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

This EGS5 User Manual is Appendix B of a document called
SLAC-R-730/KEK-2005-8, which can be obtained from the SLAC and KEK web sites.

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:

	Calls to subroutines:
PEGS5	to create media data
HATCH	to establish media data
SHOWER	to initiate the cascade
	Calls from EGS to user subroutines:
HOWFAR	to specify the geometry
AUSGAB	to score and output the results
	Altering elements of COMMON *blocks:*
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 orginal EGS4 default.

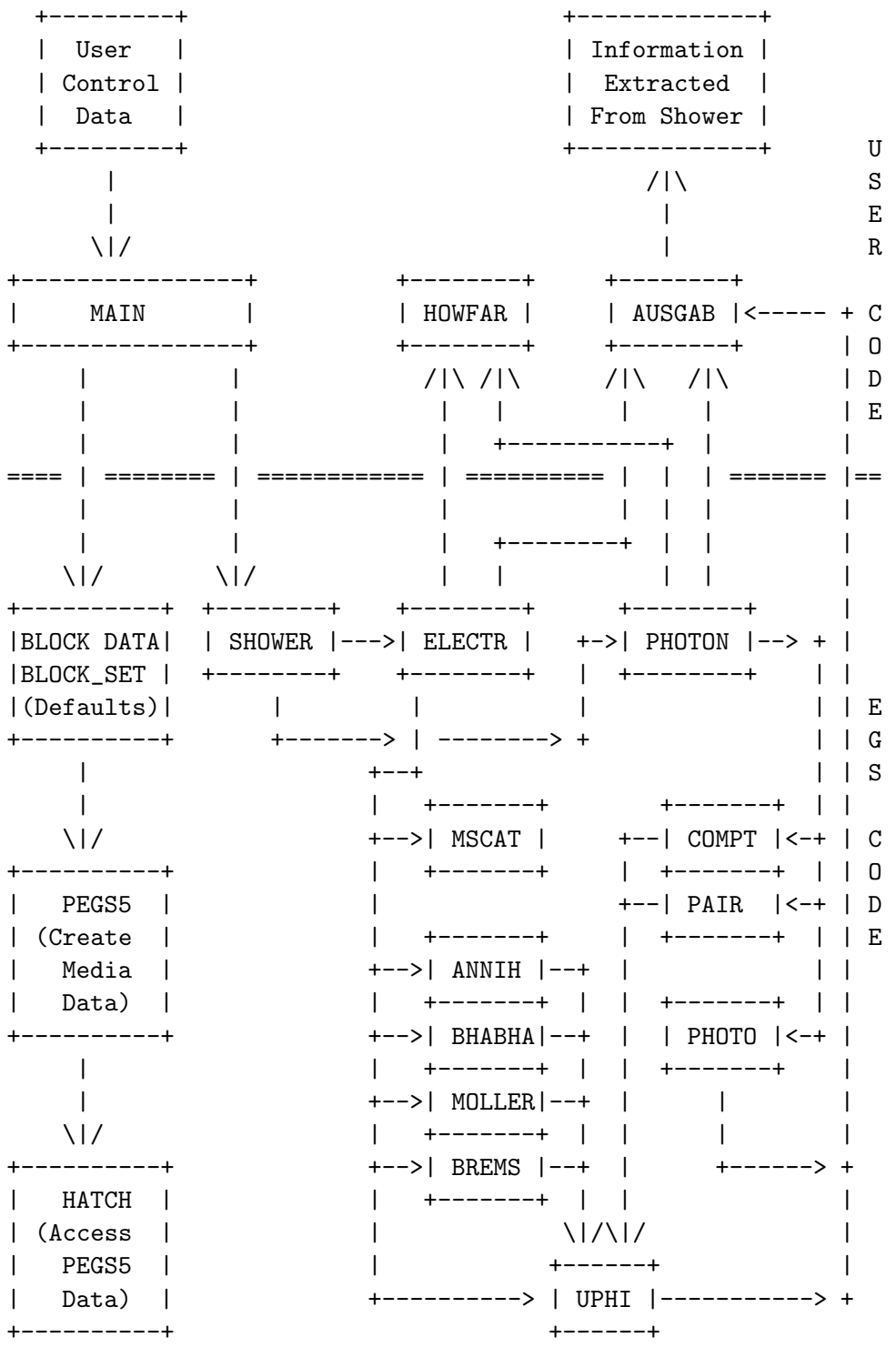

Figure B.1: EGS5 user code control and data flow diagram.

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/E).
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 distances," 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.

MSFRAC*	Fraction of initial multiple scattering step at which to force the first multiple scattering for highly differential problems with strong angular dependence on secondary particle production at the initial surface. (default=0, no forcing).
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.

`ESTEPE*`	Fractional energy loss to take before imposing an energy hinge at highest energy of the problem (default=0.05).
`ESTEPE2*`	Fractional energy loss before imposing an energy hinge at lowest energy of the problem (default=0.1).
`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).

