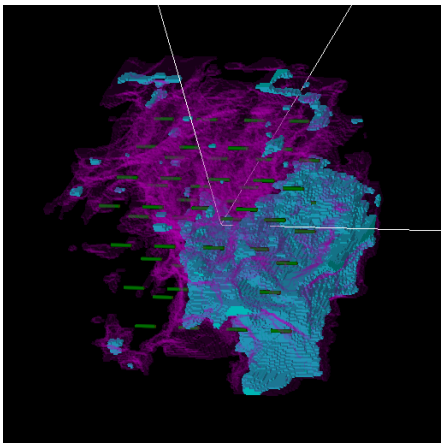
User manual for `egs_brachy`

A versatile and fast EGSnrc application for brachytherapy

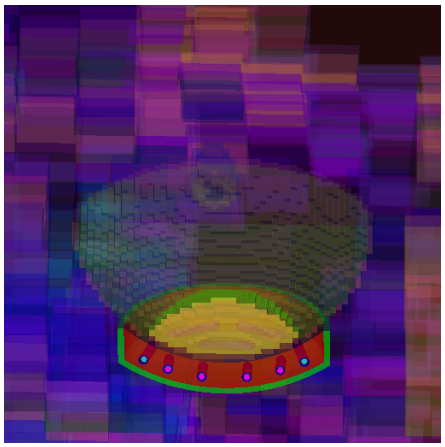
Rowan M. Thomson, Randle E. P. Taylor, Marc J. P. Chamberland,
and D. W. O. Rogers
Carleton Laboratory for Radiotherapy Physics, Department of Physics,
Carleton University, Ottawa, Ontario, K1S 5B6, Canada

Last edited 2017/09/18 at 14:19:56

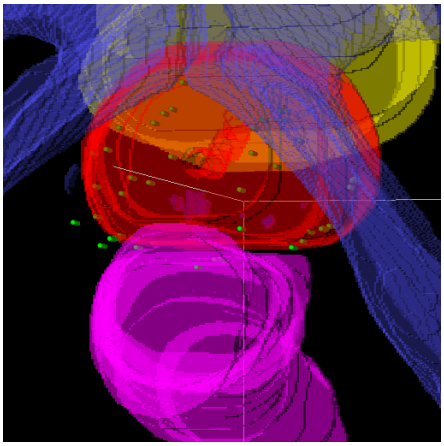
Report CLRP-17-02



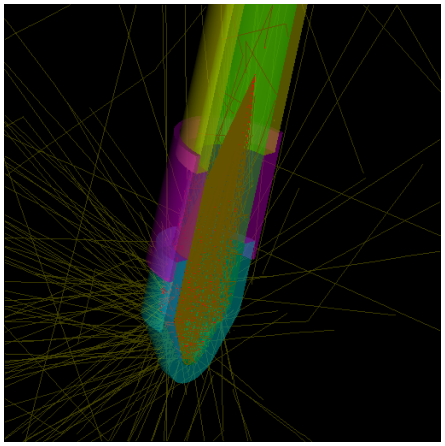
^{103}Pd breast implant



^{125}I eye plaque



^{125}I prostate implant



Electronic brachytherapy

Contents

1. Introduction

egs_brachy is a versatile and fast EGSnrc application designed for brachytherapy, employing the egs++ library for modelling geometries and sources, developed by members of the Carleton Laboratory for Radiotherapy Physics (CLRP) within the Department of Physics at Carleton University. The current document serves as a user manual for egs_brachy to complement the original 2016 egs_brachy paper [1] which provides a general overview of the code, details on egs_brachy benchmarking and characterization of simulation efficiency, and some sample calculation times for clinical scenarios. This user manual frequently refers to the html technical reference documentation (which accompanies the egs_brachy code distribution or which can be found on-line at https://clrp-code.github.io/egs_brachy/index.html or the pdf version of the same technical reference manual (>500 pages) at https://clrp-code.github.io/egs_brachy/egs_brachy_technical_manual.pdf) for details regarding input file specifications etc.

An overview of this user manual is as follows: Section 2. describes egs_brachy licence and copyright. Installation of egs_brachy is described in section 3., either with a new EGSnrc installation (simplest – section 3.1.) or when EGSnrc is already present (section 3.2.), as well as updating egs_brachy (section 3.3.). Running egs_brachy is described in section 4., followed by an overview of egs_brachy input and output files in sections 5. and 6., respectively. A more detailed description of egs_brachy, to complement our initial paper [1], appears in section 7., including radiation transport modelling (section 7.1.), geometry and source modelling (section 7.2.), run modes (section 7.3.), scoring options (section 7.4.; includes a description in section 7.4.2. of the calculation of absolute doses from doses output by egs_brachy), and features to enhance simulation efficiency (section 7.5.). Data and geometries distributed with egs_brachy are described in sections 8. and 9., respectively. Section 10. provides an overview of a suite of example input files distributed with egs_brachy, and these provide a good starting point for egs_brachy users. Finally, section 11. provides a collection of helpful hints for egs_brachy simulations.

The initial release of egs_brachy is v2017.09.15 beta. We will continue to update and maintain egs_brachy, as well as this manual and the html documentation. If you are familiar with git, bug fixes and additional features are welcomed via pull requests on GitHub. Please refer to the EGSnrc Wiki Developers section at https://github.com/nrc-cnrc/EGSnrc/wiki for general recommendations about making pull requests.

Links shown in blue in the pdf file are linked and you will be taken to the link (either a url or section or whatever). Text in magenta is code and only shown to draw attention to it.

When egs_brachy is used for publications, please cite our paper [1]: M. J. P. Chamberland, R. E. P. Taylor, D. W. O. Rogers, and R. M. Thomson, egs_brachy: a versatile and fast Monte Carlo code for brachytherapy, Phys. Med. Biol. **61**, 8214–8231 (2016).

2. Licence

The egs_brachy code (all pieces of code associated with the egs_brachy code system) is copyrighted Rowan Thomson, Dave Rogers, Randle Taylor, and Marc Chamberland. egs_brachy is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose.

egs_brachy is distributed as free software according to the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option), any later version. See the GNU Affero General Public License for more details, available at http://www.gnu.org/licenses/.

The EGSnrc Code System is needed for egs_brachy, and is distributed at https://github.com/nrc-cnrc/EGSnrc by NRC independently of egs_brachy under its own separate licence, however it too follows the GNU Affero General Public License as published by the Free Software Foundation.

3. Installation

3.1. Installing egs_brachy with a new EGSnrc install

Installing and running egs_brachy currently requires the CLRP fork of the EGSnrc source code due to the inclusion of additional egs++ geometries and sources. The CLRP fork of the EGSnrc source code is based off the develop branch of EGSnrc. This fork may or may not represent the current develop branch but depends on when CLRP last updated the fork. The goal is eventually to integrate these CLRP developments directly into the EGSnrc distribution.

Even if you already have EGSnrc installed, it is more straightforward to do a fresh install. This will not affect your old install, but to switch between the different versions (if you need to) requires changing back your EGS_CONFIG, HEN_HOUSE, EGS_HOME environment variables which will be set during this installation.

To check that you have git installed on your system you can run the following command in a terminal:

```
git --version
```

If you do not have git installed, you can download it from https://git-scm.com or install it using your operating systems package manager.

Following these instructions, a version of the EGSnrc system with egs_brachy included will be installed in a new subdirectory created on whatever directory you start from ('start_directory' indicated below). It is EGSnrc_with_egs_brachy and can be created on your home directory. To start installation:

```
cd start_directory
git clone https://github.com:clrp-code/EGSnrc_with_egs_brachy.git
```

at which point you should see something like:

```
 Cloning into 'EGSnrc_with_egs_brachy'...
remote:  Counting objects:  8812, done.
remote:  Compressing objects:  100% (220/220), done.
remote:  Total 8812 (delta 264), reused 313 (delta 211), pack-reused 8380
Receiving objects:  100% (8812/8812), 32.38 MiB | 2.33 MiB/s, done.
Resolving deltas:  100% (5391/5391), done.
Checking connectivity...  done.
```

Once this is complete (there was some pause after the 'done.' on a Carleton system) your system has the EGSnrc source codes. Change to the directory containing the source code:

```
cd EGSnrc_with_egs_brachy
```

Now you must get the CLRP version of EGSnrc which includes some extra libraries required for `egs_brachy`. These are on the `'egs_brachy'` branch of this repository. Check out that branch by issuing:

```
git checkout egs_brachy
```

which should result in output like:

```
Checking out files:  100% (907/907), done.
Branch egs_brachy set up to track remote branch egs_brachy from origin.
Switched to a new branch 'egs_brachy'
```

To download the `egs_brachy` source code enter:

```
git submodule update --init --recursive
```

which should result in output like this:

```
Submodule 'HEN_HOUSE/user_codes/egs_brachy' (git@github.com:randlet
    /egs_brachy.git) registered for path 'HEN_HOUSE/user_codes/egs_brachy'
Cloning into 'HEN_HOUSE/user_codes/egs_brachy'...
remote:  Counting objects:  3788, done.
remote:  Compressing objects:  100% (254/254), done.
remote:  Total 3788 (delta 548), reused 412 (delta 351), pack-reused 3146
Receiving objects:  100% (3788/3788), 24.03 MiB | 2.54 MiB/s, done.
Resolving deltas:  100% (3061/3061), done.
Submodule path 'HEN_HOUSE/user_codes/egs_brachy':  checked out
            '89ad245ccde1149143bc73025454a1c91ff2d85a'
```

The final step is to configure EGSnrc by using the configuration shell script. To do this you must establish the values of 3 environment variables as empty. If using the `bash` shell, this requires:

```
unset HEN_HOUSE    EGS_HOME    EGS_CONFIG
```

or if using the `tcsh` shell:

```
unsetenv HEN_HOUSE    EGS_HOME    EGS_CONFIG
```

From the `EGSnrc_with_egs_brachy` directory, issue the command

```
./HEN_HOUSE/scripts/configure
```

and follow the instructions given (default values should work). You will need to provide the complete path to the directory where you want to have your applications (*i.e.,* `$EGS_HOME`) and something like `$HOME/egsnrc` is convenient, but anything can be used.

Note the final instructions in the `finalize_egs_foruser` script which is an option at the end of the `configure` script. These require that the 3 environment variables set above

to being empty ($EGS_HOME etc) must now be properly set, either in your .cshrc chain if using tcsh or csh shells, or in your .profile if using bash shell.

With this done, you should have everything required to run egs_brachy (or any other EGSnrc application). If you have executed the finalize_egs_foruser script as part of the configure script, then the following is not needed. But if not, or if you have just updated the system, one must copy the egs_brachy files from the $HEN_HOUSE to your $EGS_HOME area.

```
cd $EGS_HOME
mkdir -p egs_brachy
cd egs_brachy
cp -pr $HEN_HOUSE/user_codes/egs_brachy/egs_brachy/* .      #note final .
```

To compile the egs_brachy application:

```
make
```

After compiling egs_brachy, if you have python 2.7 or 3.4+ installed, you can run an extensive set of tests by:

```
cd $EGS_HOME/egs_brachy
python run_tests.py     or equivalently     make test
```

These take several minutes and test many aspects of the code. Note that the timing comparisons are highly variable/machine dependent and so one can safely ignore Timing: FAIL messages but Results: FAIL messages must be investigated.

