PENELOPE/penEasy

User Manual

Josep Sempau

Department of Physics
Technical University of Catalonia
josep.sempau@upc.es

Abstract: PENELOPE is a Monte Carlo simulation package that describes the transport of photons, electrons and positrons in complex geometries and materials of arbitrary composition. The associated package PENNUC allows the random generation of the radiation cascade following the decay of a radioactive nuclide. PenEasy is a general-purpose main program for PENELOPE and PENNUC that includes a set of source models, tallies and variance-reduction techniques that are invoked from a structured code. Users are required to input, through plain text files, the information needed to set up the simulation. The present document contains a detailed description of the operation of the penEasy code. Technical aspects of some of the algorithms are given in the appendices. Familiarity with the basic concepts of Monte Carlo simulation of radiation transport and with the operation of PENELOPE is assumed throughout the text. For information on these two topics the reader is referred to the PENELOPE manual. PENELOPE is freely distributed by the Nuclear Energy Agency Data Bank (https://www.oecd-nea.org/databank) and, in North America, by the Radiation Safety Information Computational Center of the Oak Ridge National Laboratory (https://rsicc.ornl.gov). PenEasy is freely available from http://www.upc.edu/inte/downloads. We refer to the combined software as PENELOPE/penEasy.

Manual version: 2019-09-26
PenEasy code version: 2019-09-21
Operates with:
PENELOPE version: 2018
PENNUC version: 2018

Copyright and disclaimers

penEasy copyright and disclaimer

penEasy
Copyright (c) 2003-2019
Universitat Politècnica de Catalunya

Permission to use, copy, modify and re-distribute copies of this software, or parts of it, and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appears in all copies and that both that copyright notice and this permission notice appear in all supporting documentation. The Universitat Politècnica de Catalunya makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

PENELOPE copyright and disclaimer

PENELOPE/PENGEOM (version 2018)
Copyright (c) 2001-2018
Universitat de Barcelona

Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appears in all copies and that both that copyright notice and this permission notice appear in all supporting documentation. The Universitat de Barcelona makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

PENNUC copyright and disclaimer

PENNUC (version 2018)
Copyright 2016-2018
CIEMAT, Laboratoire National Henri Becquerel, and Universitat de Barcelona

Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appears in all copies and that both that copyright notice and this permission notice appear in all supporting documentation. The Ciemat and the Universitat de Barcelona make no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

Contents

Acknowledgements

The following collaborators have made important contributions to the development of the code,

- Andreu Badal (U.S. Food & Drug Administration). Dr. Badal contributed with the development of penVox, which handles voxelized geometries in penEasy.
- Immaculada Martínez-Rovira (Alba Synchrotron, CELLS). Dr. Martínez-Rovira contributed with the implementation in penEasy of phase-space files in the IAEA format.

I am deeply indebted with my colleagues Francesc Salvat (Universitat de Barcelona), main author of the PENELOPE code, José M. Fernández-Varea (Universitat de Barcelona), coauthor of former versions of the same code, and with Eduardo García-Toraño (Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas, Madrid), coauthor, together with Prof. Salvat, of the PENNUC code. All of them have provided invaluable help that facilitated the link with the respective subroutine libraries.

I would like to make special mention of my close collaborators Lorenzo Brualla (West German Proton Therapy Center, Essen) and Miguel Rodriguez (Centro Médico Paitilla, Panama City), coauthors of the PRIMO code[1], which makes intensive use of the software described in this Manual. Together with Dr. Fernández-Varea, and as (very heavy) power users, their feedback, bug hunting and suggestions for improvement have significantly contributed to the present status of the code.

My colleagues at the Universitat Politècnica de Catalunya María A. Duch, Mercè Ginjaume and Anna Camp, among others, have also had a strong influence, as power users in the field of medical physics, on the development of the code and its applications.

Other people have tested the code, given advice or pointed out potential bugs and amendments. The detailed list of contributions is given in
~/documentation/changeHistory.txt
where the changes introduced in each version are described. I would like to express my gratitude to all of them for their disinterested collaboration.

I gratefully acknowledge partial financial support received over the years from the following institutions,

- CONCERT—European Joint Programme for the Integration of Radiation Protection Research, project 003-2017-PODIUM, EU Horizon 2020.
- Metrology for radiological early warning networks in Europe (MetroERM), European Association of National Metrology Institutes (EURAMET) project ENV57.
- German Deutsche Forschungsgemeinschaft (DFG), project BR 4043/3-1.

[1] See https://www.primoproject.net.

- Spanish Ministerio de Economía y Competitividad, project FIS2012-38480.
- Grup de Recerca Consolidat en Dosimetria i Radiofísica Mèdica, Spanish Generalitat de Catalunya project 2014SGR846.
- Spanish Consejo de Seguridad Nuclear (CSN).
- Spanish Ministerio de Educación y Ciencia, project FIS2006-07016.
- Fondo de Investigación Sanitaria (FIS), Spanish Ministerio de Sanidad y Consumo projects PI03/0980 and PI01/0093-02.

1 Introduction

PENELOPE (Salvat 2019; Sempau et al. 1997; Baró et al. 1995) is a Monte Carlo simulation package that describes the transport of photons, electrons and positrons in complex geometries and materials of arbitrary composition. It is freely distributed by the OECD Nuclear Energy Agency Data Bank (https://www.oecd-nea.org/databank) and, in North America, by the Radiation Safety Information Computational Center of the Oak Ridge National Laboratory (https://rsicc.ornl.gov). The core of the system is a set of Fortran subroutines that deal with the intricacies of the transport process. To be operative, this system must be completed with a steering main program which, among other things, defines the initial particle states (*i.e.*, the radiation source) and the tallies of the quantities of interest, *e.g.*, the absorbed dose in a certain spatial region. Examples of main programs are supplied with the distribution package.

PenEasy (Sempau et al. 2011) is a general-purpose main program for PENELOPE. It provides users with a set of source models, tallies and variance-reduction techniques that are invoked from a structured code. Users are required to input, through plain text files, the information needed to set up the simulation. The description of the geometry and its associated material properties are introduced via the usual PENELOPE geometry and material data files.

It is virtually impossible to devise a main program flexible enough to cope with all the imaginable situations encountered in radiation transport problems. Therefore, it is not uncommon that users find themselves in the need to adapt an existing code to accurately describe some experimental arrangement. To facilitate this task penEasy is designed as a structured code that significantly reduces the programming effort of adapting the existing routines and of creating (by duplication) new ones.

In this manual we assume familiarity with the basic concepts of Monte Carlo simulation of radiation transport and with the operation of PENELOPE. For a detailed description of both aspects the reader is referred to the PENELOPE manual (Salvat 2019).

The penEasy code is mostly written in Fortran 95, although it has recourse to some features from the Fortran 2003 standard. It is, like PENELOPE, free and open software. In return, we kindly ask users to cite our preferred reference (see below) in their publications. Please refer to the combined simulation system as PENELOPE/penEasy.

Preferred citation

When the use of penEasy leads to publications we kindly ask users to cite the reference Sempau et al. (2011), given here in BibTeX format for the user's convenience,

```
@ARTICLE{sempau2011,
  AUTHOR  ={J. Sempau and A. Badal and L. Brualla},
  TITLE  ={{A PENELOPE-based system for the automated Monte Carlo
            simulation of clinacs and voxelized geometries---application
            to far-from-axis fields}},
```

```
  JOURNAL ={Med. Phys.},
  VOLUME  ={38},
  YEAR    ={2011},
  PAGES   ={5887 - 5895},
  NOTE    ={Available at http://dx.doi.org/10.1118/1.3643029}
}
```

Where can I find the latest update? And make comments/questions?

PenEasy can be freely downloaded from
http://www.upc.edu/inte/downloads.

It is strongly recommended that you get the latest versions of PENELOPE and penEasy before attempting a new simulation project. Only the latest version is up-to-date. Outdated penEasy releases are neither supported nor maintained. The current penEasy version, and the corresponding PENELOPE version with which the former is designed to operate, are stated in the front page of the present manual.

Comments are most welcome, especially those regarding potential bugs of the code. You can send your comments or questions to josep.sempau@upc.es. Please read the documentation carefully before sending comments. Since our human resources are very scarce, failure to do so may result in a long response time.

Thank you for using PENELOPE/penEasy.

2 What you get

The penEasy package consists of an all-purpose main program and a set of subroutines. After decompressing the distribution ZIP file your working directory (represented below as ∼/) should contain the following files and directories (in alphabetical order),

∼/documentation/changeHistory.txt
List of changes with respect to older versions.

∼/documentation/dependencies.txt
List of the PENELOPE routines and data (Fortran common blocks and modules) used by penEasy. It can be of help for maintaining the penEasy code when changes are introduced in PENELOPE.

∼/FortranCode/
This directory contains penEasy's Fortran source code. Note that a compiled version for Windows systems with 64 bits architecture is included in the distribution—see below.

∼/FortranCode/penaux.F
Initialization and auxiliary routines.

∼/FortranCode/penEasy.F
The main program.

∼/FortranCode/penvox.F
Geometry package for the simulation of voxelized geometries, as described in section 9.

∼/FortranCode/penvr.F
Variance-reduction techniques (VRT) that extend the capabilities of those in PENELOPE. These VRTs are described in section 12.

∼/FortranCode/source*.F
Source models. Each model is coded in a different file, *e.g.*, sourcePhaseSpaceFile.F.

∼/FortranCode/tally*.F
Codes to tally the quantities of interest, such as fluence, absorbed energy, digital image, etc. Each tally is coded in a different file, *e.g.*, tallySpatialDoseDistrib.F.

∼/FortranCode/timing.F
Time routines compatible with the Fortran 95 and subsequent standards.

∼/gnuplotScripts/*.gpl
Gnuplot (version 4.2 or greater) scripts to represent graphically the simulation results. Each script has a name associated to a tally. See the example provided below to learn how to execute them and refer to the Gnuplot manual for details on how to modify these scripts, if needed.

∼/IAEAcode/*
C++ code and auxiliary files intended for reading and writing phase-space files (PSF) in the binary format defined by the International Atomic Energy Agency, IAEA (Capote

et al. 2006). Details on PSFs and on how to compile these files are given in section 14. Both the PSFs and the latest updates for the C++ code are distributed by the IAEA from http://www-nds.iaea.org/phsp.

~/manual.pdf
This manual.

~/run/
This directory contains the files needed to run a simulation example. See section 5 for details.

~/run/command.in
Users may change some simulation settings by editing this file while the program is running—*e.g.*, to stop the execution. More details are given below.

~/run/penEasy.exe
The executable code for Windows systems with 64 bits architecture. It has been created with the Intel Visual Fortran Compiler as described in section 14. See that section for details on the compiler version and the optimization options used.

~/run/penEasy.in
Sample input file. It also serves to run the example case described in section 5.

~/run/phantom.geo
A simple PENGEOM-conforming geometry file that is used in the example case described in section 5. PENGEOM is the set of routines included in PENELOPE to describe the geometry of objects limited by quadric surfaces. The file also contains the general layout of a PENGEOM geometry taken from the PENELOPE distribution.

~/run/sampleBIGSspectrum.spc
Example of a spectrum data file for the BIGS source model. See section 8.2 for details.

~/run/nuclides/*
Files used to define radionuclides with the BIGS source model. This feature is based on the PENNUC package (see section 8.2 for details). The content of this directory is an exact copy from the database in the PENNUC distribution, except for an adaptation of the directory name.

~/run/water.mat
This is the PENELOPE material data file for water. It is included here for your convenience, so that the example given in section section 5 of this manual can be run.

~/voxSample/visualizeVoxelsDensity.gpl
Gnuplot script to visualize the map of material densities in a voxels file.

~/voxSample/visualizeVoxelsMaterial.gpl
Gnuplot script to visualize the map of material indices in a voxels file.

~/voxSample/voxels.vox
A simple example of a voxelized geometry data file for penEasy.

All the above files, except the Windows executable, are in plain text (ASCII) format. Most of them use the Unix new line convention, which differs from that of Windows. As a result, some Windows text editors, notably Microsoft Notepad, may not interpret correctly the new line mark and display scrambled text. This problem is solved by using an editor that can read both Unix and Windows formats—such as Microsoft WordPad. Compilers accept both new line conventions quietly.

The penEasy distribution does not include the PENELOPE package, which must be obtained separately from the Nuclear Energy Agency Data Bank (https://www.oecd-nea.org/databank) or, in North America, from the Radiation Safety Information Computational Center of the Oak Ridge National Laboratory (https://rsicc.ornl.gov). The distribution does include an adapted version of the essential ingredients of the PENNUC package (see section 8.2) in ∼/FortranCode/sourcePennuc.F and ∼/run/nuclides/*.

3 How to install it

Unix-like systems

This subsection applies to any Unix-like system, which includes all Linuxes and Apple OSX. The installation is accomplished following these steps,

1. Unzip (or copy if already unzipped) all the penEasy files into a directory on your disc. Let us suppose that it has been named ∼/penEasy/.

2. Copy the files penelope.f, pengeom.f, penvared.f and rita.f from the PENELOPE package into ∼/penEasy/fortranCode/. These files are not included in the penEasy distribution; they must be obtained separately at the address indicated in the Introduction, section 1.

3. Compile and link penEasy.F. For example, with the gfortran compiler, which is freely available from http://gcc.gnu.org, the following command must be introduced at the command prompt,

   $ gfortran penEasy.F −o penEasy.x −O

   With the Intel Fortran compiler the equivalent instruction is

   $ ifort penEasy.F −o penEasy.x −O2 −fpp

   After this command is completed, a file named penEasy.x, the executable code, is created. The user must move this executable file to the directory ∼/penEasy/run/. The options "−O" (capital letter O) or "−O2" serve to produce an optimized executable code. Notice that, although the other Fortran source files must not be invoked in the compilation command, they must be present in the same directory where penEasy is located for the compilation to be successful.

This completes the installation. The compilation command differs if the user is planning to make use of phase-space files (see section 8.3) in the IAEA binary format, as explained in section 14. Please refer to that section for further details.

Windows systems

For Windows systems (64 bits) an executable file named penEasy.exe is already included in the distribution and, therefore, the steps listed above are unnecessary, except obviously for the first one to unzip the package. This executable has been generated with the capability to read and write phase-space files in the IAEA format. Further information on this feature as well as details on the compilation command to create such an executable code are given in section 14.

Windows users willing to generate their own executable should follow the same steps as those for the Unix case, except for trivial changes—*e.g.*, replacing slash symbols / by backslash symbols \ in directory trees.

Additional software

In order to be able to visualize simulation results in graphical format, the penEasy package includes a set of Gnuplot scripts. To run these scripts and get a graph on the screen users must install Gnuplot on their machines, a data plotting software freely available, both for Unix and Windows systems, from http://www.gnuplot.info. For users preferring to rely on their own plotting software, details on the output generated by the different tallies are given in section 11.

PenEasy is mostly written in Fortran 95, although it has recourse to some features from the Fortran 2003 standard. Therefore, a compiler compatible with, at least, the 2003 standard of the Fortran language is needed. The latest versions of the two compilers mentioned above, namely, gfortran and Intel Fortran, satisfy this condition.

Caveat

All the Fortran files in the distribution have extension *.F. The capitalization of the F is not superfluous. It serves to indicate that, in order to conditionally compile some portions of the code, the so-called preprocessor directives are used. These directives require that, prior to the compiler proper, a precompiler is invoked; the big F tells the compiler to do just so.

Unfortunately, not all compilers understand the big F convention. For those that do not (we have found that Intel Fortran may not), it is necessary to force the entrance of the preprocessor by adding the option "−fpp" ("/fpp" in Windows systems) to the compilation command. Alternatively, the file extension *.F can be changed to *.fpp. In any case, it is always a good idea to consult the compiler's manual for detailed instructions.

4 How to operate it

NOTE: For concreteness, in this section it is assumed that the installation has been performed on a Unix system. Trivial changes (*e.g.*, changing the slash symbol / by a backslash symbol \ in directory trees) are required for Windows systems.

To execute a simulation run follow these steps,

1. Create a new directory in your working area where all the files for your simulation will be stored. Let us suppose that this directory is named ∼/mysim/. Always keep the original files unaltered.

2. Copy the following files to ∼/mysim/.
   ∼/penEasy/fortranCode/penEasy.x
   ∼/penEasy/run/command.in
   ∼/penEasy/run/penEasy.in
   ∼/penEasy/gnuplotScripts/*

   The geometry and material files required by PENELOPE (refer to the PENELOPE manual, Salvat 2019, for details) need also be located in ∼/mysim/.

3. Edit the input file penEasy.in, where the input data for penEasy are introduced. It is structured in sections, each one starting with a string of the type "[SECTION ...]". Each source and tally has its own configuration section, which are described in detail in section 8 and in section 11, respectively, of the present manual.

4. To run, open a terminal window (or "command prompt", as it is called in Windows) and execute

   $ penEasy.x < penEasy.in > penEasy.out

   The code reads from the standard input (the keyboard) and writes to the standard output (the screen), so it must be run redirecting these devices to the appropiate external files with the symbols < and >, as it is shown in the previous command line.

   The simulation can be controlled by sending commands to the computer code while it is running. This is accomplished by editing the file command.in. The available commands are briefly explained inside the file itself. For instance, the simulation can be terminated by resetting the requested number of simulation histories to zero. The file command.in is scanned periodically while the code is running.

5. Simulation results are written in separate files, one (for some tallies, several) for each tally. These files are named after the corresponding tally and have extension *.dat. Gnuplot scripts included in the distribution use these DAT files to display simulation results graphically. A summary of the execution is also printed to the output file (penEasy.out in the example above).

DAT files are regularly written at each update interval so that users can track the progress of the simulation.

All statistical uncertainties reported by penEasy are at two standard deviations (2 sigma).

