BEAMnrc Users Manual

D.W.O. Rogers, B. Walters, I. Kawrakow

Ionizing Radiation Standards
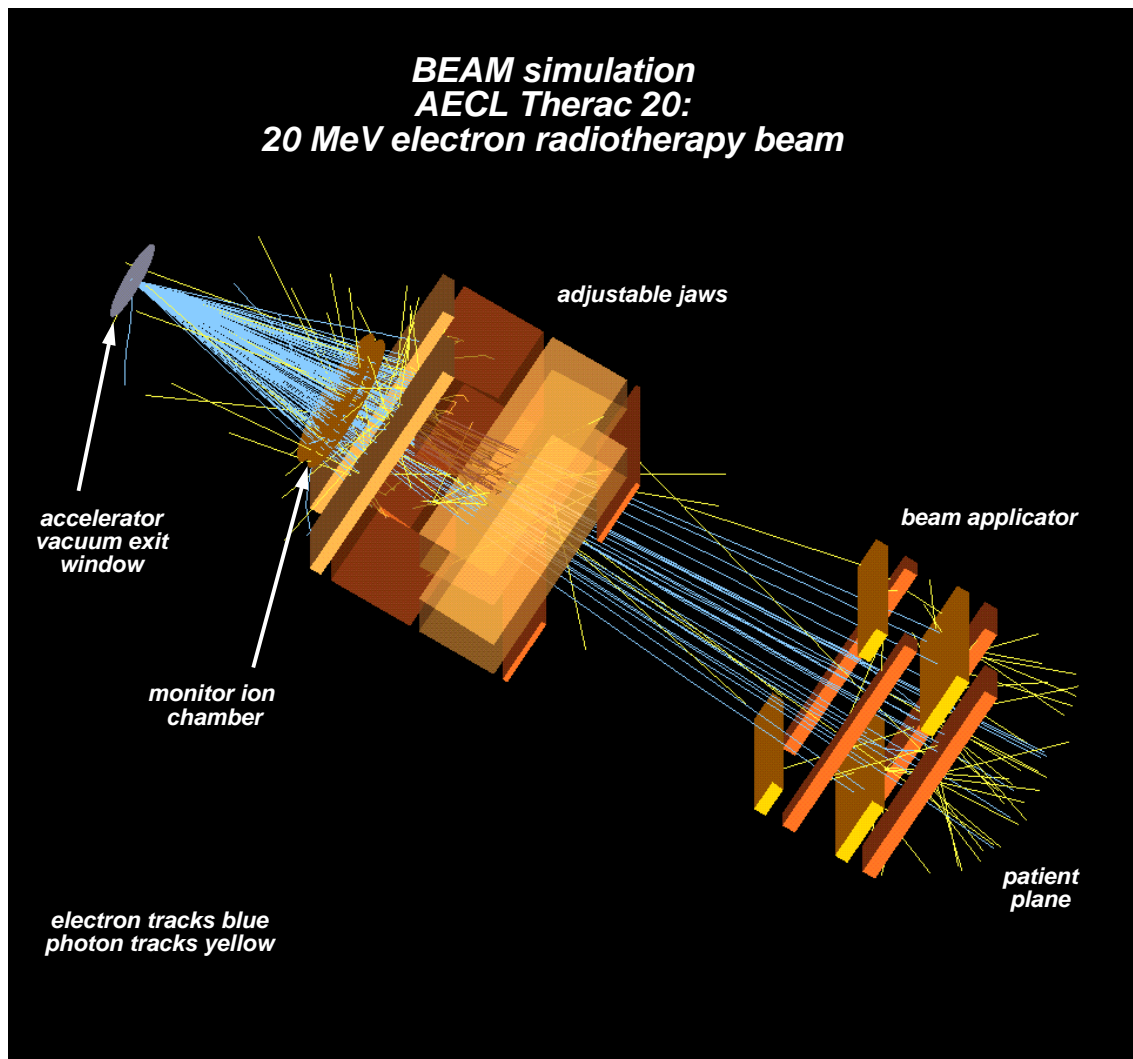National Research Council of Canada
Ottawa, K1A OR6
drogers at physics.carleton.ca

Printed: October 9, 2012(last edited 2012/10/09 17:59:26)

Source tex file is: $OMEGA_HOME/doc/pirs0509/pirs0509.tex
Available on-line via:
http://irs.inms.nrc.ca/software/beamnrc/

BEAM Code System General Licence

The BEAM code system (i.e. all pieces of code saying they are subject to the BEAM General License, including routines related to the 1999 and later versions of BEAM or BEAMnrc, DOSXYZ or DOSXYZnrc, BEAMDP, ctcreate and EGS_Windows, as well as the CMs (BEAM component modules), scripts, all the auxiliary routines except those from the EGS4 Monte Carlo system and all associated documentation) are copyrighted material owned by the National Research Council of Canada, all rights reserved.

1) The NRC grants the user a non-transferable, non-exclusive licence to use this system free of charge only for non-commercial research or educational purposes. All proprietary interest, right, title and copyright in the BEAM code system remains with NRC.

2) The express, written consent of NRC is required if the BEAM code system or any part thereof is used by an individual or an organization for developing a commercial product or service, or data generated by the code is used to develop commercial products or services.

3) The express, written consent of NRC is required if this code system or any part thereof is to be used in a fee for service application, either clinically or by consultants.

4) The code system must be obtained from NRC. Users may not copy nor distribute the code system or parts thereof.

5) NRC disclaims any warranties, expressed, implied or statutory, of any kind or nature with respect to the software, including without limitation any warranty of merchantability or fitness for a particular purpose. NRC shall not be liable in any event for any damages, whether direct or indirect, special or general, consequential or incidental, arising from the use of the software.

6) This licence supersedes all prior communications, negotiations and agreements, written or oral, concerning the BEAM code system. No amendment or waiver of terms is effective unless in writing, signed by both parties and specifically states the intention to affect this licence.

7) This licence is governed by the laws of Ontario and Canada applicable therein. The user consents to the jurisdiction of the federal court and the courts of Ontario.

——————- End of Licence ——————————-

Requests:
Please report all bugs (and corrections if possible) to `drogers at physics.carleton.ca`. Corrections will be patched into the code and explicit mention given in the documentation to the person providing a workable solution.

Abstract

BEAMnrc is a Monte Carlo simulation system (Med. Phys. **22**,1995,503 – 524) for modelling radiotherapy sources which was developed as part of the OMEGA project to develop 3-D treatment planning for radiotherapy (with the University of Wisconsin). BEAMnrc is built on the EGSnrc Code System[1]. Until 2004, BEAMnrc could only be run on Unix/Linux systems. However, with the recent port of BEAMnrc to the EGSnrcMP system[2], BEAMnrc can also run on Windows-based systems. The purpose of this manual is to be a reference/guide to someone using the BEAMnrc system.

This user's manual covers general BEAMnrc inputs and component module (CM) geometries and inputs. It discusses how to use the various variance reduction techniques which are part of the system, most importantly, range rejection, bremsstrahlung splitting, photon forcing in a specific region and Russian Roulette. It also covers the structure of the directory system used for the BEAMnrc system, the utility codes available (readphsp, addphsp, checkCM8 *etc.*), the installation procedure, the phase space file definition and it has cross references to all related BEAMnrc documentation. Appendix A gives a specification for writing new component modules.

Contents

5 Monoenergetic vs Energy Spectrum Sources

6 Variance Reduction in BEAMnrc

7 Phase Space Files

8   Tracking a Particle's History using LATCH   104

9   Calculating Dose Components   108

10   Other Input Variables   109

11   EGSnrc inputs   115

List of Figures

1   Overview of BEAMnrc

An extensive paper describing the BEAM system and its application has been published (ref [3]) and it is assumed that the user is familiar with that paper which is also available online at http://www.physics.carleton.ca/ drogers/pubs/papers/Ro95.pdf. We draw attention to the index in the present document since a great deal of effort has gone into making it and we find it a useful way to find information in this manual. This manual was originally written before the BEAM graphical user interface (GUI) was developed[4] and, thus, those sections describing input formats are somewhat redundant since the GUI makes the formatting of the input much easier. The EGSnrc system[5, 1, 6] was released in February 2000 and the BEAM code system was ported to using EGSnrc in 2001, used at the course in October 2001 and generally released in Feb 2002. In spring 2004 a version of the code was released to course participants with Directed Brem Splitting available. In fall 2004 this version of the code was made available generally with a port to the EGSnrcMP system which allows the code system to be run on Windows systems as well as the traditional Unix/Linux systems. There is a 'new version' of this manual with each release of the code (usually with the annual course) and there have been five major revisions[7, 8, 9, 10, 11] with changes in authorship to represent those primarily responsible for the current version.

1.1   The Physics in BEAMnrc

BEAMnrc uses the EGSnrc Monte Carlo system of radiation transport. The physics in EGSnrc is described in detail in the EGSnrc manual [1]. The transport physics in EGSnrc is greatly improved over that in EGS4 (the basis for all BEAM versions up to and including BEAM00). Among the improvements are the ability to include relativistic spin effects in elastic scattering cross-sections for electrons, the ability to simulate atomic relaxations after Compton and photoelectric events and improved electron transport and multiple scattering algorithms. All these improvements increase the accuracy of EGSnrc over EGS4, especially at lower energies. The user has control over the extent to which the new physics is used in a simulation and also over some parameters required for the new physics. This means there are additional inputs for BEAMnrc that did not exist in previous versions of BEAM. On the other hand, the introduction of EGSnrc has eliminated the need for PRESTA and its associated inputs. Also, the EGSnrc inputs supersede some of the main BEAM inputs (such as IDORAY for controlling Rayleigh scattering, and IFLUOR for turning on K-shell X-ray fluorescence). See section 11 for a full description of the EGSnrc inputs and a more detailed description of the improved physics. The default EGSnrc parameters used by BEAMnrc were selected to balance efficiency and accuracy for typical accelerator simulations so, *eg.*, the EGSnrc default EXACT boundary crossing algorithm (BCA) is used instead of the faster PRESTA-I BCA because the latter has been shown to result in significant overestimates of dose when the CHAMBER component module is used as a phantom[12], however, atomic relaxations in BEAMnrc default to "off" (default in EGSnrc is "on") because their effect is insignificant at accelerator energies.

BEAMnrc retains compatibility with older BEAM input files that do not include EGSnrc inputs. In this case, the EGSnrc parameters simply revert to the default values used in BEAMnrc which differ from the EGSnrc defaults (see section 11).

1.2   Other documents available

There are several other documents which describe associated subjects.   Many of these documents are available on-line via links from the BEAMnrc home page. The BEAMnrc home page URL is `http://irs.inms.nrc.ca/software/beamnrc/`.

BEAMnrc, DOSXYZnrc and BEAMDP GUI User's Manual: Describes how to install and use these graphical user interfaces to BEAMnrc and related codes[4].

BEAMDP as a General-Purpose Utility: User's manual for using BEAMDP to do simple analysis of phase space files - presenting spectra, fluence distributions, average energies, listings, angular distributions *etc.*, including the possibility of using the `LATCH` values in the phase space file[13].

QA for the BEAMnrc System: Component Modules, Variance Reduction Options and Source Routines. This 100 page document is for internal use at NRC but describes the extensive QA program carried out on component modules, variance reduction options and source routines. It also describes the automated system for on-going QA[14].

DOSXYZnrc User's Manual: DOSXYZnrc is an associated code for doing dose distribution studies in a CT voxel phantom irradiated by a beam calculated using BEAMnrc[15].

Specifications for Component Modules for BEAMnrc: This document is for code developers who need to know what is going on in the code so they can write a new CM. It is attached as Appendix A(section A.1).

EGS_Windows_4.0 User's Manual: User's manual for an X-windows based display tool for EGS histories and BEAMnrc geometries[16].

The EGSnrc Code System Manual- PIRS-701: Complete manual describing the use of the EGSnrc simulation system[1].

EGSnrcMP: the multi-platform environment for EGSnrc- PIRS-877: Manual describing the EGSnrcMP system[2]. Essential reading to understand how the current version of BEAMnrc is compiled and run.

History by history statistical estimators in the BEAM code system: Med Phys **29** (2002) 2745–2752: In-depth description of the method used to estimate uncertainty in BEAMnrc and DOSXYZnrc[17].

Large efficiency improvements in BEAMnrc using directional brem splitting: Med Phys **31** (2004) 2883 – 2898. Describes major improvement in efficiency for photon beam simulations[18].