3.2. Installing when EGSnrc already present

If the prior installation of EGSnrc was done using the zipped archive, then one must essentially install a new version using git. But if you already have a recent EGSnrc git install of the develop branch (or even the master branch), then the following should work:

```
cd $HEN_HOUSE
git remote add clrp git@github.com:clrp-code/EGSnrc_with_egs_brachy.git
git fetch clrp
```

and then follow the instructions in the previous section starting at

```
git checkout egs_brachy
```

3.3. Updating egs_brachy when it has been installed previously

On occasion you may want to update to the latest version of egs_brachy . The following will update files on your $HEN_HOUSE without touching the files on your $EGS_HOME/egs_brachy area. The assumption is that you have not changed anything on the $HEN_HOUSE. Unless you are comfortable with git it is recommended that you not change any files in the $HEN_HOUSE area to ensure you are able to easily update to the latest official CLRP version of EGSnrc and egs_brachy.

The commands up to the first make command update the EGSnrc part of the system including extensions to the egs++ classes (egs_autoenvelope, egs_glib etc).

```
cd $HEN_HOUSE
cd ..              #assumes HEN_HOUSE is just below EGSnrc_with_egs_brachy
git status         #reports changes made on your EGSnrc system
                   #please commit them or revert before continuing
```

and

```
git checkout egs_brachy     #make sure on egs_brachy branch
git pull origin egs_brachy  #gets latest version of the CLRP EGSnrc fork
git log                     #this step not essential but informative
                            #reports changes on GitHub/CLRP repository
                            #for EGSnrc system since previous update
cd $HEN_HOUSE/egs++
make                        #to compile any changes
```

Next the egs_brachy files need to be updated.

```
cd $HEN_HOUSE/user_codes/egs_brachy/
git status               #reports changes on users egs_brachy repository
                         #please commit them or revert before continuing
git pull origin master   #gets latest version of egs_brachy code
git log                  #reports changes on GitHub/CLRP repository
                         #for egs_brachy since previous update
```

The following command will overwrite any changes you have made to files anywhere on your $EGS_HOME/egs_brachy area. If you want to save them, rename those files now. This command is also tedious because you must individually allow files to be overwritten. Remove the -i to let overwriting occur silently, but be careful. To copy the updated files from the $HEN_HOUSE to your $EGS_HOME/egs_brachy area and compile them:

```
cd $EGS_HOME/egs_brachy
cp -pri $HEN_HOUSE/user_codes/egs_brachy/egs_brachy/* .    #Note final dot
                    # p means preserve the time stamp
                    # r means recursively descend the subdirectories
                    # i means ask when old files are to be overwritten
make
```

If you want to confirm everything is working, you may run the test suite again:

```
make test
```

4. Running egs_brachy

Once EGSnrc and egs_brachy are fully installed, specific simulations can be run as follows, assuming that an appropriate input file, e.g., my_input.egsinp is available on $EGS_HOME/egs_brachy. The assumption is that the .egsinp file is designed to run in pegsless mode.

```
egs_brachy -i my_input
```

or

```
ex egs_brachy my_input pegsless
```

To capture the extensive output to the terminal in either of the above cases, at the end of the command pipe the output to a file by adding > my_input.egslog.

egs_brachy runs in parallel in batch mode but only if a batch submission process is available and set up (controlled via $HEN_HOUSE/scripts/batch_options.$EGS_BATCH_SYSTEM). The user must define EGS_BATCH_SYSTEM as appropriate for their system. There are 5 examples on $HEN_HOUSE/scripts/). Once this is set up, to run in batch use:

```
exb egs_brachy my_input pegsless
```

If the input file is designed to be run using a PEGS4 datafile, then any one of the following methods work:

```
egs_brachy -i my_input -p pegsfile        #interactive
ex egs_brachy my_input pegsfile           #interactive
exb egs_brachy my_input pegsfile          #batch
```

5. Brief description of egs_brachy input files

This section presents a brief overview of the different sections of the input files required by egs_brachy. For egs_brachy details see the on-line html reference manual at https://clrp-code.github.io/egs_brachy/index.html#usage.
For general information related to egs++, see its manual[2] available on-line at http://nrc-cnrc.github.io/EGSnrc/doc/pirs898/index.html .

For convenience, in the following the input filename is considered to be filename.egsinp since the outputs(see below) are linked to filename.

egs_brachy input files have a similar look and feel to many other egs++ applications. The following describes the different input blocks of the input file which are typically required. In all cases the input blocks start and end with :start and:stop delimiter statements *e.g.,*

```
:start some input:
  my_inputs (one or more lines)
:stop some input:
```

The individual parameters are generally input using input keys with format

```
some variable name = input value(s)
```

If there is more than one input value, they are separated by spaces or commas and if they continue on the next line, there must be a comma at the end of the line to be continued. Any text on a line after a # symbol is a comment.

The order of the input blocks is generally unimportant. However, certain nested input blocks can appear multiple times within another relevant input block. For example, a :start media input: nested input block may appear multiple times within different :start geometry: input blocks and similarly multiple :start geometry: nested input blocks can appear within the :start geometry definition: input block.

5.1. run control

The :start run mode: block of the input file specifies how many histories to run (ncase) and optionally, if needed, nbatch, nchunk, egsdat file format, geometry error limit. nbatch, nchunk are discussed in section 11.3., p 33.

5.2. run mode

The :start run mode: block specifies which of the 3 possible run modes are to be used: normal, superposition, volume correction only. These are discussed in section 7.3. on p 17.

5.3. media definition

It is generally expected that users will run egs_brachy in pegsless mode and this requires a :start media definition: input block which inputs the values of AE,UE,AP and UP. To facilitate the use of pegsless mode, many materials are defined in the material.dat file distributed with the code (see section 8.1. p 23 and a list of available materials in Table 2). The material data file = input key should point to the local copy of the material.dat file.

In contrast, to run using a .pegs4dat file as input, submit the job using one of the methods shown in section 4. using the pegsfile input. In this mode, no :start media definition: input block is needed but the media names used must match those in the .pegs4dat file.

5.4. geometry definition

Section 7.2. has a general discussion of geometry modelling in egs_brachy. For details about the inputs for specific geometry classes, refer to the EGSnrc C++ class library Report PIRS-898 (2017)[2] available at http://nrc-cnrc.github.io/EGSnrc/doc/pirs898/index.html The :start geometry definition: block can be the most complex part of the input file for egs_brachy. It must define the following three egs_brachy specific input keys that are required for the geometry definition input block (in addition to the standard egs++ simulation geometry key):

source geometries = this key specifies which geometries define the actual brachytherapy source object. This may be a single geometry name (*e.g.*, when using the egs_glib shim to load one of the seed geometries suppled with egs_brachy, see section 9.1.) or a series of sub-geometry definitions used to compose a single source geometry when defining geometries inline.

phantom geometries = this tells egs_brachy which geometries to score dose in (1 or more phantom geometries are required). Currently 3 geometry types are allowed as phantoms(and many pre-defined phantoms are supplied, see section 9.2.):

EGS_XYZGeometry (library egs_ndgeometry)
EGS_RZGeometry (library egs_rz)
EGS_cSpheres (library egs_spheres)

source envelope geometry: this input is only required for superposition run mode and must name the EGS_ASwitchedEnvelope geometry that contains the sources. Note that the output .3ddose files associated with these libraries have formats described in section 6.2..

In general the :start geometry definition: input block consists of a series of :start geometry: nested blocks. Individual components of the geometry can be specified in separate :start geometry: blocks and then these can be combined in a further :start geometry: block using composite geometries (*e.g.,* egs_genvelope).

egs_brachy has extended the egs++ package with the egs_glib geometry class which allows the user to insert the contents of an externally defined geometry into an input file, thereby allowing *e.g.,* models for seeds to be distributed and used in multiple input files.

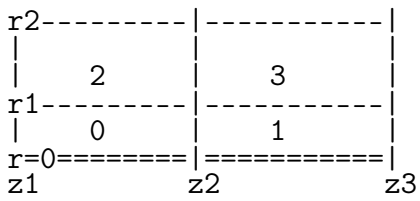
There are many geometry components which are already defined and found in seed_name.geom or phantom_name.geom files on individual subdirectories under $EGS_HOME/egs_brachy/lib/geometry. Examples of such supplied geometries are sources/OncoSeed_6711.geom, phantoms/2.0cmx2.0cmx2.0cm_1mm_xyz_water.geom. For more discussion of the pre-defined and distributed geometries see section 9. These components can have their own :start geometry definition: blocks which are then nested inside a :start geometry: block. These components can be quite complex and hence one should always use as many pre-defined geometries as possible, and also develop a local group of geometry files, either for re-use or to share with the user community. These predefined geometries use the materials from the material.dat file so it is important to use the pegsless option or ensure that all the materials used are defined in the .pegs4dat file being used..

A typical :start geometry: block will have a name = input key (so it can be referenced elsewhere in the input file) and a library = input key which specifies which type of geometry block this will be, *e.g.,* egs_spheres, egs_cylinders, egs_planes, egs_cones, egs_box, egs_rz and then a series of input keys specific to each library class. Users are urged to carefully check that geometry names, including the ones defined in external files, are unique because there is currently no check in egs++ if geometries are unique (and thus unexpected results may occur if the same name is used for two different geometries).

Within many :start geometry: blocks there will be a further nested block, :start media input: which is used to define the media in each of the local regions. The first input key, media = is used to define, in order from 0 to as many as needed, the materials for this geometry. This is then followed by as many set medium = input keys as needed. There are two input formats possible:
(i) set medium = ireg, imed means ireg is the local region which is assigned medium imed;
(ii) set medium = ireg1, ireg2, imed assigns medium imed to all regions between ireg1 and ireg2.
If there is no explicit assignment for any region, it is then assigned medium 0, *i.e.,* if there is no set medium = input key, every region is assigned medium 0 (the first one listed). (Note

that any statement `set medium = 0 0` is redundant but harmless.) Final region numbers depend on the overall geometry and how the different elements are combined. It is advisable to check things are as expected using the egs_view code.