6. In the event of an execution abort, an execution error or a program crash the execution is halted before the simulation has been completed. This can be the result of several factors, including: the user has pressed control-C to abort; there are data errors in a user-prepared input file, *e.g.*, an alphabetic symbol is introduced where the code expected a number; some data files are missing, *e.g.*, a material data file has been omitted (note that in this case the code will have created an empty file with the same name as the missing one); the code has a bug, a circumstance that happens more frequently than programmers are willing to admit.

   In all these cases the OUT file, the DAT files, or both will be incomplete. To check for possible errors and warnings look for, in the quoted order: (i) error messages issued by the operative system on your computer screen; (ii) error or warning messages written in the OUT file, which usually contains a succinct description of any problem found. On some rare occasions PENELOPE may also generate error files named FORTn, with 'n' representing an integer number.

4.1 Transport in external electromagnetic fields

The use of PENELOPE with external electromagnetic fields (penfield.f in the PENELOPE package) is not supported. Users are referred to the PENELOPE manual (Salvat 2019) for a description of the relatively simple alterations that should be introduced to adapt the code.

5 Example

> NOTE: For concreteness, in this section it is assumed that the installation has been performed on a Unix system. Trivial changes (*e.g.*, changing the slash symbol / by a backslash symbol \ in directory trees) are required for Windows systems.

Consider a 10 MeV electron beam impinging on a semi-infinite ($z > 0$) water phantom. The beam is modeled as emerging from a point source located at the Cartesian coordinates $(0, 0, -100)$, *i.e.*, at 100 cm from the water surface. Electrons are emitted with an initial direction limited to a cone with its axis along the vector $(0, 0, 1)$ and with an angular semiaperture $\alpha = 2.86°$. Since $\tan(\alpha)$ is approximately 0.05, the field on the water surface is a circle with radius equal to $100 \tan(\alpha) = 5$ cm.

We wish to calculate the energy deposited per unit depth interval, that is, a depth-dose curve. More precisely, penEasy will report $\Delta E/(\rho \Delta z)$ in eV·cm^2/g, where ΔE is the energy deposited in a bin, ρ is the mass density of water and Δz is the bin width. The dose distribution in cylindrical coordinates as a function of z and of the radial distance from the beam axis will also be computed. The depth interval from $z = 0$ up to $z = 7$ cm will be partitioned into 40 bins. For the cylindrical distribution, the radial interval $[0, 8]$ cm will be divided into 80 bins. Results are always reported per history, in this case per unit emitted electron.

Electrons, photons and positrons will be followed down to 100, 10 and 100 keV, respectively. The cutoffs for the production of secondary electrons and bremsstrahlung photons will be set to 100 and 10 keV, respectively. The simulation will be stopped after 100,000 histories, or when the average uncertainty in the depth-dose is 1% (2σ), whatever comes first. No variance-reduction technique will be applied.

To set up this simulation follow these steps,

1. Create the directory ∼/mysim/ in your working area.
2. Copy these files to ∼/mysim/
   ∼/penEasy/run/command.in
   ∼/penEasy/run/penEasy.x (See installation procedure above; on a Windows system you may use the ready-made penEasy.exe provided with the distribution.)
   ∼/penEasy/run/penEasy.in (sim input file prepared for this example)
   ∼/penEasy/run/phantom.geo (semiinfinite phantom, PENGEOM syntax)
   ∼/penEasy/run/water.mat (PENELOPE material file)
   ∼/gnuplotScripts/tallyCylindricalDoseDistrib-rz.gpl
   ∼/gnuplotScripts/tallyParticleTrackStructure.gpl
   ∼/gnuplotScripts/tallySpatialDoseDistrib-1D.gpl

   As in any other PENELOPE simulation, a material data file, water in our case, generated with the program MATERIAL (included in the PENELOPE package) is required. For convenience, this file is provided with penEasy for this example.

3. You may want to edit penEasy.in and check that the information introduced in this file reflects the definition of our depth-dose problem. Except for the ParticleTrackStructure, tallies other than those producing the requested distributions have been turned OFF to save CPU time. A detailed description of the simulation parameters EABS,C1,C2,WCC,WCR and DSMAX can be found in the PENELOPE documentation.

4. Run the program,
   $ penEasy.x < penEasy.in > penEasy.out

   On Unix systems, an ampersand "&" can be appended to the previous command to put the process to run in the background.

   When the simulation ends—it takes about a minute on a modern computer—the sought depth-dose data can be found in the file tallySpatialDoseDistrib-1D.dat. This file is updated every 100 s ("UPDATE INTERVAL" field in penEasy.in) whilst the program is running. Recall that the quoted uncertainties are at the 2 sigma level.

5. If Gnuplot is installed you can visualize the depth dose curve by executing the corresponding script from the ∼/mysim/ directory,
   $ gnuplot tallySpatialDoseDistrib-1D.gpl
   (On Windows systems enter "wgnuplot".)

   To visualize the cylindrical dose distribution use
   $ gnuplot tallyCylindricalDoseDistrib-rz.gpl

   You can also visualize a few particles tracks executing
   $ gnuplot tallyParticleTrackStructure.gpl

6. Commands can be sent to penEasy at run time by editing the file command.in, which is scanned periodically while the code is running. Please see its contents for a description of available commands and their numeric codes.

6 Structure of the input file

The input file, usually named with the extension *.in, is a plain text file divided into a number of sections. Each section can be classified into one of the following types,

- General (number of histories, allotted run time, initial random seeds, etc.)
- Source models
- Geometry
- Transport parameters
- Tallies
- Variance-reduction techniques (VRT).

Sections are concatenated and their order cannot be varied. Data fields in each section cannot be omitted or reordered either. Each data section has a version number of the form yyyy-mm-dd that is written in the corresponding section title. For example, `[SECTION TALLY SPATIAL DOSE DISTRIB v.2009-06-15]`.
Should an incorrect version be introduced, an error message would be issued and the execution halted. This avoids possible data field misinterpretations when the contents of a section is modified in newer versions.

Character strings (*e.g.*, file names) are introduced in free-format style, that is, leading and trailing blanks are allowed. Their maximum extension (except when noted) is 80 characters and they must not contain blanks. Thus, for instance, "stainless steel" could be introduced as "stainlessSteel" or "stainless_steel". Lines starting with a "#" (in column 1) and blank lines are taken as comments. Comment lines are allowed only in-between sections, not inside them.

Most data entry errors can be easily identified by looking for error messages or inconsistencies within the last lines of the penEasy output file. It is recommended that users check the output to make sure that the information written after processing each section coincides with what is expected from the input.

Hereafter it is assumed that the reader is editing the section being discussed, which is not reproduced in this manual for brevity. Text in `UPPERCASE` Courier typeface refer to the field labels appearing in a section. These texts are *not* keywords, that is, the code does not read them and, consequently, it does not identify their meaning. They are included in the file only as an aid for users.

Except when noticed, lengths are expressed in cm, angles in degrees, energies in eV and times in seconds. Rotations are specified by giving three Euler angles. The first angle, ω, defines a rotation around the z axis; the second, θ, a rotation around the original y axis; and the third, φ, a rotation around, again, the original z axis. This definition follows the standard PENELOPE convention for defining quadric geometries. For details the reader is addressed to the PENELOPE manual (Salvat 2019).

7 Input file: general

This section of the manual discusses the data fields of the `SECTION CONFIG`.

The simulation will be halted when any of the following conditions is fulfilled,

- The requested `NUMBER OF HISTORIES` has been reached.
- The `ALLOTED TIME` has been exhausted. The time is interpreted as real (*i.e.*, clock) time if it is a positive real number and as CPU (user) time otherwise.
- The `RELATIVE UNCERTAINTY REQUESTED` of all the active tallies have been reached—refer to section 11 for a discussion on tallies.

PENELOPE relies on L'Ecuyer's RANECU pseudo-random number generator (L'Ecuyer 1988; Salvat 2019). This generator uses two integer seeds. The field `INITIAL RANDOM SEEDS` provides their user-defined initial values. Alternatively, setting both random seeds equal to zero tells the code that the seeds are to be read from an external file, whose name (say, rngseeds.in) is read from the next line of input, `SEEDS FILE`. The pair of seed values must be on the first line of rngseeds.in separated by one or more blanks. This feature is useful when using parallel computing (see section 13).

Restarting a simulation

Sometimes a simulation terminates before the quantity of interest has been obtained with the desired statistical accuracy. This circumstance can be the result of a power cut, of the underestimation of the required uncetainty, etc. On these occasions users may restart a penEasy simulation and resume the calculation at exactly the point at which it stopped. To this end, a snapshot of the code state is taken at the end of the simulation and at uniform time intervals, as dictated by the value introduced in the `INTERVAL BETWEEN DUMPS` field. The snapshot is stored in binary format in the `OUTPUT DUMP FILE`; if this field contains a dash "–", no snapshots are taken.

In a subsequent simulation, setting both initial random seeds equal to -1 indicates that the simulation is to be restarted, that is, seamlessly continued from the point where a previous run ended. The dump filename of the previous run must be provided in the field `RESTART FILE`. If the random seeds are not $-1, -1$ this field is ignored.

An important comment regarding the restart feature is that, for it to work properly, the restart input file must not differ at all from the original input file that was used during the simulation that created the dump file, with the possible exception of: (i) the requested number of histories; (ii) the allotted simulation time; or (iii) the requested relative uncertainties of the active tallies. It is the user's responsibility to make sure that this condition is fulfilled—the code does not check if the file has been altered in ways other than the three cases listed above. Any alteration of any of the other fields could result in a program crash or, even worse, in the miscalculation of some tallies.

Starting a new simulation or restarting a previous one will overwrite all output data files sitting in the run directory. To preserve results from previous runs move these output files to a different location.

A simulation can be restarted as many times as desired, each time taking up from the point where the last execution ended.

8 Input file: source models

PenEasy includes two source models named Box Isotropic Gauss Spectrum (BIGS, for short) and Phase-Space File (PSF). In this section these models are described in detail.

8.1 General considerations

A general aspect of all models is that their respetive sections start with a `STATUS` switch that can be ON or OFF. An ON source is active, whereas an OFF source is inactive.

The Fortran code for BIGS has been structured with the intention of helping advanced users that want to extend its capabilities. One of such extensions for which the code is particularly well suited is source duplication, that is, the creation of one or more copies of the BIGS source. This feature can be exploited to generate several initial particle states per history, each with its own dynamical properties (particle type, location, energy spectrum, etc.). The procedure to apply source duplication is described below.

Although BIGS is in principle compatible with other source models (for instance, with its duplications) the same is not true for a PSF source. The code will issue an error and stop if users activate both the BIGS and the PSF models in the same run.

8.2 Source Box-Isotropic-Gauss-Spectrum (BIGS)

BIGS has a considerable flexibility, allowing the definition of a wide variety of source configurations.

Particle type

In the field `PARTICLE TYPE` the kind of initial particle is defined according to PENELOPE coding, that is, 1 designates electrons, 2 photons and 3 positrons. If, instead, a filename is introduced, the code assumes that a radioisotope will be used. For this latter option refer to the subsection on radioactive sources below.

Photon polarization

It is highly recommended that the relevant sections of the PENELOPE manual (Salvat 2019, scattering of polarized photons and Appendix C on photon polarization) are read before using this feature.

The use of the scattering models that include photon polarization is governed by the field named `ACTIVATE PHOTON POLARIZATION PHYSICS`. When set to 1 (activated) the three `STOKES PARAMETERS` determine the state of polarization of the photons emitted by the source. Otherwise (inactive), the state of polarization is ignored and the Stokes parameters are irrelevant.

Notice that if the polarization physics is active an initially unpolarized photon beam, defined by the Stokes parameters $(0, 0, 0)$, can become partially polarized after a scattering event. Thus, even initially unpolarized photon beams may produce different results depending on whether polarization physics is active or inactive.

It must also be borne in mind that the base vectors employed to describe the initial polarization state depend on the photon direction of flight and, therefore, if the source emits photons in different directions the corresponding vector base is rotated as described in the PENELOPE manual.

Spatial position

For the sake of brevity in this section we will refer to the reference frame implicitly used to define the geometry file as the laboratory (or lab) frame.

Particle positions are generated inside a cuboid (rectangular parallelepiped) that will be referred to as the source box. Its center is positioned as specified in the field `COORDINATES OF BOX CENTER`. Its size is determined by three `BOX SIDES` along the x, y and z directions. In addition, two Gaussian distributions, one along the x and the other along the y axis of the box reference frame, can be defined by giving their respective full width at half maximum values, `FWHM`s. The final position is the result of sampling a point inside the box and applying a Gaussian displacement along the x and y directions (in the box reference frame, see below). Hence, when the FWHMs are not zero, the final position could actually be located outside of the box.

Once the source box size and position are defined, the lab frame can be rotated and translated arbitrarily to reorientate and reposition the source box. To this end, three `EULER ANGLES` and three `TRANSLATION OF BOX CENTER POSITION` distances are given. The rotated and translated reference system is referred to as the box frame. It is in the (x, y) plane of this box frame, not in the lab frame, that the Gaussian displacement mentioned above is applied. This combination of operations covers a wide range of possible configurations of the source.

The final position of the source box, excluding the Gaussian spread, can be thought of as the result of applying four transformations in the lab frame to a unit side cube centered at the origin of coordinates. These transformations are the following, in the quoted order: (i) scaling of the three sides to arbitrary lengths; (ii) translation, as specified by the position of the box center; (iii) rotation; and (iv) a final translation.

These four transformations may appear redundant, since (ii)+(iii)+(iv) can always be reduced to an equivalent combination of a rotation and a single translation. Although this is true, it is not always convenient from the users' perspective. Indeed, consider the following three source configurations in the lab frame,

- Source 1: A linear source of length L with its middle point at located at $(1, 0, 0)$ and oriented along the direction $(1, 1, 1)$.

  This source can be defined by considering a source box with sides $(0, 0, L)$, centered

at $(0,0,0)$ (*i.e.*, no translation in step (ii)), rotated $(\omega = 0, \theta = 45, \varphi = 45)$ degrees (step (iii)) and translated to $(1,0,0)$ (step (iv)). The rotation brings the z axis to the $(1,1,1)$ direction.

- Source 2: A planar square source of side L, with its center at $(1,0,0)$ and parallel to the (y,z) plane. This source can be defined with a source box of sides $(0,L,L)$ and centered at $(1,0,0)$. This configuration could represent a model of the photon beam field shape from an x-ray tube.

- Source 3: Same as source 2, but with an additional rotation $(\omega,0,0)$, that is, rotating an angle ω around the z axis of the lab frame. The rotation that converts source 2 into source 3 illustrates the change in irradiation conditions found when taking a computerized tomography (CT) scan, in which the x-ray tube rotates around an irradiated object assumed here to be located at the origin.

Note that for source 1 the "natural" sequence of transformations is rotation→translation using transformations (iii)+(iv). For source 3 the natural sequence is instead center position→rotation, that is, (ii)+(iii). In this latter case an equivalent transformation rotation→translation could be found, but it would require additional work from the user's side to compute the required equivalent translation (iv). So, although the combination of the three transformations (ii)+(iii)+(iv) is reducible to only (iii)+(iv) in all cases, it is more convenient in practice to keep the redundant scheme.

BIGS allows users to specify a `SOURCE MATERIAL` from the quadric geometry to further restrict the initial particle positions. This is done as follows. Initial particle positions are sampled inside the source box. If `SOURCE MATERIAL` is a positive, non-zero, integer the particle is accepted only if the index of the material where it lies inside the box coincides with that number. Otherwise it is rejected and re-sampled. This procedure is useful, *e.g.*, to define radioactive sources with complex shapes, which are defined as bodies of the geometry file. A source box large enough to fully contain the radioactive source must be defined.

If `SOURCE MATERIAL` is set to 0 the particle position is sampled inside the source box and accepted regardless of the material in which it is located. If its position lies initially in vacuum it is moved forward until either it enters the simulated object or escapes to infinity. In this latter case the returned material index is zero and the main program provides safeguards to detect this. Note that for penEasy, particles emitted but not aiming at the object are effectively counted as simulated histories, a fact that must be taken into account when interpreting simulation results, which are always normalized per unit history.

Finally, if `SOURCE MATERIAL` is a negative integer m the particle direction is sampled and accepted only if it points at the material with index equal to $|m|$ or if its initial position already falls inside that material. This case is conceived to describe beams fields of arbitrary shapes. As an example consider a gamma point source immersed in air and producing a rectangular beam field on a certain distant plane. A thin rectangular body of the desired field dimensions should be included in the geometry file and in contact with

the distant plane. Let us suppose that this body is defined as filled with material 3. By setting the `SOURCE MATERIAL` to -3 only particles aimed at the rectangular field will be accepted as valid initial photon states. Care must be exercised when setting the direction and aperture (see below) to make sure that the defined cone completely "illuminates" the body intended to define the field. Notice that in order to decide whether or not the particle points at `SOURCE MATERIAL`, the presence of other intermediate materials, such as the air in the former example, is ignored.

The procedures described above apply also for voxelized geometries. If a material is selected by setting `SOURCE MATERIAL` to a positive value the source has a constant emission intensity per unit volume. For quadric geometries, and since bodies of the same material have by definition the same mass density, the source has a constant emission intensity per unit mass. When voxelized geometries are used, however, different voxels may contain the same material but with varying mass densities (see section 9). In this case, therefore, voxels inside the source box and made of the selected material emit a uniform number of particles per unit volume, but not necessarily per unit mass.

Direction of emission

The direction of emission is isotropic within a spherical window (or, more rigorously, a spherical trapezoid), which is defined as the part of a sphere limited by given polar and azimuthal angle intervals, $[\theta_0, \theta_1]$ and $[\varphi_0, \varphi_1]$, respectively.

