User's Guide to shower version 1.0, an EGS4 Interface

L. Emery

October 17, 2003

1 Introduction

Program shower is a SDDS-compliant interface program to the Monte Carlo electromagnetic shower program EGS4[1, 2]. The shower program consists of a main routine and two subroutines required by the EGS4 system, and other subroutines related to the geometry of the problem, all written in the C language. The EGS4 MORTRAN source code is not modified except for minor details [1] and is called as a subroutine.

In shower the EGS4 Monte Carlo simulation is controlled with input files describing the input particle beam and the geometry definition, and command line arguments. Program shower reads incident particles coordinates from an input date file, sets up FORTRAN data structures required by EGS4 subroutines howfar and ausgab, calls the EGS4 subroutine shower, and writes the output particle coordinates to files. The output particle coordinates files can serve as input to another shower run with a different geometry as part of a linked set of runs.

A regions summary file contains data related to each region defined in the geometry file, such as absorbed dose rate per unit input beam power.

While the geometry definition is the most difficult programming aspect in conventional EGS4, it is vastly simplified in shower because the geometry isn't hard-coded in a subroutine but rather defined in a separate geometry file with namelist type commands. In shower we restrict ourselves to defining regions of block-type or cylindrical geometries. A 3D grid defining all space for the simulation is internally generated from all the block or cylindrical definitions. It is felt that most simulations can be well approximated using these elementary shapes, however complex the problem really is.

2 Command line

Many options are available in the command line. To review the possible options, type "shower" at the shell prompt. The program returns

```
shower <SDDSinputfile> -geometry=<GeometryDefinitionFile> -root=<string>
 -summary[=<file>] [-defaultParticleType=<particleType>]
 [-seedNumber=<integer>] [-samples=<integer>]
 [-trajectories[=<file>]] [-vacuumStepSize=<distance-in-cm>] [-plotGeometry[=<file>]]
 [-outOfBounds[=<file>]] [-undefinedRegion[=<file>]]
 [-useInInput=<particleType>[,...]] [-keepInOutput=<particle-type>[,...]] [-verbose]
 [-splitting] [-describeGeometryInput] [-fixLowEnergyTransport=[{0|1}]]
```

[1] One may scan for comments in the MORTRAN code to view the changes. However none of the physics is modified from the original.

```
SDDSinputfile          input file of initial coordinates of incident particles;
                       output files produced by previous shower runs are valid input files;
-geometry              specifies file with geometry namelist commands defining materials;
-root                  string with which to construct default filenames;
-summary               SDDS file containing summary data for each region; default name
                       is <root>.summary where <root> is the value of the -root option;
-defaultParticleType   default particle type if not defined in input beam file;
<particleType>         valid values are "electrons", "positrons", or "photons";
-seedNumber            integer to initialize the random number generator;
-samples               number of times to repeat the tracking of input file particles;
-trajectories          file for storing shower trajectories; default name
                       is <root>.traj;
-vacuumStepSize        maximum step size in vacuum;
-plotGeometry          SDDS file containing line segments
                       defining solid regions edges;
-outOfBounds           SDDS file containing coordinates of out-of-bounds particles;
                       default name is <root>.outOfBounds;
-undefinedRegion       SDDS file containing coordinates of particles entering
                       undefined space; default name is <root>.undefinedRegion;
-useInInput            select which particle type or types use (read) in particle
                       coordinate input file;
-keepInOutput          select which particle type or types to keep (write) in particle
                       coordinate output files;
-splitting             enables splitting of particle for improved statistics
                       as defined in the geometry definitions;
-describeGeometryInput prints out the namelist commands for the geometry input file
-fixLowEnergyTransport flag for enabling(1)/disabling(0) the maximum limit on the fractional
                       energy loss per step defined for each material in the geometry
                       input file. The default value is 1
-verbose               spews verbiage
```

where square brackets enclose optional parameters.

3 Brief Description of Input and Output Files

Running `shower` requires three input files:

- **cross-section data:** Text data file originating from multiple `PEGS4` runs which provides the physics of the simulation.
- **input beam coordinates:** SDDS-format file containing initial coordinates, and energy of the particles. Other parameters, such as `PowerOfInitialBeam`, used to calculate absorbed dose rate, may be defined here.
- **geometry definition:** File containing namelist type commands defining the geometry of the problem, i.e. dimensions and material composition.