As an aside: for the egs_rz geometries, the region numbering starts with regions 0 to `nz-1` down the central axis of the cylinder, and `nz` to `2nz-1` down the next radial ring, etc. Note that for `nz` depth regions one needs to specify `nz+1` depths but only `nr` for the radial regions since the assumption is that `r = 0` is included.. For example, for `nz=2,nr=2`:



A phantom in the `egsphant` file format, *e.g.,* derived from DICOM-CT data, may be included in the geometry definition via the `egs_glib` library, as follows:

```
:start geometry:
    name = phantom
    library = egs_glib
    type = egsphant
    egsphant file = lib/geometry/phantoms/egsphant/fileName.egsphant.gz
    density file = lib/media/material.dat
:stop geometry:
```

The `density file` indicates to `egs_brachy` the nominal density of each medium in the `egsphant`, and currently `egs_brachy` can read these data from a (a) `material.dat` file, (b) `pegs4` data file, or (c) a simple text file of the format:

```
MEDIUM=medium1
RHO=density1
MEDIUM=medium2
RHO=density2
```

Note that voxel-by-voxel densities, defined in the `egsphant` file, may vary from the nominal densities.

To combine various geometry components there are several classes. For example, the egs_genvelope class (see documentation at http://nrc-cnrc.github.io/EGSnrc/doc/pirs898/classEGS__EnvelopeGeometry.html) allows the insertion of the object specified by the inscribed geometries = input key into another object specified by the base geometry = input key where these keys must be names of one or more of the other geometries specified in the input file. For example:

```
:start geometry:
    name = phantom_with_seed
    library = egs_genvelope
    base geometry = phantom
    inscribed geometries = seed
:stop geometry:
```

5.4.1. egs_autoenvelope

A powerful geometry extension developed for egs_brachy is the egs_autoenvelope library. It automatically detects which voxels of the base geometry contain inscribed geometries, *e.g.,* seeds, thereby allowing faster transport (see section 7.2.). This requires a nested block :start region discovery: to define how to do this step.

```
:start region discovery:
       action = discover # discover (default -- ALWAYS use for egs_brachy),
                         # other options for use in other codes are
                               # discover and correct volume,
                               # discover and zero volume
       density of random points (cm^-3) = 1E6  #default is 1E8
     :start shape:
              type = cylinder
              radius = 0.04
              height = 0.45
      :stop shape:
:stop region discovery:
```

For egs_brachy, *always* use action = discover because volume correction inputs are handled separately with their own inputs. The other options of action=discover and correct volume or action=discover and zero volume are not needed for egs_brachy and are included in the egs_autoenvelope class for its potential use in other applications. Note that the density of random points = input key can be lower than that required for accurate volume correction (for which there is a separate input density of random points) – section 5.6..

The egs_autoenvelope library has two geometry types EGS_AEnvelope (default) and EGS_ASwitchedEnvelope. The egs_autoenvelope library must be used with type = EGS_ASwitchedEnvelope for simulations employing run mode = superposition so only one source is modelled at a time (*e.g.,* for TG-43-like and HDR simulations with example egsinp files described in sections 10.6. and 10.9., respectively).

5.5. source definition

The :start source definition: input block specifies the required info about the radioactivity in the seeds/sources (the geometry of the seed itself is handled in the geometry definition input block – section 5.4.). Within this input block there is a nested input block :start source: which specifies the charge (0, -1, 1) via an input key charge = and must specify the shape of the radioactivity. This can be quite complex and will usually bear considerable similarity to at least part of the geometry specification of the source. For the distributed sources, the relevant files are named seed_name.shape as opposed to seed_name.geom which defines the overall seed geometry. See lib/geometry/sources/ for distributed sources which can be used as examples if you need to create your own.

The energy spectrum of the source must be specified in a :start spectrum: nested block. Most often, one of 2 types of spectral inputs is used, either a tabulated spectrum or monoenergetic. The former uses a spectrum file = input key to specify the required

spectrum (see section 8.2. for a list of those available on `lib/spectra/`). The latter uses a simple `energy =` input key (in MeV).

Next one needs to specify the location of the source(s). This is done in the `:start transformations:` nested block. This can specify many replications of the source via `:start transformation:` blocks. One `:start transformation:` block is required for each seed; `lib/geometry/transformations` gives examples of various arrangements. Note that at least one `:start transformation:` block is needed, even if a null transformation, to tell there is a single source (by default there are no sources). If none is specified, it is assumed there is a single source at the origin but there is a warning message printed. Note that these transformations must be identical to those indicated for geometry specification of the source locations in the in the geometry block associated with the sources; see section 10.4. for a relevant example `egsinp`.

The `:start source definition:` block must also explicitly specify the `simulation source =` input key which normally corresponds to the `name =` key in the `:start source:` block.

5.6. volume correction

The `:start volume correction:` input block gives the information required to do a volume correction which basically subtracts the volume of a seed or source from the phantom voxel it is in so that the dose is properly determined. The `correction type =` input key specifies which type is used, `correct, none, zero dose`. The volume is determined using Monte Carlo and the `density of random points (cm^ -3) =` input key specifies the density of points to use. Values between 1E6 and 1E8 are common, but it depends on the size of the seed/source and the accuracy required. Recall that the dose immediately adjacent to any seed will be very high and is not often of much interest. Within this input block there is a nested block, `:start shape:` which specifies the outer boundary of the seed/source (*e.g.*, using the `boundary.shape` files found in the seed/source definition subdirectories). This shape is then transformed to each of the locations specified for the sources (see previous section).

Alternatively, the simulation can make use of a predetermined set of volume corrections in a `file.volcor` file (see section 5.7. for how to create it). Using this file is controlled by a `:start volume correction from file:` nested block with an input key `phantom file = phantom_name oldfile.volcor` where `phantom_name` is the name of the phantom described in this input and `oldfile.volcor` is the previously calculated corrections file appropriate to this phantom and seed combination.

5.7. scoring options

The `:start scoring options:` input block specifies different scoring options. Scoring options in general are discussed in section 7.4.. For a detailed listing of scoring options, see the on-line html Technical Reference Manual for `egs_brachy` at https://clrp-code.github.io/egs_brachy/index.html#scoringopts.

For scoring dose using the tracklength estimator, only the input keys `muen file =` and `muen for media =` are needed. The former specifies where the μ_{en}/ρ data are stored for the medium the dose is to be scored in (the file `lib/muen/brachy_xcom_1.5MeV.muendat` contains a large selection, see Table 3 for a list). Any medium which you wish to score dose in needs to be listed in the 'muen for media' key (order unimportant).

If a volume correction output is desired, this is specified in this block (as well as in the `:start volume correction:` input block). Use (a) the input key `output volume correction files for phantoms =` to specify the name of the phantom for which the corrections are required (phantom and name specified elsewhere in the input file) and (b) the key `volume correction file format =` to specify whether `text or gzip` format is desired. This output also required the `:start volume correction:` input block.

If phase space output is required, a nested `:start phsp scoring:` block is required where the input keys `phsp output directory =, access mode =, print header =, kill after scoring =` are input. The `access mode` can be `write or append` where the former is for a new file and the latter for appending. The `print header` should always be yes for a new file. If the `kill after scoring` input is 'yes', there is no transport done in the phantom, which is efficient for just scoring the phase space for later use, but it could be 'no' to score the dose as well as saving the phase space. Note that phase space scoring does not work for parallel jobs.

Another scoring option is specified in the `:start spectrum scoring:` nested block. The inputs are explained in `ex_single_source_phase_space_and_spectrum_scoring.egsinp`.

```
:start spectrum scoring:
    type = surface count      # options are surface count,
                              # energy weighted surface,
                              # energy fluence in region
    particle type = photon    # photon, electron, positron;
                              # note 'energy fluence in region' type is
                              #       only compatible with photons
    minimum energy = 0.001    # defaults to 0.001 MeV
    maximum energy = 1.00     # defaults to max energy of initial source
    number of bins = 1000     # defaults to 100
    output format = egsnrc    # xmgr (default), csv, egsnrc
                              # egsnrc => format to use as input
                              # spectrum
    file extension =          # (optional)
    # for 'egsnrc' output format only
    egsnrc format mode = 2    # 0, 1, 2 standard EGSnrc options. 2=>line
:stop spectrum scoring:
```

Another useful input key is `dose scaling factor =` which can be used to scale the output dose results, *e.g.,* to being dose in Gy, rather than the dose per effective number of histories, N_{eff} (see discussion related to Table 1 below). An example is in `ex_PeppaBreastHDR192Ir.egsinp` (see section 10.9. below).

The two input keys `score tracklength dose =` and `score energy deposition =` are only needed if their defaults (yes and no respectively) need to be changed. Both options can be used at the same time. The latter scores energy deposited by electrons,

so if the source is photons, this is only when a photon interaction occurs and hence is much less efficient. This is referred to as 'interaction scoring' in section 7.4.1.. The outputs are found in the files `filename.phantom.3ddose` for the tracklength scoring and `filename.phantom.edep.3ddose` for the energy deposition scoring.

5.8. variance reduction

The `:start variance reduction:` is used to turn on some of the variance reduction techniques which are not inbuilt (such as pathlength scoring, see section 7.4.1.).

As discussed in section 7.5.1., it is often advantageous to reuse a particle which escapes from the seed/source for `run mode = normal`. For cases with more than one seed, this should always be done using a `:start particle recycling:` nested block which by default just uses the recycled particles once for each seed. But it is further possible to recycle that particle n_r times. The `:start variance reduction:` input block looks like:

```
:start variance reduction:
    :start particle recycling:
        times to reuse recycled particles = 2
        rotate recycled particles = yes
    :stop particle recycling:
:stop variance reduction:
```

Note that recycling is not compatible with `run mode = superposition`.

Another type of variance reduction is splitting brems photons when modelling an x-ray source, to split each brems photon uniformly 50 times and optionally to also enhance the brems cross section by a factor of 100 (*i.e.*, use BCSE) *e.g.*,

```
:start variance reduction:
    split bremsstrahlung photons = 50 # uniform bremsstrahlung splitting
    bcse medium = W 100     # brems cross-section enhancement factor
                            # of 100 in the tungsten target
:stop variance reduction:
```

Range rejection is automatically on outside sources and off inside sources (if electrons are being transported at all). This can be controlled within the variance reduction block (see https://clrp-code.github.io/egs_brachy/index.html#rangerejection).

5.9. MC transport parameters

`EGSnrc` requires a series of transport parameters to be defined. These are specified in a `:start MC transport parameter:` input block. The easiest approach is to use one of the distributed files found at `lib/transport/` by using an `include file =` input key. These include:

```
high_energy_default
low_energy_default
xray_source
electron_transport_10keV
high_energy_sk_calc
low_energy_sk_calc
```

6. Brief description of egs_brachy outputs

6.1. The .egslog output

The user is encouraged to carefully examine the output to the screen (interactive runs) or to `filename.egslog` when run in batch. Even in interactive mode it is useful to pipe the screen output to `filename.egslog`, *i.e.,* add "`> filename.egslog`" to the end of the execution command. Carefully examining the output is to ensure that the code is calculating what is intended. The output is quite extensive but worth examining. Note that when running interactively, there can be error messages right after the submit statement.

The initial output repeats much of what is in the input file, including a specification of the materials used and the cross sections used. Eventually it gets to a summary of the geometry information which must be very carefully analyzed to ensure no mistakes were made since the required inputs are complex.