The polar interval is introduced by the user by giving its two limiting values, that is, θ_0 and θ_1, in the field `DIRECTION POLAR ANGLE INTERVAL`. θ_1 must not be smaller than θ_0 and both values must be contained in $[0, 180]^\circ$. The azimuthal interval is introduced by giving φ_0 and the *interval width* $\Delta\varphi$, not φ_1, in the field `DIRECTION AZIMUTHAL ANGLE INTERVAL`. φ_0 can take any value in $[0, 360)^\circ$, whereas $\Delta\varphi$ can take values in $[0, 360]^\circ$. The azimuthal interval is obtained by rotating $\Delta\varphi$ counterclockwise around the z axis, as determined by the right hand rule (the thumb pointing along the positive z axis).

The reference system S' in which these spherical coordinates are given can be different from the lab frame S used to define the geometry. To allow for this, an (unnormalized) `DIRECTION VECTOR` can be introduced by the user. This vector represents the direction of the z' axis of the S' system. The corresponding x' and y' axis are implicitly defined by the polar $\theta_{z'}$ and azimuthal $\varphi_{z'}$ angles of the z' axis, as seen from S, as follows: first, a rotation of angle $\theta_{z'}$ around the y axis of S is applied; second, a rotation of angle $\varphi_{z'}$ around the z axis of S is applied. Note that, as intended, this brings the z axis of S into the z' of S'. This convention is the same as that used by PENELOPE when considering a polarized photon beam (see the PENELOPE manual, Salvat 2019, Appendix C).

As an illustrative example, one that is very frequently found in practice, suppose that we want to define a source emitting isotropically but limited to a cone of semi-aperture θ and with the cone axis given by a certain direction in space $\mathbf{u} = (u_x, u_y, u_z)$. For concreteness, let us take $\theta = 30^\circ$and $\mathbf{u} = (1, 1, 1)$. The input data to produce this source is

```
1.0  1.0  1.0    DIRECTION VECTOR
0.0  30.0        DIRECTION POLAR ANGLE INTERVAL [THETA0,THETA1]
0.0 360.0        DIRECTION AZIMUTHAL ANGLE PHI0 AND DeltaPHI
```

Note that the position of the (x', y') axis of S' is here irrelevant because the azimuthal interval covers a full circle ($\Delta\varphi = 360°$).

A "cone" with semi-aperture $\theta = 180°$defines the full sphere (4π solid angle). Another common example is a pencil beam, which is defined by setting both polar angles $\theta_0 = \theta_1$ equal to zero; in this case the azimuthal angle interval is irrelevant.

Energy

An energy spectrum can be introduced, after the `DUMMY HEADER`, as a piecewise function. Each entry in the spectrum contains two numbers, namely, the starting energy of a channel and its unnormalized probability. Probabilities must be non-negative and they do not need to be normalized to unity. The list ends whenever a negative probability (-1) is found. To sample an energy value, a channel is firstly selected according to the relative probabilities. The particle energy is then sampled uniformly inside that channel.

For example, the two most prominent gamma lines emitted by ^{60}Co are located at 1.17 and 1.33 MeV with relative yields 99.97% and 99.99%, respectively. This spectrum can be defined as follows,

```
Energy(eV) Probability (Probabilities do not need to be normalized to 1)
1.17e6     99.97       1st channel: [1.17,1.17]MeV
1.17e6      0.0        2nd channel: [1.17,1.33]MeV, no emissions
1.33e6     99.99       3rd channel: [1.33,1.33]MeV
1.33e6      -1         '-1' signals the end of the spectrum
```

An arbitrary continuous spectrum with a gamma line at 80 keV would look like this,

```
Energy(eV) Probability (not normalized)
30e3       2.0         1st channel: [30,60]keV
60e3       3.0         2nd channel: [60,70]keV
70e3       1.0         3rd channel: [70,80]keV
80e3       5.0         monoenergetic gamma line at 80 keV
80e3       1.0         Nth channel: [80,90]keV
90e3       2.0         Last channel:[90,130]keV
130e3      -1          '-1' signals the end of the spectrum
```

Note that channel widths need not be constant. In particular, the combination of continuous and discrete spectra is achieved by mixing channels of finite and null widths. In Appendix B further details on how to define the spectrum for some practical situations is given.

A Gaussian widening of the spectrum can be defined by introducing a value of FWHM different from zero. The actual particle energy is the result of applying a Gaussian displacement to the energy resulting from the sampling process described before. For a monoenergetic spectrum (*i.e.*, a single line) the end result is a pure Gaussian distribution. In general, the resulting energy distribution is the convolution between the piecewise spectrum introduced before and the Gaussian. Recall that, for a Gaussian distribution, FWHM $= \sigma\sqrt{8\ln(2)} \approx 2.35482\,\sigma$, where σ is the standard deviation.

For instance, the table

```
Energy(eV) Probability
1.0e6      1.0
1.0e6      -1
0.2e6             FWHM (eV)
```

defines a Gaussian with mean 1.0 MeV and a FWHM of 0.2 MeV ($\sigma = 84.9$ keV).

The Gaussian is truncated at $\pm 5\sigma$, with the additional restriction that no negative energies are produced; whenever the sampling returns a negative energy, its value is set to zero. These null-energy primary particles are counted as simulated histories, a fact that must be taken into account when interpreting simulation results, which are always normalized per unit history.

As an alternative to specifying the spectrum in the input file directly, an ENERGY SPECTRUM FILE NAME can be entered. If a dash "−" is input the spectrum is read from the input file as described above. If, instead, a valid file name is given the spectrum will be read from that file. In this case the header line and the lines containing the spectrum must be removed from the input file. The energy subsection would in this case look as follows,

```
SUBSECTION FOR PARTICLE ENERGY:
 spectrum.dat    ENERGY SPECTRUM FILE NAME
 0.0             FWHM(eV) OF GAUSSIAN
```

The expected format of the data in the external spectrum file is identical to the one described above for the input file with an additional feature, namely, lines starting with "#" or blank lines are allowed as comment lines. An example of an external spectrum file is given in ∼/run/sampleBIGSspectrum.spc.

Radioactive sources

As an alternative to introducing the PARTICLE TYPE and the energy spectrum of emission in the SUBSECTION FOR PARTICLE ENERGY, a radioactive source can be defined. This option is activated by entering a filename (*e.g.*, Co-60.nuc) in the field PARTICLE TYPE instead of the numeric code (1, 2 or 3) discussed above.

The introduced filename identifies the radionuclide to be used as the source, *e.g.* Co-60.nuc indicates ^{60}Co. This feature relies on the PENNUC package (García-Toraño et al. 2017; García-Toraño et al. 2019), a routine library that allows the random sampling of the complete decay cascade of a radionuclide, including x-rays and Auger electrons produced after the atomic de-excitation of the daughter nucleus. Radionuclide files (Co-60.nuc in the previous example) can be obtained from the Laboratoire National Henri Becquerel (LNHB) at
http://www.nucleide.org/DDEP_WG/DDEPdata.htm
(download using the format option named "pennuc"). This LNHB database is already included in the PENNUC distribution and, for your convenience, it has also been copied into the penEasy directory ∼/run/nuclides/ so that no further action from the user's side is required. The nuclides directory, with the exact same name, must be copied under your simulation working directory (*e.g.*, ∼/mysim/nuclides/) if the radionuclide option is selected, that is, if a radionuclide filename is introduced in the `PARTICLE TYPE` field. Although it is usually simpler to copy the full content of the nuclides directory, the only files actually required to be present for any given simulation are: (i) the file defining the radionuclide used, *e.g.*, Co-60.nuc; (ii) pdatconf.p14; and (iii) pdrelax.p11. The other files can be deleted if desired. The files named pdatconf.p14 and pdrelax.p11, which are identical copies of the corresponding files from the PENELOPE material database, contain information related to the atomic structure and atomic relaxation data, respectively. Refer to the PENELOPE manual for additional details (Salvat 2019). The radionuclides available in the database, 220 approximately, are listed in ∼/run/nuclides/isotope.list.

When this option is activated the whole `SUBSECTION FOR PARTICLE ENERGY` must be removed from the BIGS section of the input file. A convenient way to do this is by moving the said subsection after the `END OF BIGS SECTION` mark and comment it out by inserting '#' symbols in the first column.

Each history entails one single call to penEasy's source routine and, for a radionuclide, this implies that each history simulates the cascade produced after one disintegration of the selected radioactive nucleus. The daughter nucleus may reach, in the course of its de-excitation, a metastable level, which is treated as an effective halt of the decay process. Radiations emitted between successive halts constitute one cascade and, therefore, are considered to define a single history. PENNUC parameter DRTIME, in ∼/FortranCode/sourcePennuc.F of the penEasy distribution, defines the resolution time in seconds, that is, the time limit used to determine whether a given state is metastable.

A related and important caveat regarding the direction of the emitted particles and the efficiency of the radioactive source is here in order. Since a radionuclide may emit several particles in arbitrary random directions in a single cascade, it is not possible to guarantee, nor convenient to force, that a given disintegration generates particles within the user-defined spherical trapezoidal window restricting the direction of emission in the BIGS source model (see subsection on direction of emission above). This means that if the window aperture is too narrow, a large fraction of the simulated histories may produce no particles in the angular range of emission. PenEasy detects this situation when it happens and aborts the execution with an error message. The remedy is to enlarge the angular intervals that define the spherical window so as to decrease the

probability that a cascade generates no particles in the selected angular range.

8.3 Source Phase-Space File (SPSF)

In this source model initial particle states are read from an external phase-space file (PSF) specified in the field `PSF FILENAME`. Data in each record are read with the following Fortran instruction, which reveals the variables present in the file and their ordering,

```
read(file,*) kpar,e,x,y,z,u,v,w,wght,dn,ilb(1),ilb(2),ilb(3),ilb(4),ilb(5)
```

The meaning of these variables is described in the PENELOPE manual (Salvat 2019). They are contained in module `MODULE TRACK_mod` of the penelope.f file, except `dn`. The variable `dn` in a given record is the increment in history number with respect to the previous record. Thus, for instance, `dn`= 0 indicates that the current particle record belongs to the same history as the previous one and `dn`= 3 indicates that it was necessary to simulate three additional histories until a particle reached again the PSF scoring region. This information is essential to correctly estimate the statistical uncertanty associated to the tallied quantities of interest.

PSFs with formats previous to the one used in the 2008-05-15 version (referred to as pre-2008) are abandoned in favor of the new version, which includes all the `ilb` variables. The PSF source is capable of identifying whether the 2008 or pre-2008 formats are used, and act accordingly. In either case the first line of the PSF must contain the appropriate header, as produced by the corresponding version of the PSF tally (see section 11).

The standard penEasy format, `PSF FORMAT`=0, is in plain text mode. Each variable must be separated from the next by one or more spaces. Given that for some applications the time needed to access the information in the PSF record may be non-negligible, this format is obviously not optimized for speed. It was chosen because it is human-readable and directly amenable to graphical representation, *e.g.*, with Gnuplot. However, for production runs, a binary format ("unformatted", in the Fortran jargon) is more convenient.

To use PSFs in binary format, `PSF FORMAT`=1, users may have recourse to routines included in penEasy. These routines are an interface to a library of C++ routines created by the International Atomic Energy Agency (IAEA) in the framework of a project to set up a public database of PSFs[2]. For the users' convenience the Windows executable file included in the distribution has this capability already built-in. For other systems users are required to compile the C++ library and link it to the penEasy Fortran code. Detailed instructions on how to do this are given in section 14.

To increase the simulation efficiency particles in the PSF can be split a number times, as specified in the field `SPLITTING FACTOR`. Their statistical weight is reduced accordingly

[2] A description of this project and public access to the database are provided at http://www-nds.iaea.org/phsp.

to keep simulation results unbiased. Users must be aware that the clones of the original particle are stored in the particle stack and, therefore, a very large splitting factor may lead to a stack overflow. A possible way out of this is to increase the stack by editing the penelope.f file in the PENELOPE distribution and changing the parameter `NMS`, which defines the stack size. It is recommended, however, to try instead to reduce the splitting factor to a lower value so as to avoid the problem. A stack overflow can be identified by the message, in the output file, "WARNING: (STORES) not enough storage for secondaries".

After being read from the PSF, each particle position vector and direction of flight is rotated according to the `EULER ANGLES`, as defined in the PENELOPE manual and also in previous sections of the present manual. After the rotation, a `TRANSLATION` can also be applied to all particle positions.

PENELOPE routines need to know the maximum (kinetic, in the case of massive particles) energy that will be used in the simulation. Users may enter this value in the field `MAX PSF ENERGY`. This datum is unnecessary, and therefore ignored, if the option `VALIDATE BEFORE SIMULATION` is set to "yes", in which case penEasy automatically analyzes the maximum energy in the PSF prior to starting the transport.

If at least one positron is present in the PSF the maximum energy entered must exceed the maximum energy present in the PSF by $2m_{\rm e}c^2$ ($\sim$1022 keV) to account for the possibility of in-flight positron annihilation (mass $m_{\rm e}$). This process generates two photons, one of which may reach the reported energy. When automatic PSF validation is turned on this addition is performed automatically by penEasy. When the automatic validation is off and the user provides the maximum energy manually, the failure to provide a valid value may cause the program to crash due to the inappropriate setting of the maximum energy in PENELOPE's lookup tables. Note that the validation process may take a non-negligibe amount of computing time and, therefore, it is recommended that it be performed only once for a given PSF.

The Fortran routine for the PSF source model modifies the current history number N to adapt it to the actual number in the "parent" simulation that created the PSF. To do this, the `dn` variable discussed above is employed. For instance, if `dn`=5 then N will be increased by $(5-1)=4$ units. One unit is subtracted to account for the fact that N is always already increased by one unit by the main program every time the source routine is called.

Information regarding the state of polarization of photons, even if present in the PSF, is ignored. Consequently, the transport of polarized photons is not possible with the current version of the PSF source.

8.4 Duplication of a source

The BIGS source can be easily duplicated to create more than one primary particle per history. An example of a possible application is the simultaneous generation of the two

gammas (1.173 and 1.133 MeV approximately) of a Co-60 source[3]. This will allow for a sum peak (both photons fully absorbed) to appear in the pulse-height spectrum of a detector positioned next to the cobalt source, a peak that is not present if a single standard BIGS source emitting either one or the other photon is defined.

Although the duplication procedure is simple, care must be taken to avoid name clashes between different copies of the same routines and variables. To duplicate the BIGS source follow these steps,

1. Make a copy of the file sourceBoxIsotropicGaussSpectrum.F (call it, *e.g.*, sourceBoxIsotropicGauss-Spectrum2.F). Edit the copied file and change the names of all BIGS internal Fortran modules and routine names. This can be readily done by simply making a global text substitution to replace the string `BIGS` by `BIGS2`.

2. Edit penEasy.F and

   (a) Duplicate the line
   `#include "sourceBoxIsotropicGaussSpectrum.F"`
   to include the copied file.

   (b) Duplicate the line
   `call BIGSinisrc(BIGSactive,emax1,dmem)`
   and replace `BIGS` by `BIGS2` in the copy; duplicate also the next 2 lines setting the variables `emax` and `mem`.

   (c) Duplicate the line `call BIGSsource`
   and replace `BIGS` by `BIGS2` in the copy.

3. Edit the file penaux.F and locate the definition of the variable `BIGSactive`. Add the variable name `BIGS2active` next to it, preceded by a comma.

4. Compile and link the adapted code following the instructions provided in section 3.

5. Edit the input file and duplicate the entire BIGS source section. Replace
   `[END OF BIGS SECTION]`
   by
   `[END OF BIGS2 SECTION]`
   in the copy.

Now each BIGS source section in the input file can be used to define a different initial particle state. This duplication scheme can be repeated as many times as wished. Notice that different relative source intensities can be artificially created by defining, for one of the two sources, a photon with zero energy and with a probability of emission adjusted so as to generate the real photons at the appropriate rate.

As a precautionary measure keep the original penEasy files unaltered. The adaptations described above should be done on copies of the originals placed in a separate directory.

[3]However, see a better alternative for radionuclides in section 8.2.

Calls to routines from libraries external to the penEasy package cannot be duplicated as easily as described here because they do not follow the naming convention that allows the simple global substitution of, *e.g.*, `BIGS` by `BIGS2`. This affects the PSF source (see section 14) and the use of radionuclides in the BIGS source model (see section 8.2).

9 Input file: geometry

Three possible geometry models are possible with penEasy, namely,

- quadric geometries
- voxelized geometries
- a mixture of quadrics and voxels.

Although in this manual we shall restrict ourselves to these three models, it should be mentioned that other authors have extended these capabilities by developing modules that, when linked to penEasy, allow also the simulation of triangle meshes (Badal et al. 2009), such as those used in computer-aided design (CAD) systems.

Quadric geometries

Quadric geometries are defined by means of the PENGEOM library, included in the PENELOPE distribution. The elements used in this geometrical model are surfaces, and bodies (and modules), which are introduced through a plain text file typically named with extension *.geo. Users are referred to the PENELOPE manual (Salvat 2019) for details on the structure and syntax of these GEO files.

The name of the quadric geometry file is entered in the field `QUADRICS FILE NAME` of the input file. The field `VOXELS FILE NAME` must contain a dash "–" to indicate that no voxels are defined.

Voxelized geometries

A voxelized geometry is a geometry model in which objects are described in terms of a (usually large) collection of (usually small) volume elements, or voxels for short. Each voxel is a cuboid, a rectangular parallelepiped, with a homogeneous material composition and mass density. The voxels bounding box (VBB) is defined as an imaginary large cuboid containing all voxels.