1.3   Overview of the directory structure

The OMEGA/BEAM system has a well defined structure of directories. It can be thought of as having two or possibly three general parts. The first sub-system is generally referred

to as `$OMEGA_HOME` and contains all the source code needed to run BEAMnrc and associated codes such as readphsp, BEAMDP *etc.*. In practice `$OMEGA_HOME` is the directory `$HEN_HOUSE/omega` (more about `$HEN_HOUSE` below). Figure 1 outlines the `$OMEGA_HOME` subsystem. Note that this part of the system contains no execute modules related to beam simulation. These all reside on the user's `$EGS_HOME` area described below.
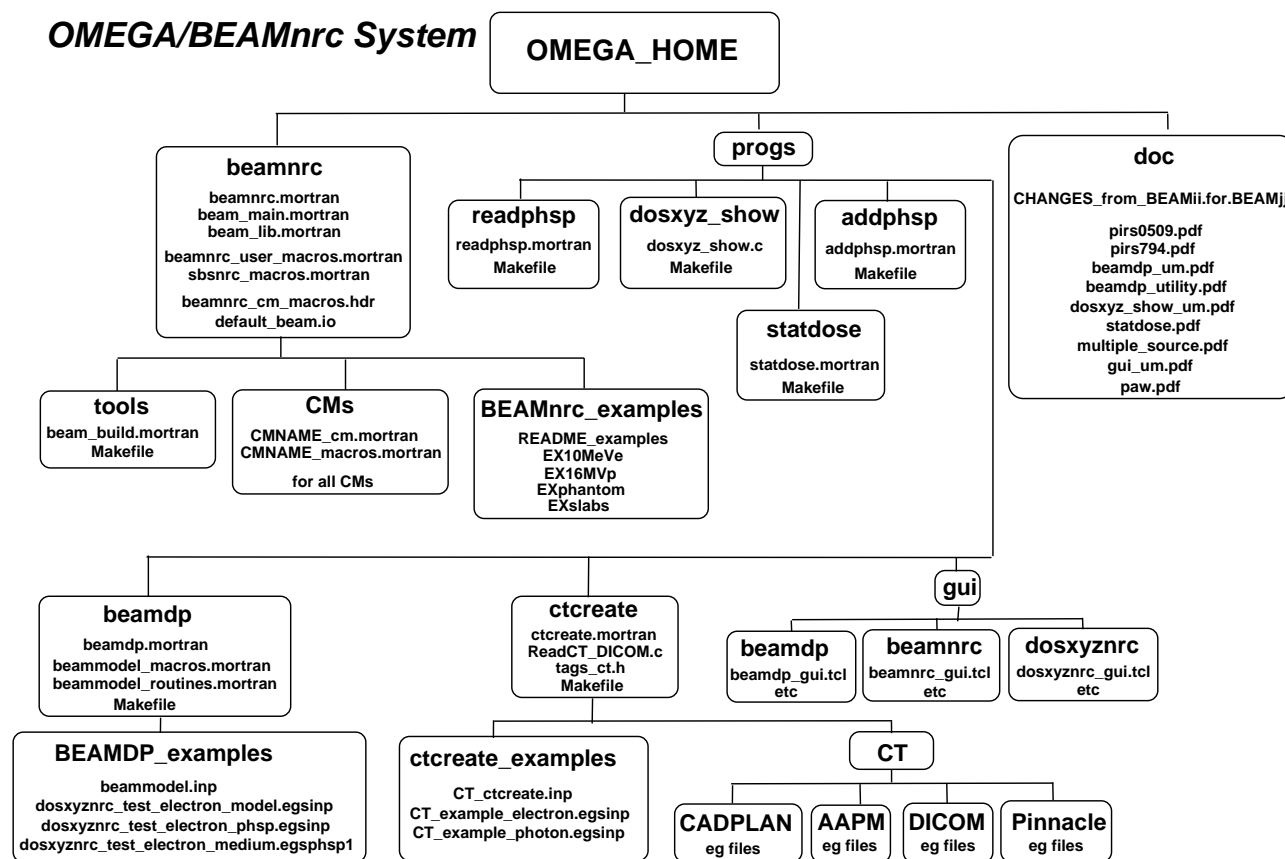
Figure 1: Main components of the `$OMEGA_HOME` directory. `$OMEGA_HOME` is a subdirectory, called `omega`, of `$HEN_HOUSE` which is given in fig 2

`$OMEGA_HOME` resides (as directory `omega`) within the `$HEN_HOUSE`, which contains the EGSnrcMP system. This is shown in more detail in fig 2. Originally, EGSnrc was a Unix-based script system developed primarily by Alex Bielajew and Dave Rogers at NRC. With the advent of EGSnrcMP[2] and the requirement for compiling on Windows-based systems in addition to Unix-based systems, we have largely eliminated the scripts and, in the case of compilation, replaced them with the GNU `make` utility. For more detail about the EGSnrcMP system and how it works, see the EGSnrcMP manual [2]. For users of the previous BEAM/BEAMnrc systems, note that in the past the `$HEN_HOUSE` was a component of `$OMEGA_HOME` since the EGS system was distributed as a component of BEAM whereas in the EGSnrcMP system, BEAMnrc is a special subcomponent of `$HEN_HOUSE` and DOSXYZnrc is treated as just another user-code.

The final component of the OMEGA/BEAM structure is the user's area which is shown in fig 3. This is known as `$EGS_HOME` and is usually the subdirectory `$HOME/egsnrc_mp`. The

HEN_HOUSE   *EGSnrcMP System*

bin

lib

makefiles
beam_makefile
standard_makefile

config

mortran3.dat
mortran3.exe
beam_build.exe
pegs4.exe
beamdp*
other executables

config

machine.mortran
machine.macros

egs_config1.h
egs_c_utils.o
read_write_pardose.o

specs

config1.conf
config2.conf
unix.spec
windows.spec

all_common.spec
beamnrc.spec
dosxyznrc_config1.spec
dosxyznrc_config2.spec

scripts

egsnrc_cshrc_additions
egsnrc_bashrc_additions
beamnrc_bashrc_additions
beamnrc_cshrc_additions

mortran_compile*
run_user_code*
run_user_code_batch*

batch_options.at
batch_options.nqs
batch_options.pbs

switch_config_bashrc*
switch_config_cshrc*

finalize_egs_foruser
finalize_beam_foruser

src

egsnrc.mortran
egs_utilities.mortran
get_inputs.mortran
egs_parallel.mortran
ranmar.mortran
ranlux.mortran
lnblnk1.mortran

egsnrc.macros
ranmar.macros
ranlux.macros
transportp.macros

utils

nrcaux.mortran
timing.macros
xvgrplot.mortran
phsp_macros.mortran
iaea_phsp_macros.mortran

**one for each
config supported**

omega
See description
of OMEGA_HOME

user_codes

pegs4

inputs
*.pegs4inp

data
*.pegs4dat

**density_
corrections**
compounds
elements

data

incoh.data
msnew.data
nist_brems.data
photo_cs.data
photo_relax.data
spinms.data
eii_ik.data

mortran3

Makefile
mortran3.f
mornew77.raw
check77.raw

spectra

bareco60.spectrum
mohan24.spectrum
ir192bare.spectrum

other ensrc format
files

cutils

egs_c_utils.c
egs_c_utils.h
load_beamlib.c

iaea_phsp

iaea_config.h
iaea_header.cpp
iaea_header.h
iaea_phsp.cpp
iaea_phsp.h
iaea_record.cpp
iaea_record.h
utilities.cpp
utilities.h

Makefile

iaea_phsp.a

dosxyznrc

Makefile
dosxyznrc.make

dosxyznrc.mortran
srcxyznrc.mortran

dosxyznrc_user_macros.mortran
srcxyznrc.macros
dosxyznrc.io

DOSXYZnrc_examples

Figure 2: The structure of the EGSnrc system. Note that the $OMEGA HOME subsystem shown in fig 1 is included in this structure.

BEAM installation will set up much of this automatically if it is not in place. One of the

Figure 3: The user's $EGS_HOME area (usually $HOME/egsnrc_mp). A $*$ indicates an executable file. In the example shown, the user has 3 accelerator models and the disk system is being used with three different configurations, *eg.*, gcc, pgf77 and Windows. The complete directory contents are only shown for accel2 and config2. Other EGSnrc user codes (eg DOSXYZnrc) also reside in the $EGS_HOME area.

main complications is that the EGSnrcMP and BEAMnrcMP systems are set up to use one disk system to support multiple configurations and their associated compilers.   Thus, all execute modules and various compiler options *etc.*   must be handled separately for each configuration. Within the `bin` area, the modules within each subdirectory apply only to the configuration shown.

1.4   Overview of running BEAMnrc

Figure 4 presents a schematic of the overall steps required to do an accelerator simulation. At the specify accelerator and build accelerator steps, the user is instructing the system how to pull together the source code and make an execute module. At this stage, only a   broad

BEAMnrc

```
                    ┌─────────────────┐
                    │     Specify     │
                    │   Accelerator   │
                    └─────────────────┘
                             │
                             ▼
                    ┌─────────────────┐
                    │  Build/Compile  │
                    │   Accelerator   │
                    └─────────────────┘
                             │
   ┌──────────────┐          ▼
   │ cross-section│─────▶┌──────────────┐
   │     data     │      │ Do Simulation│
   └──────────────┘  ┌──▶└──────────────┘
   ┌──────────────┐  │          │
   │users input file│─┘          │
   │ -geometry    │              │
   │ -incident beam│             │
   │ -output spec │              │
   │ -simulation  │              │
   │  parameters  │              │
   └──────────────┘              │
             ┌──────────────────┼──────────────────┐
             ▼                  ▼                  ▼
      ┌─────────────┐   ┌─────────────┐   ┌─────────────┐
      │ phase space │   │   output    │   │  graphics   │
      │    files    │   │   listing   │   │             │
      └─────────────┘   └─────────────┘   └─────────────┘
          ┌─────┴─────┐
          ▼           ▼
   ┌─────────────┐ ┌─────────────┐
   │  analysis   │ │   patient   │
   │  programs   │ │ simulation  │
   └─────────────┘ └─────────────┘
```
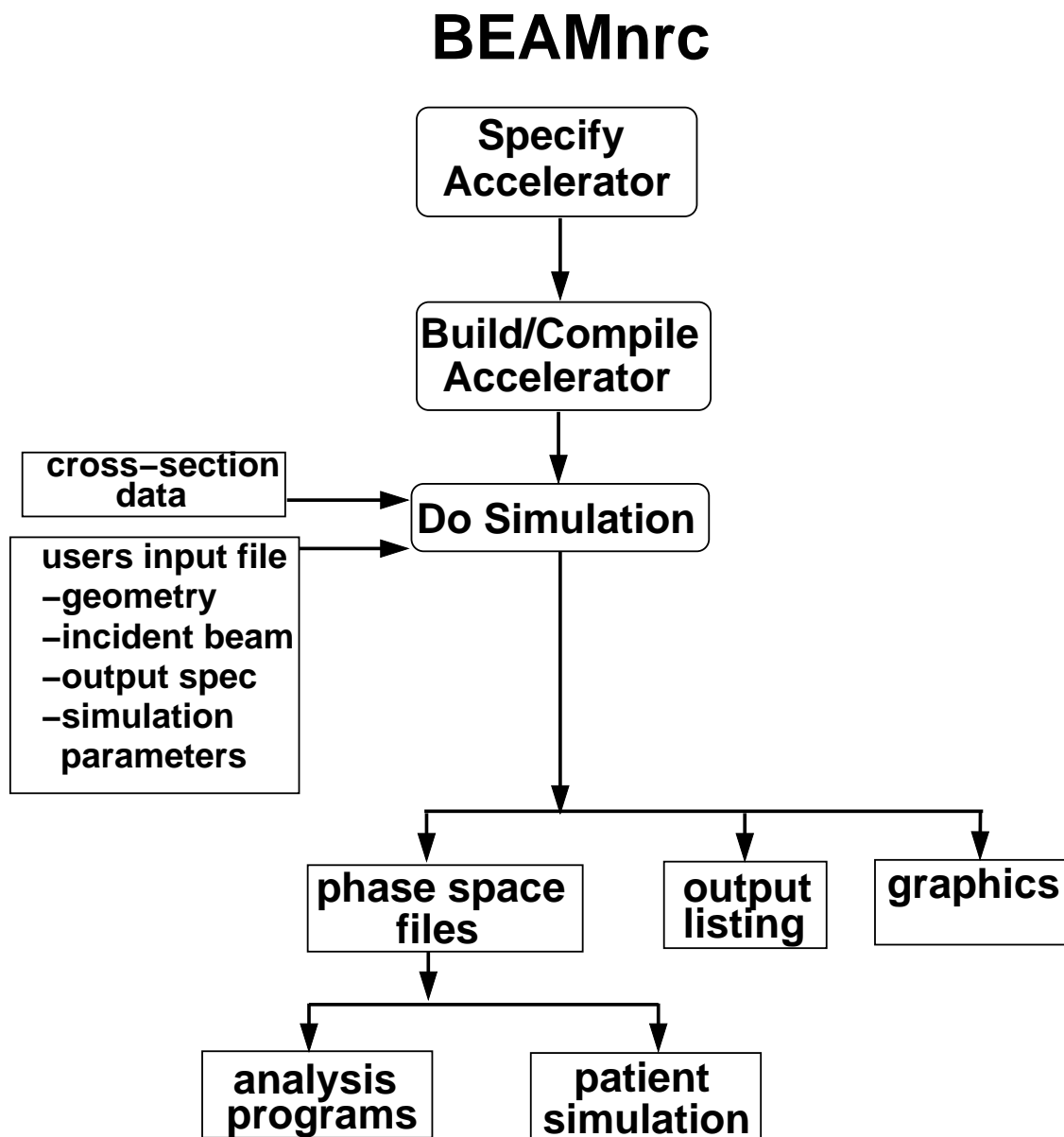
Figure 4: The steps involved in using the BEAMnrc system, from ref [3].

class of accelerator is defined (*eg.* whether the flattening filter is before or after the primary collimator). During the execution stage, the program reads in a large quantity of data related to photon and electron cross sections for the specific materials in this accelerator model. These data are generated by a code, called PEGS4 (which is included with the EGSnrc system see section 16) or read from data files such as `msnew.data`, `spinms.data`, etc, which are contained in the `$HEN_HOUSE/data` subdirectory. Also the user creates an input

file which specifies all the details about the particular accelerator (*eg.*, there are 4 scattering foils located at specified distances from the vacuum exit window, made of certain materials and of particular thickness). Also the user must specify all the parameters controlling the radiation transport modelling and must also select and control the various variance reduction techniques being used.