There is an explicit list of all the files that were read in as required by the input file.

The starting and incremental time for each batch along with the dose and uncertainty in region 0 are presented as the run progresses. The region the dose is specified in can be changed using the `current result phantom region = phantom_name ireg` key in the `:start scoring options:` input block. This allows the choice of which phantom result to monitor. If the `ireg` entry is not present, it is taken as 0.

Note that the CPU time reported is that for simulating the transport only, the total times are listed later.

When the simulations are finished, `egs_brachy` outputs some summary times and statistics on how many steps etc. It also reports how many geometry errors were encountered.

There are some summary statistics on the energy initiated, escaping the sources and escaping the geometry. Make sure these make sense since they can be a good diagnostic tool.

If a phase space file has been requested, there is a detailed summary of the data in that file. The number of particles in the file is listed. The file size will be 29 bytes per particle so be careful not to run too many histories.

The 20 highest doses are reported along with the volumes they were recorded in. Note that in simulations with seeds, the 20 highest doses will be in those regions immediately surrounding the seeds and have smaller uncertainties than many other doses. The average doses and their uncertainties are reported next.

The next section reports the amount of time spent in various parts of the run. For short runs, the initialization for the run, including creating the cross section data and correcting the volumes can dominate the total time.

To repeat, it is critically important to study the output log file carefully to ensure that the inputs were done properly.

6.2. .3ddose and other outputs

In addition to the .egslog outputs, the main output is in the .3ddose files. The pathlength doses are found in filename.phantom.3ddose and if requested, the energy deposition doses are in filename.phantom.edep.3ddose. This file format is used in other EGSnrc applications such as DOSXYZnrc and there are various tools distributed with EGSnrc for manipulating them (*e.g.,* statdose and dosxyz_show). The format of .3ddose files (from section 12 of the DOSXYZnrc Manual[3]) is:

Row/Block 1: number of voxels in x,y,z directions (*e.g.,* nx, ny, nz)
Row/Block 2: voxel boundaries (cm) in x direction(nx +1 values)
Row/Block 3: voxel boundaries (cm) in y direction (ny +1 values)
Row/Block 4: voxel boundaries (cm) in z direction(nz +1 values)
Row/Block 5: dose values array (nx.ny.nz values)
Row/Block 6: error values array (fractional errors, nx.ny.nz values)

The above format assumes an xyz rectilinear phantom. If the egs_rz cylindrical RZ geometry has been specified (section 5.4.), then the .3ddose file reports the z coordinates in the x slots, the radial coordinates in the y slots and the z slots are unused and shown as one region from -999 to +999. If the egs_spheres spherical geometry has been specified, then the .3ddose file reports the radial component in the x slots and the y and z slots are unused and shown as two regions, each from -999 to +999.

There is also a filename.egsdat file output which contains the information needed to do a restart. This can be a large file and should be discarded once the final run has been done/analyzed. In the :start run control: block one could specify
egsdat file format = gzip in which case these large files will be output in compressed format. This can be important when many parallel runs are being done.

Finally, there is a filename.mederr output file which summarizes the inputs for cross section data.

7. Description of egs_brachy

7.1. Radiation transport modelling

egs_brachy is an EGSnrc [4, 5] application, allowing for both photon and electron transport to be modelled. This means that both electronic sources (*e.g.,* miniature x-ray tubes and beta-emitting eye plaques) and photon (radionuclide) brachytherapy sources may be modelled. While electron transport may be modelled, the relatively low energy of photons from brachytherapy sources means that dose is often well approximated by collision kerma for many brachytherapy scenarios (section 7.4.1.). Thus, often only the simulation of photon transport is necessary and electron transport is not modelled via appropriate choice of the electron cutoff energy (ECUT). All the physics underlying EGSnrc may be modelled, *e.g.,* Rayleigh scattering, bound Compton scattering, photoelectric absorption, and fluorescent photons from atomic relaxations (K, L, M, N shells). As with other EGSnrc applications,

the user may select the photon cross section dataset to use in calculations and specify the raw source spectrum.

7.2. Geometry and source modelling

Geometry modelling is handled by egs++ [6] (manual available at http://nrc-cnrc.github.io/EGSnrc/doc/pirs898/index.html). Using the built-in elementary and composite geometries of egs++, complex phantom, source, and applicator geometries can be modelled. egs_brachy is distributed with a library of pre-defined phantom, applicator, and source model geometries to facilitate ease of use (section 9.) and to give working examples.

egs_brachy supports scoring in rectilinear voxels (through the use of the EGS_XYZGeometry class) and in cylindrical and spherical shells using the EGS_rz and EGS_cSpheres classes, which are additions to the egs++ library developed for egs_brachy. These and other additions (further described below - EGS_AEnvelope, EGS_ConicalShellStackShape, EGS_SphericalShellShape) to egs++ are available as part of the egs_brachy distribution. They will eventually be part of the core EGSnrc distribution but may currently be found at https://github.com/randlet/EGSnrc/tree/feature-geometries. Documentation for the additional classes can be found in the class listing of PIRS-898 included with EGSnrc_with_egs_brachy, located at $HEN_HOUSE/doc/src/pirs898-egs++/html/annotated.html.

As brachytherapy treatments may involve many sources, a new egs++ geometry class has been implemented, egs_autoenvelope. This class is a fast envelope geometry used to inscribe one or more copies of another geometry inside a base geometry, such as multiple source geometries inside a phantom. During the geometry initialization, voxels within the phantom containing part of a source geometry are identified ('region discovery'): points inside each source geometry are randomly selected and the phantom regions containing these points are determined, thus generating a list of phantom regions containing part of a source. The time for this process depends on, *e.g.,* the number of sources, voxel sizes, number of voxels, phantom size, density of random points [1]. The list of phantom voxels containing sources is used during particle transport to determine whether the current voxel contains a source; if it does not, then there is no check of the boundaries of source geometries since not checking reduces computation time. See section 10.4. for an example egsinp file demonstrating the use of egs_autoenvelope for a prostate LDR simulation involving 100 seeds.

A subtype of egs_autoenvelope, EGS_ASwitchedEnvelope, is a 'switched' egs_autoenvelope which activates and deactivates inscribed geometries (either in a sequential or arbitrary order) in custom egs++ applications. This geometry class was developed for egs_brachy to investigate the effects of interseed attenuation and scatter, as well as to simulate a source stepping through dwell positions in an HDR treatment. It must be used in conjunction with the 'superposition' run mode (section 7.3.). See section 10.9. for an egsinp example.

Within a source, the distribution of radioactivity is specified by sampling random points from a user-specified shape. Two new egs++ shape classes were developed for egs_brachy and can be used by other egs++ applications. EGS_ConicalShellStackShape and EGS_SphericalShellShape sample random points within a stack of conical shells and within spherical shells, respectively. EGS_SphericalShellShape can also sample points within hemispherical shells and in shells truncated by a conical section, which is useful when defining, *e.g.,* the activity distribution within a beta-emitting eye plaque.

The egsphant file format (also used with the EGSnrc application DOSXYZnrc [3]) may be used to model a a rectilinear phantom derived from CT data; note that 'gzipping' egsphant files can result in compression of these potentially large files by $\sim 99\%$ and these files can be directly used in egs_brachy. CT data in the DICOM format may be converted to the egsphant format using ctcreate (distributed with the EGSnrc code) or similar tools.

7.3. Run modes

egs_brachy has three run modes: normal, superposition, and volume correction only.

The default run mode for simulations is normal, in which one or more brachytherapy sources are modelled. Particles may be initiated within the source with a user-defined spectrum or using phase-space data as a source of particles initialized on the surface of the source (referred to as a 'phase-space source').

The superposition run mode may be used with more than one brachytherapy source and requires the use of the EGS_ASwitchedEnvelope geometry class (section 7.2.). In this mode, one source is 'active' at a time; while particles from this source are being transported through the geometry, all other sources are 'virtual', *i.e.,* photons are transported through the material in which the source is embedded. This run mode is suitable for simulating HDR treatments in which there is only one source present in the geometry at a time and the source steps through dwell positions (example egsinp – section 10.9.). This feature can be applied to LDR simulations to remove interseed effects and investigate interseed attenuation or mock-up TG-43 calculations (example egsinp – section 10.6.).

For simulations using multiple instances of the same source model with either the normal or superposition run modes, sources may be assigned different statistical weights; emitted photons will be weighted accordingly. This feature may be used to model sources of differing air-kerma strengths for LDR treatments or varying dwell times at different source positions for HDR treatments (example egsinp – section 10.9.). The overall normalization for user-specified weights is arbitrary as egs_brachy will divide all weights by the highest weight value such that all weights are ≤ 1. See also section 7.4.1. for details regarding weights and dose calculation.

Dose calculations in egs_brachy typically require the volume (or mass) of the voxel in which energy is deposited (section 7.4.1.). The volume correction only run mode may be used to run the volume correction routines of egs_brachy (see section 7.4.1.), output the results, then quit; no radiation transport is done. The output of a volume correction only run may be used as a pre-calculated volume correction file. This is especially useful for re-

peated simulations or simulations being run on multiple cores in parallel with lengthy volume corrections to perform, *e.g.,* in the case of large applicators: a single `volume correction only` job is run to calculate the volume corrections. Subsequent simulations use the volume correction file, eliminating the redundant, potentially time-consuming calculation of volume corrections. See section 10.8. for a relevant example `egsinp` file.

7.4. Scoring options

7.4.1. Dose

Although `egs_brachy` can model both electron and photon transport (section 7.1.), the low energy of photons from most brachytherapy sources means that the range of secondary electrons is very short [7, 8]. Charged particle equilibrium can often be assumed and collision kerma can be considered equal to absorbed dose for many applications of interest in brachytherapy. By default, collision kerma in voxels is scored using a tracklength estimator (see, *e.g.,* ref [9]):

$$D^j = K^j_{\rm col} = \frac{\sum_i E_i t_i \left(\frac{\mu_{\rm en}}{\rho}\right)_i}{V_j} \tag{1}$$

where D^j and $K^j_{\rm col}$ are the dose and collision kerma in the $j^{\rm th}$ voxel, E_i is the energy of the $i^{\rm th}$ photon crossing the voxel, t_i is the tracklength of that photon in the voxel, $\left(\frac{\mu_{\rm en}}{\rho}\right)_i$ is the mass energy absorption coefficient for energy E_i, and V_j is the volume of the voxel. When a voxel in a scoring phantom is partially occupied by another geometry (source, applicator, etc.), energy deposition is only scored in a portion of the voxel, the volume of which must be used to calculate dose. Voxel volume corrections are performed using a Monte Carlo technique to estimate the volume of the voxel occupied by another geometry.