In the model implemented in penEasy all voxels have the same dimensions and they are adjacent, that is, each voxel has its six faces in contact with those of its six neighbors, except of course for those voxels located in the VBB periphery. The VBB and its voxels are assumed to have each side parallel to one of Cartesian axis of the lab reference frame S. The VBB lies in the first octant of S, *i.e.*, in the region $\{x > 0, y > 0, z > 0\}$. The voxel with indices $(i, j, k) = (1, 1, 1)$ has its corner at the origin of S.

The voxels definition is supplied by users through a plain text file, typically named with extension *.vox. A sample file, which contains detailed instructions about its syntax and data formats, is provided with the penEasy distribution in ∼/penEasy/vox-Sample/voxels.vox. Note, however, that in most cases and due to its size the VOX file

will be created by a computer program processing the information provided by, *e.g.*, a computed tomography (CT) scan.

The name of the voxels geometry file is entered in the field `VOXELS FILE NAME` of the input file. The field `QUADRICS FILE NAME` must contain a dash "–" to indicate that no quadrics are defined.

Not all tallies are compatible with voxelized geometries. For instance, fluence spectra are not reliable when reporting data that refers to the voxelized region. Furthermore, to compute the dose distribution inside a voxelized region the tally specifically tailored for voxels is recommended. Please see section 11 for more details.

Mixing quadric and voxelized geometries

PenEasy allows the combination of quadric (GEO) and voxelized (VOX) geometries. To understand how this mix is interpreted by the code, the following considerations can be of help.

- Objects defined in the quadric geometry prevail over voxels. One way to visualize this using a 2D analogy is to think that the quadric geometry is a layer superimposed on top of the voxelized geometry layer, which lies underneath. So, the quadrics layer effectively blocks the "sight" of the voxels layer.
- However, in the input file a certain material in the quadric geometry is identified by the user as the "transparent" material. The voxels layer underneath the quadrics can only be seen (*i.e.*, exist) through the transparent material. In other words, wherever the transparent material and a voxel overlap, the voxel prevails.
- It should be noticed that at the boundary of a quadric body containing the transparent material a voxel can be partially overlapped by an adjacent non-transparent material. Thus, in that region, only the fraction of the voxel under the transparent material will effectively exist from the viewpoint of the simulation code. That is, at the edge of a transparent body non-cubic (irregular) voxels might be implicitly defined. See for example the voxels intersected by the circle in fig. 1.

When both quadrics and voxels are defined a GEO and a VOX filenames must be provided in the fields `QUADRICS FILE NAME` and `VOXELS FILE NAME`, respectively. Furthermore, the transparent material must be declared in the field `TRANSPARENT QUADRIC MAT`. This material cannot be zero, meaning that the transparent material cannot be vacuum. The declared transparent material is irrelevant if only quadrics or only voxels are defined.

As discussed above, the lab frame S is completely determined by the voxels bounding box. Thus, when voxels and quadrics are combined the quadric geometry must also be referred to S by having recourse, if necessary, to the appropriate rotations and translations of the quadric objects.

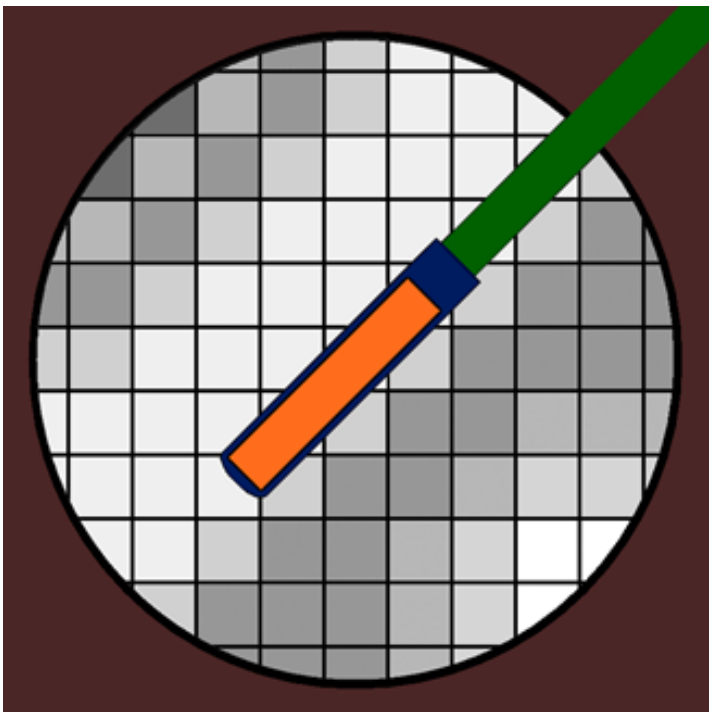


Figure 1: Combination of quadrics and voxels. In this example the quadric "transparent" material is the invisible material filling the circle. The brown, blue, orange and green materials are all part of the quadric geometry. Different gray levels represent different mass densities in the voxels. (Adapted from Badal 2008.)

Voxels mass: granularity

When quadrics and voxels are mixed some voxels might be partially overlapped by a quadric body, as it happens in the voxels partially covered by the circular quadric in fig. 1. In these cases the computation of their mass is not trivial. This computation is done during penEasy's initialization by integrating the mass density over the existing portion of the voxel volume. The accuracy with which this integration is performed can be controlled by the user via the integer value entered in the `GRANULARITY` field. More specifically, a number of threads equal to the granularity squared is cast through each voxel along the z axis to compute the voxels volume. Thus, the granularity determines how densely the voxel is scanned to compute the integral. As a rule of thumb, the granularity should be set to $3\sqrt{A_v/A_q}$, where A_v is the area of the voxel face perpendicular to the z axis and A_q is the typical cross section, as viewed from the z axis, of the smallest quadric element inside a voxel. A larger granularity performs a more accurate mass integration by detecting smaller quadric details inside voxels, but at the price of longer initialization times.

A reasonably low value for the granularity is 10; larger values should be used for spiky quadric geometries. The granularity cannot be smaller than 2. The granularity value is ignored if only quadrics or only voxels are defined.

10 Input file: transport parameters

This section defines the transport parameters that PENELOPE will use for each material present in the simulation. For a detailed description of the meaning and effect of these parameters please refer to the PENELOPE manual (Salvat 2019).

The section contains a table with one material per line. Each line starts with the material index (`MAT#`), which should be a sequencial integer starting from 1. Setting `MAT#` to -1 denotes the end of the list. 20 characters can be entered at most for the material file name. Blanks or special characters are not allowed in file names; thus, instead of "stainless steel.mat" an alternative such as "stainlessSteel.mat" must be used.

If for a certain material the transport parameters after the file name are left empty, then they are set automatically as follows,

- EABS for charged particles are set to 0.01 (*i.e.*, 1%) of the initial source energy (E), with the limiting values of 50 eV (min) and 1 MeV (max).
- EABS for photons is set to $0.001E$ with the limiting values of 50 eV and 1 MeV.
- C1 and C2 are both set to 0.1.
- WCC is set to min[EABS(e^-) , $0.01E$].
- WCR is set to min[Eabs(photons) , $0.001E$].
- DSMAX is set to infinity (10^{30}).

The table header (`MAT# FILE ...`) is a dummy line, that is, it is not interpreted by the code. However, the code expects a line (any line) of text to be there, so do not remove it to prevent the code from stopping with an error message.

This, admittedly somewhat arbitrary, recipe is an adaptation of the prescription given by Sempau and Andreo (2006) with the speedup parameter defined therein set to unity.

11 Input file: tallies

PenEasy includes a number of tallies to score quantities of interest (*e.g.*, absorbed dose distributions, fluence spectra, pulse-height spectra, pixelated images, etc.). Each tally is configured in a dedicated section of the input file that is identified with an acronym. For instance, PHS stands for Pulse-Height Spectrum.

The configuration of the available tallies is described below in alphabetical order.

11.1 General considerations

A general feature of all tally sections is that they start with a `STATUS` switch that can be ON or OFF. An ON tally is active, whereas an OFF tally is inactive and will produce no data reports nor require any significant amount of CPU time. Another common feature for most tally sections is that they end with a field named `RELATIVE UNCERTAINTY (%) REQUESTED`. The simulation will stop if *all* the requested relative uncertainties, expressed as a percentage of the mean value of the quantity of interest, have been attained. If the requested uncertainty is set to zero the condition is never fulfilled and, therefore, the simulation will not stop for this cause—but it will stop if other conditions are met, see section 7.

Another feature common to all tallies involving spatial distributions (*e.g.*, absorbed dose distributions) is that they define volume elements, which shall be called spatial bins, where the energy deposition is collected. Depending on the tally type these bins can be shaped as cuboids, cylindrical shells, spherical shells, etc. It is important to remark that spatial bins *do not exist in the geometrical description* of the material object given in the GEO file and, therefore, they do not represent a hard boundary for the transport of particles. Thus, a particle arriving at the border of a bin will not stop as it would do at the boundary separating two different materials—the only exception being when the voxel dose tally is used, see section 11.15, which does define hard boundaries. This scoring scheme is made possible by the peculiarities of PENELOPE's multiple scattering algorithm for charged particles.

Apart from spatial bins, tallies may also involve the use of energy bins to compute energy distributions, that is, spectra. Both for spatial and energy bins, and except when otherwised noted, the quantity of interest is reported per unit bin size. In the case of, *e.g.*, a 3D absorbed dose, the reported quantity is energy per unit mass, that is per unit bin volume times mass density. For an energy distribution the equivalent quantity is expressed per unit energy interval.

The graphical representation of the resulting (spatial or energy) distributions is usually done using step-like graphs, that is, by discretizing the underlying continuous distributions. The value in each bin, represented graphically as a horizontal segment between the bin end points, is an estimation of the mean value of the distribution in the bin interval. The associated statistical uncertainty is plotted as an uncertainty bar at the bin middle point. To facilitate the preparation of these step-like graphs two coordinate

values are reported for each bin in the tally output data file, namely, its low end and its middle point.

The Fortran code for most tallies has been structured with the intention of helping advanced users that want to extend its capabilities. One of such extensions for which the code is particularly well suited is tally duplication, that is, the creation of a copy of an existing tally. This feature can be exploited, for instance, to compute PHSs in several different material detectors simultaneously. The procedure to apply tally duplication is described below.

11.2 Gnuplot scripts

PenEasy outputs text data files in a format suitable for the plotting software Gnuplot[4]. Script files containing Gnuplot instructions are included in the penEasy distribution under the directory ∼/gnuplotScripts/. If Gnuplot is installed in your computer these scripts can be used to obtain a graphical representation of the output data.

For each tally there are one or more Gnuplot scripts. Which script corresponds to which tally can be easily identified by the file name. For instance, the tally Energy Deposition, which when executed creates an output data file named tallyEnergyDeposition.dat, has an associated Gnuplot script named tallyEnergyDeposition.gpl. Some DAT files may have more than one GPL file available. For example, the tally spatial dose distribution in 3D (see section 11.13) generates a data file named tallySpatialDoseDistrib-3D.dat. There are two GPL files with similar root names, namely, tallySpatialDoseDistrib-3Dmap.gpl and tallySpatialDoseDistrib-3Dsurf.gpl. The GPL file ending in "map" plots a 2D color map for each z value in the 3D data, whereas the file ending in "surf" plots a surface in 3D for each z value.

To execute a script the GPL file has to be copied to the same directory where the DAT file sits, usually the directory where the code is run. If the Gnuplot program is associated with the *.gpl extension ("open with" menu) double clicking on the GPL file will generate the graph automatically.

11.3 Duplication of a tally

Most tallies in penEasy can be easily duplicated to create more than one report of the same type. This, for example, permits the simulation of two simultaneous pulse-height spectra produced by two spectrometers located in different spatial positions.

Although the duplication procedure is simple, care must be taken to avoid name clashes between different copies of the same routines and variables. The steps to be followed are analogous to the ones described in section 8.4 to duplicate a BIGS source. Users are referred to that section for details.

[4]Freely available from http://www.gnuplot.info.

One exception is the PSF tally (see below), which cannot be duplicated so easily because it relies on a set of external routines (see section 14).

11.4 Tally: Cylindrical Dose Distribution

The absorbed dose per history is tallied in the radial (r) interval `[RMIN,RMAX]` using `NRBIN` bins, with the radial distance given by $r = \sqrt{x^2 + y^2}$. Corresponding intervals and bins are defined for the z coordinate. The resulting volume elements allow the scoring of a 2D (r, z) cylindrical dose distribution that is reported in eV/g per history.

To facilitate the preparation of step-like graphs two values for each coordinate are printed in each data row. For the z coordinate these values are the low end and the middle point of each bin. For the r coordinate they are the low end and an average radial distance $r_{\rm avg}$, the latter being a weighted average of r over the interval, with weight proportional to r. This results in

$$r_{\rm avg} = \frac{2}{3}\,\frac{r_{\rm hi}^3 - r_{\rm lo}^3}{r_{\rm hi}^2 - r_{\rm lo}^2},$$

where $r_{\rm lo}$ and $r_{\rm hi}$ are the low and high ends of the bin, respectively. The rationale for this definition is the following. Assuming that the dose $D(r)$ varies linearly in the interval $(r_{\rm lo}, r_{\rm hi})$, $r_{\rm avg}$ corresponds to the point at which $D(r_{\rm avg}) = D_{\rm avg}$, where $D_{\rm avg}$ is the average dose in the interval.

In order to obtain its mass density, it is assumed that each cylindrical bin contains a single material, *i.e.*, that it is homogeneous. The mass density of each bin is evaluated at $(x = r_{\rm avg}, y = 0, z_{\rm mid})$.

This tally should not be used to compute absorbed dose inside voxelized regions. This is because, as mentioned above, it may report erroneous results if bins are not homogeneous. Inhomogeneities may arise because some bins may overlap with more than one voxel or with voxels and quadric bodies, if both are used. The tally specific for voxelized geometries should be used instead, section 11.15.

The 2D (r, z) distribution can be integrated over z to obtain a 1D distribution that depends only on r, that is, the scoring region along the z axis is taken to be $(-\infty, +\infty)$. This is achieved by setting `NZBIN` to zero. In this case the distribution is reported in cm·eV/g per history. If `NZBIN`=0 the values `ZMIN,ZMAX` are irrelevant, except for the fact that the middle point $z_{\rm mid} = (z_{\rm min} + z_{\rm max})/2$ is used to evaluate the mass density of the entire region $z = (-\infty, +\infty)$.

If `PRINT COORDINATES` is set to 1, (r, z) values are printed in the report. Notice that, in order to interpret the various columns in the output file, the Gnuplot scripts for this tally assume that this option has been set to 1.

The `RELATIVE UNCERTAINTY REQUESTED`, specified as a percentage, is used as a criterion to stop the simulation.

11.5 Tally: Energy Deposition

The energy deposited (*i.e.*, absorbed) per history in each material is reported in eV per history. The `RELATIVE UNCERTAINTY REQUESTED`, specified as a percentage, is used as a criterion to stop the simulation. This uncertainty refers to the declared `DETECTION MATERIAL` only, although values for all materials are reported.

11.6 Tally: Fluence Track Length

The fluence spectrum integrated over the volume of the declared `DETECTION MATERIAL` is computed. To obtain it, the track length estimator is used, that is, the distance travelled by each particle in the detector is tallied as a function of its energy. The spectrum is obtained by splitting the interval `[EMIN,EMAX]` (in eV) in the specified `No. OF E BINS`. Optionally, a logarithmic energy scale can be used, which is indicated by adding the keyword `LOG` after the number of bins. To facilitate the preparation of step-like graphs two energies are printed in each row, namely, the low end and the middle point of each bin. The `RELATIVE UNCERTAINTY REQUESTED`, specified as a percentage, is used as a criterion to stop the simulation.

The result is reported in units of particles·cm/eV, or simply cm/eV, per history. This deserves some further clarification[5], as it may not be obvious where these units come from.

The fundamental quantity fluence, defined in terms of number of particles per unit cross-sectional area, has units

$$[\Phi] = \frac{1}{\mathrm{cm}^2} \ .$$

The energy-fluence Ψ, related to fluence by $\Psi = E\Phi$, where E is the particle energy, has units

$$[\Psi] = \frac{\mathrm{eV}}{\mathrm{cm}^2} \ .$$

Consequently, the units of the two quantities differential in energy[6] are

$$\left[\frac{d\Phi}{dE}\right] = \frac{1}{\mathrm{eV\ cm}^2}$$

and

$$\left[\frac{d\Psi}{dE}\right] = \frac{1}{\mathrm{cm}^2} \ ,$$

respectively. Note that the unit of energy-fluence differential in energy, $d\Psi/dE$, coincides with that of fluence, Φ, which might be prone to confusion.

The output of the present tally, which is a quantity differential in energy, corresponds to tracks integrated over the detector volume V, so that its units are those of the product

[5] We thank Prof. Pedro Andreo for providing the discussion that follows.

[6] The fluence differential in energy, $d\Phi/dE$, is sometimes represented as Φ_E, particularly in medical physics contexts.

$[dV][d\Phi/dE]$, *i.e.*, cm/eV. Since PENELOPE cannot determine the volume of a body limited by arbitrary quadric surfaces, users must divide the score output by the body volume in order to get the correct units for the average fluence differential in energy,

$$\left[\frac{d\Phi}{dE}\right] = \left[\frac{1}{V}\right] \left[V\,\frac{d\Phi}{dE}\right]_{\text{penEasy}} = \frac{1}{\text{eV cm}^2} \ .$$

11.7 Tally: Particle Current Spectrum

This tally reports, classified according to the particle type and per unit history, the energy spectrum of particles entering the specified `DETECTION MATERIAL`. The total number of each type of particle, per history, is also reported.

Particles reentering the detection material are counted as many times as they enter. If this is not wanted, the absorption energy of the material in question must be set to infinity so that incoming particles are absorbed at once. Source initial particles that are born inside the detector are counted, but secondary particles generated inside the detector as a result of an interaction are not.

