1.032,PZ(1)=1,PZ(2)=1.1 &END
COMP3   POLYSTYRENE SCINTILLATOR
COMP4   C  H
```

5. Material - Mixture is lead glass, consisting of five specified elements (and 1 per cent of the trace elements unspecified). Sternheimer-Peierls default.

```
                             Column
       Line    1234567891123456789212345678931234567894112345678..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
```

6. Material - Mixture is U-235, U-238, and carbon (not a real material). Sternheimer-Peierls default.

```
                             Column
       Line    1234567891123456789212345678931234567894112345678..etc.
       MIXT1   MIXT
       MIXT2   &INP NE=3,RHO=16,WA=235,238,RHOZ=50,30,10 &END
       MIXT3   JUNK
       MIXT4   U  U  C
```

7. Material - Element is Iron with shell-wise Compton profile.

```
                             Column
       Line    1234567891123456789212345678931234567894112345678..etc.
       ELEM1   ELEM
       ELEM2   &INP IBOUND=1,INCOH=1,ICPROF=-3 &END
       ELEM3   IRON                            FE
       ELEM4   FE
```

8. Material - Element is Iron with EII cross section by Casnati's formula.

```
                             Column
       Line    1234567891123456789212345678931234567894112345678..etc.
       ELEM1   ELEM
       ELEM2   &INP IMPACT=1 &END
       ELEM3   IRON                            FE
       ELEM4   FE
```

C.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 parameter is entered by the user with a negative value, it is multiplied by -0.511 MeV; that is, the absolute magnitude of the input energy is assumed to be 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 Møller thresholds (RM+AP and AE+TM, respectively), are then computed and printed out.

The following are examples of sets of input data lines that can be used with the ENER option:

1. 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
    Line    123456789112345678921234567893123456789412345678..etc.
    ENER1   ENER
    ENER2   &INP AE=1.5,UE=100000.,AP=0.01,UP=100000. &END
```

2. Same as above, except AE=3*RM.

```
                              Column
    Line    123456789112345678921234567893123456789412345678..etc.
    ENER1   ENER
    ENER2   &INP AE=-3,UE=100000.,AP=0.01,UP=100000. &END
```

C.3.4   The PWLF Option

The PieceWise Linear Fit option performs a simultaneous piecewise linear fit (vs. $\ln(E - RM)$) of ten or twelve electron functions over the energy interval (AE,UE) and a simultaneous piecewise linear fit (vs. $\ln E$) 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 sub intervals are used for all of the functions of a set.)

The PWLF1 subroutine is an executive routine that calls the function QFIT. Function QFIT 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. Suppose NFUN is the number of components to the vector function F(IFUN,E(J)) (where IFUN runs from 1 to NFUN) and E(J) is a sequence of points (with J from 1 to NI) covering the interval being fitted. The number of points NI must be about ten times the number of fit intervals NINT so 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 flow of the logical function `QFIT` may be shown as follows:

```
LOGICAL FUNCTION QFIT(NINT)
(commons and declarations)
QFIT=.TRUE.
REM=0.0
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)) THEN
     IF(AF.NE.0.0) REM=AMAX1(REM,AER/AF)
   ELSE
     IF(AER.GT.ZEP(IFUN)) QFIT=.FALSE.
 END DO
END DO
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 Møller thresholds), and computing fit parameters in bins flanking the main interval to guard against truncation errors in sub-interval index computations.

The net result of the fit is the determination of coefficients `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.

Note that the coding of EGS is 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 IFUNth function. This kind of fit has the advantage that it yields a better fit with a smaller amount of stored data than the default fit computed by `PWFL1`. Such a scheme has never been implemented in PEGS, however, despite that more data than necessary is currently used in describing the functions at higher energies where they vary quite smoothly.

The following is an example of the input data lines that can be used with the PWLF option:

```
                        Column
 Line    12345678911234567892123456789312345678941234 5678..etc.
 PWLF1   PWLF
 PWLF2   &INP &END
```

C.3.5   The DECK Option

The DECK option, with the aid of subroutine `LAY`, prints 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 anything that might be of use to EGS in simulating showers or to users in analysis routines. 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 when users alter `LAY` to write additional material data to PEGS output files, `HATCH` must be be updated appropriately.

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 Figure C.9). The following is an example of the input data lines that can be used with the DECK option:

```
                        Column
 Line    12345678911234567892123456789312345678941234 5678..etc.
 DECK1   DECK
 DECK2   &INP &END
```

C.3.6 The TEST Option

The TEST option is used as an easy way to obtain data for plots of some of the functions (not the fits) that the PWLF option fits. Output is printed to a file "pgs5job.pegs5plot" in the working user code directory, and includes sequential sets of data from PEGS functions ESIG, PSIG, EDEDX, PDEDX, EBR1, PBR1, PBR2, TMXS, TMXDE2, GMFP, GBR1, GBR2 and CRATIO. Thus the TEST option provides yet another useful PEGS tool for examining and studying raw physics data.

The following is an example of the input data lines that can be used with the TEST option:

```
                         Column
Line    12345678911234567892123456789312345678941234567 8..etc.
TEST1   TEST
TEST2   &INP NPTS=50 &END
```

C.3.7 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 input data lines 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 data lines for Lead).

```
                         Column
Line    12345678911234567892123456789312345678941234567 8..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 a user code accessing PEGS would be:

```
OPT = CALL
Function call:     1.95522     = GMFP   OF     49.9900
OPT = CALL
Function call:     1.97485     = GMFP   OF     50.0100
```

Table C.19: Distribution functions available with PLTI and PLTN.

IDFNAM	Purpose
ALIN	Linear plot.
DLOG	Natural log plot.
ALKE	Natural log of electron kinetic energy plot.
ADFMOL	Approximation to Møller and Bhabha distributions

C.3.8 The PLTI and PLTN Options

The PLTI and PLTN options may be used to obtain printer plots of any of the functions in the PEGS function table. These options are somewhat analogous to the TEST option in that they call the same PEGS output routine (subroutine PLOT) and write output data in the same format to the same file ('pgs5job.pegs5plot' in the working user code directory). However, the information which can be extracted using the TEST option is restricted to the energy-dependent default functions over the ranges from AE to UE and AP to UP, while PLTI and PLTN can access arbitrary PEGS functions and produce physics data over arbitrary ranges. 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 C.17) the function name we would use is BRMSRZ, the distribution function name to be specified is IDFNAM=ALOG, the static arguments needed are XP(1)=2., XP(3)=0.0, XP(4)=1.5, the independent variable to be accessed 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 data set we specific NPTS=100. The input data lines necessary to accomplish this are:

```
                     Column
 Line     123456789112345678921234567893123456789412345678..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 listed and described in Table C.19.

C.3.9   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 with the EGS5 distribution. This option provides a way to test the EGS methods used to sample differential distributions functions by looking for discrepancies between actual differential physics data and histograms of sampled data during simulations.

Suppose that a probability density function, $PDF(X)$ (see Chapter 2 of SLAC-R-730/KEK-2005-8), 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 $PDF(X)$. 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 user code and `HPLT1` subroutine in PEGS. 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))
END DO
RETURN
END
```

All of the input data lines for a specified HPLT option are printed directly by UCTESTSR to standard output. This data can then be used in a subsequent run of PEGS to produce line-printer style plots of the actual and sample functions . Users 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).

C.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:

1. Minimal options for a material data set creation (for later use by EGS):

   ```
   1.  ELEM (or MIXT, or COMP)
   2.  ENER
   3.  PWLF
   4.  DECK
   ```

2. Same as above, with default data dumps of functions for which the PWLF option provides fits:

   ```
   1.  ELEM (or MIXT, or COMP)
   2.  ENER
   3.  TEST
   4.  PWLF
   5.  DECK
   ```

3. Comparison of theoretical and sampled distributions by means of the HPLT option:

   ```
   1.  ELEM (or MIXT, or COMP)
       Note: Data lines should agree with those used
             with the UCTESTSR run that generated the HPLT input.
   2.  HPLT
       Note: Output data from UCTESTSR run.
   ```

4. 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.
   ```

Appendix D

EGS5 INSTALLATION GUIDE

Hideo Hirayama and Yoshihito Namito
Radiation Science Center
Advanced Research Laboratory
High Energy Accelerator Research Organization (KEK)
1-1 Oho Tsukuba-shi Ibaraki-ken 305-0801 JAPAN

Alex F. Bielajew and Scott J. Wilderman
Department of Nuclear Engineering and Radiological Sciences
The University of Michigan
2355 Bonisteel Boulevard
Ann Arbor, MI 48109, USA

Walter R. Nelson
Department Associate in the Radiation Physics Group (retired)
Radiation Protection Department
Stanford Linear Accelerator Center
2575 Sand Hill Road Menlo Park, CA 94025, USA

This EGS5 Installation Guide is Appendix D of a document called
SLAC-R-730/KEK-2005-8, which can be obtained from the SLAC and KEK web sites.

D.1 Installation of EGS5

The current version of the EGS5 code system run only on Unix-based operating system. On Windows platforms, this release of EGS5 can be run on top of **Cygwin** or any other Linux environment emulator for PCs. Copies of **Cygwin**, along with setup programs, installation instructions, users guides, and other documentation, can be obtained for free at the **Cygwin** website, http://www.cygwin.com.

The EGS5 system is distributed as a compressed (with **gzip**) Unix "tar" archive, **egs5.tar.gz**. It is trivially "installed" by typing:

```
% gunzip -c egs5.tar.gz | tar vxf -
```

from the command line in any desired directory into which a user places **egs5.tar.gz** (to somewhat mimic the organizational structure of EGS4, this command should be run from the user's top level directory). When this command is executed, a directory called **egs5** will be created, under which the subdirectories containing all of the EGS5 FORTRAN, data, tutorial, and sample problem files will be placed.

Although it employs some non-standard extensions to FORTRAN-77 the basic source code in EGS5 avoids the use of all system-dependent utilities, and all that is required to run EGS5 on any Unix-like platform is a FORTRAN compiler[1]. Further, all of the tutorial, sample, and auxiliary user codes provided in the EGS5 distribution should run without modification if the FORTRAN library routines `FDATE` and `ETIME` are available. Likewise, the sample shell scripts for controlling the compilation and execution of EGS5 jobs will run after minimal modification on the part of the user.

D.2 Sample Scripts for Running EGS5

The EGS5 distribution comes with two shell scripts which can be used (with minimal modification) to compile, run, and re-run generic EGS5 user codes. The five steps which are required to use the sample script "egs5run" are listed below. Example implementations follow.

1. A copy of egs5run (or a symbolic link to a global copy of egs5run) must be placed in the subdirectory containing the user code.

2. In the egs5run file, the user must specify the directory in which the main EGS5 distribution was installed. This is done by setting the shell script variable `BASKET` to the name of the

---

[1]The one system-dependent FORTRAN implementation issue which an EGS5 user may encounter is with the `NAMELIST` extension. Different compilers may require different leading characters in the end-of-list tag, usually either `&END` or `/END`.

egs5 directory (preferably using the full Unix pathname). `BASKET` is analogous to `HEN_HOUSE` of EGS4, in that all of the basic EGS5 system resides in this single place.

3. Also in egs5run, the user must specify the operating system type under which the current computation is to be performed by defining the variable `MY_MACHINE`. This is required primarily so that egs5run can select the appropriate compiler options.

4. If the user's operating system is not among the defaults included with the egs5run script, the user will have to modify the section of egs5run in which the name of the FORTRAN compiler and the relevant compiler options are set. An example is given below.

5. Lastly, and also in egs5run, the user must select the level of optimization to be used, if any, during compilation, by specifying the value of the variable `OPT_LEVEL`.

The preamble to egs5run contains some examples of how the variables described above might be set in egs5run:

```
#    BASKET=/home/wrn/egs5/
#    MY_MACHINE=Linux
#    OPT_LEVEL=O
#
#    BASKET=/home/Ralph/egs5/
#    MY_MACHINE=Cygwin-Linux
#    OPT_LEVEL=O2
#
#    BASKET=/afs/slac.stanford.edu/g/rp/egs5
#    MY_MACHINE=sparc
#    OPT_LEVEL=
```

The first example corresponds to a user with login id "wrn" on machine running Linux, with EGS5 installed from his top level (home) directory. The second example is representative of what a PC user named "Ralph" might see if he were running **Cygwin**, again with EGS5 installed from his top level directory. The final example is typical of what a user accessing a globally available version of EGS5 across an afs file system might see.

Note that the variable `MY_MACHINE` can be defined as anything the user wishes, provided that compiler options are included for that OS name later in egs5run. The version of egs5run in the current EGS5 distribution has defined compiler options for four operating systems: Linux, Cygwin-Linux, sparc and Digital-Unix. If EGS5 is to be used on some other Unix-like platform, the user must edit egs5run to specify the name of the FORTRAN compiler and any compiler options which are necessary. This can be done quite simply by changing the 5 instances of `user_defined_*` with the appropriate parameters in the section of egs5run shown below:

```
elif test "$MY_MACHINE" = "user_defined_machine"
```

398

```
then
  COMPILER="user_defined_compiler"
  DEBUG="user_defined_debug_flags"
  CFLAGS="user_defined_compilation_flags"
  if test "$OPT_LEVEL" = ""
    then
      OPTIMIZED=""
  else
      OPTIMIZED="-$OPT_LEVEL user_defined_optimizations"
  fi
```

Running egs5run: There are five options for executing egs5run, invoked by specifying a single command line argument, as shown below:

Command line	Action of egs5run
egs5run	Compile user code and execute.
egs5run comp	Compile user code but do not execute.
egs5run pegs	Compile pegs-only user code and execute.
egs5run db	Compile user code for debug (does not execute).
egs5run cl	Clear out files (and links) and exit egs5run.

When egs5run is executed in normal mode (*i.e.*, with no argument), the following sequence of actions is initiated:

1. Various files and links created in previous runs (if any) are deleted.

2. The specified system name MY_MACHINE and compiler are echoed.

3. The user is asked to key-in the name of the FORTRAN file containing the user code for this run. The file name provided by the user *must* have a .f extension. Is the file is not found, the script will exit.

4. A file named egs5job.f is created using the specified user code file, and all files with .f extensions in the following directories:

user_auxcode	user defined auxiliary subroutines, if present.
$BASKET/egs	EGS related subroutines.
$BASKET/pegs	subroutines related to PEGS.
$BASKET/auxcode	auxiliary subroutines provided with EGS5.

   Note that file extensions should not be keyed-in when specifying file names.

5. Symbolic links are created pointing to the following directories containing files of COMMON blocks included by the various source code files:

$BASKET/include	COMMON block files for EGS.
$BASKET/auxcommons	COMMON block files for EGS5 auxiliary subroutines.
$BASKET/pegscommons	COMMON block files for PEGS.

399

6. The user is then asked to key-in the name of the EGS input file for this run. Note that for many applications, all of the necessary problem-specific data can be specified in the user code itself, and so an input problem data file need not be used. When this is the case, the user may simply enter a carriage return. If an EGS problem data file is to be used, it must have a `.data` extension. If the data file has the same name as the user code file (apart from the extensions `.f` and `.data`, of course), this can be specified by keying in a carriage return. If the user specified file exists, it is copied to file called "egs5job.inp." If the file does not exist, egs5run will create an empty (dummy) file called egs5job.inp, which is opened as unit 4 for input in many EGS5 user codes.

7. The user is next asked to key-in the name of the input data file used by PEGS. This file must have a `.inp` extension. A carriage return issued in response to this question will cause egs5run to look for `.inp` files named similarly to first the user code and then the data file. If any of the files exist, a symbolic link named "pgs5job.pegs5inp" will be created, as this file is opened by default on unit 25 in PEGS. If no appropriate `.inp` file is found, the script will exit.

8. The egs5job.f will then be compiled, with executable named "egs5job.exe."

9. If egs5job.exe does not exist because of compilation errors, the script will exit.

10. Next, a symbolic link to the data directory, $BASKET/data, will be created.

11. The user will then be asked if this user code requires input to be keyed-in from the terminal (on unit 5, standard input).

12. egs5job.exe will be executed either in the foreground (if interactive input was specified) or in the background. If the job is executed in the background, standard output (unit 6) is redirected to a file named "egs5job.log." Note that the user may elect to explicitly open unit 6 in the user code, if desired.

When egs5run is executed with one of the three arguments, "cl," "db," or "comp," the script follows the basic flow described above except that it exits either immediately after cleaning out old links (with the "cl" option) or just prior to executing the compiled code (with the "db" or "comp" arguments). When egs5run is executed with "pegs" as an argument, only the **pegs** files and the `BLOCK DATA` FORTRAN source files from **egs** are included with the user code when **egs5job.f** is created. With that exception, egs5run with "pegs" as an argument runs exactly as it does with no argument. Note that because none of the main physics routines of EGS are included in **egs5job.f**, user codes based on PEGS only need not contain either `HOWFAR` or `AUSGAB` subroutines.

Example transcript from egs5run   Below is a transcript of an egs5run session using the tutorial problem in **tutor1**. (Note that input keyed-in by the user is denoted by `<-------- USER_INPUT`.)

```
% egs5run
============================
```

```
egs5run script has started
============================


working directory is /afs/slac.stanford.edu/g/rp/egs5/tutorcodes/tutor1


-------------------------------------------------------------
Erasing files (and links) from previous runs (if they exist)
-------------------------------------------------------------
                  OS_TYPE  = sparc
Your Compiler is f77


----------------------------------------
       Enter name of User Code
     (file extension must be '.f')
----------------------------------------
tutor1                                    <-------- USER_INPUT


-------------------------------------------
     Enter name of READ(4) data file
     (file extension must be '.data')
   (<CR> for same file name as User Code)
-------------------------------------------
                                          <-------- USER_INPUT (<CR>)
  --> Empty file created as egs5job.inp


-------------------------------------------
  Enter name of UNIT(25) (pegs input file)
     (file extension must be '.inp')
   (<CR> for same file name as data file
     or same file name as User Code)
-------------------------------------------
                                          <-------- USER_INPUT (<CR>)
  --> tutor1.inp linked to pgs5job.pegs5inp


------------------------------------
Compiling (with optimization of O2)
------------------------------------


-------------------------------------------------
  Does this user code read from the terminal?
  (Enter 1 for yes, anything else for no)
-------------------------------------------------
0                                         <-------- USER_INPUT

       ***************************************************
```

```
        * User code tutor1.f has been compiled and is starting *
        ********************************************************


 Running egs5job.exe in background


========================
egs5run script has ended
========================
```

Running run5again: If a user wishes to run previously compiled user code with different input data on either unit 4 (EGS problem data) or unit 25 (PEGS material data) or both, the script run5again can be executed instead of egs5run. The run5again script takes no arguments, and does all of what egs5run does except that it does not construct or compile **egs5job.f**. run5again will execute either an existing egs5job.exe program or any previously compiled egs5job.f file has been renamed and has a `.exe` file extension. When run5again (which requires no set up on the part of the user) is executed, the following actions are performed:

1. Symbolic link pgs5job.pegs5inp and file egs5job.inp from previous runs are deleted.

2. The user is asked to key-in the EGS data file name to be copied to egs5job.inp and opened on unit 4, as in egs5run, above.

3. The user is asked to key-in the name of the PEGS data file to be linked to pgs5job.pegs5inp and opened on unit 25, as with egs5run.

4. The user is then asked to specify the name of an existing executable (with file extension `.exe` for this run. If a carriage return is entered, run5again will look for an existing egs5job.exe executable to use. If neither the specified executable nor egs5job.exe exist, the script will exit.

5. The user will finally be asked if this user code requires input that is to be keyed-in (to unit 5, standard input).

6. Either the specified executable or egs5job.exe will be executed, either in the foreground (if interactive input was specified) or in the background. If the job is executed in the background, standard output (unit 6) is redirected to a file named "egs5job.log," as with egs5run.

Example transcript from run5again Below is a sample transcript from a run5again job:


```
% run5again
============================
run5again script has started
============================
```

```
working directory is /home/user/egs5/tutorcodes/tutor1


-------------------------------------------
      Enter name of READ(4) data file
      (file extension must be '.data')
-------------------------------------------
tutor1                                      <-------- USER_INPUT

  --> Empty file created as egs5job.inp

-------------------------------------------
  Enter name of UNIT(25) (pegs input file)
      (file extension must be '.inp')
   (<CR> for same file name as data file)
-------------------------------------------
                                            <-------- USER_INPUT (<CR>)

  --> d4file used, tutor1.inp linked to pgs5job.pegs5inp

------------------------------------
    Enter name of the executable
  (file extension must be '.exe')
     (<CR> to use egs5job.exe)
------------------------------------
                                            <-------- USER_INPUT (<CR>)


-------------------------------------------------
  Does this user code read from the terminal?
  (Enter 1 for yes, anything else for no)
-------------------------------------------------
0                                           <-------- USER_INPUT
      *******************************************
      * Previously compiled user code is starting *
      *******************************************

 Running egs5job.exe in background

==========================
run5again script has ended
==========================
```

Appendix E

CONTENTS OF THE EGS5 DISTRIBUTION

Hideo Hirayama and Yoshihito Namito
Radiation Science Center
Advanced Research Laboratory
High Energy Accelerator Research Organization (KEK)
1-1 Oho Tsukuba-shi Ibaraki-ken 305-0801 JAPAN

Alex F. Bielajew and Scott J. Wilderman
Department of Nuclear Engineering and Radiological Sciences
The University of Michigan
2355 Bonisteel Boulevard
Ann Arbor, MI 48109, USA

Walter R. Nelson
Department Associate in the Radiation Physics Group (retired)
Radiation Protection Department
Stanford Linear Accelerator Center
2575 Sand Hill Road Menlo Park, CA 94025, USA

This EGS5 Distribution Listing is Appendix E of a document called
SLAC-R-730/KEK-2005-8, which can be obtained from the SLAC and KEK web sites.

The EGS5 distribution archive contains six main groups of files: documentation; EGS FOR-TRAN source codes; PEGS FORTRAN source codes; material data files; example and tutorial run-time scripts and user codes (including FORTRAN source codes, input files and sample output); and FORTRAN source codes for auxiliary functions which users may find useful.

E.1 Documentation

Documentation has been included within the EGS5 distribution for two reasons. First, the most heavily used parts of SLAC-R-730/KEK-2005-8 are expected to be the user manuals, and so .pdf versions have been placed in the archive so that they can be accessed (or printed) at the discretion and convenience of the user. Second, it is anticipated that when changes and additions are made to the EGS5 code base, the user manuals will be updated appropriately, while SLAC-R-730/KEK-2005-8 may not be. Documentation files includes with the EGS5 distribution are:

egs5_user_manual.pdf	Appendix B of SLAC-R-730/KEK-2005-8
pegs_user_manual.pdf	Appendix C of SLAC-R-730/KEK-2005-8
installation_guide.pdf	Appendix D of SLAC-R-730/KEK-2005-8
distribution_contents.pdf	Appendix E of SLAC-R-730/KEK-2005-8
Writing_HOWFAR.pdf	Presentation describing construction of user code subroutine `HOWFAR`

These files can be found in the subdirectory **docs** of the under the main **egs5** directory in the distribution. In addition, a complete copy of SLAC-R-730/KEK-2005-8 is included in the main **egs5** directory, as the .pdf file **slac730.pdf**.

E.2 EGS-Related FORTRAN Source Files

There are two types of files which are part of the EGS Monte Carlo shower simulation package. As described in the EGS5 User Manual (Appendix B of SLAC-R-730/KEK-2005-8), some of the utility of MORTRAN macro substitutions used in EGS4 has been retained in EGS5 by having all FORTRAN `COMMON` blocks be defined in the subprograms of the source code through `include` statements which reference files containing just the FORTRAN listing of each `COMMON` block. This makes changing variables and parameters in any `COMMON` block a global process, similar to what could be done with a MORTRAN macro in EGS4. Each EGS5 `COMMON` block is thus contained in a unique file which is named after the `COMMON` block as in **egs5_media.f** for the file containing the declarations for `COMMON` block `MEDIA`. All of the FORTRAN `PARAMETERS` used to the specify array dimensions in the various `COMMON` blocks have been collected in a single "header" file called **egs5_h.f**, and all of the EGS5 `COMMON` block files have been placed in a directory called **include** under the main **egs5** directory. These files contain nothing other than the declarations of the variables in each `COMMON`, along with some documentation.

All of the actual FORTRAN source code files used in simulating showers have been collected in a subdirectory of **egs5** called **egs**. Each file contains the source code for one EGS5 subroutine (some source code files actually contain more than one subroutine, when groups of subprograms are very closely related), and the naming convention used with the `COMMON` block files in **include** has been followed. Descriptions of the functionality of the various subprograms of **egs** can be found both in Chapter 2 of SLAC-R-730/KEK-2005-8 and in the EGS5 User Manual. List below are the names of all of the files in **egs** and **include**, which are subdirectories of **egs5** in the EGS5 distribution.

EGS FORTRAN source files:

counters_out.f	egs5_eii.f	egs5_mscat.f
egs5_annih.f	egs5_electr.f	egs5_pair.f
egs5_aphi.f	egs5_hardx.f	egs5_photo.f
egs5_bhabha.f	egs5_hatch.f	egs5_photon.f
egs5_block_data.f	egs5_kauger.f	egs5_raylei.f
egs5_block_data_atom.f	egs5_kshell.f	egs5_rk1.f
egs5_block_set.f	egs5_kxray.f	egs5_rmsfit.f
egs5_brems.f	egs5_lauger.f	egs5_shower.f
egs5_collis.f	egs5_lshell.f	egs5_uphi.f
egs5_compt.f	egs5_lxray.f	randomset.f
egs5_edgbin.f	egs5_moller.f	rluxinit.f

EGS "included" `COMMON` block files:

counters.f	egs5_edge.f	egs5_mscon.f	egs5_useful.f
egs5_bcomp.f	egs5_eiicom.f	egs5_mults.f	egs5_userpr.f
egs5_bounds.f	egs5_elecin.f	egs5_photin.f	egs5_usersc.f
egs5_brempr.f	egs5_epcont.f	egs5_scpw.f	egs5_uservr.f
egs5_cdcsep.f	egs5_h.f	egs5_stack.f	egs5_userxt.f
egs5_cdcspl.f	egs5_media.f	egs5_thresh.f	randomm.f
egs5_coefgs.f	egs5_misc.f	egs5_uphiin.f	
egs5_csplcf.f	egs5_ms.f	egs5_uphiot.f	

Also included in the **egs** subdirectory is the EGS5 copyright file, **COPYRIGHT**, in a version suitable for inclusion with FORTRAN user codes.

E.3   PEGS-Related FORTRAN Source Files

As with EGS, there are two types of files comprising PEGS, those containing actual FORTRAN source code (in a subdirectory of **egs5** called **pegs**) and those containing just `COMMON` block declara-

tions (in a subdirectory called **pegscommons**). While all of the `COMMON` blocks in PEGS are found in unique files (named after the `COMMON` block with no prefix) all of the subroutines and functions previously a part of PEGS4 are contained in one file **pegs5.f**. All of the other source code files in the **pegs** subdirectory are new to EGS5 and are either required for calculating the scattering strength and scattering power data needed for the new electron transport mechanics of EGS5, or are part of the implementation of the new multiple scattering distribution, both of which are described in Chapter 2 of SLAC-R-730/KEK-2005-8. The full lists of the PEGS-related files found in **pegs** and **pegscommons**, which are both subdirectories of **egs5** in the EGS5 distribution, are given below.

PEGS FORTRAN source files:

```
csdar.f       elinit.f      g1e.f        k1e.f         sumga.f
dcsel.f       esteplim.f    gauleg.f     legenp.f      wmsfit.f
dcsn.f        estepmax.f    gscoef.f     makek1.f
dcsstor.f     findi.f       gsdist.f     pegs5.f
dcstab.f      fitms.f       inigrd.f     prelastino.f
elastino.f    g1dedx.f      integ.f      spline.f
```

PEGS "included" `COMMON` block files:

```
bcom.f     elemtb.f    lbhabm.f    mimsd.f     radlen.f
bremp2.f   elmtbc.f    lbremr.f    mixdat.f    rngspl.f
cohcom.f   epstar.f    lbremz.f    molvar.f    rslts.f
cpcom.f    funcs.f     lcompm.f    mscom.f     scpspl.f
dbrpr.f    funcsc.f    legacy.f    mxdatc.f    sfcom.f
dcsstr.f   k1spl.f     lpairr.f    phpair.f    spcomc.f
dercon.f   lamolm.f    lpairz.f    pmcons.f    spcomm.f
eimpact.f  lanihm.f    lspion.f    pwlfin.f    thres2.f
```

E.4   Material Data Files

The EGS5 distribution contains six primary data files plus five subdirectories containing material dependent data files, as described below. All files and directories are found in the subdirectory **data** of **egs5**. More complete descriptions of the data contained in the files can be found in Chapter 2 of SLAC-R-730/KEK-2005-8.

Data files: The main material data files in **data** are briefly described below:

aprime.data	Data for empirical bremsstrahlung correction.
bcomp.dat	Bound total Compton cross section data (σ_{bC}) for elements $Z = 1 \sim 100$.
incoh.dat	Incoherent scattering function data ($S(x, Z)$) for all elements $Z = 1 \sim 100$.
K1.dat	Tables of optimal initial scattering strengths as a function of geometry region size at various energies for various reference materials.
pgs5form.dat	Rayleigh scattering form factor data ($F(x, Z)$), identical to **pgs4form.dat** from PEGS4.
pgs5phtx.dat	Photo-electric cross section data, pair-production cross section data (for photon energies less than 50 MeV), and Rayleigh scattering cross section data from PHOTX for all elements $Z = 1 \sim 100$.

Data subdirectories: Data subdirectories of **data** are briefly described as follows:

dcslib	Differential nuclear elastic scattering cross sections for electrons and positrons for elements $Z = 1 \sim 95$.
density_corrections	Explicit values of the density effect correction to stopping power which can be used to reproduce ICRU-37 values. Placed in two subdirectories, **elements** and **compounds**.
int_coherent_cs	Coherent scattering cross sections (with interference effects modeled) for selected materials.
int_form_factor	Form factors (including interference effects) for selected materials.
shellwise_Compton_profile	Shellwise Compton profiles for elements $Z = 1 \sim 100$.

Files in dcslib: Files containing elemental partial-wave differential elastic scattering cross section data for both electrons and positrons are included in the EGS5 distribution in subdirectory **dcslib** of **data**. A straight-forward convention has been used to name these data files, which are listed below

```
eeldx001.tab   eeldx039.tab   eeldx077.tab   peldx020.tab   peldx058.tab
eeldx002.tab   eeldx040.tab   eeldx078.tab   peldx021.tab   peldx059.tab
eeldx003.tab   eeldx041.tab   eeldx079.tab   peldx022.tab   peldx060.tab
eeldx004.tab   eeldx042.tab   eeldx080.tab   peldx023.tab   peldx061.tab
eeldx005.tab   eeldx043.tab   eeldx081.tab   peldx024.tab   peldx062.tab
eeldx006.tab   eeldx044.tab   eeldx082.tab   peldx025.tab   peldx063.tab
eeldx007.tab   eeldx045.tab   eeldx083.tab   peldx026.tab   peldx064.tab
eeldx008.tab   eeldx046.tab   eeldx084.tab   peldx027.tab   peldx065.tab
eeldx009.tab   eeldx047.tab   eeldx085.tab   peldx028.tab   peldx066.tab
eeldx010.tab   eeldx048.tab   eeldx086.tab   peldx029.tab   peldx067.tab
eeldx011.tab   eeldx049.tab   eeldx087.tab   peldx030.tab   peldx068.tab
eeldx012.tab   eeldx050.tab   eeldx088.tab   peldx031.tab   peldx069.tab
eeldx013.tab   eeldx051.tab   eeldx089.tab   peldx032.tab   peldx070.tab
```

```
eeldx014.tab   eeldx052.tab   eeldx090.tab   peldx033.tab   peldx071.tab
eeldx015.tab   eeldx053.tab   eeldx091.tab   peldx034.tab   peldx072.tab
eeldx016.tab   eeldx054.tab   eeldx092.tab   peldx035.tab   peldx073.tab
eeldx017.tab   eeldx055.tab   eeldx093.tab   peldx036.tab   peldx074.tab
eeldx018.tab   eeldx056.tab   eeldx094.tab   peldx037.tab   peldx075.tab
eeldx019.tab   eeldx057.tab   eeldx095.tab   peldx038.tab   peldx076.tab
eeldx020.tab   eeldx058.tab   peldx001.tab   peldx039.tab   peldx077.tab
eeldx021.tab   eeldx059.tab   peldx002.tab   peldx040.tab   peldx078.tab
eeldx022.tab   eeldx060.tab   peldx003.tab   peldx041.tab   peldx079.tab
eeldx023.tab   eeldx061.tab   peldx004.tab   peldx042.tab   peldx080.tab
eeldx024.tab   eeldx062.tab   peldx005.tab   peldx043.tab   peldx081.tab
eeldx025.tab   eeldx063.tab   peldx006.tab   peldx044.tab   peldx082.tab
eeldx026.tab   eeldx064.tab   peldx007.tab   peldx045.tab   peldx083.tab
eeldx027.tab   eeldx065.tab   peldx008.tab   peldx046.tab   peldx084.tab
eeldx028.tab   eeldx066.tab   peldx009.tab   peldx047.tab   peldx085.tab
eeldx029.tab   eeldx067.tab   peldx010.tab   peldx048.tab   peldx086.tab
eeldx030.tab   eeldx068.tab   peldx011.tab   peldx049.tab   peldx087.tab
eeldx031.tab   eeldx069.tab   peldx012.tab   peldx050.tab   peldx088.tab
eeldx032.tab   eeldx070.tab   peldx013.tab   peldx051.tab   peldx089.tab
eeldx033.tab   eeldx071.tab   peldx014.tab   peldx052.tab   peldx090.tab
eeldx034.tab   eeldx072.tab   peldx015.tab   peldx053.tab   peldx091.tab
eeldx035.tab   eeldx073.tab   peldx016.tab   peldx054.tab   peldx092.tab
eeldx036.tab   eeldx074.tab   peldx017.tab   peldx055.tab   peldx093.tab
eeldx037.tab   eeldx075.tab   peldx018.tab   peldx056.tab   peldx094.tab
eeldx038.tab   eeldx076.tab   peldx019.tab   peldx057.tab   peldx095.tab
```

Files in density_corrections: To access values of collision stopping powers derived from ICRU Report 37, the user must provide density effect values explicitly for each material. Files containing the appropriate data for 450 elements and compounds are provided in the EGS5 distribution in subdirectories **elements** and **compounds** of **data** subdirectory **density_corrections**. Files are titled using a unique and explicit name for each material along with a **.density** extension. The full list of elemental material files (without the **.density** extension) is given below. A list of the 345 files containing data for compounds can be found in a **README** file provided in **density_corrections**.

```
actinium                germanium               protactinium
aluminium               gold                    radium
aluminum                hafnium                 radon
americium               helium                  rhenium
antimony                holmium                 rhodium
argon                   hydrogen                rubidium
arsenic                 hydrogen_liquid         ruthenium
astatine                indium                  samarium
barium                  iodine                  scandium
berkelium               iridium                 selenium
```

beryllium	iron	silicon
bismuth	krypton	silver
boron	lanthanum	sodium
bromine	lead	strontium
cadmium	lithium	sulfur
calcium	lutetium	sulfur_1.92g_cm3
californium	magnesium	sulfur_2.07g_cm3
carbon_graphite_1.700g_cm3	manganese	tantalum
carbon_graphite_2.000g_cm3	mercury	technetium
carbon_graphite_2.265g_cm3	molybdenum	tellurium
cerium	neodymium	terbium
cesium	neon	thallium
chlorine	neptunium	thorium
chromium	nickel	thulium
cobalt	niobium	tin
copper	nitrogen	titanium
curium	osmium	tungsten
dysprosium	oxygen	uranium
einsteinium	palladium	vanadium
erbium	phosphorus	xenon
europium	platinum	ytterbium
fermium	plutonium	yttrium
fluorine	polonium	zinc
francium	potassium	zirconium
gadolinium	praseodymium	
gallium	promethium	

Files in int_coherent_cs: For computations in which coherent scattering is modeled with interference effects included, the user must provide the interference coherent scattering cross section data. Included in subdirectory **int_coherent_cs** of **data** in the EGS5 distribution are files of coherent scattering cross section data with interference effects for seven materials: water, PMMA, fat, muscle, kidney, liver and blood. The data files are provided are named **ics_** followed by the first letter of the material, as below.

```
ics_b.dat     ics_k.dat     ics_m.dat     ics_w.dat
ics_f.dat     ics_l.dat     ics_p.dat
```

Files in int_form_factor: When coherent scattering interference effects are modeled, the user must also provide the appropriate form factors. Sample data files containing form factors with interference effects included are provided in subdirectory **int_form_factor** of **data** of the EGS distribution for the same seven materials in **int_coherent_cs**. The naming convention analogous to that described above is employed, and the exact EGS5 file names are listed below.

```
iff_b.dat        iff_k.dat        iff_m.dat        iff_w.dat
iff_f.dat        iff_l.dat        iff_p.dat
```

Files in shellwise_Compton_profile: The elemental shellwise Compton profile data files included in the EGS5 distribution in subdirectory **shellwise_Compton_profile** of **data** are named as follows. This data is used in modeling Compton scattering from bound atomic electrons on a shell-by-shell basis. The naming convention employed is intuitive.