Scoring collision kerma efficiently using equation (1) requires rapid access to mass energy absorption coefficients. Mass energy absorption coefficients for phantom media in which dose is scored are supplied in a specified external file for a given simulation; these data may be calculated using the EGSnrc application *g*. Data are stored on the same energy grid as used by the EGSnrc system for materials data and the same lookup algorithms are used as for the cross section data. For accurate calculations, mass energy absorption data must be provided for a reasonable grid of energies; the current default is 2000 points. A file containing mass-energy absorption coefficients for various media is distributed with `egs_brachy` (`egs_brachy/lib/muen/brachy_xcom_1.5MeV.muendat`) and these coefficients were calculated using the XCOM photon cross section dataset. For consistency the user of `egs_brachy` should use the same data set for photon cross sections. The user may consult the file `egs_brachy/lib/media/material.dat` for the list of elemental compositions. As of September 2017 the `.muendat` file contains the materials listed below in Table 3 for an energy range from 1 keV to 1.5 MeV.

The user may also choose to employ interaction scoring to compute absorbed dose or collision kerma, depending on the cutoff energy chosen for electron transport. Computation of absorbed dose may be necessary for situations in which the collision kerma approximation

is inadequate, *e.g.,* for distances less than 2 mm from the central axis of an ^{192}Ir source (see [10] and references therein) or near other higher-energy brachytherapy sources; in situations where there are significant media heterogeneities; or if the dimensions of scoring voxels are not significantly longer than the secondary electron range.

Whether dose is approximated as collision kerma or absorbed dose is calculated, statistical uncertainties on dose are evaluated using history-by-history statistics [11]

$$s_{D^j} = \sqrt{\frac{1}{N_h - 1}\left[\sum_{i=1}^{N_h}\frac{(D_i^j)^2}{N_h} - \left(\sum_{i=1}^{N_h}\frac{D_i^j}{N_h}\right)^2\right]}. \tag{2}$$

Here, s_{D^j} is the statistical uncertainty per history on the dose D^j deposited in voxel j, while the summation is over the N_h statistically independent starting particles (histories); D_i^j is the dose deposited in voxel j from all particles originating from history i. When particle recycling is used, recycled particles are considered part of the same history as the original particle.

Doses and statistical uncertainties are output within the phantoms specified by the user in the `3ddose` format developed for the BEAMnrc system [3](see section sec-3ddose). The statistical uncertainty on dose in each voxel is output as a fractional uncertainty. Dose in voxel j is output normalized as dose (in Gray) per effective number of histories per source (or source positions, for `superposition` run mode):

$$\bar{D^j} = \sum_{i=1}^{N_h}\frac{D_i^j}{N_{eff}/n_s} \tag{3}$$

The effective number of histories, N_{eff}, is defined in Table 1. In simulations for which starting particles are initiated within the radioactivity distribution within the source (herein referred to as 'ab initio'), N_{eff} is just the number of histories, N_h. When a phase-space source or the particle recycling feature are used (see section 7.5.), N_{eff} is the number of independent histories that would have to be simulated in an ab initio simulation to obtain the same number of scoring particles. Table 1

Dose may be scored separately for primary, single-scattered, and multiple-scattered particles according to the Primary Scatter Separated (PSS) dose formalism [12] at the user's request. Any particle escaping the source encapsulation is considered a primary until it interacts outside the source. The primary and scatter separated doses are normalized to the source's total radiant energy, which is the sum of the energies of all particles escaping source encapsulation.

7.4.2. Calculation of absolute dose from doses output by `egs_brachy`

Calculations of absolute dose for photon sources require values of air-kerma strength. In the following, the initial or starting (time $t = 0$) air-kerma strength of source i, $i = 1, ..., n_s$, is denoted $S_K(i)$; the air-kerma strength per history is denoted S_K^{hist} (units: Gy·cm^2/history) - see Table 5 (section 9.1.) for values corresponding to source models distributed with `egs_brachy`.

Table 1: Effective number of independent histories, N_{eff}, used for dose normalization (equation (3)), where N_h is the number of initial histories, n_s is the number of sources in the simulation, n_r is the number of times to recycle particles at each source location, N_1 is the number of particles to initialize from the phase-space source, and N_{emit} is the the number of photons emitted from the source from simulation of N_h (initial) histories when generating the phase space data.

Simulation	Description	N_{eff}
(1) Ab initio (default)	All particles initialized within each source	N_h
(2) Ab initio with recycling	Particles initialized within one source; particles escaping that source are recycled n_r times at each of the n_s source locations	$N_h\, n_s\, n_r$
(3) Phase-space source	Initialize $N_1 \leq N_{emit}$ particles from a file containing data for N_{emit} particles from simulation of N_h (initial) histories	$\frac{N_1 N_h}{N_{emit}}$
(4) Phase-space source and recycling	Phase-space source as in (3) with recycling as in (2)	$\frac{N_1 n_s n_r N_h}{N_{emit}}$

Low-dose rate (LDR) brachytherapy with photon sources:

As described in section 7.3., doses scored by `egs_brachy` are weighted by the statistical weight of the photon from the particular source. For LDR treatments, source weights should be based on the *starting* ($t = 0$) air-kerma strength of each source, *e.g.,* weights will be input as $S_K(i)$. Then, with the maximum *starting* ($t = 0$) air-kerma strength amongst the n_s sources denoted $[S_K]_{max}$, `egs_brachy` normalizes weights internally as $w_i = S_K(i)/[S_K]_{max}$ such that the source with the highest air-kerma strength has weight unity. If all sources have the same air-kerma strength then there is no need to input weights (all sources equally weighted). Then taking $t = 0$ and t_{end} as the treatment start and end times, respectively, the total dose in voxel j is:

$$D^j_{tot} = [S_K]_{max} \frac{\bar{D}^j}{S^{hist}_K} \tau \left(1 - e^{-\lambda t_{end}}\right), \tag{4}$$

where τ is the radionuclide mean lifetime with $\tau = 1/\lambda = t_{1/2}/\ln 2$ where standard notation is used for the decay constant, λ, and half-life, $t_{1/2}$. For a permanent implant, taking $t_{end} \to \infty$,

$$D^j_{tot} = [S_k]_{max} \frac{\bar{D}^j}{S^{hist}_K} \tau\,. \tag{5}$$

For clarity, `egs_brachy` (by default) outputs the values of $\bar{D}^j$ and it is up to the user to apply the relevant equations for their case either as a post-processing step or using the `dose scaling factor` input.

High dose rate (HDR) brachytherapy with photon sources:

Assuming the source air-kerma strength is constant over the treatment and that the dwell

times are much shorter than the mean (radionuclide) lifetime, then internally the weights are taken as $\Delta t_i/\Delta t_{max}$ where $i = 1, ..., n_s$ enumerate the source dwell positions, Δt_i is the dwell time for source position i, and Δt_{max} is the maximum individual dwell time. However, the user is free to input the dwell times directly as the weights, or the fractional dwell times at each position and egs_brachy internally calculates the required weights. Given these weights, the total dose in voxel j is:

$$D^j_{tot} = [S_K]_{max} \frac{\bar{D}^j}{S^{hist}_K} \Delta t_{max}, \tag{6}$$

where $[S_K]_{max}$ is the starting ($t = 0$) source air-kerma strength and the user will need to know Δt_{max} to use the outputs. As the half-life of ^{192}Ir is 73.8 days and dwell times are often on the order of seconds, these approximations may well be adequate for many applications. However, without these assumptions, the (un-normalized) weights should be $S_K(t_{i,1})\left(e^{-\lambda t_{i,1}} - e^{-\lambda t_{i,2}}\right)/\lambda$ (where $t_{i,1}$ and $t_{i,2}$ are the start and end times, respectively, for dwell position i, $i = 1, ..., n_s$) which will be normalized by egs_brachy via division by the maximum weight over all the source positions, $\max\left\{S_K(t_{i,1})\left(e^{-\lambda t_{i,1}} - e^{-\lambda t_{i,2}}\right)/\lambda \,|\, i = 1, ...n_s\right\}$. The total dose in voxel j is then:

$$D^j_{tot} = \frac{\bar{D}^j}{S^{hist}_K} \max\left\{S_K(t_{1i})(e^{-\lambda t_{i,1}} - e^{-\lambda t_{i,2}})/\lambda \,|\, i = 1, ...n_s\right\}. \tag{7}$$

7.4.3. Phase space

A particle's position, charge, and energy, as well as the direction of its trajectory comprise its phase-space data. egs_brachy can tabulate phase-space data on the surface of a source for all particles emitted from the source. Optionally, particles can be discarded immediately after having been scored to the phase space to increase the speed of the generation of the phase-space data. Phase-space data are written in the IAEA format [13] which is a generalization of the original BEAMnrc phase space format since it also allows the z position to be included.

7.4.4. Spectrum

egs_brachy has three spectrum scoring options:

1. an absolute count, on the surface of the source, of the particles (photons or charged particles) escaping a source;
2. an energy-weighted spectrum of particles (photons or charged particles) scored on the surface of a source; and
3. a photon energy fluence spectrum scored in a voxel.

For the second option, energy is scored when particles are emitted from the surface. For the third option, the photon fluence in a user-specified voxel is calculated as the total photon

pathlength in a given energy bin divided by the volume of the region, which is equivalent to the fluence averaged over the voxel volume [14, 15].

Scored spectra may be output in a file formatted for plotting with `xmgrace`, or in the three possible EGSnrc input spectrum 'modes' enumerated as: 0 (histogram in counts per bin); 1 (histogram in counts per MeV); and 2 (line spectrum).

Further details at
https://clrp-code.github.io/egs_brachy/index.html#specscoring .

7.5. Features to enhance simulation efficiency

`egs_brachy` was designed to be a fast MC code for brachytherapy. Geometry handling is designed to be efficient, *e.g.,* by identifying phantom voxels containing part of a source during geometry initialization (section 7.2.). By default, dose is approximated as collision kerma scored using a tracklength estimator (section 7.4.1.), and hence electron transport is not typically needed. Other features to enhance simulation efficiency are discussed in the current section.

7.5.1. Recycling of photons emitted from sources

For simulations of $n_s > 1$ sources of the same model with the `normal` run mode, the simulation of photons within sources need not be repeated for every history. Using the particle recycling feature, the first source in the simulation acts as a particle generator such that particles initiated in this source are tracked until they are either absorbed within it or they escape the source encapsulation. If the particle does not escape the source, the history counter is incremented by one and another particle is initiated in the first source. This process is repeated until a particle escapes the first source. The escaping particle is then translated and initiated for each source at the same relative position as it escaped the first source or rotated by a random angle about the source axis before being re-initiated at each source location. This is called recycling. Each generated particle may be recycled more than once (n_r times) at each source location; in this case, the rotation is not optional and particles are rotated by a random angle about the source axis before each reuse so they don't both go through the same voxels.

Recycled particles from the same primary history are not statistically independent. The computation of statistical uncertainties accounts for correlations between particles: the history counter is incremented by one for each set (i.e., $n_s n_r$) of recycled particles to preserve the history-by-history statistics (Table 1, p20).