Energy bins are determined by the `[EMIN,EMAX]` interval, in eV, and by the `No. OF E BINS`. To facilitate the preparation of step-like graphs two energies are printed in each row, namely, the low end and the middle point of each bin.

The `RELATIVE UNCERTAINTY REQUESTED`, specified as a percentage, is used as a criterion to stop the simulation.

11.8 Tally: Particle Track Structure

This tally allows the representation of particle tracks in a 3D graph. To this end, particle coordinates after each interaction are written to an external file. This file is used by the corresponding Gnuplot script to display the tracks (see section 11.2).

To prevent creating huge files with myriads of particle tracks that would be virtually impossible to discern when displayed on the computer screen, only a few simulated histories are reported. The number is set by the user in the input file. A reasonably small number is 100.

11.9 Tally: Phase-Space File

The dynamic state of all particles entering (or created inside) the specified `DETECTION MATERIAL` is written to the file indicated in the field `PSF FILENAME`. More precisely, the variables written are, in the quoted order: KPAR, E, X, Y, Z, U, V, W, WGHT, DN, ILB(1), ILB(2), ILB(3), ILB(4) AND ILB(5), *i.e.*, particle type, energy, position, direction, statistical weight, and all ILB labels.

The meaning of each variable is explained in the PENELOPE manual (Salvat 2019), except DN. This variable contains the increment in the number of primary histories since the previous record in the PSF was written. Thus DN=0 indicates that the current particle record belongs to the same history as the previous one and DN=3, for instance, implies that it was necessary to simulate three additional primary histories until a particle reached again the PSF scoring region. DN is necessary to produce accurate estimations of the statistical uncertainties.

The sum of all the DNs of a PSF should coincide with the number of simulated histories employed to generate that PSF. This consistency check may fail if the last simulated history (or histories) do not score any particle in the PSF. Although this match is not critical for most applications, it may cause interpretation problems with the bookkeeping of the number of histories. To avoid this issue, when the last history does not contribute to the PSF one last particle is artificially appended. This particle is identical to the last real scored particle, except that its statistical weight (WGHT) is set to zero. This will effectively make nil the contribution to any quantity of interest that could be subsequently calculated with the PSF in question and that takes WGHT into account. We refer to this last "fake" particle as an *endino*. Endinos can be created only if the penEasy standard format is used; if, instead, the IAEA format is used endinos are not appended since in this case the total number of histories is already stated in the header file that must accompany the PSF—see section 14 for details.

When the standard penEasy text format is selected in the field `PSF FORMAT`, the PSF is written in a plain text (ASCII) file. In this format each variable is separated from the next by one or more spaces. Given that a non-negligible fraction of the time needed to simulate each history is spent accessing the information in the PSF record, this format is obviously not optimized for speed nor for storage volume. Its main advantage is that it is human-readable and directly amenable to graphical representation, *e.g.*, with Gnuplot. However, for production runs a binary format ("unformatted", in the Fortran jargon) is more convenient. A discussion on how to implement the latter following the so-called IAEA standard format is deferred until section 14.

The simulation of particles reaching the PSF detector must be terminated to ensure the consistency of subsequent simulations that employ this PSF as the source of initial particle states. Failing to do so would allow a scoring particle to generate secondaries downstream which, in turn, could also reach the PSF and be scored. These secondary particles would then be doubly counted since, in the subsequent simulation, the primary read from the PSF would re-generate them again. To ensure that these situations do not occur, all absortion energies for the detection material must be set to infinity, namely, to a very large value. The program stops and an error message is issued if this condition is not satisfied.

The generated PSF does not include information about photon polarization states which, if present, is lost when particles are scored. This must be taken into account if the polarization state is likely to have an effect on results produced when PSFs are used as the source term of subsequent simulations.

11.10 Tally: Photon Fluence Point

This tally computes the photon fluence spectrum at a detection point specified by its `DETECTOR POSITION COORDINATES`. The spectrum is obtained in the interval [`EMIN`,`EMAX`] specified by the user and with the selected `No. OF E BINS IN SPECTRUM`. Its estimator is called "next collision" by some authors, although we shall prefer the term *detection forcing estimator*.

To facilitate the preparation of step-like graphs two energies are reported per bin, namely, the low end and the middle point of the bin. The `RELATIVE UNCERTAINTY REQUESTED`, specified as a percentage, is used as a criterion to stop the simulation.

The relative complexity of this tally deserves additional comments. In Appendix C a detailed description of the conceptual basis and the tally main features is given. It is strongly recommended that users read the appendix and take into consideration the approximations and limitations described therein. The following list highlights the consequences from the point of view of the tally usage.

- This tally is designed to work in conjuction with the BIGS source model. There is, however, one caveat: when use is made of the beam field feature of this source model (see section 8.2) the reported results will be biased if photons generated at the source have a non negligible probability of reaching the detection point unscattered. This is due to the fact that the tally assumes the source emits in a solid angle determined by the spherical trapezoidal window described in section 8.2, thus ignoring any possible solid angle reduction introduced by the beam field shape.

- The Phase-Space File source model, or other alternative source models created by users, may bias results due to the incorrect scoring of contributions from the source term. These other source models should nevertheless produce accurate results if the detector is far enough from the location of the primary particles to ensure that their fluence contributions are negligible.

- Photon sources emitting particles in vacuum can introduce a bias caused by an incorrect estimation of the contribution to the unscattered fluence from the source term. To prevent this artifact photon sources must be fully embedded in a material different from zero (*i.e.*, from vacuum). If necessary, a rarified material (*e.g.*, hydrogen with mass density equal to 10^{-10} g/cm^3) can be defined as an alternative to real vacuum.

- Detection forcing is known to have a finite expected value but, when the detector is not in vaccum, infinite variance. Apart from precluding the estimation of meaningful statistical uncertaintites, this feature has additional implications: (i) the estimator does not converge to the expected value with the usual $N^{-1/2}$ dependency, where N is the number of histories, but as $N^{-1/3}$; (ii) the evolution of the estimator with N may show sudden jumps from one history to the next.

- To restore the usual $N^{-1/2}$ convergence, and to mitigate the said variations, users can define an exclusion sphere surrounding the detection point. Its radius, defined in the field `RADIUS OF EXCLUSION`, is interpreted in cm if a positive value is entered and as the number of mean free paths at the maximum photon energy if the value is negative.

- Contributions produced by interactions occurring inside the exclusion sphere are neglected, which underestimates the true fluence by an amount roughly proportional to the sphere radius, if it is small enough. The output report includes a separate estimation of the contribution of the exclusion sphere based on a semi-analytical approximation. To obtain this estimate the code assumes that the sphere is homogeneous, filled with the material found at its center. It is the user responsibility to ensure that this is the case.

- Although not recommended, the radius of the exclusion sphere can be set to zero, which is equivalent to not having any exclusion sphere. Due to the said divergence at the detector position, the statistical uncertainty is then infinite and its reported values are meaningless. The only exception is when the detector is located in vacuum, there is no divergence and the variance is finite. In this case the exclusion sphere does not operate and its radius is irrelevant, provided the sphere does not intersect any material object. Hence, when the detector is in vacuum setting the radius to zero is recommended.

- Different approaches can be adopted to obtain the total fluence spectrum at the detection point. One approach is to add the contribution from the exclusion sphere as reported by the code. An alternative approach is to perform a set of simulations with increasingly reduced sphere radii in order to extrapolate the fluence for null radius.

11.11 Tally: Pixelated Imaging Detector

This tally creates a pixelated image of the radiation field impinging upon a detector.

Detector position and reference frame

PenEasy scans the list of bodies (and modules) introduced in the geometry file (GEO file hereafter). The detector is identified as the body or module that is made of the material introduced in the field `DETECTION MATERIAL`. If more than one body or module are made of the detection material, the last one listed in the GEO file is chosen. Hereafter we shall refer to this body or module as DETBODY.

In order for DETBODY to be a valid detector the following considerations apply.

- In the GEO file DETBODY must be limited by, at least, six planes (P1, P2, ..., P6) forming a rectangular box, that is, by three mutually perpendicular pairs of parallel planes. The space delimited by these planes will be called the detector box.

- In the GEO file these planes must be the first six entries appearing in the list of surfaces limiting DETBODY—additional limiting surfaces and bodies can be introduced after them. If we call P1 the first introduced plane, P2 the second, up to P6, then
  - P1 and P2 must be parallel,
  - P3 and P4 must be parallel,
  - P5 and P6 must be parallel,

  and P1, P3 and P5 must be mutually perpendicular, thus defining an orthogonal frame.

  The following example for a GEO file illustrates the idea,

```
BODY    (  37)                     <- Detector body
MATERIAL(   2)                     <- Detecting material
SURFACE (   9), SIDE POINTER=(+1)  <- First limiting surface is P1
SURFACE (  10), SIDE POINTER=(-1)  <- P2; parallel to P1
SURFACE (  11), SIDE POINTER=(+1)  <- P3; perpendicular to P1
SURFACE (  12), SIDE POINTER=(-1)  <- P4; parallel to P3
SURFACE (   7), SIDE POINTER=(+1)  <- P5; perpendicular to P1 and P3
SURFACE (   8), SIDE POINTER=(-1)  <- P6; parallel to P5
```

  It must be emphasized that the identification of P1...P6 bears no relation with the user-defined surface labels. It is only the order in which planes are declared inside the body definition what determines the said identification. Additional limiting surfaces, to be declared after P1...P6, can be used if needed.

- The use of quadric equations of the type $x^2-1=0$ to define pairs of parallel planes ($x=+1$ and $x=-1$ in the given equation) is not allowed. If attempted this will cause the program to issue an error message and stop during the initialization.

A reference frame, named S_{det} hereafter, attached to the detector box is automatically defined by the system in the following manner. The x axis of S_{det} runs perpendicularly from P1 to P2, with P1 being the plane $x=0$ (the yz plane). The y axis runs from P3 to P4, with P3 being the $y=0$ (xz) plane of S_{det} . Finally, the z axis runs from P5 to P6, with P5 being the $z=0$ (xy) plane of S_{det}. The origin of coordinates of S_{det} is therefore located at the intersection of the (perpendicular) planes P1, P3 and P5.

The previous scheme provides a natural, simple, intuitive and automatic way of defining the detector and its associated reference frame. The image reported by the tally and displayed by the corresponding Gnuplot script on the computer screen is referenced to the $x-y$ pair of axis of the S_{det} frame, with x running horizontally from left to right and y running vertically from bottom to top.

Photon interaction filter

Photons arriving at the detector can be filtered according to the interactions they have undergone. This is controled via the field `FILTER PHOTON INTERACTION`, with the following options:

- 0: no filter is applied. This is the option by default.
- -1: only unscattered photons are accepted, *i.e.*, those that are emitted from the source and reach the detector without having undergone any interaction.
- 1: selects photons that have undergone exactly one Rayleigh interaction before reaching the detector.
- 2: selects photons that have undergone exactly one Compton interaction before reaching the detector.
- 3: selects secondary photons, that is, those that are not emitted from a source. Note: initial photons that are created from a PSF source do not qualify for this filter; they are considered unscattered particles even if they were created as a result of an interaction (*e.g.*, bremsstrahlung radiation) during the simulation that produced the PSF.
- 9: selects photons that have experienced two or more interactions, or secondaries (in the sense described in the previous item) that have experienced one or more.

For example, setting the filter equal to 2 means that the image produced corresponds to photons that have been Compton scattered exactly once. Charged particle contributions are also included in the image, regardless of their origin. If desired, the latter can be avoided by defining a body covering the detector with an infinite (very large) absorption energy for charged particles.

The activation of any of these filters requires that the absorption energies in the detection material, both for photons and charged particles, be set to infinity. Failure to do so will cause the program to stop with an error message. The activation also implies the internal use of the variable ILB(5) in PENELOPE, which is therefore rendered unsuitable for other possible uses. This fact is relevant only for users planning to develop their own sources and tallies.

Pixels size and number

The image is reported as a matrix of pixels (picture elements). Each pixel, which can be thought of as an elemental detector, is a (usually small) rectangular portion of the detector box with its sides parallel to the x and y axis of S_{det}. Its dimensions along these axis will be called Δx and Δy. Pixels are located in the first quadrant of the xy plane of S_{det}, with the first pixel covering the rectangular region $(0 < x < \Delta x, 0 < y < \Delta y)$.

The value of Δx is introduced in the field `X-PIXEL SIZE`. The field `No. X-PIXELS` determines the number of pixels N_x along the x axis of S_{det}. Notice that $G_x = N_x \Delta x$ is the size along x of the grid of pixels. Alternatively, one of the two values N_x or Δx (but not both) can be set equal to zero. In this case the size G_x is set equal to the distance between the planes P1 and P2 (see above) and the parameter that was set to zero (either N_x or Δx) is automatically recomputed from $G_x = N_x \Delta x$. Analogous fields and definitions are provided for the y axis.

Detection modes

The field `DETECTION MODE` is used to select among energy integrating, photon counting or pulse-height spectrum (energy discriminating) modes. Their operation is described below.

- In energy integrating mode the reported image "signal" is the energy deposited in each pixel per unit pixel area and per history. Signal units are eV/cm^2 per history.
- In photon counting mode the image signal is the number of counts per history. A count is scored if, after the completion of a history, the energy deposited in a pixel falls in the interval `[EMIN,EMAX]`. `EMIN` is therefore the threshold for photon counting. Signal units are counts per history.
- In energy discriminating mode a pulse-height spectrum is tallied for each pixel, that is, the energy deposited is classified into energy bins. The quantity reported for each pixel is the number of counts, per history, in each energy bin divided by the bin width. Its units are counts/eV per history. Energy bins cover the interval `[EMIN,EMAX]` and their number is defined in the field `No. OF E BINS`.

It should be mentioned that some variance-reduction techniques may bias results in photon counting and energy discriminating modes. This is not a limitation of the code, but an unavoidable consequence of the nature of these tallies.

Report formats

The field `REPORT FORMAT` is used to select among columnar, matrix or binary formats.

In columnar format an output file containing the image signal for each pixel and its associated statistical uncertainty is written in plain text (ASCII) format. Each record (line) includes pixel indices and coordinates. To facilitate the preparation of step-like graphs two values are printed for each coordinate of each pixel, namely, the low end and the middle point of the pixel. In energy discriminating mode energy bins are also printed.

In matrix format an output file containing the image signal for each pixel is written in plain text. Each line represents a line of pixels with constant x coordinate in the S_{det} frame. No indices, coordinates or uncertainties are printed next to each datum. This format is not available in energy discriminating mode, since this mode produces an output that is essentially 3D and, therefore, not naturally amenable to a printout in matrix format.

The main advantage of the former two formats is that they are human-readable and well suited for graphical representation, *e.g.*, with Gnuplot. However, given the fact that commercially available detectors are already in the tens of megapixels range, these formats may not be optimal when disk storage is an issue. In this case the binary format is more convenient as it produces an output file of considerably smaller size.

Stopping criteria

The value entered in the field `RELATIVE UNCERTAINTY REQUESTED` is used as a criterion to stop the simulation.

Signal collection effects

An ideal detector reproduces precisely the radiation flux map that arrives at its entrance face. Real imaging detectors, however, introduce a certain distortion due to, mainly, two factors: (i) ionizing radiation is transported inside the detector itself, which implies that the energy-flux entrance map differs from the energy deposition map; and (ii) signal collection effects introduce additional distortion in the final image. In an indirect (scintillating) imaging detector signal collection effects are caused by the spread of light photons from the point in the phosphor where they are created; in a direct detector, these effects are attributable to the transport of charges (electrons and holes) driven by the electric field. Hereafter we shall use the generic term "signal" to refer to light or charge collection indistinctly. The sensitive material is identified in the field `DETECTION MATERIAL`.

In the simulation the behavior of an ideal detector can be mimicked assuming that: (i) the detection material is a perfect absorbent, *i.e.*, its absorption energies EABS are set to infinity; and (ii) there are no signal collection effects, *i.e.*, the field `ACTIVATE SIGNAL COLLECTION EFFECTS` is off.

When signal collection is on, the model in penEasy describes two different types of effects. These are: (i) the variation of signal collection efficiency with the depth of the energy deposition event inside the detector; and (ii) the lateral spread of the signal around the position of that event. More precisely,

- the field `CE0,CE1` defines the coefficients in the linear equation $\eta(z) = \texttt{CE0} + \texttt{CE1}\ z$, where η is the collection efficiency. $\eta(z)$ is the probability that an optical photon created at a depth z in the scintillating material reaches the photodiode and, therefore, produces some signal. In terms of the quantity reported by the present tally, the energy deposited by the ionizing particle is multiplied by $\eta(z)$.
- the field `FW0,FW1` defines the coefficients in the linear equation $\mathrm{FWHM}(z) = \texttt{FW0} + \texttt{FW1}\ z$. The quantity $\mathrm{FWHM}(z)$ is the full-width at half maximum of the signal profile produced when optical photons are generated at a depth z in the scintillating material. Following Freed et al. (2010), penEasy assumes that the shape of the signal profile is a truncated, normalized to unit area, Cauchy-Lorentz distribution,
$$L(r;z) \sim \frac{1}{r^2 + (\mathrm{FWHM})^2/4},$$
where $r^2 = x^2 + y^2$.

The quantity tallied in a given pixel of center coordinates (x_i, y_j) (i and j represent the

integers used to index the pixels) after an energy deposition $E_{\rm dep}$ has taken place at (x, y, z) is therefore

$$E_{\rm tally} = E_{\rm dep}\eta(z)L(r; z),$$

where

$$r^2 = (x_i - x)^2 + (y_j - y)^2.$$

The coordinates of the pixel and the energy deposition event are referred to the $S_{\rm det}$ frame, not the lab.