There are three types of output files:

- **output beam coordinates:** Coordinates of all particles crossing a bounding plane of a region defined in the geometry file. The file name is specified in region definition. The data may be used for quantitative analysis of a shower, or used as input file to another run of `shower`. Particles that go out-of-bounds or entering an undefined region of space, may be recorded in separate files defined in the command line.
- **shower trajectory:** Coordinates of trajectories of all particles produced in a `shower` run, including those that are created and eventually get absorbed.
- **summary:** Lists quantities related to each region defined, such as the total energy absorbed.
- **line segment geometry:** Contains coordinates of all line segments of the shapes defining the geometry of the problem. Most useful when overlayed on shower trajectory plots.

4 Input Files

The input file will be covered in more detail here.

4.1 Cross-section Data File

This file is required by the `EGS4` subroutines, and contains most of the physics of the simulation. The cross-section data is created by program `PEGS4`, the `EGS4` preprocessor. `PEGS4` has to be run once for each material type in an `EGS4` simulation. The material can be an element, compound or mixture. The resulting data file is an ascii text file that an `EGS4` subroutine will parse and store as cross-section data. One normally pre-calculates a number of materials and combine the individual cross-section files into a larger file. A file with several material types that have been used at APS is provided in the distribution.

Program `shower` looks for this file in the environment variable `EGS4_CROSSSECTION`. If `EGS4_CROSSSECTION` is undefined, the program looks for the file `fort.12` in the current directory. (The name `fort.12` comes from the conventional implementation of `EGS4`.)

4.2 Input Beam File

The file for the input beam is specified in the command line. This file may be created using an editor or the `SDDS` toolkit, or may be an output of previous `shower` runs. The data may even originate from other SDDS-compliant physics simulation programs. Converting from one type of data to another is relatively easy with `SDDS` toolkit programs.

A short introduction to the `SDDS` format is given here with specific definitions required in the `shower` input beam file.

An `SDDS`-format file consists of a header and zero or more pages of data represented as follows:

```
SDDS1
description-command
column-command
.
.
column-command
parameter-command
.
```

```
.
parameter-command
data-command
! end of header, beginning of data page 1
parameter data
tabular data
! beginning of data page 2
parameter data
tabular data
.
.
```

The header contains namelist commands describing the structure of the data pages using parameter and column commands. The tabular data consists zero or more rows of n columns, where n is the number of columns defined in the header. Each column of data column must appear in the same order as the definitions of the header. The description command is optional. Another data type, array, is not mentioned here since it is not used in `shower`.

The first line of the file must be `SDDS1`, indicating that the file is an `SDDS` version 1 file. The description command follows, for example:

```
&description text = "Coordinates of particles entering ‘‘concrete’’ region",
 contents = "Particle coordinates", &end
```

The `text` and `contents` field are intended to be a general description to aid a user in identifying the file. This particular description command came from a output file created by a `shower` simulation with a region labeled "concrete". The `contents` field is intended to represent a formal way to identify the type of file. The quantities in the description don't have any particualr significance to `shower` but may be used in other `SDDS` toolkit programs, such as `sddsplot`.

In the input beam file `shower` recognizes the columns `x`, `y`, `z`, `u`, `v`, `w`, and `Energy`. The first three are the cartesian coordinates of incident particle with units of meters. The next three are direction cosines with no units. The last coordinate is the total energy (kinetic plus rest mass), in MeV. Note that only the `Energy` column need not be defined at all, as the rest of the coordinates default to zero (except for `w` which defaults to 1) when the simulation starts. Additional columns may exist in this file, but will be ignored by `shower`. An example of a column definition is

```
&column
 name = "Energy",  symbol = "Energy",  units = "MeV", description = "Energy",
 type=double,
&end
```

The minimum set of fields for a valid column definition in general consists of `name` and `type`. For `shower`, an additional requirement is the specification of the units, "`m`" for coordinates and "`MeV`" for `Energy`. All coordinate data types should be `double`. Other columns could be defined in the input file for informational purposes, but they will be ignored by `shower`.

Parameters are quantities assigned for each data page. For the input beam file the parameter `Type` indicates whether the page of data correspond to electrons, positrons or photons. The definition of parameter `Type` is

```
&parameter name = "Type", symbol = "Type",
 description = "Particle type", type = "string",
&end
```

Note that while the definition of the parameter belong in the header, the actual value of the parameter appears at the start of each page. Program shower will recognize string values equal to `electrons`, `positrons`, and `photons`. The string data may appear in quotes.