In general when using particle recycling with $n_r = 1$ (*i.e.,* when using multiple sources), rotating leads to slightly less efficient simulations than recycling particles without rotation since photons are less likely to be in the same voxel if they all start in the same direction from different seeds. The COMS eye plaque simulations are more efficient even when $n_r = 1$ if particles are rotated prior to being recycled at each source location because of the particular

spatial arrangement of seeds within the plaque; as the statistical uncertainty accounts for correlations between recycled particles from the same primary history, when the rotation option is not on, several of these photons deposit dose in the same voxels, resulting in an increase in computing time with little improvement in statistical uncertainty. Prostate and breast LDR simulations are generally more efficient when rotation is off as photons from the same history are less likely to deposit dose in the same voxels. In general, recycling particles more than once at each source location does not lead to further efficiency gains. See Table 6 of ref[1] for specific examples.

7.5.2. Use of phase-space source

The user specifies the number of particles, N_1, to be initiated from the phase-space data. If this number exceeds the number of particles for which data exist in the phase-space file (N_{emit}), then `egs_brachy` generates a warning that the phase-space file is being used more than once. This situation may lead to biasing as the calculation of statistical uncertainties will not account for the fact that particles are not all independent. If the user wishes to simulate more starting particles than there are data for in the phase space (i.e., $N_1 > N_{emit}$), particle recycling should be used in conjunction with the phase-space source since this will properly account for correlations between particles from the same primary history (Table 1).

7.5.3. Features for electronic brachytherapy sources

In miniature x-ray tube sources, electrons set in motion impinge on a bremsstrahlung target. As discussed by Ref. [16], the probability of bremsstrahlung emission from low-energy electrons decelerating in target material is very small, resulting in a large fraction of CPU time spent tracking electrons which stop without giving off photons and hence create inefficient simulations. `egs_brachy` users have the option to use bremsstrahlung cross section enhancement (BCSE), uniform bremsstrahlung splitting (UBS), and Russian roulette to enhance simulation efficiency. With the BCSE feature, the user specifies a factor f_{enh} by which to scale up the bremsstrahlung production cross section in the target material. Using UBS, the user specifies N_{split} and each generated bremsstrahlung photon is split into N_{split} photons. Russian roulette is automatically activated if BCSE or UBS are used, and then secondary charged particles produced have a survival probability of $1/(f_{enh}N_{split})$. The original BCSE paper [16] suggests optimal values for the required parameters, $f_{enh} = 500$ and $N_{split} = 100$, for the $(1\text{ mm})^3$ array of voxels they considered (note that optimal values will depend on the simulation under consideration) [1].

8. Data distributed with `egs_brachy`

8.1. Material data definitions distributed with `egs_brachy`

The user may opt to run the `egs_brachy` application using the traditional `.pegs4dat` data file or in `pegsless` mode. For the latter, the materials' properties are defined in the file

egs_brachy/lib/media/material.dat. The user can easily add any material needed to the egs_brachy/lib/media/material.dat file. Two typical definitions are:

```
          medium = AIR_TG43
          rho = 0.0012
          elements = H, C, N, O, AR
          mass fractions = 0.0732, 0.0123, 75.0325, 23.6077, 1.2743
          gas pressure = 1.0
          bremsstrahlung correction = NRC
-------------------------------------------------------------------------
          medium = WATER_0.998
          rho = 0.998
          elements = H, O
          number of atoms = 2, 1
          bremsstrahlung correction = NRC
```

Note that, in our experience, using the traditional .pegs4dat data file results in slightly faster total calculation times for simulations involving many media. For simulations with many histories and relatively long calculation times, the time saved using the traditional .pegs4dat data file is probably negligible, however, it may be beneficial to run with the .pegs4dat data file for sub-minute calculation times of simulations involving many media.

As distributed, the file egs_brachy/lib/media/material.dat contains the definitions of the materials in Table 2. A material definition is required for all materials being transported in. In addition to the information in these files, the .egsinp file must contain information on the upper and lower energy thresholds for creation of charged particles (AE,UE) and photons (AP,UP) in the :start media definition: block of the .egsinp file (see section 5.3. above). Note that the photon cross sections by default are the defaults used by EGSnrc, in particular the xcom-based cross sections. This can be changed in the input file but care must be taken to then match the mass energy absorption coefficients used by egs_brachy.

As distributed the file egs_brachy/lib/muen/brachy_xcom_1.5MeV.muendat contains the mass energy absorption coefficients for those media in which the dose is typically scored using pathlength scoring. Table 3 lists the available materials in this file. They were created using the XCOM photon cross sections. Not all materials in the material.dat file are included here and there are some additional materials (usually tissues) which are not included in the material.dat file.

Table 2: As distributed, the file egs_brachy /lib/media/material.dat contains the following materials definitions. The compositions of each material can be seen in this file and are given in the reference manual (https://clrp-code.github.io/egs_brachy/md_media.html)

25GLAND-75ADIPOSE	50GLAND-50ADIPOSE	75GLAND-25ADIPOSE
ADIPOSE2_WW86	ADIPOSE_PHANT	ADV_PD_POLY
AIR_PHANT	AIR_TG43	AIR_TG43_LD
AQUEOUS_MAR	Ag	AgBrAgI_6.20
AgBrAgI_6.20	AgI	AgI_6.003
Al	Al2O3	AlN_Y2O3_0.05_3.26
AlSilicate	Alumina_2.88	Ar
Au	Au80Cu20	BRAIN_PHANT
BRAIN_WMATTER_WW1986	C12H18NCl	C85.7H14.3
CALCIFICATION_ICRU46	CORNEA_COLLSTRUCMECH_ZIE	CARTILAGE_PHANT
CORTICAL_BONE_WW86	CRANIUM_PHANT	CS10_polymer
Co	Cu	DENSIMET_D176
EYES_PHANT	F_SOFT_TISSUE_ICRU46	GLAND2_WW86
GLAND_PHANT	HEART_BLOODFILLED_WW86	Graphite2.26
I	IPlant_active	Ir
Ir70Pt30	KOVAR	LENS_ICRU
LUNG_BLOODFILLED_WW86	MANDIBLE_PHANT	MINERALBONE_PHANT
MODULAY	MUSCLE2_WW86	MUSCLE_PHANT
M_SOFT_TISSUE_ICRU46	Mo	Ni
P50C50	PMMA	PROSTATE_WW86
Pb	Pd	Pollucite
PolySty	Poly_Best2335_0.5gpcm	Poly_for_BestPd103_1gpcm
Pt	Pt70Ir30	Pt75Ir25
Pt90Ir10	Pyrex_2.4	Pyrex_2.4_Cs
RECTUM_ICRP23	SCLERA_COLLSTRUCMECH_ZIE	SILASTIC
SKIN2_WW86	SKIN_PHANT	SS_AISI301
SS_AISI304	SS_AISI304_p5.6	SS_AISI316L
SS_AISI316L_p5.0	SS_AISI316L_p6.9	SS_AISI316L_p7.8
SS_AISI316L_p8.02	SS_AISI316L_rho4.81	SS_AISI321
SS_PROBE_RHO8.0	SdVBC24H24	SiO2
TEETH_PHANT	Ti	Ti10W90
Ti44.4Ni55.6	Ti_grade2	URETHRA_WW86
URINARY_BLADDER_FULL	URINARY_BLADDER_EMPTY	W
VITREOUS_TUMRAM_MAR	WATER_0.998	Yb169
melanoma1_maughan	quartz_2.21	

Table 3: The file egs_brachy/lib/muen/brachy_xcom_1.5MeV.muendat contains the mass energy absorption coefficients of the following materials iand more. Complete current list at https://clrp-code.github.io/egs_brachy/md_media.html

WATER_0.998	AIR_TG43	AIR_TG43_LD
WATER_1.000	50GLAND-50ADIPOSE	25GLAND-75ADIPOSE
75GLAND-25ADIPOSE	10GLAND-90ADIPOSE	90GLAND-10ADIPOSE
CALCIFICATION	TISSUE_ICRU33	SOFT_BONE_ICRU44
ADIPOSE1_WW86	ADIPOSE2_WW86	ADIPOSE3_WW86
HEART1_WW86	HEART2_WW86	HEART3_WW86
HEART_BLOODFILLED_WW86	LUNG_BLOODFILLED_WW86	MUSCLE1_WW86
MUSCLE2_WW86	MUSCLE3_WW86	CARTILAGE_WW86
SPONGIOSA_WW86	RED_MARROW_WW86	YELLOW_MARROW_WW86
SKIN1_WW86	SKIN2_WW86	SKIN3_WW86
RIBS_2_6_WH87	RIBS_10_WH87	CORTICAL_BONE_WW86
LUNG_UNITRHO	EYELENS_ICRU46	M_SOFT_TISSUE_ICRU46
F_SOFT_TISSUE_ICRU46	GLAND2_WW86	PROSTATE_WW86

8.2. Initial radionuclide spectra

egs_brachy is distributed with initial spectra for a variety of radionuclides as shown in Table 4. They are read in via a spectrum file = input key in the :start spectrum: nested block. These spectra are in the standard ensrc format used by many EGSnrc applications, *viz*:

```
TITLE      spectrum title  (80 char)
NENSRC, ENMIN, MODE (on 1 line)
   NENSRC  # energy bins in spec. histogram
   ENMIN   lower energy of first bin
   MODE    =0 => cts/bin; =1  => cts/MeV; =2 => line spectra
ENSRCD(I),SRCPDF(I)  I=1,NENSRC pair/line
  -modes 0 & 1: top of energy bin and probability of initial particle being
   in this bin.  Probability does not need to be normalised.
  -mode 2: energy of spectral line and its probability.
```

Table 4: Spectra files available in the egs_brachy distribution for different radionuclides. Found on $EGS_HOME/egs_brachy/lib/spectra. Photon only spectra unless otherwise stated. If there are more than two spectra for a radionuclide, the recommended spectrum is indicated by an asterix (∗) after the file name.

Nuclide	Label(.spectrum)	Ref.	Comments
^{60}Co	bareco60_line		
^{103}Pd	Pd103_NNDC_2.6_line ∗	[17]	Based on NNDC 2000 evaluation
^{103}Pd	Pd103_TG43	[18]	Spectrum above fits meas. better.
^{106}Rh	Rh106_ICRU72_line	[19]	β spectrum
^{125}I	I125_NCRP_line ∗	[20, 17]	Fits meas. better than TG43[a)]; used by BIPM.
^{125}I	I125_TG43	[18]	Above spectrum is better.
^{131}Cs	Cs131_NNDC_2.6_line	[21]	Khazov et al. (2006).
^{137}Cs	Cs137_NNDC_2.6_line	[21]	Browne & Tuli, Nucl Data Sh. 108(2007)2173.
^{169}Yb	Yb169_NNDC_2.6_line	[21]	Baglin (2008)
^{192}Ir	Ir192_NNDC	[21]	Photons only.
^{192}Ir	Ir192_NNDC_2.6_line ∗	[21]	Baglin (2012) beta- and EC decays.
^{192}Ir	Ir192_bare_1993	[22]	Duchemin and Coursol (1993)

[a)] Only significant difference from TG-43 spectrum is 31 keV peak(s).