In the photon counting and energy discriminating modes an additional effect is simulated. The signal deposited in each pixel is convolved, before the count is tallied, with a Gaussian function. This imitates the effect of signal processing in a real spectrometer, where the effect of the generation of light photons (or electron-hole pairs), its conversion into electric charges at the photodiode (or collection of charges) and the amplification of the resulting electrical signal can be modeled as a random process with Gaussian dispersion.

Since signal units are given in terms of energy (eV), the full width at half maximum (FWHM) of the said Gaussian is also given in eV. It is defined as

$$\text{FWHM} = \sqrt{A + BE}$$

where E is the signal deposited in a pixel and A and B are user-defined parameters introduced in the field `A,B FOR SIGNAL GAUSSIAN NOISE`. A constant width can be specified by setting $B = 0$.

Finally, if the field `WRITE POINT-SPREAD FUNCTION TO A FILE` is set to 1 the code writes an output file with a tabulation of the function $\eta(z)L(r; z)$, which represents the point response function at r when an energy deposition occurs at $(0, 0, z)$—cylindrical symmetry is assumed.

The strategy adopted by penEasy to score counts in the photon counting and energy discriminating modes deserves one further comment. At the end of the simulation of a given history one count is produced in any given pixel if the signal in that pixel is within the energy limits set by the user in the input file, independently from the signal collected in neighborg pixels. A consequence of this "independent" counting scheme is that a single ionizing particle could produce a count in more than one pixel simultaneously. A possible alternative behavior is that each particle history produces exactly one count in a single pixel. That pixel could be chosen, for example, to be the centroid of all the detected signals, or the pixel with the maximum detected signal. This "single count to a single pixel" method is in principle preferable to reduce image noise, but it is not currently available in any commercial detector that we are aware of, likely due to the increased difficulty in signal processing that it demands.

11.12 Tally: Pulse-Height Spectrum

In this tally the energy deposited by each history in the specified `DETECTION MATERIAL` field is classified into energy bins. The quantity reported is, for each energy bin, the

number of counts scored divided by the bin width and per unit history. The reported units are thus eV^{-1}. This definition coincides with the normalized probability distribution of the deposited energy.

All energy bins have equal width, which is determined by the energy interval defined in the field `[EMIN,EMAX]` and by their number, defined in `No. OF E BINS`. To facilitate the preparation of step-like graphs two energies are reported for each bin, namely, its low end and its middle point.

The simulated pulse-height spectrum can be convolved with a Gaussian function of variable width. This mimics the behavior of a real spectrometer, where the effect of the generation and transport of light (in a scintillation detector) or of electrical charges (in a semiconductor) and its conversion into a measurable electrical signal can be modeled as a random process with Gaussian dispersion.

The full width at half maximum (FWHM) of the Gaussian is defined as

$$\mathrm{FWHM}(E) = \sqrt{A + BE}$$

where E (in eV) is the energy deposited whereas A (eV^2) and B (eV) are user-defined parameters. A constant width can be specified by setting $B = 0$. The values of these parameters are usually determined by performing an experimental resolution calibration, where measured values of FWHM^2 are plotted vs. E and fit to a straight line.

The `RELATIVE UNCERTAINTY REQUESTED` is used as a criterion to stop the simulation.

It should be noticed that pulse-height spectra may be biased by some variance-reduction techniques. This is not a limitation of penEasy or PENELOPE, but an unavoidable consequence of the nature of the tally.

11.13 Tally: Spatial Dose Distribution

The 3D absorbed dose distribution per history is tallied in the x interval `[XMIN,XMAX]` using `NXBIN` bins. Corresponding intervals and bins are defined for the y and z Cartesian coordinates. The code assumes that each bin is homogeneous and determines its mass density from the value found at the bin center. It is the user responsibility to ensure that bins are indeed homogenous, since this condition is not checked by the code. The units used to report a 3D distribution are eV/g.

This tally can also report 2D and 1D distributions. For instance, if `NXBIN` is set to zero a single scoring region $-\infty < x < +\infty$ is defined thus yielding a 2D distribution in y and z. Analogous definitions apply when `NYBIN`=0 and `NZBIN`=0. If two out of the three `NBIN` values are set to zero, the remaining coordinate defines a 1D distribution. The units used to report 2D and 1D distributions are cm·eV/g and cm^2·eV/g, respectively.

When `NXBIN`=0 the user-defined interval `[XMIN,XMAX]` is still used to determine the center point $x_{\mathrm{mid}} = (x_{\mathrm{min}} + x_{\mathrm{max}})/2$ at which the mass density of the entire region $-\infty < x < +\infty$ is evaluated.

This tally should not be used to compute dose distributions inside voxelized regions. This is because, as mentioned above, it may report erroneous results if bins are not homogeneous. Inhomogeneities may arise because some bins overlap with more than one voxel, if their sizes do not match, or with voxels and quadric bodies, if both are used. To compute dose distributions in voxelized geometries the voxel dose tally should be used instead, see section 11.15.

If the field `PRINT COORDINATES` is set to 1 the values of x, y, z are printed in the output report. If it is set to 0 they are not. If it is set to -1 no coordinates are printed and the data is written in binary format to save disk space and CPU time. In the last case a pair of values (dose, uncertainty) is written in binary for each bin following the same order in which it is written when printing in text mode. For Gnuplot scripts to operate correctly this option has to be set to 1.

The `RELATIVE UNCERTAINTY REQUESTED` is used as a criterion to stop the simulation.

11.14 Tally: Spherical Dose Distribution

The absorbed dose distribution per history is tallied in the radial ($r = \sqrt{x^2 + y^2 + z^2}$) interval `[RMIN,RMAX]` using `NRBIN` bins. The code assumes that each one of these spherical shells is homogeneous and determines their mass density from the value found at the point of coordinates ($x = 0, y = 0, z = r_{\rm mid}$), where $r_{\rm mid}$ is the midpoint of the radial interval. It is the user responsibility to ensure that bins are indeed homogenous, since this condition is not checked by the code. The units used in the output report are eV/g.

To facilitate the preparation of step-like graphs two radial distances are printed per bin, namely, the low end and an average radius $r_{\rm avg}$ of the spherical shell. The average is weighted proportionally to r squared, resulting in

$$r_{\rm avg} = \frac{3}{4}\,\frac{r_{\rm hi}^4 - r_{\rm lo}^4}{r_{\rm hi}^3 - r_{\rm lo}^3},$$

where $r_{\rm lo}$ and $r_{\rm hi}$ are the low and high ends of the bin, respectively. The rationale for this choice is as follows. Assuming that the dose $D(r)$ varies linearly in a small interval, $r_{\rm avg}$ is the point at which $D(r_{\rm avg}) = D_{\rm avg}$, where $D_{\rm avg}$ is the average dose in the interval.

This tally should not be used to compute dose distributions inside voxelized regions. This is because, as mentioned above, it may report erroneous results if bins are not homogeneous. Inhomogeneities may arise because some bins overlap with more than one voxel, if their sizes do not match, or with voxels and quadric bodies, if both are used. To compute dose distributions in voxelized geometries the voxel dose tally should be used instead, see section 11.15.

If the field `PRINT COORDINATES` is set to 1 the values of r are printed in the output report. If it is set to 0 they are not. For Gnuplot scripts to operate correctly this option has to be set to 1.

The `RELATIVE UNCERTAINTY REQUESTED` is used as a criterion to stop the simulation.

11.15 Tally: Voxel Dose Distribution

The absorbed dose per history in each volume element (voxel) is tallied for the region of interest (`ROI INDEX MIN,MAX`) defined in the input file. The ROI is defined in terms of a range of x, y and z indices. If the ROI for a given coordinate is set to $(0, 0)$ the effective ROI goes from 1 up to the last voxel along that coordinate.

If the field `PRINT MASS` is set to 1 the mass of each voxel is also printed. The mass is evaluated during the tally initialization as the product of the voxel mass density times the voxel volume. Most commonly, the voxel volume is simply the product of the three voxel sides. However, as discussed in section 9, when quadric and voxelized geometry models are mixed a voxel can be partially "obstructed" by a quadric and, in that case, its irregular volume can be evaluated by a numerical integration. Hence, printing the voxel mass can be useful to discern which voxels are blocked by quadrics and by how much.

If the field `PRINT COORDINATES` is set to 1 the Cartesian coordinates of all voxels in the ROI are also printed in the output report. If it is set to 0, they are not. If it is set to -1 no coordinates are printed and, furthermore, the data is written in binary format to save disk space and CPU time. In this last case a pair (dose,uncert) is written in binary for each bin in the same order in which those pairs are written when printing in text mode. For Gnuplot scripts to operate correctly this option has to be set to 1.

The field `RELATIVE UNCERTAINTY REQUESTED` is used as a criterion to stop the simulation.

12 Input file: variance-reduction techniques

Some ill-conditioned problems take an unreasonably long simulation time to complete. In these cases it may be useful to have recourse to the so-called variance-reduction techniques (VRT), intended to reduce the CPU time required to produce a given statistical uncertainty. Most VRTs accomplish this by artificially increasing the probability of those events that contribute to the quantity of interest, thus producing more "counts" and, therefore, better counting statistics. To keep simulation results unbiased a so-called statistical weight (variable `WGHT` in the PENELOPE code) is assigned to each particle and used to weight its contributions in a manner that preserves average values unaltered.

Three VRTs are provided in penEasy, namely, interaction forcing, particle splitting and Russian roulette. Their principles and application are described in the PENELOPE manual (Salvat 2019). As explained below, penEasy extends the capabilities of these methods as implemented in PENELOPE.

VRTs can reduce the time considerably if applied correctly. Beware, however, that when used incorrectly they can bias results and produce artifacts. A general comment in this respect is that VRTs may bias tallies producing pulse-height spectra. This is not a limitation of PENELOPE or penEasy, but a consequence of the nature of that particular kind of tally.

12.1 Interaction forcing

Interaction forcing artificially increases interaction cross sections so as to produce a larger number of interaction events and, therefore, better counting statistics. If the cross section of a particle species is increased by the factor `FORCING` then its statistical weight is reduced by the same factor.

Forcing is not applied if the particle's statistical weight is below the value specified in the input file field `WGHTMIN`. This feature can be used to prevent the repetitive application of interaction forcing to secondary particles that had already been generated in forced events, since those secondary particles will have a reduced statistical weight. It is advisable to define a prudent `WGHTMIN` to prevent the occurrence of extremely small statistical weights, which may give rise to numerical precision problems. For instance, assuming that no other VR technique is applied, a `WGHTMIN` equal to 1.0 prevents the re-application of interaction forcing to all forced secondary particles.

Different forcing factors can be used for different combinations of material, particle species and type of interaction. The configuration section for this VRT contains a table with the following fields,

- The material number (`MAT`).
- Particle type ((`KPAR`). Following the standard PENELOPE numbering, the values 1,2,3 correspond to electrons, photons and positrons, respectively.

- Type of interaction (`ICOL`). PENELOPE labels the interaction mechanisms in the following way (see the PENELOPE manual for details):

  For electrons and positrons:

```
ICOL = 1 artificial soft event (hinge)
     = 2 hard elastic collision
     = 3 hard inelastic collision
     = 4 hard bremsstrahlung emission
     = 5 inner-shell ionization
     = 6 positron annihilation
     = 7 delta interaction
     = 8 'auxiliary' fictitious interactions
     = -4 value reserved for Bremsstrahlung splitting (see below)
```

  For photons:

```
ICOL = 1 coherent (Rayleigh) scattering.
     = 2 incoherent (Compton) scattering.
     = 3 photoelectric absorption.
     = 4 electron-positron pair production.
     = 7 delta interaction.
     = 8 'auxiliary' fictitious interactions.
```

  If `ICOL` is set to 0 all interactions are forced by the same factor.

- Forcing factor (`FORCING`) to be applied to interactions `ICOL` when particles of type `KPAR` interact inside material `MAT`.

The last line entered before `END of SECTION` must have `MAT` $= -1$ to signal the end of the list.

In the particular case of electrons the setting `ICOL` $= -4$ has the effect of activating Bremsstrahlung splitting as described in the PENELOPE manual (Salvat 2019). In essence, the generated photons are distributed at random azimuthal angles—with respect to the direction of flight of the emitting electron. Although penEasy will only accept the value -4 when `KPAR` $= 1$ (that is, for electrons), Bremsstrahlung splitting will be applied both to electrons and positrons.

It must be noticed that the term "interaction forcing" is used by some other authors to denote a VRT that differs from the one defined in PENELOPE.

12.2 Particle splitting

In particle splitting copies (or clones) of the original particle are artificially created with a statistical weight equal to that of the original particle divided by the number of clones. Splitting should be applied when particles approach the region of interest.

Splitting is applied only if the particle's statistical weight is above the value specified in the input file field `WGHTMIN`. This feature can be used to prevent the repetitive application of splitting to already cloned particles, since those clones will have a reduced statistical weight. It is advisable to define a prudent `WGHTMIN` to prevent the occurrence of extremely small statistical weights, which may give rise to numerical precision problems. For instance, assuming that no other VR technique is applied, a `WGHTMIN` equal to 1.0 prevents the re-application of splitting to clones.

Splitting is applied when a particle *enters* the declared `SPLITTING MATERIAL`. It is not applied when the particle is born in it, so it is not applied to primary, secondary or cloned particles generated inside that material.

Three different splitting modes are possible, namely,

- "Simple", which creates copies that are identical to the original particle.
- "Rotational", which creates copies that are rotated around the z axis (or any other axis, see below) so that they are uniformly distributed within a user-defined azimuthal interval.
- "XY", which creates three copies of the original particle, one with inverted sign for the x coordinate, one with inverted y and one with inverted x and y. The corresponding components of the direction of flight vector are also rotated.

Rotational splitting should only be used when the problem has azimuthal symmetry, whereas XY splitting should only be used for problems with fourfold symmetry with respect to reflections on the x-z and y-z planes. These two situations occur frequently in the simulation of the patient-independent part of clinical linear accelerator (linac) heads. It is the user's responsibility to ensure that these symmetries exist, since penEasy assumes no change in the body or material assigned to the newly created copies—although it does compute the new voxel indices if cloned particles are in a voxelised geometry.

The number of copies is specified as the `SPLITTING FACTOR`. This datum is ignored for XY splitting because in that case the number of copies is always 4.

In the case of rotational or XY symmetry the reference system in which these symmetries are apparent can be different from the lab reference frame employed in the description of the geometry. An example is in a linac radiotherapy treatment when the gantry, table or collimator angles are different from zero. To still be able to take advantage of the symmetry and apply splitting a set of `EULER ANGLES` and a `SHIFT` can be introduced. The rotation and translation defined by these parameters are those needed to convert the original lab reference system into the system in which the symmetry is manifest. The usual PENELOPE convention for the Euler angles is used for the rotation.

Splitting can be conditionally applied depending on the `SIGN OF W`, the third component of the direction of flight of the particle, as it enters the splitting material. For particle beams, this allows to discern between particles flying downstream from those going

upstream because they have been backscattered. Notice that the value of `W` is to be understood in the rotated reference system as defined by the Euler angles commented before. So, if the $+z$ axis of the rotated frame coincides with the initial direction of the beam, the character "+" in the `SIGN OF W` field identifies particles flying downstream, that is, along the positive z axis; the character "−" identifies backscattered particles; and "0" means "apply to all directions".

In the case of rotational splitting the cloned particles can be distributed in an arbitrary azimuthal interval that can be smaller than 360°. The interval is specified by giving, in degrees, the initial azimuthal angle, `PHI0`, and the interval width, `DeltaPHI`. For instance, the pair $\{0.0, 360.0\}$ defines a full circle and $\{90.0, 20.0\}$ defines the interval $(90.0, 110.0)°$.

12.3 Russian roulette

In Russian roulette particles are killed with a given probability and survive with the complementary probability. Users define the latter in the field `SURVIVAL PROBABILITY`. Surviving particles are affected by an increased statistical weight set equal to the original weight divided by the survival probability. The roulette should be applied when a particle is going away from the region of interest.

This VRT is not applied if the particle's statistical weight is above the value specified in the input file field `WGHTMAX`. This feature can be used to prevent the repetitive application of the roulette to surviving particles. It is advisable to define a prudent `WGHTMAX` to prevent the occurrence of extremely large statistical weights, which may give rise to numerical precision problems. For instance, assuming that no other VR technique is applied, a `WGHTMAX` equal to 1.0 prevents the re-application of the roulette to particles that have already survived once.

Russian roulette is played when a particle *enters* the `RUSSIAN ROULETTE MATERIAL`. It is not applied when the particle is born in it, so it is not applied to source initial particles generated inside that material nor to secondary particles created in the material as a result of an interaction.

13 Parallel execution

Monte Carlo simulations can be parallelized relatively easily. Various strategies and tools have been developed to facilitate the implementation of parallel algorithms, *e.g.*, MPI, openMP, PVM and openMOSIX. All these have been successfully used with Monte Carlo, but they suffer from the drawback of requiring the modification of the sequential code and/or the installation of additional software, which may be inconvenient for some users.

An alternative solution for penEasy that does not have these limitations is provided by the package clonEasy[7]. Its principles and usage are described by Badal and Sempau (2006). In brief: the same job is distributed to several CPUs (the clones) but each clone is fed with a different set of initial seeds for the random number generator so as to initiate seemingly independent sequences. The resulting partial results are collected from the clones and averaged appropriately. The secure shell (ssh) protocol is used to distribute the jobs among the clones. This protocol is usually embedded in Unix-like systems, including the most popular Linux distributions. This means that any accessible Unix-like computer—connected to the Internet with a permanent IP address and with a valid user account—can be used as a clone for the parallel computation. This scheme does not require the installation of any additional software.