If the parameter `Type` isn't defined in the input file, then `shower` uses the value given for the command line option `-defaultParticleType`. If no value was assigned to this option, then `shower` gives an error message, and aborts.

The parameters listed below are all optional. Most of them are generated in output files. They do not take part in the Monte-Carlo simulation themselves, but in combination of the simulation results these parameters are used to calculate absorbed doses.

- `PowerOfInitialBeam` — Double. Units of W. Default value of 1 W. This is defined in the input beam file by the user. This parameter is passed to all output files. If linked `shower` runs are used to transport radiation through different parts of a complex simulation, this parameter serves as a mechanism to pass the information required to calculate a radiation dose in the final run.
- `Generation` — Long. Default value of 0. Its value in an output beam file is one more than its value in an input beam file. For a primary beam, its value should be set to zero. Therefore, in an output beam file, its value is the number of `shower` runs that have been done starting from the primary beam. This parameter is involved in defining parameter `TotalIncidentEnergyOfFirstRun`.
- `CumulativeSamples` — Long. Default value of 1. This is the cumulative number of samples that precedes the input file. In an output file, its value is multiplied by the number of samples in the current `shower` run. This parameter is used in the calculation of the radiation absorbed dose.
- `TotalIncidentEnergyOfFirstRun` — Double. No default values. This parameter is the total energy of all particle used in the primary beam run ("generation" zero). Its value is generated in the output files of the generation zero run. This quantity is used in calculating the radiation absorbed dose.

For the `&data` command, one sets `mode` equal to `ascii`, and `no_row_counts` flag to 1. Here is an example of a minimal input file used as the primary beam of a Monte Carlo calculation of radiation dose at the APS. The beam power at the safety envelope is 308 W.

```
SDDS1
&description
 text="One-particle 7000 GeV positron beam",
 contents="Particle coordinates" &end
&parameter
 name = "PowerOfInitialBeam" description="Initial beam power",
 symbol = "P$b0$n", type="double" units="W" &end
&parameter
 name = "Type" type="string"  &end
&column
 name="E" units="MeV" type=double &end
&data
 mode = "ascii" no_row_counts=1 &end
! page number 1
```

```
! Initial beam power of 308 W
308
! parameter Type
positrons
! start of tabular data
7000
```

Here, only the energy column is defined. The other column quantities are assumed to be zero. The parameter `PowerOfInitialBeam` is defined with value 308 W, which happens to be relevant for safety simulations at APS. The particle type is "positrons". Only one particle of 7 GeV is defined.

Note that command line option `-chooseInInputfiles=particleType[,...]` can be used to select particles of one or more types in the input files, and discard the particles of the unnamed types. Thus if one had `-chooseInInputfiles=electrons` for the input file above, then no particle would be tracked.

4.3 Geometry File

The required geometry file is created by the user. Three namelist commands are available, `&initial_coordinates`, `&block`, and `&cylinder`. Since the regions can only be of block type or cylindrical type, the commands `&block` and `&cylinder` can't be mixed in the same geometry file. The fields for the first command are:

```
&initial_coordinates
   double x_cm = 0.0
   double y_cm = 0.0
   double z_cm = 0.0
   double xp = 0.0
   double yp = 0.0
&end
```

- Defaults are shown on the right hand side of "`=`". The word "`double`" in the above syntax definition isn't part of the actual command and should be omitted in a command. It is shown here only to specify the data ytpe.
- `x_cm,y_cm,z_cm` — Amount by which to change the coordinates of all particles defined in all input files. Units are in centimeters.
- `xp,yp` — Amount by which to change the slope with respect to the z-axis of all particles defined in all input files.

The `&block` command defines a block-like region of material:

```
&block
   STRING label = NULL
   STRING material = NULL
   double x_lower_cm = 0.0
   double x_upper_cm = 0.0
   double y_lower_cm = 0.0
   double y_upper_cm = 0.0
   double z_lower_cm = 0.0
```

```
   double z_upper_cm = 0.0
   double Bx = 0.0;
   double By = 0.0;
   double Bz = 0.0;
   double Ex = 0.0;
   double Ey = 0.0;
   double Ez = 0.0;
   long x_slices = 1;
   long y_slices = 1;
   long z_slices = 1;
   STRING outputfile = NULL;
   double photonEnergyCutoff = 0.0;
   double leptonEnergyCutoff = 0.0;
   long split = 1;
   double minimumWeight = 0.0;
   double maximumDeltaE = 0.02;
&end
```

- `label` — An optional label identifying the region. The label value will appear in the output file.