```
z001.dat   z019.dat   z036n.dat  z054.dat   z072.dat   z090.dat
z002.dat   z020.dat   z037.dat   z055.dat   z073.dat   z091.dat
z003.dat   z021.dat   z038.dat   z056.dat   z074.dat   z092.dat
z004.dat   z022.dat   z039.dat   z057.dat   z075.dat   z093.dat
z005.dat   z023.dat   z040.dat   z058.dat   z076.dat   z094.dat
z006.dat   z024.dat   z041.dat   z059.dat   z077.dat   z095.dat
z007.dat   z025.dat   z042.dat   z060.dat   z078.dat   z096.dat
z008.dat   z026.dat   z043.dat   z061.dat   z079.dat   z097.dat
z009.dat   z027.dat   z044.dat   z062.dat   z080.dat   z098.dat
z010.dat   z028.dat   z045.dat   z063.dat   z081.dat   z099.dat
z011.dat   z029.dat   z046.dat   z064.dat   z082.dat   z100.dat
z012.dat   z030.dat   z047.dat   z065.dat   z083.dat   z101.dat
z013.dat   z031.dat   z048.dat   z066.dat   z084.dat   z102.dat
z014.dat   z032.dat   z049.dat   z067.dat   z085.dat
z015.dat   z033.dat   z050.dat   z068.dat   z086.dat
z016.dat   z034.dat   z051.dat   z069.dat   z087.dat
z017.dat   z035.dat   z052.dat   z070.dat   z088.dat
z018.dat   z036.dat   z053.dat   z071.dat   z089.dat
```

E.5   Sample User Codes and Run Scripts

The EGS5 distributions includes three sets of example user codes and two sample shell scripts for setting up and executing EGS5 simulations, as described below:

egs5run   Sample script for compiling and executing user code.

run5again   Sample script for rerunning previously compiled user code.

tutorcodes   Directory containing a series of sample problems, each in a unique subdirectory with FORTRAN source codes, input files, and output files for the step-by-step tutorials on using EGS5 found in Chapter 3 of SLAC-R-730/KEK-2005-8.

samplecodes   Directory containing subdirectories with advanced user codes and input files for example problems described in Chapter 4 of SLAC-R-730/KEK-2005-8.

extra_ucodes Directory containing subdirectories with additional sample user codes.

The tutorial problems and advanced user codes are discussed in detail in Chapters 3 and 4 of SLAC-R-730/KEK-2005-8, respectively, while the programs (based on PEGS) included as "extra codes" are described briefly in Appendix C. Listed below are the input, source, and sample output files included in the EGS5 distribution in each of these subdirectories.

Files in subdirectories under egs5/samplecodes:

```
uc_lp/uc_lp.f          uccyl/uccyl.out        ucsampl5/ucsampl5.inp
uc_lp/uc_lp.inp        ucbend/ucbend.f        ucsampl5/ucsampl5.out
uc_lp/uc_wlp.out       ucbend/ucbend.inp      ucsampcg/ucsampcg.data
uc_lp/uc_wolp.out      ucbend/ucbend.log      ucsampcg/ucsampcg.f
uccyl/uccyl.f          ucbend/ucbend.pic      ucsampcg/ucsampcg.inp
uccyl/uccyl.inp        ucsampl5/ucsampl5.f    ucsampcg/ucsampcg.out
```

Files in subdirectories under egs5/tutorcodes:

```
tutor1/tutor1.f        tutor3/tutor3.out      tutor6/tutor6.inp
tutor1/tutor1.inp      tutor4/tutor4.f        tutor6/tutor6.out
tutor1/tutor1.out      tutor4/tutor4.inp      tutor7/tutor7.f
tutor2/tutor2.f        tutor4/tutor4.out      tutor7/tutor7.inp
tutor2/tutor2.inp      tutor5/tutor5.f        tutor7/tutor7.w.out
tutor2/tutor2.out      tutor5/tutor5.inp      tutor7/tutor7_wo.out
tutor3/tutor3.f        tutor5/tutor5.out
tutor3/tutor3.inp      tutor6/tutor6.f
```

Files in subdirectories under egs5/extra_ucodes:

```
uc_examin/uc_examin.f        ucpegs/ucpegs.f          uctestsr/uctestsr.out
uc_examin/uc_examin.inp      ucpegs/ucpegs.inp        uctestsr/hplt1.inp
uc_examin/uc_examin.out      ucpegs/ucpegs.pegs5dat   uctestsr/hplt1.pegs5lst
uc_examin/uc_e_mfps_AL.xvgr  uctestsr/uctestsr.f
uc_examin/uc_ph_mfp_AL.xvgr  uctestsr/uctestsr.inp
```

E.6 Auxiliary Subprogram FORTRAN Source Files

The final set of files included with the EGS5 distribution are the "auxiliary" FORTRAN source codes (in subdirectory **auxcode** of **egs5**) and files containing the COMMON blocks (in subdirectory **auxcommons**) associated with these codes. The auxiliary codes contain functions and subroutines

that are useful in performing input, geometry setup, distance-to-boundary, and scoring computations for many generic problems, and are referenced by some of the advanced user codes provided with the EGS5 distribution. Detailed descriptions of the functionality of the subprograms can be found in the comments in the source code and in Chapter 3 of SLAC-R-730/KEK-2005-8. Lists of the files provided with the distribution are given below.

Auxiliary FORTRAN source files found in auxcode:

```
cg_related.f  cyl2.f        edistr.f      plan2p.f     sph2.f
chgtr.f       cylndr.f      fintrn.f      plan2x.f     sphere.f
cone.f        decod_xyz.f   finval.f      plane1.f     sphtrn.f
cone2.f       decodeir.f    geomout.f     plotxyz.f    swatch.f
cone21.f      ecnsv1.f      ntally.f      rdistr.f     xyzbound.f
```

"Included" auxiliary COMMON block files found in auxcommons:

```
aux_h.f              etaly2.f        instuf.f     sphdta.f
condta.f             etaly3.f        lines.f      trndta.f
cyldta.f             geom_common.f   nfac.f       voxel.f
dataconst_common.f   geortz.f        ntaly1.f     watch.f
edata.f              georz.f         pladta.f
etaly1.f             geoxyz.f        rdata.f
```

Bibliography

[1] R. G. Alsmiller Jr. Private communication with W.R. Nelson, 1970.

[2] R. G. Alsmiller Jr and J. Barish. High-energy (<18 GeV) muon transport calculations and comparison with experiment. *Nucl. Instrum. Methods*, 71:121–124, 1969.

[3] R. G. Alsmiller Jr and J. Barish. Energy deposition by 45 GeV photons in H, Be, Al, Cu, and Ta. Report ORNL-TM-4933, Oak Ridge National Laboratory, Oak Ridge, TN, 1974.

[4] R. G. Alsmiller Jr, J. Barish, and S. R. Dodge. Energy deposition by high-energy electrons (50 to 200 MeV) in water. *Nucl. Instrum. Methods*, 121:161–167, 1974.

[5] R. G. Alsmiller Jr and H. S. Moran. Electron-photon cascade calculations and neutron yields from electrons in thick targets. Report ORNL-TM-1502, Oak Ridge National Laboratory, Oak Ridge, TN, 1966.

[6] R. G. Alsmiller Jr and H. S. Moran. Electron-photon cascade calculations and neutron yields from electrons in thick targets. *Nucl. Instrum. Methods*, 48:109–116, 1967.

[7] R. G. Alsmiller Jr and H. S. Moran. The electron-photon cascade induced in lead by photons in the energy range 15 to 100 MeV. Report ORNL-TM-4192, Oak Ridge National Laboratory, Oak Ridge, TN, 1968.

[8] R. G. Alsmiller Jr and H. S. Moran. Calculation of the energy deposited in thick targets by high-energy (1 GeV) electron-photon cascades and comparison with experiment. *Nucl. Sci. Eng.*, 38:131–134, 1969.

[9] P. L. Anthony, R. Becker-Szendy, P. E. Bosted, M. Cavalli-Sforza, L. P. Keller, L. A. Kelley, S. R. Klein, G. Niemi, M. L. Perl, L. S. Rochester, and J. L. White. An accurate measurement of the Landau-Pomeranchuk-Migdal effect. Report SLAC-PUB-6796, Stanford Linear Accelerator Center, Stanford, CA, 1995.

[10] T. W. Armstrong and Jr. R. G. Alsmiller. An approximate density effect correction for ionization loss of charged particles. *Nucl. Instrum. Methods*, 82:289, 1970.

[11] W. N. Asaad. Intensities of the K-LL Auger lines. *Nucl. Phys.*, 44:399–414, 1963.

[12] J. J. Aubert, U. Becker, P. J. Biggs, J. Burger, M. Chen, G. Everhart, P. Goldhagen, J. Leong, T. McCorriston, T. G. Rhoades, M. Rohde, S. C. C. Ting, S. L. Wu, and Y. Y Lee. Experimental observation of a heavy particle *J. Phys. Rev. Lett.*, 33:1404–1406, 1974.

[13] J. E. Augustin, A. M. Boyarski, M. Breidenbach, F. Bulos, J. T. Dakin, G. J. Feldman, G. E. Fischer, D. Fryberger, G. Hanson, B. Jean-Marie, R. R. Larsen, V. Luth, H. L. Lynch, D. Lyon, C. C. Morehouse, J. M. Paterson, M. L. Perl, B. Richter, P. Rapidis, R. F. Schwitters, W. M. Tanenbaum, and F. Vannucci. Discovery of a narrow resonance in e^+e^- annihilation. *Phys. Rev. Lett.*, 33:1406–1408, 1974.

[14] J. Baró, J. Sempau, J. M. Fernández-Varea, and F. Salvat. PENELOPE: An algorithm for Monte Carlo simulation of the penetration and energy loss of electrons and positrons in matter . *Nucl. Instrum. Meth. B*, 100:31–46, 1995.

[15] M. J. Berger. Monte Carlo calculation of the penetration and diffusion of fast charged particles. In B. Adlu, S. Fernbach, and M. Rotenberg, editors, *Methods in Computational Physics Vol. I*, pages 135–215. Academic Press, New York, 1963.

[16] M. J. Berger. Private communication with W.R. Nelson, 1976.

[17] M. J. Berger and S. M. Seltzer. Tables of energy losses and ranges of electrons and positrons. Report NASA-SP-3012, National Aeronautics and Space Administration, 1964. (also National Academy of Sciences, National Research Council Publication 1133, 1964, Second Printing 1967).

[18] M. J. Berger and S. M. Seltzer. Calculation of energy and charge deposition and of the electron flux in a water medium bombarded with 20-MeV electrons. *Ann. N.Y. Acad. of Sci.*, 161:8–23, 1969.

[19] M. J. Berger and S. M. Seltzer. Bremsstrahlung and photoneutrons from thick tungsten and tantalum targets. *Phys. Rev. C*, 2:621–631, 1970.

[20] M. J. Berger and S. M. Seltzer. Stopping powers and ranges of electrons and positrons (2nd Ed.). Report NBSIR 82-2550-A, U. S. Department of Commerce, National Bureau of Standards, 1983.

[21] H. A. Bethe. Theory of passage of swift corpuscular rays through matter. *Ann. Physik*, 5:325, 1930.

[22] H. A. Bethe. Scattering of electrons. *Z. für Physik*, 76:293, 1932.

[23] H. A. Bethe. Molière's theory of multiple scattering. *Phys. Rev.*, 89:1256–1266, 1953.

[24] H. A. Bethe and J. Ashkin. Passage of radiation through matter. In E. Segre, editor, *Experimental Nuclear Physics, Vol. I, Part II*. Wiley & Sons, New York, 1953.

[25] H. J. Bhabha. Scattering of positrons by electrons with exchange on Dirac's theory of the positron. *Proc. R. Soc. Lon. Ser.-A)*, 154:195, 1936.

[26] A. F. Bielajew. Improved angular sampling for pair production in the EGS4 code system. Report PIRS-0287, National Research Council of Canada, 1991.

[27] A. F. Bielajew. Plural and multiple small-angle scattering from a screened Rutherford cross section. *Nucl. Instrum. Meth. B*, 86:257–269, 1994.

[28] A. F. Bielajew. HOWFAR and HOWNEAR: Geometry modeling for Monte Carlo particle transport. Report PIRS-0341, National Research Council of Canada, 1995.

[29] A. F. Bielajew, R. Mohan, and C. S. Chui. Improved bremsstrahlung photon angular sampling in the EGS4 code system. Report PIRS-0203, National Research Council of Canada, 1989.

[30] A. F. Bielajew and D. W. O. Rogers. Photoelectron angular distribution in the EGS4 code system. Report PIRS-0058, National Research Council of Canada, 1986.

[31] A. F. Bielajew and D. W. O. Rogers. PRESTA: The Parameter Reduced Electron-Step Transport Algorithm for electron Monte Carlo transport. Report PIRS-0042, National Research Council of Canada, 1986.

[32] A. F. Bielajew and D. W. O. Rogers. PRESTA: The Parameter Reduced Electron-Step Transport Algorithm for electron Monte Carlo transport. *Nucl. Instrum. Meth. B*, 18:165–181, 1987.

[33] 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, 1989.

[34] A. F. Bielajew and S. J. Wilderman. Innovative electron transport methods in EGS5. In *Proceedings of the Second International Workshop on EGS4*, pages 1–10, Japan, 2000. (KEK Proceedings 2000-20).

[35] F. Biggs, L. B. Mendelsohn, and J. B. Mann. Hartree-fock compton profiles for the elements. *Atom. Data Nucl. Data*, 16:201–309, 1975.

[36] F. Bloch. Stopping power of atoms with several electrons. *Z. für Physik*, 81:363, 1933.

[37] H. Burfeindt. Monte-Carlo-Rechnung für 3 GeV-Schauer in Blei. Report DESY-67/24, Deutsches Elektronen-Synchrotron, 1967.

[38] J. C. Butcher and H. Messel. Electron number distribution in electron-photon showers. *Phys. Rev.*, 112:2096–2106, 1958.

[39] J. C. Butcher and H. Messel. Electron number distribution in electron-photon showers in air and aluminum absorbers. *Nucl. Phys.*, 20:15–28, 1960.

[40] J. W. Butler. Machine sampling from given probability distributions. In H. A. Meyer, editor, *Symposium on Monte Carlo Methods*. Wiley & Sons, New York, 1966.

[41] L. L. Carter and E. D. Cashwell. Particle-transport simulation with the monte carlo method. Report TID-26607, National Technical Information Service, U. S. Department of Commerce, Springfield, VA, 1975.

[42] E. D. Cashwell and C. J. Everett. *Monte Carlo Method for Random Walk Problems*. Pergamon Press, New York, 1959.

[43] 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.

[44] E. Casnati, A. Tartari, and C. Baraldi. An empirical approach to K-shell ionisation cross section by electrons. *J. Phys. B*, 16:505, 1983.

[45] A. Clark (Lawrence Berkeley Laboratory). Private communication with W. R. Nelson and R. L. Ford, August 1977.

[46] H. M. Colbert. SANDYL: A computer program for calculating combined photon-electron transport in complex systems. Report SCL-DR-720109, Sandia Laboratories, Livermore, CA, 1973.

[47] M. Conti, A. Del Guerra, D. Mazzei, P. Russo, W. Bencivelli, E. Bartolucci, A. Messineo, V. Rosso, A. Stefanini, U. Bottigli, P. Randaccio, and W. R. Nelson. Use of the EGS4 Monte Carlo code to evaluate the response of HgI$_2$ and CdTe detectors for photons in the diagnostic energy range. *Nucl. Instrum. Meth. A*, 322:591–595, 1992.

[48] A. J. Cook. Mortran3 user's guide. Technical Memorandum CGTM 209, SLAC Computation Research Group, 1983.

[49] H. Davies, H. A. Bethe, and L. C. Maximon. Theory of bremsstrahlung and pair production. II. Integral cross section for pair production. *Phys. Rev.*, 92:788, 1954.

[50] C. M. Davisson and R. D. Evans. Gamma-ray absorption coefficients. *Rev. Mod. Phys.*, 24:79–107, 1952.

[51] A. Del Guerra, W. R. Nelson, and P. Russo. A simple method to introduce K-edge sampling for compounds in the code EGS4 for X-ray element analysis. *Nucl. Instrum. Meth. A*, 306:378–385, 1991.

[52] S. Duane, A. F. Bielajew, and D. W. O. Rogers. Use of ICRU-37/NBS collision stopping powers in the EGS4 system. Report PIRS-0173, National Research Council of Canada, 1989.

[53] R. D. Evans. *The Atomic Nucleus*. McGraw-Hill, New York, 1955.

[54] 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.

[55] E. L. Feinberg and I. Pomeranchuk. High energy inelastic diffraction phenomena. *Nuovo Cimento*, 3:652, 1956.

[56] J. M. Fernández-Varea, R. Mayol, J. Baró, and F. Salvat. On the theory and simulation of multiple elastic scattering of electrons. *Nucl. Instrum. Meth. B*, 73:447–473, 1993.

[57] R. B. Firestone and V. S. Shirley, editors. *Table of Isotopes*. Wiley & Sons, New York, 8th edition, 1996.

[58] J. Fischer. Beiträge zur Theorie der Absorption von Röntgenstrahlen. *Ann. Physik*, 8:821–850, 1931.