In order to feed different clones with different seeds it is necessary to exploit, as explained in section 7, the feature of penEasy that allows the introduction of the initial seed values through an external file. Seeds for the PENELOPE pseudo-random number generator (named RANECU) that initiate disjoint sequences can be obtained with the help of the seedsMLCG code[8].

[7] clonEasy can be freely downloaded from http://www.upc.edu/inte/downloads.
[8] seedsMLCG can be freely downloaded from http://www.upc.edu/inte/downloads.

14 Phase-space files in IAEA format

PenEasy includes both a source and a tally called phase-space file (PSF). A PSF is basically a record of the dynamical state of all particles arriving at (tally) or emerging from (source) a certain region in space. By dynamical state it is meant the type, energy, position, direction, etc. of those particles. The penEasy code allows users to store this information in two different formats, namely, in plain text (ASCII) or in a binary format specified by the International Atomic Energy Agency (IAEA), see Capote et al. (2006).

While the plain text format has the advantages of being human-readable and compatible with all platforms, it also involves the drawback of requiring considerably more disc space than the alternative. Furthermore, the use of the IAEA-standardized binary format permits the use in penEasy of PSFs obtained from the IAEA database for linear accelerators and cobalt therapy units (see http://www-nds.iaea.org/phsp), even if these PSFs were not computed with penEasy.

To create or read PSFs in IAEA format it is necessary to select this option in the corresponding source or tally sections in the penEasy input file. Although users are requested to provide a single PSF filename with no extension, two files are actually involved in the process. One is a header file, which is written in plain, human-readable text format. The other is the actual PSF, in binary format. These files are given filenames that are formed by the filename root provided by the user and the terminations ".IAEAheader" and ".IAEAphsp" for the header and the binary PSF, respectively.

To facilitate the processing of IAEA-compliant PSFs, the Agency provides a package of routines written in the C++ programming language. PenEasy invokes these routines if the user requests, in the input file, that PSFs are handled in IAEA format, but to do so a compilation procedure different from that presented in section 3 must be followed. The compilation command is somewhat more elaborate due, in part, to the fact that it is necessary to combine code in Fortran (PENELOPE/penEasy) and C++ (IAEA routines).

Assuming a Linux system and the gfortran compiler, the compilation procedure is as follows. First, steps 1 and 2 (copy the necessary Fortran files) of section 3 must have been successfully completed. Then, the next steps to be taken are,

3. Copy the files
   ~/penEasy/IAEAcode/iaea_config.h
   ~/penEasy/IAEAcode/iaea_header.cpp
   ~/penEasy/IAEAcode/iaea_header.h
   ~/penEasy/IAEAcode/iaea_phsp.cpp
   ~/penEasy/IAEAcode/iaea_phsp.h
   ~/penEasy/IAEAcode/iaea_record.cpp
   ~/penEasy/IAEAcode/iaea_record.h
   ~/penEasy/IAEAcode/utilities.cpp
   ~/penEasy/IAEAcode/utilities.h
   into the directory ~/fortranCode/. Other files included in ~/penEasy/IAEAcode/

come with the IAEA package and are included also in penEasy for completeness, but they must not be copied into in the compilation directory.

4. Compile and link penEasy.F,

   $ gfortran penEasy.F *.cpp −o penEasy.x −O −lm −lstdc++ −DIAEAPSF

   This generates the executable file penEasy.x. The option "−DIAEAPSF" ensures that the IAEA routines are invoked properly.

The gfortran compiler automatically invokes the C++ compiler for the *.cpp files and the Fortran compiler for the *.F files. Other compilers may require a separate compilation of the Fortran and C++ codes. For example, with Intel Fortran steps 1-3 are the same, but then

4. Compile the IAEA routines,

   $ icl *.cpp −c −O2

   This generates a set of *.o files.

5. Compile penEasy.F and link with the IAEA routines,

   $ ifort penEasy.F *.o −fpp −o penEasy.x −O2 −lm −lstdc++ −DIAEAPSF

This generates the executable penEasy.x. For the Intel case, some issues deserve further comments,

- Depending on the system, the options "−lm" and "−lstdc++" must be present in both compilation commands. Note that "−DIAEAPSF" is needed only for the Fortran code.
- The Fortran files in the penEasy distribution all have extension *.F. The capitalization of the "F" is not superfluous and must be kept as it is. See section 3 for further details on this matter.
- We have found that the optimization option "−O2" gives best results, but this may depend on a number of factors such as the computer architecture. Knowledgeable users may opt for other optimization levels and options to produce a faster code.

Executable for Windows

The executable code for Windows systems included in the official distribution, named penEasy.exe, has been created using the Intel Visual Fortran 64 Compiler XE version 12.0.2.154 with the options −fpp −O2 −DIAEAPSF.

Appendix A

Use of the TALLY routine for advanced users

This section is intended for users that are planning to modify an existing tally or that wish to create a new one.

Notice that the `TALLY` routine, which takes the arguments `MODE` (integer) and `ARG` (real*8), is called at various points of the penEasy main program where it is expected to perform different actions. The possible values of `MODE` identify the point at which the tally is called so that the appropiate action, if any, can be carried out.

The following list specifies the situation that corresponds to each mode and the value passed as `ARG` in each case. In the list, E represents the particle kinetic energy, for massive particles, or the total energy for photons. By a "history" we mean the simulation of a primary particle (or particles) produced by a single call to the source routine and the simulation of all the secondaries generated by it (or them).

Generally speaking the action to be performed depends on the nature of the tally. For the particular case of tallies reporting quantities related to the deposited energy the required action is described below in detail. It should be noticed that in all modes less or equal to zero the value of `ARG` should be added to these energy counters.

- Mode -99

  Situation:

  A new particle (either primary or secondary) has been retrieved from the stack and its simulation is about to begin.

  Required action:

  Its energy E should be subtracted from deposited energy counters.

  Passed argument:

  $-E$. (This value should be *added* to energy counters.)

- Mode -98

  Situation:

  The particle has been absorbed.

  Required action:

  Its energy E should be added to deposited energy counters.

  Passed argument:

  E.

  Comment:

KNOCK sets $E = 0$ after absorption and, thus, scoring E in energy deposited counters seems irrelevant. However, in penEasy a particle can also be absorbed right after it enters a medium with an E smaller than $E_{\rm abs}$, even if no interaction occurs in that medium—*i.e.*, not as a result of a call to KNOCK.

- Mode −97

  Situation:

  A positron has been absorbed with $E > 0$, *i.e.*, not as a result of a call to KNOCK but because it entered a region with lower $E_{\rm abs}$. Two photons, with an energy of $m_{\rm e}c^2 \approx 511$ keV each, have been pushed to the stack.

  Required action:

  $2m_{\rm e}c^2$ should be added to deposited energy counters.

  Passed argument:

  $2m_{\rm e}c^2$.

- Mode −96

  Situation:

  One or more particles have been added to the stack as a result of a particle splitting event.

  Required action:

  E*(NumberOfParticlesAdded) should be added to deposited energy counters. This mode is not used in case the splitting is produced by the PSF source, which uses mode= 0 instead (see below).

  Passed argument:

  E*(NumberOfParticlesAdded).

- Modes from −1 to −8

  Situation:

  An interaction, simulated by KNOCK, has occurred. The type of interaction is returned by KNOCK through the variable ICOL, which takes on values between 1 and 8. The calling MODE is set to ICOL with reversed sign. ICOL labels are as follows:

  Electrons (KPAR=1) and positrons (KPAR=3):

  ICOL =
  = 1 artificial soft event (hinge)
  = 2 hard elastic collision
  = 3 hard inelastic collision
  = 4 hard bremsstrahlung emission
  = 5 inner-shell ionization
  = 6 positron annihilation

= 7 delta interaction
= 8 “auxiliary” fictitious interactions

Photons (`KPAR`=2):

`ICOL` =
= 1 coherent (Rayleigh) scattering
= 2 incoherent (Compton) scattering
= 3 photoelectric absorption
= 4 electron-positron pair production
= 7 delta interaction
= 8 “auxiliary” fictitious interactions

Required action:

The energy $\Delta E > 0$ lost by the particle should be added to deposited energy counters.

Passed argument:

ΔE.

- Mode 0

  Situation:

  A source (*e.g.*, BIGS or PSF) has pushed a new particle to the stack.

  Required action:

  The particle energy E should be added to deposited energy counters.

  Passed argument:

  E.

  Comment:

  In the case of particle splitting in a PSF source only one call is made, although several particles will be added to the stack. The passed argument is then E * (NumberOfSplitCopies), which accounts for all the energy deposited.

- Mode 1

  Situation:

  A new history is about to begin.

  Required action:

  None specific.

  Passed argument:

  N, the current history number.

- Mode 2

  Situation:

Obsolete, not used anymore. Previously used to signal that the history number N had been changed by the PSF source. Read ∼/documentation/changeHistory.txt for further details.

- Mode 3

  Situation:

  The particle has been moved a certain distance Δs by `STEP` without any interface crossing and it is about to interact.

  Required action:

  None specific. Used by track length estimators to obtain the fluence.

  Passed argument:

  Δs, the distance travelled.

- Mode 4

  Situation:

  The particle has been moved a distance Δs_{eff} by `STEP` and an interface crossing has ocurred at the end of the flight.

  Required action:

  None specific. Used by track length estimators to obtain the fluence.

  Passed argument:

  Δs_{eff}, the distance travelled.

- Mode 5

  Situation:

  A particle has been killed. This implies that its simulation is finished without any further action. No energy is deposited and, in the case of a positron, no annihilation radiation is created.

  Required action:

  None specific.

  Passed argument:

  None (a dummy value).

- Mode 6

  Situation:

  A history has been finished.

  Required action:

  Perform bookkeeping procedures to store the average and variance of the quantities being calculated.

  Passed argument:

  N, the history number just finished.

- Mode 7

  Situation:

  Obsolete, not used anymore. Previously used to signal that a source routine had created a new particle and had moved it (using `STEP`) up to the object. Read ∼/documentation/changeHistory.txt for further details.

Appendix B

BIGS source spectrum: common situations[9]

In penEasy the source energy distribution must be coded as a histogram in the form

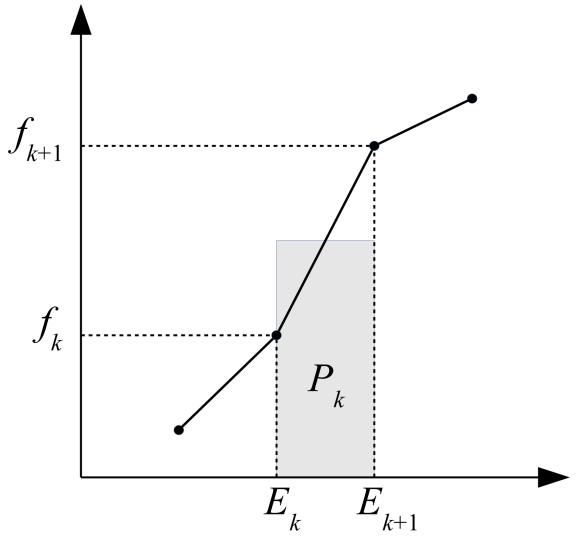
$$\begin{array}{ll} E_1 & P_1 \\ \vdots & \vdots \\ E_k & P_k \\ E_{k+1} & \vdots \\ \vdots & \vdots \\ E_K & P_K \\ E_{K+1} & -1 \end{array}$$

Let us suppose that the source spectrum is known as a, possibly unnormalized, *continuous* probability distribution $p(E)$. Then the P_k's can be set to the (unnormalized) probability of the interval (E_k, E_{k+1}), that is,

$$P_k = \int_{E_k}^{E_{k+1}} dE\, p(E) \qquad \text{with} \quad k = 1, \ldots, K.$$

The E_k's should be selected so as to create an energy grid dense enough to reproduce the main features of the spectrum. Note that P_{K+1} must be set to -1 to end the table.

If, instead, the (again possibly unnormalized) source spectrum is known in tabular form, say (E_k, p_k) with $k = 1, \ldots, K+1$, the previous method may be implemented using a suitable interpolation algorithm. For instance, a simple option is to adopt the same energy grid, $\{E_k\}$, and apply a linear interpolation scheme (see figure) to obtain the interval unnormalized probabilities,



[9] We thank Dr. José M. Fernández-Varea for providing the ideas and text for this appendix.

$$P_k = \frac{1}{2}\,(f_{k+1} + f_k)\,(E_{k+1} - E_k), \qquad k = 1, \ldots, K$$

In particular, if the energy grid is uniform ($E_{k+1} - E_k$ =constant) then the P_k's can be defined, ignoring the irrelevant change of normalization factor, as

$$P_k = \frac{1}{2}\,(f_{k+1} + f_k), \qquad k = 1, \ldots, K$$

The value of P_{K+1} must be set to -1 to terminate the spectrum.

Appendix C

The Photon Fluence Point tally

This tally estimates to photon fluence spectrum at a detection point D. To this end, the probability per unit area that a photon reaches D is scored.

Conceptual basis

Let us consider the estimation of the photon fluence at a detection point D embedded in a non-trivial geometry. For instance, in environmental dosimetry studies D might be located in mid air and an extended photon source is distributed on an irregular soil around it, a problem for which no symmetry can be exploited to increase the simulation efficiency.

We recall that (see, *e.g.*, International Commission on Radiation Units and Measurements 2011) the fluence Φ can be defined as the number of photons per unit area reaching a small test sphere of cross sectional area ΔA centered at D. A direct application of this definition to the problem described above is impractical, since for an extended and isotropic source the number of photons that reach the sphere is only a tiny fraction of the primary histories generated, which implies that unreasonably long computation times are required to attain low statistical uncertainties.

A more effective approach is to use a scoring method called next-event estimator by some authors (see, *e.g.*, Shultis and Faw 2011); we shall use the expression *detection forcing* to refer to it. The idea underlying the method is that, instead of counting photons that reach the test sphere, the probability per unit cross sectional area $p(dA)/dA$ of reaching the sphere is tallied. To avoid excessive complexity and to speed up the calculation some approximations (described below) are introduced and, therefore, strictly speaking the fluence produced by the present tally is not equivalent to the one produced with the tally Fluence Track Length, which is fully consistent with PENELOPE's physics except for the unavoidable statistical fluctuations inherent to Monte Carlo.

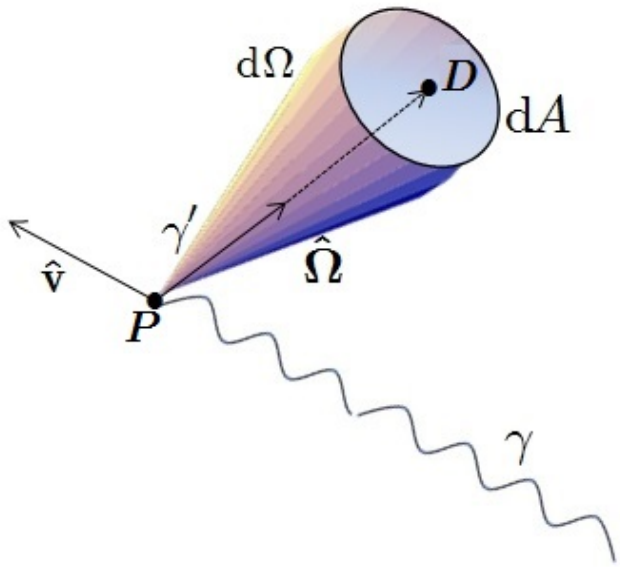
The practical implementation of the tally is as follows. Conventional detailed photon simulation is performed. After a photon experiences a Rayleigh or Compton interaction the probability per unit area $p(dA)/dA$ that a scattered photon is directed towards a small sphere centered at D is computed. Note that the photon direction does not, in general, coincide with the result of the actual interaction and so we shall term the photon pointing to the detector as "virtual".

Consider for instance a Rayleigh, or elastic ('el'), event. The probability per unit solid angle $d\Omega$ that the emerging virtual photon γ' is directed along the unit vector $\hat{\mathbf{\Omega}}$ that points at D is

$$p_{\rm el}(\hat{\mathbf{\Omega}}) = \frac{1}{\sigma_{\rm el}} \frac{d\sigma_{\rm el}(\hat{\mathbf{\Omega}}\cdot\hat{\mathbf{v}})}{d\Omega},$$

where $d\sigma_{\rm el}/d\Omega$ is the cross section differential in the solid angle evaluated at the angle

formed by $\hat{\mathbf{\Omega}}$ and the direction of flight $\hat{\mathbf{v}}$ of the primary photon.



The probability that γ' arrives at D without further interactions is

$$\exp\left(-\sum_m \mu_m r_m\right) \equiv \mathrm{e}^{-\Lambda},$$

where μ_m is the linear attenuation coefficient (*i.e.*, inverse mean free path) of the m-th material encountered by the virtual photon along its straight line path towards D and r_m the distance traveled in that material. For notational simplicity we have introduced the quantity

$$\Lambda \equiv \sum_m \mu_m r_m.$$

Seen from P, the solid angle $d\Omega$ subtended by a differential test area dA normal to $\hat{\mathbf{\Omega}}$ is

$$d\Omega = \frac{dA}{r^2},$$

where $r = \sum r_m$ is the total distance $\overline{PD}$. Hence, the probability per unit area that the virtual photon reaches dA is

$$\Delta\Phi_{\mathrm{el}} = \frac{p_{\mathrm{el}}(dA)}{dA} = \frac{p_{\mathrm{el}}(\hat{\mathbf{\Omega}})\, d\Omega}{dA}\,\mathrm{e}^{-\Lambda} = \\ = \frac{1}{\sigma_{\mathrm{el}}}\frac{d\sigma_{\mathrm{el}}}{d\Omega}\,\frac{\mathrm{e}^{-\Lambda}}{r^2}.$$

A completely analogous analysis can be done for the Compton case except that, being this interaction inelastic, Λ must be evaluated at the energy E of the scattered virtual photon γ', and not at the energy E_0 of the primary photon γ. Thus, for Compton,

$$\Delta\Phi_{\text{in}} = \frac{1}{\sigma_{\text{in}}(E_0)} \frac{d\sigma_{\text{el}}(\hat{\mathbf{\Omega}}; E_0)}{d\Omega} \frac{\mathrm{e}^{-\Lambda(E)}}{r^2}$$

where the values of the arguments at which the various terms are to be evaluated have been explicitly stated.