The final stage of the simulation is the analysis of the outputs which consist of raw phase space files (measured in tens if not hundreds of Mbytes), an output listing and optionally a 3D graphics file to be displayed by EGS_Windows[16].

2   Building/compiling/running BEAMnrc

See section 17 at the end of this manual for instructions for installing BEAMnrc and the rest of the OMEGA/BEAM codes.

If you have installed BEAMnrc on a Linux/Unix system then the first step towards running the code is to go into your $HOME/.cshrc or $HOME/.bashrc (depending on whether you're running in a C-shell or Bourne-shell, respectively) file and, at the end of the file, add the following statements:

```
setenv EGS_HOME /full directory path to $HOME/egsnrc_mp/
(or export EGS_HOME=/full directory path to $HOME/egsnrc_mp for .bashrc)
setenv EGS_CONFIG /full directory path to $HEN_HOUSE/specs/config.conf/
(or export EGS_CONFIG=/full directory path to $HEN_HOUSE/specs/config.conf/
                                                    for .bashrc)
source /full directory path to $HEN_HOUSE/scripts/egsnrc_cshrc_additions
(or source /full directory path to $HEN_HOUSE/scripts/egsnrc_bashrc_additions
                                                    for .bashrc)
source /full directory path to $HEN_HOUSE/scripts/beamnrc_cshrc_additions
(or source /full directory path to $HEN_HOUSE/scripts/beamnrc_bashrc_additions
                                                    for .bashrc)
```

where config is the name of the particular configuration (eg gcc, pgf77) that you have installed BEAMnrc on. The first three statements are required for the EGSnrcMP system (see the EGSnrcMP Users Manual[2]), while $HEN_HOUSE/scripts/beamnrc_cshrc_additions (or $HEN_HOUSE/scripts/beamnrc_cshrc_additions) define aliases for the OMEGA/BEAM system. Once you have added these statements you must source the .cshrc (or .bashrc) file to bring them into effect.

Instructions for adding these statements are given explicitly at the end of the EGSnrcMP and BEAMnrc installations (see section 17)

No such additions are required when installing BEAMnrc on a Windows system.

Previous to the current version of BEAM (BEAMnrcMP), the user had the option to specify, build, compile and run a BEAM simulation from within a Linux/Unix script, called beamnrc. This script was eliminated with the advent of BEAMnrcMP as it was seen to be of limited use for a system that is also required to work on Windows (which does not support scripts). However, you will note in the descriptions that follow that all of the functions of

the `beamnrc` script can be performed (in a much more user-friendly manner) from within the BEAM GUI.

2.1   "Specifying" Accelerators

Before compiling and running a BEAM accelerator simulation, you must "specify" which component modules are to be used and in what order. The available `CMs` are discussed in detail in section 15. Each CM can be used in a wide variety of applications and user should not restrict themselves by the names. For example, the JAWS CM is well suited to simulating a wedge. It is useful to select identifiers (8 characters or less) which are physically meaningful (eg xit_win, scatfoil, jaws *etc.*) since these will appear throughout the code and listing files and thereby help the user understand what is going on in various applications. The user may use the same CM as often as needed, as long as the identifiers used are unique. One restriction is that the identifier name should not start with the word "exit" since the `MORTRAN` compiler gets confused with this.

Accelerators are stored in a very simple format in `myaccel.module` files that reside in the user's `$EGS_HOME/beamnrc/spec_modules` subdirectory (see Figure 3 above). When built and compiled, the executable code will have the name `BEAM_myaccel`, so it is useful to have `myaccel` indicate the machine you are modeling.

The `myaccel.module` file created from scratch using the BEAM GUI (just select "Specify a new accelerator" from the "File" menu), or you can copy an existing `.module` file (eg one of the examples included with the distribution) into `myaccel.module` and modify the latter for your own purposes using an editor.

Any number of specified accelerators can coexist at the same time.

2.2   Building an Accelerator: The `beam_build` Code

Once the accelerator has been specified, it must be "built", which corresponds to concatenating all of the relevant source code for the `CMs` and editing it to avoid duplicate variable names (this is discussed below for those who need details). The resulting files are on the user's area and are `$EGS_HOME/BEAM_myaccel/BEAM_myaccel_cm.mortran` and `$EGS_HOME/BEAM_myaccel/BEAM_myaccel_macros.mortran`. Note that the name of the subdirectory and the file names themselves are just the name of the specification module with the prefix `BEAM_`.

If you are modifying the code at all, an accelerator must be re-built every time any of the `CMs` are modified, but not when the main `beamnrc.mortran` code is modified.

An accelerator is built using the MORTRAN code `beam_build.mortran`. The command for running this code to build the accelerator specified by `myaccel.module` is:

```
beam_build.exe myaccel
```

`beam_build.exe` creates the `BEAM_myaccel_cm.mortran` and `BEAM_myaccel_macros.mortran` files and puts them in the `BEAM_myaccel` subdirectory (creating the subdirectory on the way

if it does not already exist).

In addition, `beam_build` creates the files `Makefile`, `modules.make` and `sources.make` and puts them in the `BEAM_myaccel` subdirectory. These files are used for compiling beam. More details are given in section 2.12.3, on page 31, below.

If the `beam_build` utility has not already been compiled for your particular configuration (this may be the case if you have switched configurations after installing BEAMnrc), then before typing the above command line, go into `$HEN_HOUSE/omega/beamnrc/tools/beam_build` and type `make`. This will put a copy of the `beam_build.exe` executable in the appropriate subdirectory of `$HEN_HOUSE/bin`.

2.3   Compiling an Accelerator using `make`

Once specified and built, the accelerator must be compiled. This requires using the `MORTRAN` compiler (which is supplied as part of the EGSnrc system) followed by the `FORTRAN` compiler.

In previous versions of BEAM, compiling of accelerators was completely driven by Unix scripts. However, with the necessity to run on non-Unix platforms, EGSnrcMP now uses the GNU `make` utility. For more detailed information on `make`, see the EGSnrcMP manual [2].

To compile an accelerator, go into the `BEAM_myaccel` subdirectory and type:

`make [options]`

The options for `make` are:

```
make                Compile the accelerator executable with default optimization
make opt            turned on.  Default optimization is level 3 (-O3).

make noopt          Compile the accelerator executable with optimization
                    turned off.

make debug          Compile an accelerator executable for debugging.

make fortran        Do mortran compilation only, leaving behind the fortran
                    file BEAM\_myaccel\_config.F.

make clean          Remove the fortran file, mortjob file, mortlst file
                    and the various executables.
```

Upon typing `make` you will see output to the screen indicating files that are being concatenated together to create the final .mortran code that is then compiled.

After compiling an accelerator, the following files will be left behind:

In the accelerator directory:

    `mortjob.mortran` The concatenated MORTRAN file which is MORTRAN compiled.

BEAM_myaccel_config.mortlst Listing from the MORTRAN compiler. Contains the concatenated MORTRAN code, formatted with levels of loops and IF blocks indicated. Output at the end of the file indicates if there were any MORTRAN errors and, if so, how many and of what type.

BEAM_myaccel_config.F Fortran output from the MORTRAN compiler. This is the f77 code that is ultimately Fortran compiled. The use of capital "F" indicates that some C routines are to be linked with the Fortran code at compile time. In the case of BEAM, these C routines pertain to parallel job submissions.

In the $EGS_HOME/bin/config directory:

BEAM_myaccel* The executable code.

If none of the MORTRAN source coding or macros have changed since the last compilation and the file BEAM_myaccel_config.F exisits in the accelerator directory and there is an executable in your $EGS_HOME/bin/config directory, then typing make will not recompile the accelerator.

If you want to include the beam characterization source in the accelerator, see the instructions in section 4.16 page 83.

2.3.1   Compiling an Accelerator as a Shared Library

DOSXYZnrc[15] and other EGSnrc user codes[19] allow the user to use a full BEAM simulation as a particle source (instead of a stored phase space file). In order to use this source, the BEAM accelerator code must first be compiled as a shared library.

Once an accelerator has been built, it can be compiled as a shared library by going into $EGS_HOME/BEAM_myaccel and typing:

```
make library
```

The codes concatenated to create the mortjob.mortran file are the same as those used for a standard BEAM compilation with the exception that
$OMEGA_HOME/beamnrc/beam_lib.mortran is used in place of
$OMEGA_HOME/beamnrc/beam_main.mortran (see section 2.12.4, page 33). The library compilation also leaves behind libBEAM_myaccel_config.mortlst (output from MORTRAN compiler) and libBEAM_myaccel_config.F (Fortran source code), where config is the name of your configuration.

The BEAM shared library is archived as the file libBEAM_myaccel.so (for unix/Linux machines) or BEAM_myaccel.dll (for Windows machines) in directory $EGS_HOME/bin/config.

Note that previous versions of BEAMnrc required the static g2c library, libg2c.a, when compiling shared libraries on Unix/Linux machines. Use of this library prevented confusion of I/O units between BEAM and the code using BEAM as a source (the "driving" code). In the current version of BEAMnrc, however, the opening of I/O units has been recoded so that only those units not already being used by the driving code are available to BEAMnrc.

When using the BEAM shared library as a source, you must also supply a working input file and the pegs data for the BEAM simulation. The input file must exist in your $EGS HOME/BEAM myaccel directory and be set up to write a phase space file at a scoring plane. See the DOSXYZnrc Manual[15] and the manual for EGSnrc user codes[19] for more details about this source.

2.4   Compiling an Accelerator using `mf` (Unix/Linux specific)

To preserve compatibility with old usage, an accelerator can also be compiled using `mf`, which is aliased to the Unix script `$HEN HOUSE/scripts/compile user code`. To compile using `mf`, go into the `BEAM myaccel` directory and type:

```
m[f] BEAM_myaccel [a] [opt|noopt|debug]
```

The options for `mf` are: `mf`   => Mortran and Fortran compile and then link
`m`   => Mortran compile and create the Fortran file
`opt`   => use optimization
`noopt`   => use no optimization
`debug`   => create executable ready for a debug run
The parameter "`a`" is not used and is only present for compatibility with the previous version of `mf`.

Once the `mf` command is issued, the `compile user code` script then calls `make` with the appropriate options.

2.5   Building/Compiling with the BEAM GUI

Building and compiling are done in a single step when using the BEAM GUI. Once an accelerator `.module` file has been specified or read into the GUI, then it can be built/compiled by selecting "Compile" from the "Execute" menu. This opens up a window in which you can choose any of the `make` options outlined in section 2.3 above. Then press the "BUILD & COMPILE" to run `beam build` followed by `make`. If there is no `beam build.exe*` executable in the appropriate `$EGS HOME/bin/config` directory, then the GUI will automatically compile `beam build` for your configuration.

2.6   Internal Documentation and Input Description

Since a given accelerator model can have an arbitrary structure, defining the input file for a given accelerator must be done after the accelerator is specified. Extensive descriptions of the main BEAM inputs and individual CM inputs are given at the top of the `beamnrc.mortran` and `CM cm.mortran` codes, however the BEAMnrc graphical user interface (GUI) provides extensive descriptions of each input variable in addition to being able to display graphical representations of component module and accelerator geometries as input by the user. Thus, the GUI is the preferred method for generating input files. For more information on the BEAMnrc gui, see the BEAMnrc, DOSXYZnrc and BEAMDP GUI User's Manual[20].

2.7   Running an Accelerator Simulation

Once the accelerator model is compiled, the user may run a simulation. To run a simulation from the command line, type:

`BEAM_myaccel -i inputfile -p pegsdata`

where `inputfile` is the name of the BEAM inputfile (with no `.egsinp` extension) and `pegsdata` is the name of the PEGS4 material data file (with no `.pegs4dat` extension). These two files are discussed in more detail below.