9. Geometries distributed with egs_brachy

This section provides some brief details regarding some source, phantom, and applicator geometries distributed with egs_brachy. To get a complete list of available geometries, cd $EGS_HOME/egs_brachy/lib/geometry and explore or issue ls -l * . The file names should be self-explanatory. For some geometries, the use of egs_view is useful for seeing the geometry.

9.1. Sources

The initial (beta) release of egs_brachy includes the three sources benchmarked in the egs_brachy paper [1]: ^{103}Pd TheraSeed 200 (Theragenics), ^{125}I OncoSeed 6711 (Amersham), and ^{192}Ir microSelectron-v2 HDR (Nucletron), in addition to the generic ^{192}Ir source developed by the joint AAPM-ASTRO-ABG Working Group on MBDCA (MBDCA-WG)[23]. Note that we have modelled many other seeds/sources which will be included once we have completed a paper describing them (23 ^{125}I, 11 ^{103}Pd, 12 ^{192}Ir HDR, 2 ^{192}Ir LDR, 5 ^{192}Ir PDR, 1 ^{169}Yb HDR, 1 ^{137}Cs LDR, 1 ^{131}Cs131 LDR and 2 ^{60}Co HDR sources).

Note that there are two versions of some seed geometries. This occurs when there is a 'hole' at the end of the seed (*e.g.,* TheraSeed 200). In one model this region is filled with air and should be used for in-air calculations. In the other model, this region is filled with water for use when doing calculations in a water phantom. If used in another phantom material, strictly speaking the region should be filled with that material, but since these regions are so small, little difference should be found by leaving them filled with water.

A summary of dose-rate constants and air-kerma strengths per history for the distributed source models are given in table 5. egs_brachy calculates in the WAFAC geometry and then we correct to what the S_K value would be on-axis assuming the source were isotropic using equation 9 in Ref. [24] or the closed form expression in equation 3 of Ref. [17] (with L=0). Table 5

Table 5: Source models, dose-rate constants (DRCs), and air-kerma strengths per history for source models distributed with egs_brachy. For the low-energy sources, values of DRC and air-kerma strength per history were determined using the scoring geometry approximating the WAFAC at NIST. For the high-energy sources, the DRC values use the 10×10×0.05 cm^3 scoring voxel, corrected to a point. Please cite the original egs_brachy paper [1] if using the ^{125}I OncoSeed 6711, ^{103}Pd TheraSeed 200, and ^{192}Ir microSelectron-v2 HDR source models.

Radionuclide	Source model	DRC (cGy h^{-1} U^{-1})	S_K^{hist} (Gy cm^2/hist)
^{103}Pd LDR	TheraSeed 200	0.6840 ± 0.0009	$(6.4255 \pm 0.0017) \times 10^{-14}$
^{125}I LDR	OncoSeed 6711	0.931 ± 0.001	$(3.7651 \pm 0.0010) \times 10^{-14}$
^{192}Ir HDR	MicroSelectron-v2	1.1085 ± 0.0006	$(1.1517 \pm 0.0002) \times 10^{-13}$
^{192}Ir HDR	MBDCA-WG	1.1109 ± 0.0009	$(1.1592 \pm 0.0007) \times 10^{-13}$

9.2. Phantoms

Diverse phantoms are distributed with egs_brachy (see /lib/geometry/phantoms/); a small subset include:

- lib/geometry/10.0cmx10.0cmx10.0cm_2mm_xyz_water.geom: all-water cubic phantom spanning -5.0 cm$< x, y, z <$5.0 cm consisting of a $50 \times 50 \times 50$ array of (0.2 cm)^3 voxels.

- `lib/geometry/10.1cmx10.1cmx10.1cm_1mm_xyz_water.geom`: all-water cubic phantom spanning -5.05 cm< x, y, z <5.05 cm consisting of (0.1 cm)^3 voxels.
- `lib/geometry/30cm_0.1mm_sph_water.geom`: all-water spherical phantom with outer radius 15 cm with 0.01 cm thick shells for radii≤ 1.015 cm.
- `lib/geometry/30cmx30cm_r101xz203_0.1mm_rz_water.geom`: all-water cylindrical phantom with total length 30 cm and outer radius 30 cm with 0.01 cm thick cylindrical shells for $0 \leq r \leq 1.015$ cm and -1.015 cm $\leq z \leq 1.015$ cm.
- `PeppaBreastHDR192Ir_MBDCA-WG.egsphant.gz`: Derived from DICOM-CT data for the breast phantom in case 3 of Peppa *et al.* [25].

9.3. Eye plaques

Note that we have modelled many eye plaques (all COMS; many Ru-106 beta plaques) and these will be distributed once we have completed a paper describing them.

9.4. Applicators

Applicators included in the `egs_brachy` distribution are found in the directory `lib/geometry/applicators`:

- `lib/geometry/applicators/TG186`
  Generic TG-186 Ir-192 shielded applicator[26] developed by members of the joint AAPM-ESTRO-ABG working group on MBDCA, design based on three commercially available gynecological applicators. See section 10.8..

10. Example input files distributed with `egs_brachy`

A suite of example input files is distributed with `egs_brachy` (found in `lib/examples`). These provide a good starting point for users new to `egs_brachy` and are described in the following subsections. Note that the input files (plus results) in the test suite (found in `egs_brachy/tests`) are also useful.

In these input files, various references to other files are given relative to `$EGS_HOME/egs_brachy`. This will work when the examples are being run interactively. However, when running in batch mode, the complete path to these files must be used, so wherever you see `lib/geometry(etc)` in the example input files, insert `$EGS_HOME/egs_brachy/` immediately before `lib` where **you insert** the full path on your system for `$EGS_HOME`.

These examples almost all use the `pegsless` form of input for cross section data. The example in section 10.5. uses a `.pegs4dat` file for input.

When starting to learn egs_brachy, the examples should be studied in the order presented here in the sense that the earlier examples have somewhat more in-line commenting which is not repeated in the later examples.

10.1. ex_single_source.egsinp

A single LDR Pd-103 TheraSeed200 seed in a 30x30x30 cm^3 water phantom; dose is scored in a 2x2x2 cm^3 volume made up of (1 mm)^3 voxels.

10.2. ex_single_source_phase_space_and_spectrum_scoring.egsinp

A single Pd-103 TheraSeed200 seed is modelled in a water phantom; the phase space and spectrum of photons emitted from the source are scored.

10.3. ex_single_source_from_phase_space.egsinp

Same simulation as ex_single_source.egsinp (section 10.1.) except that particles are initialized on the source surface using the phase space generated using ex_single_source_phase_space_scoring.egsinp (section 10.2.) – note that that the phase space file must be copied into the directory lib/phsp/ (that the user must create if it does not already exist). This example also makes use of recycling and rotating the recycled particles.

10.4. ex_prostate_permanent_implant.egsinp

A phantom of prostate-tissue material contains 100 Amersham OncoSeed 6711 seeds; dose is scored in a rectangular 'PTV' 3.4x2.8x3.8 cm^3 in (2 mm)^3 voxels. Includes use of egs_autoenvelope involving multiple copies of the same source in different locations, as well as particle recycling.

10.5. ex_prostate_permanent_implant_pegs4.egsinp

Same simulation as ex_prostate_permanent_implant.egsinp (section 10.4.) except that it uses an input .pegs4dat file rather than the input for a pegsless run normally found in a :start media definition: block. The specified .pegs4dat file must contain all the media used by the inserted (geometry) files, in this case Ti, AIR_TG43, AgBrAgI_6.20, Ag, PROSTATE_WW86. The .pegs4dat file is found in: $HEN_HOUSE/pegs4/data/brachy_xcom_1.5MeV.pegs4dat.

10.6. ex_prostate_permanent_implant_TG43.egsinp

A 'TG43' version of the simulation defined in `ex_prostate_permanent_implant.egsinp` (section 10.4.): 100 LDR Amersham OncoSeed 6711 seeds; dose is scored in a rectangular 'PTV' 3.4x2.8x3.8 cm^3 in $(2\ mm)^3$ voxels. The phantom and scoring voxels are water, and there are no interseed effects through the use of the `run mode = superposition` and EGS_ASwitchedEnvelope option for the `egs_autoenvelope` geometry. Note that there is no particle recycling for this file as it is not compatible with `run mode = superposition`.

10.7. ex_mbdca-wg_center.egsinp, ex_mbdca-wg_center_water_only.egsinp, and ex_mbdca-wg_x7cm.egsinp

Test cases of the MBDCA-WG: single MBDCA-WG generic ^{192}Ir source in phantoms as described by Ballester *et al.* [23]:

- `ex_mbdca-wg_center.egsinp`: Single source at the centre of a 20.1 cmx20.1 cmx20.1 cm water phantom comprised of $(1\ mm)^3$ voxels, surrounded by air
- `ex_mbdca-wg_center_water_only.egsinp`: Single source at the centre of a water phantom (no surrounding air): $(1\ mm)^3$ scoring voxels span 20.1 cmx20.1 cmx20.1 cm and total phantom is $(50\ cm)^3$.
- `ex_mbdca-wg_x7cm.egsinp`: Same as previous but source is displaced 7 cm in the x-direction.

10.8. ex_hdr_source_with_applicator.egsinp and ex_hdr_source_with_applicator_create_volume_correction.egsinp

These examples model the generic MBDCA working group Ir-192 source with generic shielded applicator [23, 26]. As the volume occupied by the applicator is relatively large, the volume correction can take several minutes; for parallel runs, the initialization time can be greatly reduced by pre-computing volume correction data using `ex_hdr_source_with_applicator_create_volume_correction.egsinp` and then, in subsequent computations using `ex_hdr_source_with_applicator.egsinp`. See discussion at end of section 7.3..

10.9. ex_PeppaBreastHDR192Ir.egsinp

Simulates 'case 3' of Peppa *et al* [25] using an `egsphant` file giving the patient geometry (derived from distributed DICOM-CT data [25]) with MBDCA-WG generic ^{192}Ir source

stepping through multiple positions with varying dwell times. The dose scaling factor included normalizes output doses (3ddose file) in units of Gy. The source angles are specified.

10.10. ex_xray_sph.egsinp

A simple x-ray source consisting of an isotropic, monoenergetic, electron point source impinging on a thin target tungsten. Electron transport is enabled within the source geometry. Variance reduction techniques are used.

10.11. ex_brem_cyl.egsinp

This is an egs_brachy input file for benchmarking against a DOSRZnrc x-ray source calculations, *i.e.,* it uses the egs_rz geometry. It is a 1 cm^2 parallel beam passing through 1 cm of air and then incident on a 0.0001 cm target of Ti10W90. Tracklength and energy deposition doses are scored in $r = 5.0$ cm / $\Delta z = 0.1$ cm slabs of water extending 4 mm. There is no electron transport in the water so electrons coming from the target deposit all their energy in the first water slab. Note that the tracklength dose in the $z = 0.0$ to 0.1 cm voxel scored by egs_brachy will significantly underestimate this dose since it is not captured. The dose in the air will also be significantly underestimated since only photon dose is captured by tracklength scoring.