Contributions from characteristic x-rays and annihilation quanta (resulting from a positron annihilation) are scored whenever these photons are generated. To this end, when such a photon is recovered from the stack (PENELOPE stores secondary radiation in a memory stack), and before its simulation actually begins, its contribution to the fluence at D is tallied. The generation is assumed to be isotropic, which results in

$$\Delta\Phi_{\text{x or ann}} = \frac{1}{4\pi} \frac{\mathrm{e}^{-\Lambda(E)}}{r^2}$$

where E is the energy of the x-ray or of the annihilation photon, depending on the case. The assumption of isotropy is only approximate for annihilation photons, since a positron has a finite, albeit relatively small, probability to annihilate in flight. In the case of characteristic x-ray emission, and due to the fact that in some cases ionization of atoms by particle impact or by photons leads to ion alignment (see Mehlhorn 1968), the assumption of isotropy is also only approximate, although in this case our approach is equivalent to the one adopted in PENELOPE.

An additional contribution arises from photons that, after being generated at the source with energy E_s, arrive unscattered at D. The BIGS source model has a constant probability per unit solid angle equal to

$$p_{\text{s}}(\hat{\mathbf{\Omega}}) = \frac{1}{\Omega_s},$$

where Ω_s is the source solid angle aperture (see section 8.2). For each new photon generated at the source the fluence contribution is therefore

$$\Delta\Phi_s = \frac{\mathrm{e}^{-\Lambda(E_s)}}{\Omega_s\, r^2}.$$

The various $\Delta\Phi$ contributions are all independent of the extent of the test area. By letting dA tend to zero the estimator gives the fluence at exactly the point D, in contradistinction with the Fluence Track Length tally, which requires the definition of a detector with finite volume. Since all photons contribute to the score at every single step, detection forcing is usually a very effective technique to reduce the computation time.

Approximations

The present tally uses the following approximations.

- Virtual annihilation photons are assumed to be emitted isotropically, which is exact only if the positron annihilates at rest. This approximation could introduce a sizeable bias for directed high-energy photon beams.

- In Compton events, and for the purpose of computing the energy E of the virtual photon that points at D, Doppler broadening and electron binding effects are disregarded. This is expected to have a negligible influence when: (i) photon energy is well above the K edge; or (ii) the fluence spectrum is convolved with mass energy absorption coefficients in order to compute the absorbed dose; or (iii) the fluence spectrum is convolved with a broadening detector response.

- This tally is primarily designed for photon transport and, hence, the algorithm does not account for the fluence produced by bremsstrahlung photons that arrive at the detector unscattered. However, contributions from scattered bremsstrahlung photons are, as with any other photon, tallied when they scatter.

- The use of this tally with the phase-space file (PSF) source model can produce biased results. This is because contributions due to photon interactions occurring *before* photons reached the PSF "detector" (the PSF is previously generated in a separate simulation) are obviously not accounted for. Users should be aware of this effect and make sure that the PSF is stored far enough from the fluence detection point D to minimize the error.

- The BIGS photon source model can cause an apparent singularity at the detector under some circumstances. These are: (i) the source is extense, that is, it has a finite volume V_s; and (ii) photons are emitted as a parallel monodirectional (or nearly parallel) beam; and (iii) a straight line that passes through the detection point D and is aligned with the direction of flight of the source photons intersects the source volume V_s, *i.e.*, unscattered photons emitted from the source can reach the detector.

  To see the origin of the problem, and for simplicity, let us assume that there is no scattering. The intersection between the line and V_s defines a chord C inside V_s. It can be seen that, under these conditions, the contribution to the fluence Φ per unit history from unscattered source photons is

  $$\Phi = \frac{\text{length of } C}{V_s}.$$

  PENELOPE cannot determine the volume of a body limited by arbitrary quadric surfaces and, therefore, an alternative approach must be adopted.

  A simple workaround is to define a small non-null solid angle aperture of emission. It should be noted that penEasy computes the unscattered contribution from points in V_s that emit with a solid angle aperture Ω_s with an estimator that behaves as $\sim \Omega_s^{-1}$. The estimator converges to the correct value when Ω_s tends to zero because, although the contribution tends to infinity, the number of contributions vanishes as the region of source points that can reach the detector shrinks to nothing. However, and due to the randomness inherent to the Monte Carlo

algorithm, when such an unlikely point is generated by the source, a spurious very large contribution will appear, giving rise to a large jump in the tallied fluence. To prevent these rare events from creating huge tally jumps it is recommended that the solid angle aperture of the source be set to a not too small value.

The exclusion sphere

Consider a small spherical shell of radius r and thickness dr centered at the detection point D. The number of interactions in the shell vary as $r^2 dr$. Contributions to the fluence estimator, apart from attenuation, which is negligible as r tends to zero, vary as r^{-2}. Hence, the expected contribution $\Delta\Phi$ of a sphere with finite small radius R is finite and decreases linearly with R,

$$\langle \Delta\Phi \rangle_R \sim \int_0^R dr\ r^2\, r^{-2} = R.$$

However, the expected value of the square, and therefore the variance, diverges. Indeed,

$$\langle \Delta\Phi^2 \rangle_R \sim \int_0^R dr\ r^2 \left[r^{-2}\right]^2 = +\infty.$$

In other words, the variance of the fluence-at-a-point estimator does not exist[10]. As it stands, it is impossible to provide a meaningful estimation of the associated statistical uncertainty.

A possible remedy to the variance divergence at the origin is to define an exclusion sphere of radius R around the detection point. Virtual-photon contributions from interactions inside the sphere are ignored, thus effectively avoiding the divergence in the variance at the expense of underestimating the true fluence value. Notice that the error is (approximately) proportional to R, for small R. Note also that the exclusion sphere is unnecessary if the detection point D is in vacuum, since the number of interactions in a small sphere surrounding D is then identically zero and the divergence disappears.

In penEasy the radius R is a user-defined parameter, possibly null, which is the recommended value if, and only if, the detector is in vacuum. For any other material a finite radius should be introduced, preferably small compared with the mean free path. In practice, a trial and error process is advisable with exclusion spheres of decreasing radii to test the stability of the solution. Given the possible ocurrence of close-to-the-origin events, large fluctuations in some energy bins of the spectrum are to be expected from time to time. It should be mentioned that a number of alternative finite variance methods with no bias have been devised (Kalos 1963; Kalli and Cashwell 1977) to escape the divergence.

[10] It can be seen that the detection-forced fluence estimator converges as $N^{-1/3}$ with the number of histories N, instead of the usual $N^{-1/2}$ dependency (Kalos 1963).

Approximate contribution from the exclusion sphere

Photon interactions occuring inside the exclusion sphere do not contribute to the tally. To estimate the missing contribution an analytical approximation is used and the result, for each energy bin, is indicated separately in the tally report.

In order to provide a simple analytical formula for this contribution it is assumed that, (i) the material composition, and (ii) the photon fluence are constant inside the sphere. Condition (i) is satisfied by ensuring that the sphere is small enough not to intersect any interface separating different materials; therefore, the detection point must not be positioned exactly on an interface. One condition for (ii) to be valid is that the sphere radius R be small compared to the photon mean free path $1/\mu$ at all energies,

$$\mu R \ll 1,$$

so that the incoming photon fluence Φ_0 is not appreciably attenuated. Another condition is that Φ_0 be itself constant over the sphere surface, which fails to be true for emission points close to the surface. All in all, it seems clear that a relatively small sphere is preferable if assumptions (i) and (ii) are to be any good, although the accuracy of the estimator introduced below can only be determined by comparing results obtained with varying radii.

Consider Rayleigh scattering for concreteness. As discussed above, for each Rayleigh interaction in a volume element dV of the sphere a contribution to the fluence at the sphere center, where the detector D is located, is scored. This contribution is

$$\Delta\Phi_{\rm el}^{\rm (oneEvent)} = \frac{1}{\sigma_{\rm el}} \frac{d\sigma_{\rm el}}{d\Omega} \frac{{\rm e}^{-\mu R}}{r^2},$$

where μ is the total linear attenuation coefficient for the sphere material at the energy of the scattered photon, which in the elastic case coincides with E_0, the energy of the incoming fluence Φ_0. Furthermore, the number of Rayleigh events inside dV is

$$dI_{\rm el} = \Phi_0 \; \mu_{\rm el} \; dV$$

with $\mu_{\rm el}$ evaluated also at E_0. The contribution $d\Phi_{\rm el}$ to the fluence at D is the product of the two quantities,

$$\begin{aligned} d\Phi_{\rm el} &= \Phi_0 \; \frac{\mu_{\rm el}\, dV}{\sigma_{\rm el}\, r^2} \; \frac{d\sigma_{\rm el}}{d\Omega} \; {\rm e}^{-\mu r} = \\ &= \Phi_0 \; \mathcal{N} \; \frac{d\sigma_{\rm el}}{d\Omega} \; {\rm e}^{-\mu r} \; dr \; d\Omega \end{aligned}$$

where the volume element has been expressed in polar coordinates as

$$dV = r^2 \, dr \, d\Omega$$

and the relation

$$\mu_{\rm el} = \mathcal{N} \, \sigma_{\rm el}$$

has been used, with $\mathcal{N}$ equal to the number of molecules per unit volume.

Integrating over the sphere volume,

$$\Delta\Phi_{\rm el} = \Phi_0 \ \left(\frac{\sigma_{\rm el}}{\sigma}\right) \ \left(1 - {\rm e}^{-\mu R}\right).$$

Thus, for each contribution $\Phi_0(E_0)$ to the main (out-of-the-sphere) fluence counter another contribution $\Delta\Phi_{\rm el}$ from the sphere is tallied at the same energy E_0. Note that the sphere contribution does not depend on the shape of the differential cross section because all scattering angles produce the same scattered energy $E = E_0$.

Pair production can be analyzed in a similar manner. If positron annihilation is assumed to occur at rest, only virtual photons with energy $E = m_{\rm e}c^2 \approx 511$ keV need to be considered. Thus, for E_0 above the threshold, we have

$$d\Phi_{\rm pp} = \Phi_0 \ \mathcal{N} \ \frac{2d\sigma_{\rm pp}}{d\Omega} \ {\rm e}^{-\mu(E)\,r} \ dr \ d\Omega$$

where the factor 2 is introduced to account for the production of two annihilation photons per event. Integrating over the sphere,

$$\Delta\Phi_{\rm pp} = \Phi_0 \ \frac{2\,\sigma_{\rm pp}(E_0)}{\sigma(E)} \ \left(1 - {\rm e}^{-\mu(E)\,R}\right).$$

The energies (either E_0 or $E = m_{\rm e}c^2$) at which the quantities are evaluated have been explicitly stated.

When the energy E of the scattered photon depends on the scattering angle a continuum of energies needs to be considered. Thus, for Compton collisions

$$d\Phi_{\rm in} = \Phi_0 \ \mathcal{N} \ \frac{d\sigma_{\rm in}}{d\Omega} \ {\rm e}^{-\mu(E)\,r} \ dr \ d\Omega$$

and the integral over r can be carried out as before, but the solid angle $d\Omega$ has to be integrated over an interval such that scattered photons inside the subtended volume dV contribute to a single energy bin in the tally. For this purpose the relation

$$d\Omega \, \frac{d\sigma_{\rm in}(\hat{\mathbf{\Omega}}; E_0)}{d\Omega} = dE \, \frac{d\sigma_{\rm in}(E; E_0)}{dE}$$

can be used, where it has been made explicit that the cross section on the right hand side is differential in the energy E of the scattered photon.

Integrating over a single energy bin (all bins have the same width ΔE) centered at a given energy E and assuming that ΔE is small enough so that variations of $d\sigma_{\rm in}/dE$ and μ can be neglected in the interval, yields

$$\Delta\Phi_{\rm in}(E) = \Phi_0 \ \frac{1}{\sigma(E)} \ \frac{d\sigma_{\rm in}(E; E_0)}{dE} \ \left[1 - {\rm e}^{-\mu(E)\,R}\right] \ \Delta E.$$

Note that the contribution due to Compton events does depend on the angular distribution associated to the differential cross section. For the purpose of its calculation

penEasy takes the total cross section from the PENELOPE model but the angular distribution is approximated by the simpler Klein-Nishina formula, that is, the contribution actually scored is

$$\Delta\Phi_{\rm in}(E) = \Phi_0 \ \left[\frac{\sigma_{\rm in}(E_0)}{\sigma(E)}\right] \ \left[\frac{1}{\sigma_{\rm KN}(E_0)}\,\frac{d\sigma_{\rm KN}(E;E_0)}{dE}\right] \ \left[1-{\rm e}^{-\mu(E)\,R}\right] \ \Delta E,$$

where the subindex KN indicates the Klein-Nishina terms.

To save computing time, penEasy calculates the sphere contribution resulting from all interaction mechanisms by post-processing the spectrum after the simulation is done. Thus, for each energy bin centered at E_i ('i' being the index), the combined fluence contribution from all other bins E_j is added. Due to the linear relation between $\Delta\Phi(E_i)$ and the $\Phi_0(E_j)$'s, this contribution can be expressed as a matrix product. Obviously, only bins with $E_j \geq E_i$ (*i.e.*, $j \geq i$) need to be considered.

References

Badal, A. (2008). *Development of advanced geometric models and acceleration techniques for Monte Carlo simulation in Medical Physics*. Ph. D. thesis, Universitat Politècnica de Catalunya. Available at http://www.tdx.cat/TDX-0523108-095624.

Badal, A., I. Kyprianou, D. P. Banh, A. Badano, and J. Sempau (2009). penMesh—Monte Carlo Radiation Transport Simulation in a Triangle Mesh Geometry. *IEEE Transactions on Medical Imaging 28*, 1894–1901. Available from https://www.fda.gov/about-fda/cdrh-offices/penmesh-monte-carlo-simulation-radiation-transport-triangle-mesh-geometry.

Badal, A. and J. Sempau (2006). A package of Linux scripts for the parallelization of Monte Carlo simulations. *Comput. Phys. Commun. 175*, 440 – 450.

Baró, J., J. Sempau, J. M. Fernández-Varea, and F. Salvat (1995). 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.

Capote, R., R. Jeraj, C.-M. Ma, D. W. O. Rogers, F. Sánchez-Doblado, J. Sempau, J. Seuntjens, and J. V. Siebers (2006). Phase-Space Database for External Beam Radiotherapy. Technical Report INDC(NDS)-0484, International Atomic Energy Agency, Nuclear Data Section.

Freed, M., S. Park, and A. Badano (2010). A fast, angle-dependent, analytical model of CsI detector response for optimization of 3D x-ray breast imaging systems. *Med. Phys. 37 6*, 2593 – 2604.

García-Toraño, E., V. Peyres, M.-M. Bé, C. Dulieu, M.-C. Lépy, and F. Salvat (2017). Simulation of decay processes and radiation transport times in radioactivity measurements. *Nucl. Inst. Meth. B 396*, 43 – 49.

García-Toraño, E., V. Peyres, and F. Salvat (2019). PenNuc: Monte Carlo simulation of the decay of radionuclides. *Computer Physics Communications*. In press.

International Commission on Radiation Units and Measurements (2011). Fundamental Quantities and Units for Ionizing Radiation (Revised) — ICRU Report 85. *Journal of the ICRU 11*.

Kalli, H. J. and E. D. Cashwell (1977). Evaluation of Three Monte Carlo Estimation Schemes for Flux at a Point. Technical Report Report LA-6865-MS, Los Alamos Scientific Lab., NM, USA. Available at https://www.osti.gov/servlets/purl/7280869.

Kalos, M. H. (1963). On the estimation of flux at a point by Monte Carlo. *Nuc. Sci. and Eng. 16*, 111 – 117.

L'Ecuyer, P. (1988). Efficient and Portable Combined Random Number Generators. *Communications of the ACM 31*, 742 – 774.

Mehlhorn, W. (1968). On the polarization of characteristic X radiation. *Physics Letters 26A*, 166 – 167.

Salvat, F. (2019). *PENELOPE-2018: A Code System for Monte Carlo Simulation of Electron and Photon Transport*. OECD Nuclear Energy Agency. Available at http://www.oecd-nea.org.

Sempau, J., E. Acosta, J. Baró, J. M. Fernández-Varea, and F. Salvat (1997). An algorithm for Monte Carlo simulation of coupled electron-photon transport. *Nucl. Instrum. Meth. B 132*, 377 – 390.

Sempau, J. and P. Andreo (2006). Configuration of the electron transport algorithm of PENELOPE to simulate ion chambers. *Phys. Med. Biol. 51*, 3533 – 3548.

Sempau, J., A. Badal, and L. Brualla (2011). A PENELOPE-based system for the automated Monte Carlo simulation of clinacs and voxelized geometries—application to far-from-axis fields. *Med. Phys. 38*, 5887 – 5895. Available at http://dx.doi.org/10.1118/1.3643029.

Shultis, J. K. and R. E. Faw (2011). An MCNP primer. Technical report, Dept. of Mechanical and Nuclear Engineering, Kansas State University. Available at https://www.nucleonica.com/wiki/images/6/6b/MCNPprimer.pdf.