To preserve compatibility with old usage, accelerators can also be run with the "`ex`" command:

`ex BEAM_myaccel inputfile pegsdata`

"`ex`" is aliased to the Unix script `$HEN_HOUSE/scripts/run_user_code`. This script uses the non-script command line shown above to actually run the accelerator, but first it checks for the existence of the relevant directories (`$EGS_HOME` and the `BEAM_myaccel` subdirectory). It also checks for the existence of the `BEAM_myaccel*` executable and compatibility between the `config.conf` file you are using and the actual configuration that you are running on.

The users input file, *i.e.* `inputfile.egsinp`, defining the geometry of the accelerator and the BEAM simulation parameters, must exist in `$EGS_HOME/BEAM_myaccel`. The best way for a new user to begin creating a new input file is to use the BEAM GUI[20]. From the GUI, you can either begin a new input file from scratch or read in an existing input file (for the same accelerator) and modify the parameters accordingly. More experienced users may find it faster to use an editor to modify an existing input file.

Note that the input file extension is `.egsinp`. Theoretically, BEAMnrc can run a BEAM00 `.egs4inp` input file, however, depending on the parameters, there may be new inputs that will cause an error on reading the older file. It is safer to read the old `.egs4inp` file into the GUI and resave it. The GUI will then write to the file all the necessary inputs for the current version of BEAM. Note that `.egs4inp` files do not contain any EGSnrc transport parameter inputs, so standard BEAMnrc defaults for these parameters will be used when older files are read into the GUI and/or used in a BEAM simulation.

Much of the rest of this document is devoted to discussing the meaning of all the options available through the input file.

In order to run an accelerator, you also have to enter the name of the `.peg4dat` file containing the cross section data for the materials in the model. Note that all of the material names used in the input file must exist in the `.pegs4dat` file used. There is more information in section 16 but basically the files `700icru.pegs4dat` and `521icru.pegs4dat` contain data for a large number of commonly used materials. The numbers in the names correspond to AE=521 and AE=700, *i.e.* thresholds for secondary electron production of 10 and 189 keV kinetic energy respectively. The data sets go up to 55 MeV in both cases. These data sets are on the distribution on area `$HEN_HOUSE/pegs4/data`. If the user wishes to make their own data files using the program PEGS4, these files can be put either on that area, or on the user's pegs4 area, *viz.* `$EGS_HOME/pegs4/data`.

2.7.1   Batch Runs

Thus far, we have described interactive BEAM runs, in which the BEAM input and job execution information is echoed to the screen and where the window in which the job is running cannot be used for anything else while the job is executing. However, if you are running on a Unix/Linux system, then it is often desirable to run simulations in batch mode. For example, batch job submission can make use of a network queuing system (eg PBS or NQS) and is required for parallel jobs (see section 13 in this manual for more information on parallel jobs).

Submitting a batch job uses the `exb` command, which is aliased to the Unix script `$HEN_HOUSE/scripts/run_user_code_batch`. The syntax of the `exb` command is:

`exb BEAM_myaccel inputfile pegsdata [short|medium|long] [batch=batch_system] [p=N]`

The input `[short|medium|long]` determines the name of the queue to be used (the default is `long`). With our naming system at the NRC, the `short` queue has the lowest maximum CPU time but the highest priority, while `long` has an unlimited CPU time with lowest priority.

The `batch=batch_system` input determines the name of the queuing system to use. The `run_user_code_batch` script sources the file `$HEN_HOUSE/scripts/batch_options.batch_system`, which defines the batch submission commands for the particular queuing system chosen and may redefine the names of the queues available on that system (this means that the queue names `short|medium|long` are not necessarily general). Currently, `batch_system` can be set to `at` (the standard Unix batch command), `pbs` (for the PBS queuing system) or `nqs` (for the NQS queuing system). This means that the files `batch_options.at`, `batch_options.pbs` and `batch_options.nqs` are included with the EGSnrcMP distribution. However, the batch submission commands in these files are for our system at the NRC and you may have to make some changes for your system. The default value for `batch_system` is `at`, unless you have the environment variable `$EGS_BATCH_SYSTEM` set to something else.

Finally, the `p=N` input is used if you want to split the simulation into `N` parallel jobs. For more information on parallel runs, see section 13, page 123.

In addition to the files which are output from an interactive accelerator simulation, a batch run will also output a `inputfile.egslog` and file and a `inputfile.eo` file. The `.egslog` file contains all the output that would be echoed to the screen during an interactive run. **IT IS IMPORTANT TO LOOK AT THE `.egslog` FILE TO SEE IF ANY INPUT OR RUN TIME ERRORS OCCURRED**. The `.eo` file contains messages from the queuing system. If, for some reason, a job did not get submitted properly, then this file may contain clues to the problem. More information on BEAM output files is given in section 2.10, page 25 below.

For more information on submitting batch jobs, see the EGSnrcMP manual [2].

2.8   Running BEAMnrc using the GUI

It is possible to use the BEAM GUI to submit interactive, and if your system is set to handle batch jobs, batch and parallel runs. To do this, select "Run" from the "Execute" menu. This opens a running window which will give you the option to switch from an interactive to a batch (and parallel) run, and will allow you to select the queue you wish to run on. If you have not set the environment variable `$EGS_BATCH_SYSTEM`, then the GUI automatically submits to a PBS network queuing system (ie sets `batch_system=pbs`). When you press the "Execute" button, the GUI then assembles and executes the appropriate command line as described in the sections above. Output which would normally appear on the screen during an interactive run is now displayed in the GUI running window.

2.9   A Note on Temporary Working Directories

Before a BEAM run (batch or interactive), a temporary working directory,
`$EGS_HOME/BEAM_myaccel/egsrun_pid_inputfile_hostname` is created, where `pid` is the process ID number, `inputfile` is the name of the input file (no `.egsinp` extension), and `hostname` is the name of the computer the job is running on. All output files from a run (more about output files in the next section) are written to this directory with the exception of phase space (`.egsphsp#`) files, which are written directly to the `$EGS_HOME/BEAM_myaccel` directory. Once the run is finished, all output files are moved from
`$EGS_HOME/BEAM_myaccel/egsrun_pid_inputfile_hostname` into `$EGS_HOME/BEAM_myaccel` directory and the temporary working directory is removed. The reason phase space files are written directly to the accelerator directory is that they often end up being large, which makes moving them from one directory to another time consuming. If, for some reason, the simulation terminates prematurely, the temporary working directory, containing its output files, will be left behind.

2.10   BEAMnrc Output Files

Each of the BEAM output files that will be found in your `$EGS_HOME/BEAM_myaccel` directory at the end of a run is described briefly below.

.egslst This may be considered the "main output file" because it contains all of the dose and fluence results of the simulation. The file is broken up into different sections:

    Error and Warning Messages Most error and warning messages that appear on screen (or in the log file) during input are repeated at the top of the listing file.

    Summary of the Main BEAMnrc Inputs This section summaries those inputs that are not CM-specific, such as the number of histories to run, photon forcing inputs, *etc.*. The section also includes some characteristics of a phase space source (if used), such as the total number of particles in the source, the maximum kinetic energy of the source, *etc.*. If range rejection was chosen, this section also contains a table of maximum electron range versus electron energy for the various materials to be used in the simulation.

Material Data Shows relevant data (density, `AE`, `AP`, *etc.*) for all materials used in the simulation as read from the `.pegs4dat` file.

Source Parameters This is a summary of parameters such as the type of source, the Z location of the face on which the source is incident, *etc.* For a phase space source, this section repeats the source characteristics found in the first section of the file.

Region and Range Rejection Summary This is a very useful summary of all of the regions in the simulation. Regions are identified by both their absolute and local numbers. The absolute region number identifies the region within the entire simulation geometry, the local region number identifies the region within its own component module. The region summary indicates what component module a region is in, the dose zone number and bit number the user has assigned to a region (called the "bit region") and the material that the user has specified the region to be made of. If range rejection is turned on, this summary also contains the value of `ECUTRR` (the minimum energy an electron must possess in a region in order for it to reach the bottom of the simulation geometry) the residual range and the value of `ESAVE` (maximum electron energy at which range rejection is considered) for each region.

Component Module Summary This section of the listing files summaries the geometry and region information for each component module in the order that the component modules appear in the simulation geometry. This section is useful for making sure that the geometry, as interpreted by BEAMnrc, is the same as the geometry the user intended to input. This section gives specific dimensions of a component module and the specific location of each region within a component module. For each region, `ECUT`, `PCUT`, `ECUTRR`, `ESAVE`, the material, the dose zone, and the bit number are output. There is enough information in the component module summary to reconstruct the input file.

Execution Information and Warning Messages This section contains a post-run summary of some relevant run-specific information, such as the elapsed and CPU time required for the run, the total number of charged particle steps, the fraction of steps for which multiple scattering was switched off, *etc.* When a phase space source is used, this section contains a summary of the source characteristics (number of electrons, number of photons, maximum energy, *etc.*) determined from those particles actually used in the simulation. Some warning messages, such as those printed when all particles in a phase space source have been used and the source must be restarted with the first particle, are printed in this section.

Number, Fluence, Average Energy and Average Angle Results This section summaries the results for all scoring planes in the geometry. BEAMnrc outputs a summary of each scoring plane, including the plane's Z-position, the number of particles that crossed it, and the radii (half-widths) of the scoring zones, followed by the actual fluence results for the plane. The fluence is taken as the weighted sum of $1/\cos\theta$ where θ is the angle of the particle with respect to the z-axis. The number averaged energy and number averaged angle are also output. Scoring zones are numbered in order of increasing radius (half-width). Note that the fluence results may be output for one more scoring zone than the number of

scoring zones input by the user. This "extra" scoring zone represents the area of the scoring plane not covered by the scoring zones. Fluence results for particles crossing the scoring plane only once and those for particles crossing the plane two or more times ("multiple passers") are output separately. Note that all relevant results (number, fluence, dose, energy deposited) are normalized per initial source particle in the entire accelerator model, *i.e.* per initial particle which is not from a phase space file. This is done by using the variable `NINCPHSP` discussed in section 7.1. one advantage of this method is that the same results are obtained automatically, whether the calculation is split up into several components or not.

Dose Results The first table shows the total dose and total energy deposited in each of the dose scoring zones set up in the accelerator. Note that each dose scoring zone may include several geometric regions. The second table (if contaminant dose calculations are asked for) shows the dose in each region due to the contaminant particles which are defined as the charge state selected by the user as the particles cross a certain planar boundary. This was designed for calculation of dose in a phantom but is applied in general. Finally, the doses in each region with bit filters applied are given. Note that these dose components are not exclusive; that is, dose components from particles that have been in a specified scraper include contributions from particles that have been in other scrapers before and/or after passing through/interacting in the specified scraper.

.egslog The log file (created only for a batch run) contains all of the dialogue that would be appear on the screen during an interactive run. Thus, input parameters are echoed and any input errors will appear here. Below the input is a summary of the materials in the simulation (and whether or not Rayleigh scattering data is available for each material), a summary of `PRESTA` parameters, followed by the run-time execution messages (number of histories completed, CPU time for a batch, warning and error messages, *etc.*). After completion of a batch job, looking in the log file is often the easiest way to make sure that all histories were completed and that no run-time errors, such as endless loops, or negative `USTEP` errors, occurred. When using a phase space file for the source, the log file contains a warning line every time the particles in the phase space source are all used and the source file is reused from the beginning.

.egsdat This file contains all of the information required to restart a run and also the information required to obtain dose and fluence when parallel jobs are recombined. Data stored in the `.egsdat` file include:

1. $\sum_{i=1}^{nhist}(\text{energy deposited})_i$, $\sum_{i=1}^{nhist}(\text{energy deposited})_i^2$, $\sum_{i=1}^{nhist}(\text{fluence})_i$, and $\sum_{i=1}^{nhist}(\text{fluence})_i^2$ in each dose and fluence scoring zone, where `nhist` is the number of primary (non-phase space) histories completed.

2. elapsed CPU time

3. no. of histories completed, no. of primary histories completed

4. state of the random number generator at the end of the last batch

When a run is restarted, BEAMnrc reads the energy deposited and fluence data (item 1 above) for the previous run from the `.egsdat` file and then adds it to the current results

before calculating doses, fluence values and uncertainties [17]. When recombining parallel jobs, BEAMnrc reads the energy deposited and fluence data for each parallel job from its `.egsdat` file, sums the data across all jobs and (without running any further simulations) calculates the doses, fluence values and uncertainties for the complete simulation. See section 10.3 about the input variable IRESTART and Section 13 about parallel runs.

.egsrns This file contains the complete state of the random number generator at the start the current batch (`ISTORE=0`) or current history (`ISTORE=1`) in a simulation. This is only used for debugging problems which occur. These data are used to restart the run with this RNG when `ISTORE` is set to -1.

.egsplot This file contains dose *vs* depth for all dose components when a CHAMBER CM is used as a depth dose phantom. The format of the file is suitable for plotting with `xmgr` or `xmgrace`. Note that the output of the file only makes sense for doses scored in a CHAMBER depth dose phantom.