- `material` — A string specifying the material in the region. This material can be an element, a mixture, or a compound, and must be defined in the cross-section file.

  The `material` string value is converted to uppercase and passed to `EGS4` for identification of data arrays in the cross-section data file defined by environment variable `EGS4_CROSSSECTION` (see section 4.1). If this cross-section file exists, and contains the required information on the specified element, mixture, or compound, the relevant data is read into `EGS4` subroutine arrays. Otherwise, an error message is printed out, and execution stops.

  In order to track through empty space, the empty space must be defined as a region of pre-set material "`vacuum`". Otherwise, a particle entering an undefined region of space will be discarded, and optionally recorded in a coordinate file (see command line arguments).

- `{x,y,z}_lower_cm,{x,y,z}_upper_cm` — Boundaries of the block region in units of cm (the units used here is to be consistent with the units used by `EGS4`). If a later `&block` definition overlaps with the volume of one defined earlier, then the first block takes precedence. Smaller objects such as targets where the shower process usually starts should be defined first so that they can be nested inside larger regions.

- `Bx,By,Bz` — Uniform magnetic field in entire region. Only one component of B is allowed in one region. Each tracking segment in a region with a magnetic field is converted into an arc of a circle approximated by shorter line segments joined by a small angle, with the possibility of interaction with matter at each vertex.

- `Ex,Ey,Ez` — Electric field in region. Not implemented yet.

- `{x,y,z}_slices` — Number of equal divisions of the region along a particular direction. Each sub-region becomes a new region with the same label, material, magnetic field. This usually doesn't affect the Monte Carlo simulation. However, for regions with dimensions smaller than the mean free path of a particle, the larger number of steps may improve the simulation of Coulomb scattering. Also, in the regions summary file each subdivision of a region is considered as a separate region with the same material and label values. This is a convenient way to determine absorbed dose as a function of position.

The slice option does not work with the `outputfile` specification. Therefore slices should be left equal to 1 if an output file is specified.

- `outputfile` — Creates an ouput file which records coordinates of particles entering the region. Once in the output region, the particles are discarded from further tracking. Therefore the region with the output file is often the "last" region of the geometry.
- `photonEnergyCutoff, leptonEnergyCutoff` — Modifies the energy cut-off of photon and lepton transport for the particular region. The energy cut-off is defined as the minimum energy at which the particle can no longer move forward and is absorbed on the spot. The cut-off energy cannot be made smaller than the default value found in the cross-section data file. Using a larger value than the default values may reduce simulation time at the expense of reduced accuracy. The user must make his own determination.
- `split` — When greater than one, allows particle splitting when particle enters the region, in order to improve statistics for that region. The number of particles is increased by the value of `split`. At the same time the weighting of the particles are reduced by the same factor, thus unaffecting the energy deposited in the region.
- `minimumWeight` — Specifies the minimum weighting allowed when particle splitting is enabled. This prevents a runaway situation for some geometries where potentially a very large number of particles are created from a single particle.
- `maximumDeltaE` — Specifies the maximum relative energy loss a particle may suffer when it is tracked through a single step. This parameter forces the reduction of the path length such that the relative energy loss is approximately equal to the value given.

The `&cylinder` command is similar to the `&block` command:

```
&cylinder
   STRING label = NULL
   STRING material = NULL
   double r_lower_cm = 0.0
   double r_upper_cm = 0.0
   double z_lower_cm = 0.0
   double z_upper_cm = 0.0
   double Bx = 0.0
   double By = 0.0;
   double Bz = 0.0;
   double Ex = 0.0;
   double Ey = 0.0;
   double Ez = 0.0;
   long r_slices = 1
   long z_slices = 1
   STRING outputfile = NULL
   double photonEnergyCutoff = 0.0;
   double leptonEnergyCutoff = 0.0;
```

```
   long split = 1;
   double minimumWeight = 0.0;
   double maximumDeltaE = 0.02;
&end
```

The only difference with the `&block` command is the absence of the dimension variables in y, and the replacement of the variables in x with variables in r. The cylinder defined are concentric and are centered on the z-axis.

The `&block` and `&cylinder` commands can't be mixed in the same geometry file. If the first material definition command is of one type, then the rest must be of the same type.