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 isotropic emission).

```
      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 estepe/estepe2
      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`.

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(i,j)
        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. It is also used to set the energy loss hinge step sizes based on ECUT, ESTEPE and ESTEPE2.

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 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 isotropic emission.

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 modeling 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).

ESTEPE and ESTEPE2 The parameters ESTEPE and ESTEPE2 (found in COMMON/USERSC) define the fractional energy losses between electron energy loss hinges at the maximum and minimum electron energies of the problem, respectively. Fractional energy losses at other are interpolated between ESTEPE and ESTEPE2 in a log-linear scale. The default values for the two variables are .05 and .1, respectively, corresponding to energy loss hinges of 5% and 10%. Because of the complete decoupling of electron energy loss and multiple scattering in EGS5, values of ESTEPE do not play the same role in determining the accuracy of simulations at lower energies than they did in EGS4. In EGS5, the energy hinges need be short enough only so that the energy dependence of media variables (such as the hard collision cross section, scattering power, and stopping power) is roughly linearly over the energy hinge. In addition, since hard collisions result in the imposition of *de facto* energy hinges in EGS5, in simulations for which the values of AE and AP are relatively low and hard collision cross sections are high, energy loss hinges based on ESTEPE can be redundant and add only inefficiency to the simulation. The default values of ESTEPE and ESTEPE2 in EGS5 are quite conservative and should provide sufficiently accurate results in almost all applications. For simulations which require very low variance, the user is, as with all Monte Carlo simulations involving electrons, encouraged to modify the default values of ESTEPE and ESTEPE2 until results converge.

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 addition, the appropriate requests to pre-compute the distribution function tables must have been specified in the PEGS input files prior to the call to PEGS5, or the appropriate data table files must exist in the working directory. 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.

MSFRAC When MSFRAC (found in COMMON/MS) in non-zero, ELECTR scales the initial multiple scattering step for an electron by MSFRAC times the mean free path to a hard collision. The default for MSFRAC(MXMED) is zero, in which case the scaling is not done. This scaling, which is used *only* for the first step for primary source electrons, is sometimes necessary in problems which involve parallel- or pencil- beam electron sources and in which the output tallies are strongly dependence on the direction of the source electrons at the time of the first hard collision. Because of the random hinge transport mechanics in EGS5, any electron undergoing a hard collision prior to encountering a multiple-scattering hinge is moving in it's initial direction, and for problems which tally, for example, the angular distribution of forward-emitted bremsstrahlung photons, there may be a correlation. Imposing a single multiple-scattering hinge prior to the first hard collision (values of MSFRAC on the order of 25%) resolves this problem.

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 the energy hinge steps in EGS5 defined by ESTEPE and ESTEPE2 are global for all materials and regions, 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 $1/k$
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
```

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

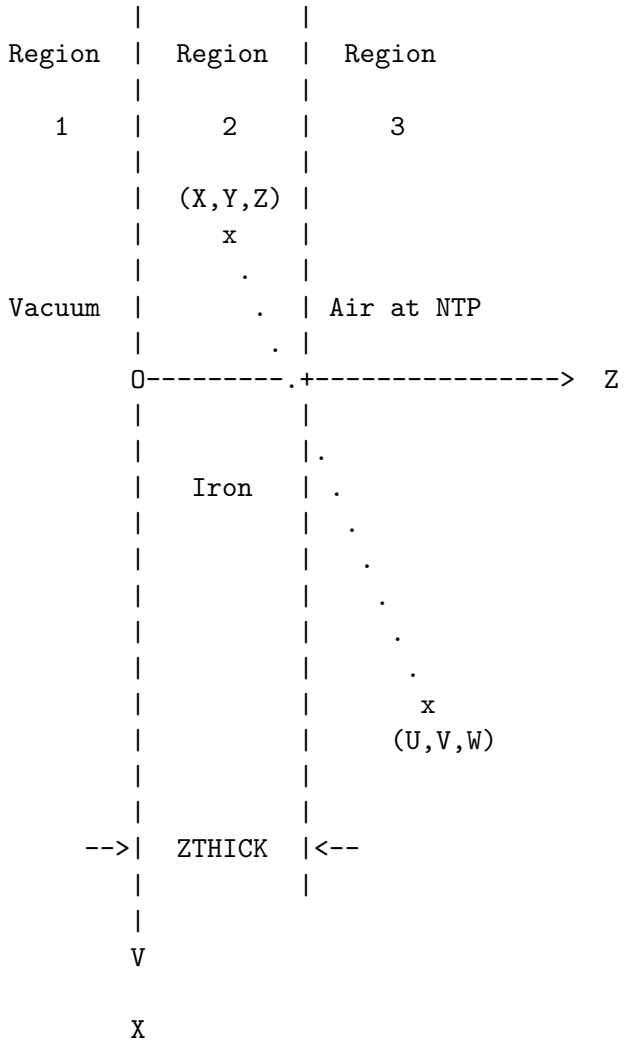

Figure B.2: A three-region geometry for a HOWFAR example code.

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:

```
      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.

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.

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.

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
```

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.

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 2004-5.
!
! 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                                            *
!                                                                     *
```

```
!*********************************************************************
!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')
```

```
!     ====================
      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
!     --------------------------------
```

```
      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

!     -------------------------------
!     Set parameter estepe and estepe2
!     -------------------------------
      estepe=0.10
      estepe2=0.20
      write(6,110) estepe, estepe2
110   FORMAT(1X,'ESTEPE at EKMAX: ',F10.5,' (estepe)',
     *       /,1X,'ESTEPE at ECUT:  ',F10.5,' (estepe2)',/)

!     ------------------------------------------------------
!     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-730, KEK-2004-5 (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-730, KEK-2004-5 (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
!-----------------------------------------------------
      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------------------------
```