.egsphsp1(2 or 3) These files contain phase space information (see section 7) for all particles crossing the scoring planes. If `IO_OPT` is set to 1 (see section 10.4), no phase space is scored, and these files are empty.

.egsgeom If a graphical output is requested (`IWATCH=4`), this file specifies the accelerator geometry for display by `EGS_Windows` [16].

.egsgph If a graphical output is requested, this file contains the track histories for display by `EGS_Windows`. Note that this file becomes large for even a few 10's of histories.

.errors Lists any errors in the EGSnrc transport parameter inputs. The contents of this file are generated by the code `$HEN_HOUSE/src/get_inputs.mortran`, which takes care of reading the EGSnrc transport parameter inputs in BEAM (and other user codes).

.eo Contains output from the network queuing system (batch job submissions only).

Since the `.egsphsp` files are often very large, and may exceed the available disk space, BEAMnrc provides options for changing the output directory for `.egsphsp` files from the default output directory, `$EGS_HOME/BEAM_myaccel` (see Section 7.2 for more information). Changing the output directory for the other output files, however, is more complicated, and involves going into `$HEN_HOUSE/src/egs_utilities.mortran` and changing the directory path specified in subroutine `egs_open_file`.

2.11 Changing the defaults

At the top of the `.egslog` file, there is a list of 10 user selectable internal parameters which the user may wish to vary. These are all found in `$OMEGA_HOME/beamnrc/beamnrc_user_macros.mortran` along with 10 or so other parameters you may change.

```
The following internal parameters are set:
Max number of CMs: 20                     Max number of media   12
Max number of regions: 250                Max stack: 10000
Max bremsstrahlung split: 2000            Max number dose zones:   50
Max number of scoring planes:  3          Max number of scoring zones:    5
Max number dose components: 12            Minimum air gap:        0.0100 cm
   All of above can be adjusted in beamnrc_user_macros.mortran
```

Some of these parameters are obviously correlated, so *eg.* the `Max stack` should have a value at least 4 times greater than `Max bremsstrahlung split`. Nonetheless, the user should feel free to vary these parameters if they need to. The default settings should be OK using directional bremsstrahlung splitting.

Once any of the parameters in `beamnrc_user_macros.mortran` have been changed, the accelerator must be recompiled to make the changes effective.

On a stand alone system you may be able to change `$OMEGA_HOME/beamnrc/beamnrc_user_macros.mortran` but this change will affect all accelerator models recompiled after that. If you wish to make the change for just one accelerator, then copy `$OMEGA_HOME/beamnrc/beamnrc_user_macros.mortran` to the directory of the local accelerator, make the changes in that copy and then edit the `sources.make` file and delete `$(BEAM_HOME)` before `beamnrc_user_macros.mortran`. When the accelerator is recompiled, it will pick up the new version of `beamnrc_user_macros.mortran`.

2.12   Some Details

This section can be skipped without problems for new users.

2.12.1   The `BEAM_myaccel.io` File

This file (which is a renamed copy of `$OMEGA_HOME/beamnrc/default_beam.io`) is put into your `BEAM_myaccel` directory by `beam_build` when you build your accelerator. This file can be used to associate file names with Fortran unit numbers. In the current version of BEAMnrc, all output files (eg `.egsphsp#`, `.egsdat`, `.egsplot`, etc) are opened explicitly from within the code and so the default `BEAM_myaccel.io` is empty (except for comments at the top). However, the file is useful if you wish to customize BEAMnrc to write your own output quantities. For example, if you wish to output `QUANTITY` to the file `inputfile.myoutput`, the easiest way is to go into `beamnrc.mortran` and add statements, `WRITE(UNIT,FORMAT)QUANTITY;` (where `UNIT` is an arbitrary Fortran unit number), where required and then add the line:

`UNIT   .myoutput`

to the `BEAM_myaccel.io` file (Note that the only the file extension needs to be specified).

Files specified in `BEAM_myaccel.io` are opened at the beginning of the simulation and are not closed until the simulation has completed. Moreover, the files specified here are ALWAYS opened (ie there are no conditions on whether the file is created or not), so if no

quantity is written an empty file will be created. For more information about `.io` files, see
the EGSnrcMP Manual [2].

2.12.2   What's in `BEAM_myaccel_macros.mortran` and `BEAM_myaccel_cm.mortran`

The file `BEAM_myaccel_macros.mortran`, created by `beam_build` when you build an acceler-
ator, comprises the following code (in this order):

- `beamnrc_cm_macros.hdr`. Copied from `$OMEGA_HOME/beamnrc`, this is a short header
  file with nothing but comments.

- `$CM_LIST` macro. This macro has the form:

  ```
  REPLACE{$CM_LIST} WITH {CMLIST(
        CMID1,
        CMID2,
        CMID3,
        CMID4
    )}
  ```

  where `CMID1` is the identifier for the first CM, `CMID2` is the identifier of the second
  CM, etc., as specified in the `myaccel.module` file. This is ultimately expanded during
  MORTRAN compilation of the accelerator to generate a list of CM identifiers (ie
  `CMLIST(1)='CMID1'`, etc) which is used in many places in the BEAM code.

- `$CM_TYPE` macro, which has the form:

  ```
  REPLACE{$CM_TYPE} WITH {CMTYPE(
        CM1,
        CM2,
        CM3,
        CM4
    )}
  ```

  where `CM1` is the name of the first CM (eg SLABS), `CM2` is the name of the second CM,
  etc, as specified in `myaccel.module`. During MORTRAN compilation, this macro gets
  expanded to a list of CM names (ie `CMTYPE(1)='CM1'`, etc), which is used in many
  places in the BEAM code.

- `CM_macros.mortran` for each CM in the accelerator model (in the order in which they
  appear in the `myaccel.module` file) with `CM` replaced by the CM identifier as speci-
  fied in `myaccel.module`. The `CM_macros.mortran` files are copied from the directory
  `$OMEGA_HOME/beamnrc/CMs`. Note that even if a given CM is used more than once in
  an accelerator, a `CM_macros.mortran` file for each occurence of the CM will be copied
  into `BEAM_myaccel_macros.mortran`, but, of course, the identifier used to replace `CM`
  will be different in each case.

The `BEAM_myaccel_cm.mortran` file, also created by `beam_build` when you build an accelerator, consists of the `CM_cm.mortran` files for every CM in the accelerator (in the order specified in `myaccel.module`) with CM replaced by the CM identifier specified in `myaccel.module`. Similar to `BEAM_myaccel_macros.mortran`, even if a CM occurs more than once in the accelerator, `CM_cm.mortran` is copied into `BEAM_myaccel_cm.mortran` for every occurence of the CM, but the identifier in each case will be different. The `CM_cm.mortran` files are copied from `$OMEGA_HOME/beamnrc/CMs`.

2.12.3   Files used during Compilation with `make`

The `make` command uses several different files to direct the concatenation and compilation of the final MORTRAN code. This section those files, along with their functions, which are likely to be relevant to BEAM users. For a more complete description of the files involved in the `make` command, see the EGSnrcMP manual [2].

`Makefile` Located in the `BEAM_myaccel` subdirectory and created by `beam_build` when the accelerator is built. This provides the "overall" instructions for compilation. It tells the compilation to include the `$HEN_HOUSE/specs/config.conf` for your particular "config" (ie gcc, pgf77, Windows, etc). It also instructs the compilation to include the files `$HEN_HOUSE/specs/beamnrc.spec` and `$HEN_HOUSE/makefiles/beam_makefile`. Finally it defines the name of the accelerator (ie `myaccel`) for use in other files.

`config.conf` Located in the `$HEN_HOUSE/specs` subdirectory, where `config` is the name of the configuration you are running (eg gcc, pgf77, win2k, etc). This file contains many definitions essential for compiling all user codes. Definitions include:

- `DSEP` The directory separator for your machine ("/" for Unix/Linux, "\" for Windows).
- `my_machine` The name of your machine.
- `make_prog` The `make` command used on installation.
- Definitions of `$HEN_HOUSE`, `$EGS_HOME` and `$SPEC_DIR` (directory where the `config.conf` file is found)
- Definitions of the f77 and C compilers to be used, along with the default compiler options (eg optimization levels).

Other variables defined in `config.conf` include:

- `CUTIL_OBJECTS` Object files compiled from the C utility codes (such as file locking functions, etc) linked with BEAM at compile time. On Unix/Linux, machines this will point to `egs_c_utils.o` if you have a working C compiler and is empty otherwise. On Windows machines, `CUTIL_OBJECTS` will point to `egs_c_utils.obj`, a precompiled object file that eliminates the need to have a working C compiler when using these functions on a Windows machine.
- `BEAMLIB_OBJECTS` The object file required at compilation if a user code (such as DOSXYZnrc) is to use a BEAM shared library as a source. On Unix/Linux

systems, BEAMLIB_OBJECTS will point to load_beamlib.o if you have a working C compiler, and will be empty otherwise. On Windows machines BEAMLIB_OBJECTS will point to load_beamlib.obj, which is a precompiled object file and elminates the need to have a C compiler when using a BEAM library source on a Windows machine.

- BEAMLIB_EXTRA_LIBS The library required at compilation if a user code is to be able to use a BEAM shared library source (-ldl on Unix/Linux machines with a working C compiler, nothing otherwise), and also $(IAEA_LIB), which, if set, points to the library required for a user code to use IAEA-format phase space files as sources (see next item).

- IAEA_LIB and IAEA_PHSP_MACROS Variables defining the library of IAEA phase space handling routines (IAEA_LIB) and the Mortran macros which use these routines (IAEA_PHSP_MACROS). If your system does not have a working C++ compiler, then the IAEA phase space handling routines cannot be compiled and these two variables will be left blank. Variables are used during the compilation of BEAM-nrc and other EGSnrc user codes and must be defined in order to be able to read/write phase space data in IAEA format.

The config.conf file also instructs compilation to include the files $HEN_HOUSE/specs/[unix.spec or windows.spec] and $HEN_HOUSE/specs/all_common.spec described below.

unix.spec **or** windows.spec Contains definitions (such as the extension to add to an executable file) unique to Unix or Windows.

all_common.spec Contains definitions common to all systems. This including the mortran sources and macros required to compile a generic EGSnrcMP user code. This file also defines the variable RANDOM, for the random number generator, but the definition of RANDOM in beamnrc.spec (see below) overrides this one.

beamnrc.spec Located in the $HEN_HOUSE/specs subdirectory, this file contains definitions required for compiling codes in the BEAM system. Among these are the definitions of directories, $OMEGA_HOME, $BEAM_HOME, $DOSXYZ_HOME and $CM_HOME. beamnrc.spec also defines the extension, FEXT to add to the Fortran code output by the MORTRAN compiler. Currently, FEXT = F, which instructs the compiler to use the C-preprocessor before compiling the code. This is necessary for the new implementation of parallel processing in BEAM. If, for some reason, you do not have a C compiler on your system, then the installation will set FEXT = f. In addition, the variable SOURCES in beamnrc.spec defines all the MORTRAN macros and codes to concatenate and the order in which to concatenate them to create the final mortjob.mortran file that is then Fortran compiled (see section 2.12.4 below for more details). beamnrc.spec also defines the variable LIB_SOURCES, which lists the macros and codes (in order) that are concatenated to create the mortjob.mortran when BEAMnrc is compiled as a shared library for use as a source in DOSXYZnrc and other user codes (see section 2.3.1 for more information). In practice, the definitions of SOURCES and LIB_SOURCES are copied into a file called sources.make in the accelerator directory when the accelerator is built, and these latter definitions supersede those in beamnrc.spec (see below for

more about `sources.make`). Finally, `beamnrc.spec` defines the variable `RANDOM`, which determines the random number generator to be used in the simulation. Two random number generators, `RANLUX` and `RANMAR` are included in the EGSnrcMP system. Currently, `RANMAR` is the default random number generator. See section 10.8 for more information about the random number generators and how to switch from the default RANMAR to RANLUX.

`beam_makefile` Located in `$HEN_HOUSE/makefiles`, this defines the final compilation of the MORTRAN and Fortran codes. All of the `make` options outlined above are taken care of here. In addition, this file sources the files `$EGS_HOME/BEAM_myaccel/modules.make` and `$EGS_HOME/BEAM_myaccel/sources.make` created when the accelerator was built (more about these files below). `beam_makefile` ensures that the accelerator will be recompiled if there is a change in any of the MORTRAN code or macro files used by the accelerator (these are compiler "dependencies"). `beam_makefile` also automatically rebuilds (by calling `beam_build.exe`) the accelerator if there is a change in any of the individual CM macros or MORTRAN codes or if `myaccel.module` has been changed to use different CMs and/or CM identifiers (these are build "dependencies").