11. Helpful hints for egs_brachy simulations

11.1. Geometry preview

It is good practice to preview simulation geometries using egs_view. For simulations with run mode = superposition and employing the type = EGS_ASwitchedEnvelope geometry option, only the first source listed will be shown in the preview. If the user wishes to view all source positions simultaneously, then *temporarily* comment out the entry type = EGS_ASwitchedEnvelope to view all sources or source dwell positions. Note, however, that for HDR treatments with source dwell locations in close proximity, it is possible that there may be overlap of sources previewed. Ensure that the type = EGS_ASwitchedEnvelope geometry option is then uncommented for egs_brachy simulations.

11.2. Use of a wrapper geometry about sources

A wrapper is an RZ geometry, made of water (or other material of choice appropriate for the simulation/phantom), that is slightly larger than the seed in diameter and in length (*e.g.,* 1 μm larger). A wrapper may be used to prevent geometry errors which may occur, *e.g.,* when seed and voxel boundaries end up overlapping. Thus, while not generally necessary, if the user finds that there are many geometry errors due to overlapping boundaries, then

wrappers may be defined directly in the input file (see example egsinp file described in section 10.9.).

11.3. nbatch and nchunk options

EGSnrc applications break simulations into nchunk 'chunks' which are useful during parallel runs since it allows for balancing the computing time taken on each computer, even if they have very different speeds. For simple runs with no parallel processing, by default nchunk=1 whereas for parallel runs, by default nchunk=10. For parallel runs the job is divided into ncase/(npar*nchunk) 'chunks' of histories where ncase is the total number of histories to be done on npar machines in parallel. These 'chunks' are then allocated to the various machines when the previous 'chunk' on that machine is completed. This way faster machines do more 'chunks'.

Within each 'chunk' there are, by default, nbatch=10 'batches' which are not required for statistical analysis, but are checkpoints at which time the current results are written to a data file. This can be very useful since these data files can be used to restart or to analyze partial results if the system stops for any reason during the run. However, for shorter runs with a large number of dose scoring regions, writing these intermediate files can represent a significant part of the run time and setting nbatch=1 can save considerable time.

If all computers being used have the same processing speed, then even if nbatch=1, setting nchunk=1 can also save some time since the data are written to disk at the end of each chunk(i.e., at the end of its last batch) and having only 1 chunk saves time.

11.4. Source orientation

By default, egs_brachy source models are aligned in the positive z-direction, $\hat{k}$, however, the user may wish to model sources with a particular orientation. Supposing that the user knows the source orientation in terms of its direction cosines, *e.g.,* the source orientation is described by a (unit) vector $\vec{v} = \cos a\,\hat{\imath} + \cos b\,\hat{\jmath} + \cos c\,\hat{k}$, one method for specifying source orientation is via three angles (α, β, γ) which are used by egs++ to determine the rotation matrix. These three angles may be determined from the direction cosines as described in the following. Following the convention used in egs++, the rotation matrix is

$$R = R_x(\alpha)R_y(\beta)R_z(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

With sources aligned along the z-axis and models generally symmetric under rotations about the z-axis, we take $\gamma = 0$ and $R_z(\gamma = 0)$ is the identity matrix. Applying the resulting rotation matrix to the unit vector in the z-direction and setting equal to $\vec{v}$, we obtain:

$$R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = R_x(\alpha)R_y(\beta) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -\sin\beta \\ \sin\alpha\cos\beta \\ \cos\alpha\cos\beta \end{bmatrix} = \begin{bmatrix} \cos a \\ \cos b \\ \cos c \end{bmatrix} \tag{9}$$

Hence the angles α, β may be solved via the equations:

$$\alpha = \cos^{-1}\left(\frac{\cos c}{\cos \beta}\right) = \cos^{-1}\left(\frac{\cos c}{\cos\left[\sin^{-1}(-\cos a)\right]}\right), \qquad \beta = \sin^{-1}\left(-\cos a\right). \tag{10}$$

The third equation, $\cos b = \sin \alpha \cos \beta$, may serve as a check of results. The user is encouraged to verify that rotations are as expected via visualization in `egs_view`.

See section 10.9. for an example `egsinp` file involving rotated sources.

12. Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada, Canada Research Chairs program, and Ministry of Research and Innovation of Ontario. The authors thank Compute/Calcul Canada and the Shared Hierarchical Academic Research Computing Network (SHARCNET) for access to computing resources. Thanks to individuals who contributed to `egs_brachy` testing and development: Christian H. Allen, Stephen Deering, Gavin Hurd and Martin Martinov.

References

[1] M. Chamberland, R. E. P. Taylor, D. W. O. Rogers, and R. M. Thomson, egs_brachy: a versatile and fast Monte Carlo code for brachytherapy, Phys. Med. Biol. **61**, 8214 – 8231 (2016).

[2] I. Kawrakow, E. Mainegra-Hing, F. Tessier, R. Townson, and B. R. B. Walters, The EGSnrc C++ class library, Technical Report PIRS–898(2017), National Research Council Canada, Ottawa, Canada. http://nrc-cnrc.github.io/EGSnrc/doc/pirs898/index.html, 2017.

[3] B. R. B. Walters, I. Kawrakow, and D. W. O. Rogers, DOSXYZnrc Users Manual, NRC Report PIRS 794 (rev B) (2005).

[4] I. Kawrakow, Accurate condensed history Monte Carlo simulation of electron transport. I. EGSnrc, the new EGS4 version, Med. Phys. **27**, 485 – 498 (2000).

[5] I. Kawrakow, E. Mainegra-Hing, D. W. O. Rogers, F. Tessier, and B. R. B. Walters, The EGSnrc code system: Monte Carlo simulation of electron and photon transport, Technical Report PIRS-701, National Research Council Canada, 2017.

[6] I. Kawrakow, E. Mainegra-Hing, F. Tessier, and B. R. B. Walters, The EGSnrc C++ class library, Technical Report PIRS–898 (rev A), National Research Council Canada, Ottawa, Canada. http://www.irs.inms.nrc.ca/EGSnrc/PIRS898/, 2009.

[7] L. Beaulieu, A. C. Tedgren, J.-F. Carrier, S. D. Davis, F. Mourtada, M. J. Rivard, R. M. Thomson, F. Verhaegen, T. A. Wareing, and J. F. Williamson, Report of the Task Group 186 on model-based dose calculation methods in brachytherapy beyond the TG-43 formalism: Current status and recommendations for clinical implementation, Med. Phys. **39**, 6208 – 6236 (2012).

[8] R. M. Thomson, Å. Carlsson Tedgren, and J. F. Williamson, On the biological basis for competing macroscopic dose descriptors for kilovoltage dosimetry: cellular dosimetry for brachytherapy and diagnostic radiology, Phys. Med. Biol. **58**, 1123 – 1150 (2013).

[9] J. F. Williamson, Monte Carlo evaluation of kerma at a point for photon transport problems, Med. Phys. **14**, 567 – 576 (1987).

[10] R. E. P. Taylor and D. W. O. Rogers, EGSnrc Monte Carlo calculated dosimetry parameters for ^{192}Ir and ^{169}Yb brachytherapy sources, Med. Phys. **35**, 4933 – 4944 (2008).

[11] B. R. B. Walters, I. Kawrakow, and D. W. O. Rogers, History by history statistical estimators in the BEAM code system, Med. Phys. **29**, 2745 – 2752 (2002).

[12] K. R. Russell, A. K. Carlsson-Tedgren, and A. Ahnesjö, Brachytherapy source characterization for improved dose calculations using primary and scatter dose separation, Med. Phys. **32**, 2739 – 2752 (2005).

[13] R. Capote, R. Jeraj, C.-M. Ma, D. W. O. Rogers, F. Sánchez-Doblado, J. Sempau, J. Seuntjens, and J. V. Siebers, Phase-space Database for External Beam Radiotherapy: Summary Report of a Consultants' Meeting, Technical Report INDC(NDS)-0484, International Nuclear Data Committee, INDC, IAEA, Vienna, Austria, 2006.

[14] A. B. Chilton, A Note on the Fluence Concept, Health Phys. **34**, 715 – 716 (1978).

[15] A. B. Chilton, Further Comments on an Alternative Definition of Fluence, Health Phys. **36**, 637 – 638 (1979).

[16] E. S. M. Ali and D. W. O. Rogers, Efficiency improvements of x-ray simulations in EGSnrc user-codes using Bremsstrahlung Cross Section Enhancement (BCSE), Med. Phys. **34**, 2143 – 2154 (2007).

[17] M. Rodriguez and D. W. O. Rogers, On determining dose rate constants spectroscopically, Med. Phys. **40**, 011713 (10pp) (2013).

[18] M. J. Rivard, B. M. Coursey, L. A. DeWerd, M. S. Huq, G. S. Ibbott, M. G. Mitch, R. Nath, and J. F. Williamson, Update of AAPM Task Group No. 43 Report: A revised AAPM protocol for brachytherapy dose calculations, Med. Phys. **31**, 633 – 674 (2004).

[19] ICRU, Dosimetry of Beta-Rays and Low-Energy Photons for Brachytherapy with Sealed Sources, ICRU Report 72, ICRU, Washington D.C., 2004.

[20] NCRP Report 58, A Handbook of Radioactivity Measurements Procedures, NCRP Publications, 7910 Woodmont Avenue, Bethesda, MD. 20814 USA (1985).

[21] Brookhaven National Laboratory, National Nuclear Data Center, http://www.nndc.bnl.gov/nudat2.

[22] B. Duchemin and N. Coursol, Reevaluation de l'^{192}Ir, Technical Note LPRI/93/018, DAMRI, CEA, France (1993).

[23] F. Ballester *et al.*, A generic high-dose rate ^{192}Ir brachytherapy source for evaluation of model-based dose calculations beyond the TG-43 formalism, Med. Phys. **42**, 3048 – 3062 (2015).

[24] R. E. P. Taylor, G. Yegin, and D. W. O. Rogers, Benchmarking BrachyDose: voxel-based EGSnrc Monte Carlo calculations of TG–43 dosimetry parameters, Med. Phys. **34**, 445 – 457 (2007).

[25] V. Peppa, E. Pantelis, E. Pappas, V. Lahanas, C. Loukas, and P. Papagiannis, A user-oriented procedure for the commissioning and quality assurance testing of treatment planning system dosimetry in high-dose-rate brachytherapy, Brachytherapy **15**, 252–262 (2016).

[26] Y. Ma *et al.*, A generic TG-186 shielded applicator for commissioning model-based dose calculation algorithms for hig-dose-rate ^{192}Ir brachytherapy, Med. Phys. **44**, in press (2017).

Index