[59] K. Flöttmann. Investigations toward the development of polarized and unpolarized high intensity positron sources for linear colliders . Report DESY-93-161, Deutsches Elektronen-Synchrotron, 1993. Dissertation zur Erlangung des Doktorgrades des Fachbereichs Physik der Universität Hamburg.

[60] R. L. Ford, B. L. Beron, R. L. Carrington, R. Hofstadter, E. B. Hughes, G. I. Kirkbridge, L. H. O'Neill, and J. W. Simpson. Performance of large, modularized NaI(Tl) detectors . Report HEPL-789, Stanford University High Energy Physics Laboratory, 1976.

[61] R. L. Ford and W. R. Nelson. The EGS code system: Computer programs for the Monte Carlo simulation of electromagnetic cascade showers (version 3). Report SLAC-210, Stanford Linear Accelerator Center, 1978.

[62] M. L. Goldberger. The interaction of high energy neutrons and heavy nuclei. *Phys. Rev.*, 74:1269–1277, 1948.

[63] S. A. Goudsmit and J. L. Saunderson. Multiple scattering of electrons. *Phys. Rev.*, 57:24–29, 1940.

[64] S. A. Goudsmit and J. L. Saunderson. Multiple scattering of electrons. II. *Phys. Rev.*, 58:36–42, 1940.

[65] M. Gryziński. Classical theory of atomic collisions. I. Theory of inelastic collisions. *Phys. Rev. A*, 138:336–358, 1965.

[66] M. Gryziński. Two-particle collisions. I. General relations for collisions in the laboratory system. *Phys. Rev. A*, 138:305–321, 1965.

[67] M. Gryziński. Two-particle collisions. II. Coulomb collisions in the laboratory system of coordinates. *Phys. Rev. A*, 138:322–335, 1965.

[68] J. A. Halbleib, Sr., and W. H. Vandevender. CYLTRAN. *Nucl. Sci. Eng.*, 61:288–289, 1976.

[69] P. R. Halmos. *Measure Theory*. Van Nostrand Co., Princeton, NJ, 1950.

[70] J. H. Hammersley and D. C. Handscomb. *Monte Carlo Methods*. Wiley & Sons, New York, 1964.

[71] W. Heitler. *The Quantum Theory of Radiation*. Clarendon Press, Oxford, 1954.

[72] H. Hirayama and Y. Namito. Implementation of a general treatment of photoelectric-related phenomena for compounds or mixtures in EGS4. KEK Internal 2000-3, High Energy Accelerator Research Organization, Japan, 2000.

[73] H. Hirayama and Y. Namito. Implementation of a general treatment of photoelectric-related phenomena for compounds or mixtures in EGS4 (revised version). KEK Internal 2004-6, High Energy Accelerator Research Organization, Japan, 2004.

[74] H. Hirayama, Y. Namito, and S. Ban. Effects of linear polarisation and Doppler broadening on the exposure buildup factors of low-energy gamma rays. KEK Preprint 93-186, National Laboratory for High Energy Physics, Japan, 1994.

[75] H. Hirayama, W. R. Nelson, A. Del Guerra, T. Mulera, and V. Perez-Mendez. Monte Carlo studies for the design of a lead glass drift calorimeter. *Nucl. Instrum. Methods*, 220:327, 1984.

[76] H. Hirayama and D. K. Trubey. Effects of incoherent and coherent scattering on the exposure buildup factors of low-energy gamma rays. *Nucl. Sci. Eng.*, 99:145–156, 1988.

[77] J. H. Hubbell and I. Øverbø. Relativistic atomic form factors and photon coherent scattering cross sections. *J. Phys. Chem. Ref. Data*, 8:69–105, 1979.

[78] J. H. Hubbell, W. J. Veigele, E. A. Briggs, R. T. Brown, D. T. Cromer, and R J Howerton. Atomic form factors, incoherent scattering functions, and photon scattering cross sections. *J. Phys. Chem. Ref. Data*, 4:471–538, 1975.

[79] ICRU. Stopping powers for electrons and positrons. ICRU Report 37, International Commission on Radiation Units and Measurements, Washington, D.C., 1984.

[80] C. Jakoby, H. Genz, and A. Richter. *J. Phys. (Paris) Colloq.*, C9:487, 1987.

[81] F. James. A review of pseudorandom number generators. Report DD/88/22, CERN-Data Handling Division, 1988.

[82] F. James. RANLUX: A FORTRAN implementation of the high-quality pseudorandom number generator of Lüscher. *Comput. Phys. Commun.*, 79:111–114, 1994.

[83] J. M. Jauch and F. Rohrlich. *The Theory of Photons and Electrons.* Addison-Wesley, Reading, MA, 1955.

[84] P. C. Johns and M. J. Yaffe. Coherent scatter in diagnostic radiology. *Med. Phys.*, 10:40, 1983.

[85] H. Kahn. Applications of Monte Carlo. USAEC Report AECU-3259, The Rand Corporation, 1954.

[86] P. P. Kane, R. H. Pratt L. Kissel, and S. C. Roy. Elastic scattering of gamma-rays and x-rays by atoms. *Phys. Rep.*, 140:75–159, 1986.

[87] K. R. Kase and W. R. Nelson. *Concepts of Radiation Dosimetry.* Pergamon Press, New York, 1979.

[88] I. Kawrakow and A. F. Bielajew. On the condensed history technique for electron transport. *Nucl. Instrum. Meth. B*, 142:253–280, 1998.

[89] J. F. C. Kingman and S. J. Taylor. *Introduction to Measure and Probability.* Cambridge University Press, 1966.

[90] O. Klein and Y. Nishina. Über die Streuung von Strahlung durch freie Electronen nach der Neuen Relativistischen Quantum Dynamic von Dirac. *Z. für Physik*, 52:853, 1929.

[91] H. W. Koch and J. W. Motz. Bremsstrahlung cross-section formulas and related data. *Rev. Mod. Phys.*, 31:920–955, 1959.

[92] H. Kolbenstvedt. Simple theory for K-ionization by relativistic electrons. *J. Appl. Phys.*, 38:4785–4787, 1967.

[93] T. Kotseroglou, V. Bharadwaj, J. Clendenin, S. Ecklund, J. Frisch, P. Krejcik, A. Kulikov, J. Liu, T. Maruyama, K. K. Millage, G. Mulhollan, W. R. Nelson, D. C. Schultz, J. C. Sheppard, J. Turner, K. VanBibber, K. Flöttmann, and Y. Namito. Recent developments on the design of the NLC positron source. In *Proceedings of the 1999 Particle Accelerator Conference (PAC'99)*, 1999.

[94] L. Landau and I. J. Pomeranchuk. Electron-cascade processes at ultra-high energies. *Dokl. Akad. Nauk. SSSR*, 92:735–738, 1953.

[95] L. Landau and I. J. Pomeranchuk. The limits of applicability of the theory of bremsstrahlung by electrons and of creation of pairs at large energies. *Dokl. Akad. Nauk. SSSR*, 92:535–536, 1953.

[96] E. W. Larsen. A theoretical derivation of the condensed history algorithm. *Ann. Nucl. Energy*, 19:701–714, 1992.

[97] H. W. Lewis. Multiple scattering in an infinite medium. *Phys. Rev.*, 78:526–529, 1950.

[98] J. C. Liu, T. Kotseroglou, W. R. Nelson, and D. C. Schultz. Polarization study for NLC positron source using EGS4. In *Proceedings of the Second International Workshop on EGS4*, Japan, 2000. (KEK Proceedings 2000-20).

[99] M. Loeve. *Probability Theory*. Van Nostrand Reinhold Co., New York, 1950.

[100] G. A. Loew and Juwen Wang (SLAC). Private communication.

[101] C. Malamut, D. W. O. Rogers, and A. F. Bielajew. Calculation of water/air stopping-power ratios using EGS4 with explicit treatment of electron—positron differences. *Med. Phys.*, 18:1222–1228, 1991.

[102] E. J. McGuire. Atomic L-shell Coster Kronig, Auger, and radiative rates and fluorescence yields for Na-Th. *Phys. Rev. A*, 3:587–594, 1971.

[103] H. Messel and D. F. Crawford. *Electron-Photon Shower Distribution Function*. Pergamon Press, Oxford, 1970.

[104] H. Messel, A. D. Smirnov, A. A. Varfolomeev, D. F. Crawford, and J. C. Butcher. Radial and angular distributions of electrons in electron-photon showers in lead and in emulsion absorbers. *Nucl. Phys.*, 39:1–88, 1962.

[105] L. M. Middleman, R. L. Ford, and R. Hofstadter. Measurement of cross sections for x-ray production by high-energy electrons. *Phys. Rev.*, 2:1429–1443, 1970.

[106] A. B. Migdal. Bremsstrahlung and pair production in condensed media at high energies. *Phys. Rev.*, 103:1811, 1956.

[107] G. Z. Molière. Theorie der Streuung schneller geladener Teilchen. I. Einzelstreuung am abgeschirmten Coulomb-Field. *Z. Naturforsch*, 2a:133–145, 1947.

[108] G. Z. Molière. Theorie der Streuung schneller geladener Teilchen. II. Mehrfach- und Vielfach-streuung. *Z. Naturforsch*, 3a:78–97, 1948.

[109] C. Møller. Passage of hard beta rays through matter. *Ann. Physik*, 14:531, 1932.

[110] L. R. M. Morin. Molecular form factors and photon coherent scattering cross sections for water. *J. Phys. Chem. Ref. Data*, 11:1091, 1982.

[111] J. W. Motz, H. A. Olsen, and H. W. Koch. Pair production by photons. *Rev. Mod. Phys.*, 41:581–639, 1969.

[112] H. H. Nagel. *Die Berechnung von Elektron-Photon-Kaskaden in Blei mit Hilfe der Monte-Carlo Methode*. Inaugural-Dissertation zur Erlangung des Doktorgrades der Hohen Mathe-matich-Naturwissenschaftlichen Fakultät der Rheinischen Freidrich-Wilhelms-Universität, Bonn, 1964.

[113] H. H. Nagel. Elektron-Photon-Kaskaden in Blei: Monte-Carlo-Rechnungen für Primär-elektronenergien zwischen 100 und 1000 MeV. *Z. Physik*, 186:319–346, 1965. English trans-lation, Stanford Linear Accelerator Center Report Number SLAC-TRANS-28, 1965.

[114] H. H. Nagel. Private communication in the form of notes left by the author at the Stanford Linear Accelerator Center, August 1966.

[115] H. H. Nagel and C. Schlier. Berechnung von Elektron-Photon-Kaskaden in Blei für eine Primärenergie von 200 MeV. *Z. Physik*, 174:464–471, 1963.

[116] Y. Namito, S. Ban, and H. Hirayama. Implementation of linearly-polarized photon scattering into the EGS4 code. *Nucl. Instrum. Meth. A*, 322:277–283, 1993.

[117] Y. Namito, S. Ban, and H. Hirayama. Implementation of Doppler broadening of Compton-scattered photons into the EGS4 code. *Nucl. Instrum. Meth. A*, 349:489–494, 1994.

[118] Y. Namito, S. Ban, H. Hirayama, S. Tanaka, H. Nakashima, Y. Nakane, Y. Sakamoto, N. Sasamoto, and Y. Asano. Compton scattering of 20- to 40-keV photons. *Phys. Rev. A*, 51:3036–3043, 1995.

[119] Y. Namito and H. Hirayama. Improvement of low energy photon transport calculation by EGS4—electron bound effect in Compton scattering. *J. Atom. Energ. Soc. Jpn.*, page 401, 1991.

[120] Y. Namito and H. Hirayama. Implementation of the electron impact ionization into the EGS4 code. *Nucl. Instrum. Meth. A*, 423:238–246, 1999.