`modules.make` Located in `$EGS_HOME/BEAM_myaccel` and created by `beam_build` when the accelerator is built. This file is a list of the MORTRAN codes (`$HEN_HOUSE/omega/beamnrc/CMs/CM_cm.mortran`) and macros `$HEN_HOUSE/omega/beamnrc/CMs/CM_macros.mortran`) for all CMs used in the accelerator. It is through the `modules.make` file that `beam_makefile` has access to the individual CM macros and MORTRAN codes and can, thus, rebuild the accelerator if any of the CM coding changes. The user should not modify this file since it

`sources.make` Located in `$EGS_HOME/BEAM_myaccel` and created by `beam_build` when the accelerator is built. This file contains definitions of the variables `SOURCES` and `LIB_SOURCES` copied from the `beamnrc.spec` file (see above). These definitions override those in `beamnrc.spec` and allow the user to customize their accelerator, adding their own source code and/or changing the directory from which source codes are picked up. Note when adding your own coding that macros must appear above the MORTRAN coding they are used in and, in the case where there are several occurrences/versions of the same macro, the last occurrence is the one that is used. Note that changes made to this file will be lost if you execute `beam_build` or build the accelerator using the GUI. This file must be adjusted to include source 31 for beam characterization (see section 4.16.

2.12.4 Files Concatenated to Create `mortjob.mortran`

`mortjob.mortran`, the file which is actually MORTRAN compiled on the way to creating an executable for your accelerator, consists of many concatenated macros and MORTRAN files. The files which are concatenated and the order in which they are concatenated are determined by the `SOURCES` (for stand-alone accelerators) or `LIB_SOURCES` (for shared libraries) definition in the file `$EGS_HOME/BEAM_myaccel/sources.make` discussed in section 2.12.3 above.

Figure 5 below shows all the files that are concatenated to create a `mortjob.mortran` for a stand-alone accelerator simulation.

Note that macros must appear before the MORTRAN code in which they are used. Hence, the first portion of `mortjob.mortran` consists entirely of macro files.

mortjob.mortran

```
HEN_HOUSE/src/egsnrc.macros                                    macros
HEN_HOUSE/utils/timing.macros
HEN_HOUSE/lib/config/machine.macros
HEN_HOUSE/src/ranlux.macros
  (or HEN_HOUSE/src/ranmar.macros)
HEN_HOUSE/src/transportp.macros

OMEGA_HOME/beamnrc_user_macros.mortran
OMEGA_HOME/phsp_macros.mortran
OMEGA_HOME/sbsnrc_macros.mortran

EGS_HOME/BEAM_myaccel/BEAM_myaccel_macros.mortran
--------------------------------------------------------------------
HEN_HOUSE/src/egs_utilities.mortran

OMEGA_HOME/beamnrc.mortran

HEN_HOUSE/utils/xvgrplot.mortran

EGS_HOME/BEAM_myaccel/BEAM_myaccel_cm.mortran

HEN_HOUSE/src/get_inputs.mortran
HEN_HOUSE/src/ranlux.mortran
  (or HEN_HOUSE/src/ranmar.mortran)
HEN_HOUSE/utils/nrcaux.macros
HEN_HOUSE/lib/config/machine.mortran
HEN_HOUSE/src/egs_parallel.mortran
HEN_HOUSE/src/egsnrc.mortran                             MORTRAN code
```

Figure 5: Files concatenated to form the complete BEAM source code, `mortjob.mortran`, where `BEAM_myaccel_cm.mortran` and `BEAM_myaccel_macros.mortran` contain the source code related to all the selected CMs. The files and the order of concatenation are defined by the `SOURCES` variable in `$EGS_HOME/BEAM_myaccel/sources.make`.

A brief description of each of these files follows:

egsnrc.macros Macros for the EGSnrc system. This includes common blocks, such as STACK, that are used by all user codes. See the EGSnrc Manual[1] for more details.

timing.macros Definitions of timing macros used in BEAMnrc (and other user codes). Replaces the macros with calls to EGS subroutines used to determine elapsed CPU time, elapsed total time, etc.

phsp_macros.mortran Macros used by BEAMnrc to read/write data from/to phase space files. See section 7 for more details.

iaea_phsp_macros.mortran Macros used by BEAMnrc to read/write phase space data in IAEA format. This file is only present if there is a working C++ compiler (required to

compile the library of IAEA phase space handling routines). Otherwise, these macros are defined as blank (";") in `phsp_macros.mortran` and IAEA-format phase space data cannot be handled.

`machine.macros` Machine/compiler-dependent macros. Defines `$LONG_INT` as `integer*8` if it is available on this machine, the record length factor (`$RECL-FACTOR`) for a 4-byte record in a phase space file, etc. See the EGSnrcMP Manual[2] for more details.

`ranmar.macros` **(or `ranlux.macros`)** Macros used by BEAMnrc (and other user codes) to obtain random numbers, store random number states to a file, retrieve random number states from a file and initialize the random number generator.

`transportp.macros` Macros used by `get_inputs` to define text patterns searched for in the EGSnrc input section of a BEAMnrc input file (see section 11).

`beamnrc_user_macros.mortran` Macros defining many default BEAMnrc parameters such as the maximum number of dose/fluence scoring zones, the maximum stack depth, etc. These parameters often need to be changed by the user. See section 2.11 for more details.

`sbsnrc_macros.mortran` Macros used by BEAMnrc when performing selective bremsstrahlung splitting (SBS). See section 6.3.2 for more details.

`BEAM_myaccel_macros.mortran` Macros for all CMs that comprise `myaccel` concatenated together by `beam_build` with the names of the CMs replaced by their identifiers as specified in the `myaccel.module` file. See section 2.2 for more details.

`egs_utilities.mortran` Various EGSnrc utility codes. Includes the `egs_init` functioning for initializing variable arrays and opening any Fortran output units with the names given in `BEAM_myaccel.io` (see section 2.12.1). Also includes `egs_combine_runs` routine for recombining results after parallel runs (see section 13.3).

`beam_main.mortran` BEAMnrc main code. Consists of calls to the main BEAMnrc subroutines, `beam_init`, `beam_shower_loop` and `beam_finish` (in that order). These subroutines are in `beamnrc.mortran` (see below).

`beam_lib.mortran` Used instead of `beam_main.mortran` when compiling BEAMnrc as a shared library for use as a source (in DOSXYZnrc or other EGSnrc user code). Contains macro (re)definitions essential for use as a source along with subroutines for initializing and sampling the BEAMnrc source. Among these is a redefinition of the main phase space writing macro which causes phase space data that would normally be written to a phase space file to be stored in an array instead. Incident particles for the simulation using the BEAMnrc simulation as a source are then sampled from this array.

`beamnrc.mortran` Contains definitions of macros used by BEAMnrc, common blocks used by BEAMnrc and all BEAMnrc subroutines.

`xvgrplot.mortran` Subroutine for creating a data file suitable for plotting with xmgr/xmgrace. BEAMnrc uses this subroutine to output an `.egsplot` file (see section 2.10).

BEAM_myaccel_cm.mortran Subroutines for all CMs comprising `myaccel` concatenated together by `beam_build` with CM names replaced by their identifiers as specified in `myaccel.module`. See section 2.2 for more details.

get_inputs.mortran Coding for reading EGSnrc inputs from the BEAMnrc input file. See section 11 for more about these inputs.

ranmar.mortran **(or `ranlux.mortran`)** Subroutines used by the RANMAR (or RANLUX) random number generator. Many of the macros defined in `ranmar.macros` (or `ranlux.macros`) include calls to these subroutines.

nrcaux.mortran Auxiliary MORTRAN routines used by BEAMnrc (and other EGSnrc user codes). Includes `SUBROUTINE WATCH`, which outputs details of each particle step/interaction or particle tracks to the `.egsgeom` file for display using EGS_Windows, depending on the value of the input variable `IWATCH` (see section 10.1 for more on `IWATCH`).

machine.mortran Machine/configuration-dependent routines such as date/time routines and system calls. This file is created during EGSnrcMP installation. See the EGSnrc Users Manual[1] and EGSnrcMP Users Manual[2] for more info.

egs_parallel.mortran Subroutines for creating, opening, reading from and writing to the job control (`.lock`) file during a parallel run (see section 13).

egsnrc.mortran EGSnrc subroutines. See the EGSnrc Manual[1] for more details.

Note that in previous versions of BEAMnrc, all of the BEAMnrc-specific coding was contained in the file `$OMEGA_HOME/beamnrc/beamnrc.mortran`. Now, however, this coding has been broken up into two files: the BEAMnrc main code is in `$OMEGA_HOME/beamnrc/beam_main.mortran`, and the rest of the macro/subroutine definitions in `$OMEGA_HOME/beamnrc/beamnrc.mortran`. Separating out the BEAMnrc main code was necessary to allow us to compile BEAMnrc as a shared library for use as a source in DOSXYZnrc and other EGSnrc user codes, since shared libraries do not use the BEAMnrc main code. See section 2.3.1 (page 21) for more information about BEAMnrc as a shared library.

3   Description of main BEAMnrc input file

The first section of an input file (`.egsinp`) for BEAMnrc concerns all issues not related to individual `CMs`. The following is taken from the source code. More detailed discussion of most of these options is given in the following sections (use the Index to find where in most cases), but this is included here as a short summary/reference. Note that the graphical user interface (`BEAMnrc_GUI`) greatly simplifies the task of creating input files[20] and the help files in the GUI mostly duplicate what is below.

```
              FORMAT OF PARAMETER-DEFINITION(input) FILE
              *******************************************
```

```
                      GENERAL INPUT/OUTPUT PARAMETERS
                      *******************************


   First record    TITLE   80A1
   ************


   Next Record   MEDIUM for nominal air (as in pegs4dat file)
   ***********
              In many CMs, the region about the central-axis or at the
              front or back of the CM, is assumed to be this medium.
              It is thought of and refered to as air, but can be anything.
              Default is VACUUM. MEDIUM must exactly match name in pegs4dat
  ---------------------------------------------------------------------------


   Next Record
   ***********
      IWATCH,ISTORE,IRESTART,IO_OPT,IDAT,LATCH_OPTION,IZLAST


             IWATCH = 0 for normal output (the default)
                    = 1 output for every discrete interaction
                    = 2 output for every electron/photon step as well
                    = 4 outputs file for graphics
                    = -N set to 2 on history N, set to 0 on all other
                         histories (for debugging purposes)
             ISTORE = 0 store rn seeds for the 1st history of a batch
                    = 1 store initial rn status (unit 2) for each history
                         being simulated
                    =-1 start first history with rn status from file (unit 2)
                         This is a debugging tool. If run quits, rerun with
                         ISTORE=1, then again ISTORE=-1 and IWATCH = 1/2
                         and/or the debugger on.
           IRESTART = 0 first run for this data set (the default)
                    = 1 restart of a previous run
                    = 2 just create the input file and exit
                    = 3 read in the raw data from a previous run and do
                         the statistical analysis on dose etc.
                    = 4 read in the .egsdat files from parallel jobs
                         having the same base name as the input file but
                         with the extension _w#, where #
                         can be any positive integer.  These .egsdat files
                         will be summed and then the result analyzed similar
                         to IRESTART=3.
             IO_OPT = 0 phase-space output at each scoring plane(the default)
                    = 1 no phase-space output when particles cross scoring
                         plane
                    = 2 no phase-space output but do data analysis for
                         simplified source models
                    = 3 phase-space output up to 100 k particle histories
                         then do analysis only for simplified source models
```

```
                        = 4 output phase space in IAEA format
            IDAT        = 0 store data arrays for re-use (takes time but safer)
                        = 1 don't store them
      LATCH_OPTION      = 0 defaults to 2
                        = 1 LATCH for secondaries not inherited from primaries
                            Bits 1-23 set for all regions particle is in
                        = 2 LATCH bits set for all regions particle is in
                            and inherited by secondaries
                            also record bit regions where secondaries created
                            and whether they were created by brem photons
                        = 3 = option 2 but the region numbers are recorded
                            for photons where they interact rather than where
                            they pass through
            IZLAST      = 0 do not score ZLAST etc. (the default)
                        = 1 score the z-position of the last site of interaction
                            for photons and creation of electrons by a photon.
                        = 2 score the xyz-position of the last site of
                            interaction in the file $.egs4gph to be used by
                            EGS_WINDOWS. IWATCH=4 must not be used at same time.
                            Note that for phase space inputs, ZLAST is passed
                            through, but XLAST and YLAST are not.
--------------------------------------------------------------------------------


  Next Record                MONTE CARLO CONTROL INPUT
  ***********                *************************

    NCASE,IXXIN,JXXIN,TIMMAX,IBRSPL,NBRSPL,IRRLTT,ICM_SPLIT
            NCASE       = # of histories to run for this simulation
                            (min:$NCASEMIN = 100 for IWATCH=0)
            IXXIN       = 1st random number initial seed (blank or 0 OK)
                            Note that, if using the ranlux random no. generator,
                            this input is the luxury level and should have a
                            value >=0 and <=4.  Otherwise, a default luxury
                            level of 1 will be used.
            JXXIN       = 2nd random number initial seed (blank or 0 OK)
            TIMMAX      = max cpu time allowed for this run in hours
                            (default=0.99)
            IBRSPL      = 0 no brem splitting
                        = 1 with uniform brem splitting
                        = 2 with directional bremsstrahlung splitting (DBS)
                        = 29 with selective bremsstrahlung splitting
            NBRSPL      = (if IBRSPL = 1 or 2) brem splitting number AND
                            annihilation splitting number (if IRRLTT=2)
                        = (if IBRSPL = 29) max. brem splitting number
            IRRLTT      = 0 no Russian Roulette (the default).  Also, no
                            annihilation or higher-order splitting.
                        = 1 no longer used.  This defaults to IRRLTT=2
                        = 2 perform Russian Roulette--eliminates all but one
                            of secondary charged particles created by split
```