5 Output Files

The output files will be covered here. Since all output files are in `SDDS` format, the type of quantities contained in them can always be determined by typing:

```
sddsquery <file>.
```

5.1 Output Particle Files

When a particle enters a region with with `outputfile` speficied, the full coordinates are recorded into this file, and the particle is discarded. It is best to define the output region such that most particles entering this region passes through only one side. This will make the processing less confusing.

The option `-discardInOutputFiles=particleType[,...]` in the command line is used to discard particles of one or more types in the output files, which is useful in reducing the size of the files if a particular particle type is not needed in the results. This option applies to all output files produced in a `shower` run.

The file produced by the `-outOfBounds`, and `-undefinedRegion` contain the full coordinates of particles going out of bounds, or going into a volume within bounds but not defined. Usually these particles aren't interesting. These files can be used for debugging purposes. For instance, one could check whether total energy is conserved.

Beam data coordinates are buffered in internal data pages of 500 rows in length before being written to the output file. Thus one may find interleaved data pages for electrons, positrons, and photons. When processing data one should realize that the output files are split into pages. For instance before histogramming, say the energy coordinate, the command `sddscombine` should be used on the output file to merge all data pages into one.

5.2 Shower Trajectory File

This file is generated by the `-trajectories` option. If the name of the trajectories file is unspecified, then one is formed automatically by adding `.traj` to the string given by `-root`.

The shower file contains the x, y, z coordinates and the energy of each particles produced in the shower at each tracking step. The column names are `x`, `y`, `z`, and `Energy`, respectively. The shower file may grow very large depending on the number of samples requested, the energy of the initial particle, and depending on whether the shower is fully developed.

The shower output file may be plotted directly with the `sddsplot` toolkit plotting program. For instance,

```
sddsplot -col=x,z <file> -grapic=dots
```

plots the x, z projection of the shower trajectories in `<file>` as dots. Since no information on the geometry of the problem is contained in these output files it may be necessary to overlap the shower product trajectories with the outline of the regions. The outline can be generated from the geometry plot file produced by `shower`, as explained in the next section.

5.3 Geometry Plot File

This file is generated by the `-plotGeometry` option. If the name of the geometry plot file is unspecified, then one is formed by adding `.line_segments` to the string given by `-root`.

This file contains the end point coordinates of the segments forming the edges of the solids defined in the geometry file. The columns of the `SDDS` file are `xFrom`, `yFrom`, `zFrom`, `xTo`, `yTo`, and `zTo`. If two regions with the same material are touching, then the common edge, if any, is not included.

The commandline option `-vacuumStep=<distanceInCM>` allows one to change the default distance by which particles are tracked at every step in vacuum regions. In vacuum, all particles have infinite mean-free path. For cpu efficiency, the default is made a very large number such as 10 m. To create coordinates inside vacuum regions, a distance value smaller than the dimension of the regions must be given.

5.4 Summary File

A regions summary file is always created. The filename can be specified by the argument of the `-summary` option or formed automatically by adding `.summary` to the value of the `-root` argument. The columns in this file contain information pertaining to regions, such as the label, the geometrical center of each region (calculated from the limits in the definition command), the volume (including any subtractions due to overlap with previously defined regions), the energy deposited by each type of particles, the fraction of the total beam energy deposited, and, if the beam power was specified, the radiation absorbed dose rate.

6 Postprocessing the Output Files

Since the output coordinate files and the summary file are in `SDDS` format, `SDDS` toolkit programs are available for post-processing the results. The programs are listed in references [3]. Programs that may be applicable with `shower` outputs files are:

- `sddshist, sdds2hist` — Histogram program
- `sddsquery` — Lists the columns and parameter definitions in a file.
- `sddsprocess` — General processing program.
- `sddscombine` — Combines `SDDS` files, and optionally merges data pages into one.
- `sddsconvert` — Convert between ascii and binary, and other filtering functions.
- `sddscontour` — Contour and density plotting program

For instance, the program `sddsprocess` operating on the summary file can give the maximum dose rate in any region with material value matching the string "human".

7 Acknowledgement

The `EGS4` code system was obtained from A. Bielajew of NRC of Canada.

References

[1] W. R. Nelson, Hideo Hirayama, and D. W. O. Rogers, "The EGS4 Code System," SLAC 265, SLAC, December 1985.

[2] A. F. Bielajew, "egs4unix ver. 3.0." Implementation of EGS4 on Unix, private communication.

[3] M. Borland, "User's Guide for SDDS Toolkit Version 1.8"