[121] Y. Namito and H. Hirayama. Improvement of the cross-section and branching-ratio evaluation in EGS4 in the energy interval which has an absorption-edge. In *Proceedings of the Eighth EGS4 User's Meeting in Japan*, pages 1–6, Japan, 1999. (KEK Proceedings 99-15).

[122] Y. Namito and H. Hirayama. LSCAT: Low-energy photon-scattering expansion for the EGS4 code (inclusion of electron impact ionization). KEK Internal 2000-4, High Energy Accelerator Research Organization, Japan, 2000.

[123] Y. Namito, H. Hirayama, A. Takamura, and T. Sugita. EGS4 Particle trajectory and geometry display program cgview ver 1.2. KEK Internal 2004-8, High Energy Accelerator Research Organization, Japan, 2004.

[124] Y. Namito, W. R. Nelson, S. M. Seltzer, A. F. Bielajew, and D. W. O. Rogers. Low-energy x-ray production studies using the EGS4 code system . *Med. Phys.*, 17:557 (abstract), 1990.

[125] W. R. Nelson and A. F. Bielajew. EGS — A Technology spinoff to medicine. *Beam Line (#1)*, 21:7–11, 1991.

[126] W. R. Nelson, H. Hirayama, and D. W. O. Rogers. The EGS4 Code System. Report SLAC-265, Stanford Linear Accelerator Center, Stanford, CA, 1985.

[127] W. R. Nelson and T. M. Jenkins, editors. *Computer Techniques in Radiation Transport and Dosimetry*. Plenum Press, New York, 1980.

[128] D. F. Nicoli. The application of Monte Carlo cascade shower generation in lead, 1966. submitted in partial fulfillment of the requirement for the degree of Bachelor of Science at the Massachusetts Institute of Technology.

[129] J. Pach and P. K. Agarwal. *Combinatorial Geometry*. Wiley & Sons, New York, 1995.

[130] E. Parzen. *Modern Probability Theory and Its Applications*. Wiley & Sons, New York, 1960.

[131] PHOTX. Photon interaction cross-section library for 100 elements. Data Package DLC-136/PHOTX, Radiation Shielding Information Center, Oak Ridge National Laboratory, Oak Ridge, TN, 1995.

[132] S. Pollack. Private communication, 2002.

[133] J. A. Rawlinson, A. F. Bielajew, P. Munro, and D. M. Galbraith. Theoretical and experimental investigation of dose enhancement due to charge storage in electron-irradiated phantoms. *Med. Phys.*, 11:814–821, 1984.

[134] R. Ribberfors. X-ray incoherent scattering total cross sections and energy-absorption cross sections by means of simple calculation routines. *Phys. Rev. A*, 27:3061–3070, 1983.

[135] R. Ribberfors and K. F. Berggren. Incoherent x-ray-scattering functions and cross sections $(d\sigma/d\omega')_{\text{incoh}}$ by means of a pocket calculator. *Phys. Rev. A*, 26:3325–3333, 1982.

[136] D. W. O. Rogers. Low energy electron transport with EGS. *Nucl. Instrum. Methods*, 227:535–548, 1984.

[137] D. W. O. Rogers. The role of Monte Carlo simulation of electron transport in radiation dosimetry. *Int. J. Appl. Radiat. Is.*, 42:965–974, 1991.

[138] D. W. O. Rogers. How accurately can EGS4/PRESTA calculate ion chamber response? *Med. Phys.*, 20:319–323, 1993.

[139] 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. Report PIRS-0177, National Research Council of Canada, 1989.

[140] F. Rohrlich and B. C. Carlson. Positron-electron differences in energy loss and multiple scattering. *Phys. Rev.*, 93:38, 1954.

[141] B. Rossi. *High-Energy Particles*. Prentice-Hall, Englewood Cliffs, NJ, 1952.

[142] R. W. Roussin, J. R. Knight, J. H. Hubbell, and R. J. Howerton. Description of the DLC-99/HUGO package of photon interaction data in ENDF/B-V format. Report ORNL-RSIC-46/ENDF-335, Radiation Shielding Information Center, Oak Ridge National Laboratory, Oak Ridge, TN, 1983.

[143] Y. Sakamoto. Photon cross section data PHOTX for PEGS4 code. In *Proceedings of the Third EGS4 User's Meeting in Japan*, pages 77–82, Japan, 1993. (KEK Proceedings 93-15).

[144] S. I. Salem, S. L. Panossian, and R. A. Krause. Experimental K and L relative x-ray emission rates. *Atom. Data Nucl. Data*, 14:91–109, 1974.

[145] F. Sauter. Über den atomaren Photoeffekt in der K-Schale nach der relativistischen Wellenmechanik Diracs. *Ann. Physik*, 11:454–488, 1931.

[146] J. H. Scofield. Relativistic Hartree-Slater values for K and L x-rays emission rates. *Atom. Data Nucl. Data*, 14:121–137, 1974.

[147] W. T. Scott. The theory of small-angle multiple scattering of fast particles. *Rev. Mod. Phys.*, 35:231–313, 1963.

[148] S. M. Seltzer. A PC-based program EPSTAR/ESPA, 1988. private communication.

[149] 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, 1989.

[150] S. M. Seltzer. Electron-photon Monte Carlo calculations: the ETRAN code. *Int. J. Appl. Radiat. Is.*, 42:917–941, 1991.

[151] S. M. Seltzer and M. J. Berger. Evaluation of the collision stopping power of elements and compounds for electrons and positrons. *Int. J. Appl. Radiat. Is.*, 33:1189, 1982.

[152] S. M. Seltzer and M. J. Berger. Improved procedure for calculating the collision stopping powers of elements and compounds for electrons and positrons. *Int. J. Appl. Radiat. Is.*, 35:665, 1984.

[153] S. M. Seltzer and M. J. Berger. Bremsstrahlung spectra from electron interactions with screened atomic nucleii and orbital electrons. *Nucl. Instrum. Meth. B*, 12:95, 1985.

[154] S. M. Seltzer and M. J. Berger. Procedure for calculating the radiation stopping power for electrons. *Int. J. Appl. Radiat. Is.*, 33:1219, 1985.

[155] J. Sempau, S. J. Wilderman, and A. F. Bielajew. DPM, a fast, accurate Monte Carlo code optimized for photon and electron radiotherapy treatment planning dose calculations. *Phys. Med. Biol.*, 45:2263–2291, 2000.

[156] Y. A. Shreider, editor. *The Monte Carlo Method*. Pergamon Press, New York, 1966.

[157] J. Spanier and E. M. Gelbard. *Monte Carlo Principles and Neutron Transport Problems*. Addison-Wesley, Reading, MA, 1969.

[158] J. E. Spencer. Private communication with A.F. Bielajew and W.R. Nelson regarding the proposed SLAC E144 experiment, 1991.

[159] R. M. Sternheimer. The density effect for the ionization loss in various materials. *Phys. Rev.*, 88:851, 1952.

[160] R. M. Sternheimer. The energy loss of a fast charged particle by Cerenkov radiation. *Phys. Rev.*, 91:256, 1953.

[161] R. M. Sternheimer. Density effect for the ionization loss in various materials. *Phys. Rev.*, 103:511, 1956.

[162] R. M. Sternheimer. Density effect for the ionization loss of charged particles. *Phys. Rev.*, 145:247, 1966.

[163] R. M. Sternheimer. Density effect for the ionization loss of charged particles. II. *Phys. Rev.*, 164:349, 1967.

[164] R. M. Sternheimer, M. J. Berger, and S. M. Seltzer. Density effect for the ionization loss of charged particles in various substances. *At. Data Nucl. Data Tables*, 30:261, 1984.

[165] R. M. Sternheimer and R. F. Peierls. General expression for the density effect for the ionization loss of charged particles. *Phys. Rev. B*, 3:3681, 1971.

[166] R. M. Sternheimer, S. M. Seltzer, and M. J. Berger. Density effect for the ionization loss of charged particles in various substances. *Phys. Rev. B*, 26:6067, 1982.

[167] E. Storm and H. I. Israel. Photon cross sections from 1 keV to 100 MeV for elements $Z=1$ to $Z=100$. *Atom. Data Nucl. Data*, 7:565–681, 1970.

[168] E. A. Straker, P. N. Stevens, D. C. Irving, and V. R. Cain. MORSE-CG, general purpose Monte Carlo multigroup neutron and gamma-ray transport code with combinatorial geometry. Report CCC-203, Radiation Shielding Information Center, Oak Ridge National Laboratory, Oak Ridge, TN, 1976.

[169] T. Sugita, T. Torii, and A. Takamura. Incorporating combinatorial geometry to the EGS5 code and its speed-up. In *Proceedings of the 12th EGS Users' Meeting in Japan*, pages 7–21, Japan, 2005. (KEK Proceedings 2005-10).

[170] M. L. Ter-Mikaelyan. Bremsstrahlung radiation spectrum in medium. *Zu Eksper. Teor. Fiz.*, 25:289–296, 1954.

[171] T. Torii and T. Sugita. Development of PRESTA-CG incorporating combinatorial geometry in EGS4/PRESTA. Report JNC TN1410 2002-201, Japan Nuclear Cycle Development Institute, Japan, 2002.

[172] Y. S. Tsai. Pair production and bremsstrahlung of charged leptons. *Rev. Mod. Phys.*, 46:815, 1974.

[173] J. E. Turner. Values of I_{adj} suggested by the subcommittee. In *Studies in Penetration of Charged Particles in Matter*. National Academy of Sciences, National Research Council, 1967. Publication Number 1133.

[174] S. M. Ulam and J. von Neumann. On combination of stochastic and deterministic processes. *Bull. Amer. Math Soc.*, 53:1120, 1947.

[175] A. Van Ginneken. Lecture 14: Calculation of the average properties of electromagnetic cascades at high energies (AEGIS). In W. R. Nelson and T. M. Jenkins, editors, *Computer Techniques in Radiation Transport and Dosimetry*. Plenum Press, New York, 1980.

[176] A. A. Varfolomeev and I. A. Svetlolobov. Monte Carlo calculations of electromagnetic cascades with account of the influence of the medium on bremsstrahlung. *Soviet Physics JETP*, 36:1263–1270, 1959.

[177] U. Völkel. Elektron-Photon-Kaskaden in Blei für Primärteilchen der Energie 6 GeV. *Deutsches Elektronen-Synchrotron Report Number DESY-65/6*, 1965. English translation, Stanford Linear Accelerator Center Report Number SLAC-TRANS-41 (1966).

[178] I. Waller and D. R. Hartree. On the intensity of total scattering of x-rays. *Proc. R. Soc. Lon. Ser.-A*, 124:119–142, 1929.

[179] R. R. Wilson. Monte Carlo study of shower production. *Phys. Rev.*, 86:261–269, 1952.

[180] C. D. Zerby and H. S. Moran. A Monte Carlo calculation of the three-dimensional development of high-energy electron-photon cascade showers. Report ORNL-TM-422, Oak Ridge National Laboratory, Oak Ridge, TN, 1962.

[181] C. D. Zerby and H. S. Moran. Studies of the longitudinal development of high-energy electron-photon cascade showers in copper. Report ORNL-3329, Oak Ridge National Laboratory, Oak Ridge, TN, 1962.

[182] C. D. Zerby and H. S. Moran. Studies of the longitudinal development of electron-photon cascade showers. *J. Appl. Phys.*, 34:2445–2457, 1963.

Index