3 DESCRIPTION OF MAIN BEAMNRC INPUT FILE

```
                            photons.
                            If the surviving particle undergoes another (higher-
                            order) bremsstrahlung event or an annihilation,
                            resulting photons are split again by NBRSPL for
                            IBRSPL=1. (ie uniform splitting) and by NMIN
                            for IBRSPL=29 (i.e. selective brem splitting)

                    Note: The input IRRLTT is automatically set to 0 if IBRSPL=2
                          This is because Directional Bremsstrahlung Splitting
                          does not use the built-in Russian Roulette of EGSnrc

              ICM_SPLIT = 0 no splitting of photons and electrons as they cross
                            a plane at the start of a user-specified CM
                        > 0 Split photons and electrons a user-specified number
                            of times as soon as they cross the arbitrary
                            splitting plane at the top of this CM #.

   Next record (if IBRSPL=2 or 29)
   ***********

      FS,SSD,(NMIN),(ICM_DBS,ZPLANE_DBS,IRAD_DBS,ZRR_DBS)  (3F12.0 or 6F12.0)
                   FS = radius of field (cm) into which bremsstrahlung photons
                        must be directed if they are to be split (IBRSPL=2)
                        or length of side of square field (cm) in which
                        selective bremsstrahlung splitting probabilities are
                        calculated for IBRSPL=29.
                  SSD = distance from bremsstrahlung target where FS
                        is defined.  FS and SSD only define an angle which is
                        used (IBRSPL=29).
                 NMIN = background bremsstrahlung splitting number (ie even
                        outside the field, bremsstrahlung events will be
                        split into NMIN photons). Also equal to the
                        higher generation brem splitting number
                        and annihilation photon splitting number if IRRLTT=2.
                        NMIN is only required for IBRSPL=29.
          ICM_DBS and   These are only required to define the splitting
                        plane if IBRSPL=2.  As soon as
           ZPLANE_DBS   a fat electron reaches ZPLANE_DBS within CM number
                        ICM_DBS, it gets split NBRSPL times.  This is designed
                        to improve electron statistics in the current
                        implementation of directional bremsstrahlung
                        splitting (DBS).  If ICM_DBS=0, then no electron
                        splitting is done (recommended if only good photon
                        statistics are required).  Note that ZPLANE_DBS is
                        the index of the plane within ICM_DBS, not the Z
                        position of the plane.  Usually, ICM_DBS will
                        be the CM number of the flattening filter in the
                        accelerator.  If this is modelled using FLATFILT
                        or CONESTAK, then ZPLANE_DBS will denote the layer
```

```
                          no. (starting from the top).  If the flattening
                          filter is modelled using CONS3R, then only two
                          planes are available: ZPLANE_DBS=1 is the plane
                          at the top of the structure and ZPLANE_DBS=2 is the
                          plane at the bottom of the structure.  Currently,
                          only FLATFILT, CONESTAK and CONS3R support these
                          inputs.  Usually ZPLANE_DBS is the plane defining the
                          bottom of the flattening filter.
              IRAD_DBS    Set to 1 if you want the NBRSPL split electrons
                          to be distributed in a radially-symmetric manner
                          about the beam axis.  Note that the beam must be
                          radially symmetric above the splitting plane for this
                          to make sense.  Set to 0 (the default) otherwise.
              ZRR_DBS     Z position of the russian roulette plane (cm).  Only
                          required if IBRSPL=2.  This defines the Z
                          position of a plane within the geometry below which
                          non-fat photons about to undergo a compton, pair
                          or photoelectric event will NOT be subject to
                          russian roulette and compton, pair or photoelectric
                          events from fat photons will be split NBRSPL times.
                          This is designed to increase the number of electrons
                          (albeit with a lower weight) below this plane and
                          is only used if electron splitting is on (ie
                          ICM_DBS above is > 0).  Note that radiative events
                          (bremsstrahlung, annihilation) of non-fat electrons
                          below this plane are not split. Usually, the Russian
                          Roulette plane is above the electron splitting plane,
                          and so it is within the flattening filter
                          but somewhere above the bottom.  Note that ZRR_DBS is
                          in cm whereas the electron splitting plane must be
                          on a horizontal boundary in a CM.

   Next record (if ICM_SPLIT>0)
   ***********

      NSPLIT_PHOT,NSPLIT_ELEC (2I3)
              NSPLIT_PHOT = The photon splitting number.
              NSPLIT_ELEC = The electron splitting number.
      This input is unrelatted to bremm splittin and is designed to improve
      efficiency in phantom depth-dose calculations.
 ------------------------------------------------------------------------


                SOURCE GEOMETRICAL CONFIGURATION INPUT
                **************************************


   Next Record  specifies charge and type of source of incident particles
                The meaning of parameters depends on source type

   ISOURC = 0   PARALLEL BEAM INCIDENT FROM THE FRONT (+VE Z-AXIS)
```

3   DESCRIPTION OF MAIN BEAMNRC INPUT FILE

```
   **********

     IQIN,ISOURC,RBEAM,UINC,VINC,WINC
             IQIN    charge of the incident beam (defaults to 0)
             ISOURC  =  0
             RBEAM   radius of parallel beam in cm (defaults to max radius
                     if RBEAM < 0 or > max radius
                     max radius =RMAX_CM(1) for circular CM boundary
                                =RMAX_CM(1)*SQRT(2) for square CM )
             UINC    incident x-axis direction cosine
             VINC    incident y-axis direction cosine
             WINC    incident z-axis direction cosine
                     Note: (UINC,VINC,WINC) get automatically normalized
                           defaults to (0.0,0.0,1.0)
 -------------------------------------------------------------------------


  ISOURC = 1   POINT SOURCE ON Z-AXIS INCIDENT FROM THE FRONT
  **********                 circular or square

     IQIN,ISOURC,DISTZ,RBEAM,GAMMA,XINL,XINU,YINL,YINU
             IQIN    charge of the incident beam (defaults to 0)
             ISOURC  =  1
             DISTZ   distance of the point source above front of first
                     CM at Z=Z_min_CM(1). Defaults to  100 cm
             RBEAM   radius of the beam on front of first CM
                     defaults to max radius of first CM if GAMMA is also 0.
                     or:
                     If negative, denotes that that field on front of
                     first CM is rectangular
             GAMMA   1/2 angle about z-axis(degrees) of source,
                     ONLY if RBEAM=0.0
             XINL,XINU,YINL,YINU  Lower and upper X boundaries and
                     Y boundaries of rectangular field on first CM in
                     cm.  ONLY if RBEAM<0.
 -------------------------------------------------------------------------


  ISOURC = 3  UNIFORM ISOTROPICALLY RADIATING SOURCE WITHIN CMs
  **********          Circular: Vertical ring centred on Z-axis
                      or     horizontal cylinder centred parallel to X-axis

     IQIN,ISOURC,RMINBM,RBEAM,ZSMIN,ZSMAX
             IQIN    charge of particles from source (defaults to 0)
             ISOURC  =  3
             RMINBM  inner radius of vertical ring (RBEAM >= 0) (cm)
                     or:
                     Z position of centre of horizontal cylinder
                     (RBEAM < 0) (cm)
             RBEAM   outer radius of vertical ring (RBEAM >= 0) (cm)
                     or:
```

```
                  -radius of horizontal cylinder (RBEAM < 0) (cm)
          ZSMIN   Z of top of vertical ring (RBEAM >= 0) (cm)
                  or:
                  min. X of horizontal cylinder (RBEAM < 0) (cm)
          ZSMAX   Z of bottom of vertical ring (RBEAM >= 0) (cm)
                  or:
                  max. X of horizontal cylinder (RBEAM < 0) (cm)

        NOTE: The sign of RBEAM determines if the source will be a
                  vertical ring a horizontal cylinder.
              The Z-span of the source must be in the range
                  Z_min_CM(1)-Z_min_CM(MAX_CMs+1).
              Currently, this source is limited to being placed within
                  CONESTAK, FLATFILT or SIDETUBE.
------------------------------------------------------------------------
  ISOURC = 3a  A cylindrical, isotropically radiating Co60 source within CMs
  **********  using directional source biasing (DSB).

    IQIN,ISOURC,RMINBM,RBEAM,ZSMIN,ZSMAX,i_dsb,DSB_DELTA
          (same is source 3)
------------------------------------------------------------------------


  ISOURC = 5   NRC SWEPT BEAM SOURCE
  **********

    IQIN,ISOURC,GAMMA,RBEAM
            IQIN    charge of the incident beam (defaults to 0)
            ISOURC  =  5
            GAMMA   1/2 angle of cone in degrees
            RBEAM   radius of beam spot at Z = 0.0 (cm)

    Note apex of cone is at x=y=0,z=Z_min_CM(1)
------------------------------------------------------------------------


  ISOURC = 6   RECTANGULAR BEAM INCIDENT FROM THE FRONT
  **********

    IQIN,ISOURC,XBEAM0,YBEAM0,XBEAM,YBEAM
            IQIN    charge of the incident beam (defaults to 0)
            ISOURC  = 6
            XBEAM0  X position of centre of beam (cm)
            YBEAM0  Y position of centre of beam (cm)
            XBEAM   half-width in X direction (cm)
            YBEAM   half-width in Y direction (cm)
------------------------------------------------------------------------


  ISOURC = 7   SCANNING BEAM SOURCE (sawtooth like Therac20)
  **********
```

3 DESCRIPTION OF MAIN BEAMNRC INPUT FILE

```
      IQIN,ISOURC,FD_AT100, IRATIO_YXF, RBEAM
              IQIN     charge of the incident beam (defaults to 0)
              ISOURC  =  7
              FD_AT100   length & width of scanning field at  SSD=100cm
              IRATIO_YXF =  the number of Y scans per X scan
                           (rounds 2*IRATIO_YXF up to nearest odd
                           number--default IRATIO_YXF = 6.5)
              RBEAM    radius of the beam at Z=0, defaults to 0.01cm
-----------------------------------------------------------------------


   ISOURC = 8   SCANNING BEAM FOR MM50 (uniform circular beam from
   *********                              a point on axis at Z=0)


     IQIN,ISOURC,DISTZ,RBEAM
              IQIN     charge of the incident beam (defaults to 0)
              ISOURC  =  8
              DISTZ    SSD (default to 100 cm)
              RBEAM    radius of scanned beam at SSD.  If set <=0 or
                       too large, then RBEAM gets reset to
                       RMAX_CM(1)*DISTZ/Z_min_CM(1) (circular CM 1) or
                       SQRT(2)*RMAX_CM(1)*DISTZ/Z_min_CM(1) (square CM 1)
                       so that particles strike the front of
                       CM 1 within a circle of radius RMAX_CM(1)
                       or SQRT(2)*RMAX_CM(1).
              RBEAM0   Radius of beam spot at Z=0cm.  Defaults to 0
                       if RBEAM0<0 and gets reset to
                       RMAX_CM(1)*DISTZ/Z_min_CM(1)-RBEAM (circular CM 1)
                       or SQRT(2)*RMAX_CM(1)*DISTZ/Z_min_CM(1)-RBEAM
                       (square CM 1) if RBEAM+RBEAM0 >
                       RMAX_CM(1)*DISTZ/Z_min_CM(1) or >
                       SQRT(2)*RMAX_CM(1)*DISTZ/Z_min_CM(1).
   For this source the particles start at Z_min_CM(1) and hence
   Z_min_CM(1) must be >= 0.0
-----------------------------------------------------------------------


   ISOURC = 9    SCANNING BEAM FOR MM50 (discrete field coverage from
   *********                              a point source at Z=0)


     IQIN,ISOURC,DISTZ,NPTS_SRC9
              IQIN     charge of the incident beam (defaults to 0)
              ISOURC  =  9
              DISTZ    SSD (default to 100 cm)
          NPTS_SRC9    the number of discrete points at the SSD
                       defaults to $MAXPTS_SRC9 if NPTS_SRC9 > $MAXPTS_SRC9
                       or 1 if NPTS_SRC9 <= 0.


   Next record (if ISOURC=9)
   ************************
```

```
     Repeat for I=1,NPTS_SRC9

     X_SRC9(I),Y_SRC9(I),PROB_SRC9(I) (3F15.0)
               X_SRC9(I)     X coordinate of point I at the SSD (cm)
               Y_SRC9(I)     Y coordinate of point I at the SSD (cm)
               PROB_SRC9(I)  probability of a particle being at point I

     Note that PROB_SRC9(I) need not be normalized; they are
     automatically normalized in the source routine.
    For this source the particles start at Z_min_CM(1) and hence
    Z_min_CM(1) must be >= 0.0
 ----------------------------------------------------------------------

   ISOURC = 10    PARALLEL CIRCULAR BEAM INCIDENT FROM THE SIDE
   ***********
                  (NOTE: beam facing X-AXIS, I.E., UINC should be < 0.0   )
                  (this source should only be used together with CM XTUBE )
                  (for simulating the target of an X-ray tube.            )
                  (XTUBE should always be the first CM in the geometry.   )

     IQIN,ISOURC,RBEAM,UINC,VINC,WINC
            IQIN     charge of the incident beam (defaults to 0)
            ISOURC  =  10
            RBEAM    radius of parallel beam in cm (defaults to max radius)
            UINC     incident X-axis direction cosine (UINC < 0.0)
            VINC     incident Y-axis direction cosine
            WINC     incident Z-axis direction cosine
                     Note: (UINC,VINC,WINC) get automatically normalized
                           defaults to (-1.0,0.0,0.0)
 ----------------------------------------------------------------------

   ISOURC = 13    PARALLEL RECTANGULAR BEAM INCIDENT FROM THE SIDE
   ***********
                  (Note beam facing X-axis, i.e., UINC should be < 0.0    )
                  (this source should only be used together with CM XTUBE )
                  (for simulating the target of an X-ray tube.            )
                  (XTUBE should always be the first CM in the geometry.   )

     IQIN,ISOURC,YBEAM,ZBEAM,UINC,VINC
            IQIN     charge of the incident beam (defaults to 0)
            ISOURC  =  13
            YBEAM    half-width of parallel beam in cm (defaults to 0.2 cm)
            ZBEAM    half-height of parallel beam in cm (defaults to 0.2 cm)
            UINC     incident X-axis direction cosine (UINC < 0.0)
            VINC     incident Y-axis direction cosine
                     (incident Z-axis direction cosine WINC default to 0.0)
                     Note: (UINC,VINC) get automatically normalized
                           defaults to (-1.0,0.0,0.0)
 ----------------------------------------------------------------------
```

3   DESCRIPTION OF MAIN BEAMNRC INPUT FILE

```
   ISOURC = 15   NRC SWEPT BEAM WITH BEAM DIVERGENCE AND RADIAL INTENSITY
   ***********                                               DISTRIBUTION

      IQIN,ISOURC,GAMMA,ZFOCUS,RTHETAIN,THETAIN
               IQIN        charge of the incident beam (defaults to 0)
               ISOURC      =  15
               GAMMA       half angle of the cone swept by the beam (degrees)
               ZFOCUS      Z position of the apex of the cone (cm)
               RETHETAIN   radius at which THETAIN, the divergence angle of the
                           beam, is specified (cm).  RTHETAIN must be > 0.
               THETAIN     divergence angle of the beam (degrees).  If GAMMA
                           is not 0, then THETAIN can be set to 0; otherwise
                           it must be > 0.

      Note that particles are always incident at Z_min_CM(1), regardless of
      the value of ZFOCUS

  Next record (If ISOURC=15)
  ***********
      SPCNAM        FILENAME (with EXT) containing description of the radial
                    intensity distribution of the incident particles
                                                  (maximum 80 characters)

               -----------------------------------------------------
               FILE FORMAT for SPCNAM:
                 NRDIST
                 (RDISTF(I),RPDF(I),I=1,NRDIST)
                       NRDIST      # radial bins
                       RDISTF(I)   upper radius of bin I (cm)
                       RPDF(I)     probability of particle being in bin I.
 ----------------------------------------------------------------------


   ISOURC = 19   PARALLEL ELLIPTICAL BEAM FROM FRONT GAUSSIAN IN X AND Y
   ***********

      IQIN,ISOURC,RBEAM,UINC,VINC,WINC,sigma_src19,RBEAMY
               IQIN    charge of the incident beam (defaults to 0)
               ISOURC  = 19
               RBEAM   sigma of the 2-D gaussian distribution (RBEAM > 0)
                       in the X-direction in cm
                         or
                       -FWHM of 2-D gaussian distribution (RBEAM < 0) in
                       the X-direction in cm
                       Note: sigma of gaussian distribution is limited to
                             <RMAX_CM(1) for circular CM 1 and
                             <SQRT(2)*RMAX_CM(1) for square CM 1
               UINC    incident x-axis direction cosine
               VINC    incident y-axis direction cosine
               WINC    incident z-axis direction cosine
                       Note: (UINC,VINC,WINC) get automatically normalized
```

```
                            defaults to (0.0,0.0,1.0)
     sigma_src19    mean angular spread of particles about Z axis
                    in degrees (none if set <=0).  Overrides incident
                    direction cosines if set > 0, so that beam is assumed
                    to be centred on Z axis.
         RBEAMY     same as RBEAM but for Y-direction.  If set to 0,
                    then RBEAMY=RBEAM for a circular beam.
------------------------------------------------------------------------


  ISOURC = 21   FULL PHASE-SPACE SOURCE
  ***********

    IQIN,ISOURC,INIT_ICM,NRCYCL,IPARALLEL,PARNUM,ISRC_DBS,RSRC_DBS,
                                           SSDSRC_DBS,ZSRC_DBS
          IQIN       dummy NOT USED. Set = 9 for this source by BEAMnrc
          ISOURC     = 21
          INIT_ICM   particles start at front surface of this CM
                     (INIT_ICM is actually read as a real)
          NRCYCL     Number of times to recycle each particle in a phase
                     space source.  Each particle in the phase space
                     file is used a total of NRCYCL+1 times before
                     going on to the next particle.
                     If NRCYCL is set <=0 then NRCYCL is automatically
                     calculated to use the entire phase space file with no
                     restarts.  The calculated NRCYCL does not take into
                     account particles that are rejected because they
                     miss the geometry.
                     If NRCYCL is set > 0, then the user-input value is
                     used.
                     If NCASE > no. of particles in the phase space file,
                     then use of NRCYCL is essential for accurate
                     statistics.  If you are unsure of how many times to
                     recycle, use the automatically-calculated value of
                     NRCYCL.  If this still results in many restarts
                     (because of multiple passers being rejected and/or
                     photons rejected because they fall outside the DBS
                     splitting radius--see below)
                     then re-run the simulation with NRCYCL set manually
                     to:
                     NCASE/(NNPHSP-NNPHSP*(NPASS_ph_sp+NFAT_ph_sp)/
                               (NTOT_ph_sp+NPASS_ph_sp+NFAT_ph_sp) ) -1
                     where NNPHSP is the no. of particles in the file,
                     NTOT_ph_sp is total no. of particles used (not
                     including recycling), NPASS_ph_sp is total no.
                     of multiple passers ignored (not including
                     recycling), and NFAT_ph_sp is the no. of photons
                     rejected (not including recycling because they fall
                     outside the DBS splitting radius at the SSD (only
                     if ISRC_DBS=1--see below)). These numbers are
```

                          available in the .egslst file. Always round your
                          calculated value of NRCYCL up.
            IPARALLEL   set >1 if you are distributing the job among
                          IPARALLEL machines. IPARALLEL is used with PARNUM
                          (see below) to partition a phase space source into
                          IPARALLEL equal parts.
            PARNUM   For each of the IPARALLEL parallel jobs, PARNUM
                          should have a different integer value in the range
                          1<=PARNUM<=IPARALLEL. The partition of the phase
                          space source that is used for a particular job is
                          then given by:
                              (PARNUM-1)*(NNPHSP/IPARALLEL)<INPHSP<=
                                                    (PARNUM)*(NNPHSP/IPARALLEL)
                          where NNPHSP is the total number of particles in the
                          phsp source and INPHSP is the particle no. chosen.
            ISRC_DBS   Set to 1 if you used directional bremsstrahlung
                          splitting (DBS) in the BEAM simulation used to
                          generate this phase space source and you wish to
                          reject photons not aimed into the splitting
                          field(which are fat). These fat photons compromise
                          statistics. Set to 0 otherwise.
            RSRC_DBS   DBS splitting radius in BEAM simulation used to
                          generate this source (cm). Only used if ISRC_DBS=1.
            SSDSRC_DBS   SSD at which RSRC_DBS was defined in the BEAM sim.
                          used to generate this source (cm). Only used if
                          ISRC_DBS=1.
            ZSRC_DBS   Z where the phase space source was collected in
                          the BEAM simulation used to generate this source(cm).
                          Only used if ISRC_DBS=1.

       Photons are projected from ZSRC_DBS to SSDSRC_DBS and if they will fall
       outside of RSRC_DBS (based on their trajectory) then they will
       be rejected. This prevents fat photons from compromising statistics.

   Next record (If ISOURC=21)
   ****
      SPCNAM   FILENAME (with EXT) contains phase space information
                      (maximum of 80 characters)

   ISOURC = 23   BEAM SIMULATION SOURCE
   ****

       IQIN,ISOURC,INIT_ICM,ISRC_DBS,ALPHA24,BETA24,DIST24
              IQIN   dummy NOT USED. Set = 9 for this source by BEAMnrc
              ISOURC   = 23
              INIT_ICM   particles start at front surface of this CM
                          (INIT_ICM is actually read as a real)
              ISRC_DBS   Set to 1 if you are using directional bremsstrahlung

                       splitting (DBS) in the BEAM simulation source
                       and you wish to reject fat photons (not aimed into
                       the splitting field). These fat photons compromise
                       statistics. Set to 0 otherwise.
          ALPHA24   Angle of rotation of source plane about X-axis
                       in degrees. Positive angle is clockwise rotation.
                       (-90 < ALPHA24 < 90)
           BETA24   Angle of rotation of source plane about Y-axis
                       in degrees. Positive angle is counter-clockwise
                       rotation. (-90 < ALPHA24 < 90)
           DIST24   Distance of point of rotation above INIT_ICM.

    Note restriction that if ALPHA24˜=0 and/or BETA24 ˜=0, then INIT_ICM
    must be > 1. This is because the rotation will result in some
    particles incident within INIT_ICM-1. Also, the following CMs
    currently do not handle the general case of a forward-directed particle
    incident from within them:
    APPLICAT,ARCCHM,CHAMBER,CIRCAPP,CONESTAK,DYNJAWS,JAWS,MESH,PYRAMIDS,
    SIDETUBE
    Thus, if ALPHA24˜=0 and/or BETA24 ˜=0 and INIT_ICM or INIT_ICM-1 is
    one of these, negative ustep errors may result.

  Next record (If ISOURC=23)
  ****

    the_beam_code, the_pegs_file, the_input_file (3A80)

       the_beam_code   The name of the BEAM code you are running as a
                          source (ie BEAM_sourceaccelname). This must have
                          been compiled as a shared library
                          (libBEAM_accelname.so or BEAM_accelname.dll) and
                          exist in EGS_HOME/bin/config.
       the_pegs_file   The pegs data set used by the BEAM simulation source
                          (no .pegs4dat extension). This must be in
                          HEN_HOUSE/pegs4/data or EGS_HOME/pegs4/data.
       the_input_file   The input file used to run the BEAM simulation
                          source (no .egsinp extension). This must exist in
                          your EGS_HOME/BEAM_sourceaccelname directory. It
                          must be a working input file and must be set up to
                          write a phase space file at a single scoring plane.
                          This plane becomes where particles are sampled from
                          for the second-stage BEAMnrc simulation (no phase
                          space file is scored, however).

  ISOURC = 24   FULL PHASE-SPACE SOURCE INCIDENT FROM USER-DEFINED ANGLE
  ****

  Inputs identical to ISOURC = 21 with the following additional line of

---

3   DESCRIPTION OF MAIN BEAMNRC INPUT FILE

    inputs after SPCNAM:

    ALPHA24,BETA24,DIST24

              ALPHA24   Angle of rotation of source plane about X-axis
                         in degrees. Positive angle is clockwise rotation.
                         (-90 < ALPHA24 < 90)
               BETA24   Angle of rotation of source plane about Y-axis
                         in degrees. Positive angle is counter-clockwise
                         rotation. (-90 < ALPHA24 < 90)
               DIST24   Distance of point of rotation above INIT_ICM.

    Note restriction that if ALPHA24~=0 and/or BETA24 ~=0, then INIT_ICM
    must be > 1. This is because the rotation will result in some
    particles incident within INIT_ICM-1. Also, the following CMs
    currently do not handle the general case of a forward-directed particle
    incident from within them:
    APPLICAT,ARCCHM,CHAMBER,CIRCAPP,CONESTAK,DYNJAWS,JAWS,MESH,PYRAMIDS,
    SIDETUBE
    Thus, if ALPHA24~=0 and/or BETA24 ~=0 and INIT_ICM or INIT_ICM-1 is
    one of these, negative ustep errors may result.

    The initial idea and much of the coding for Source 24 is courtesy of
    Patrick Downes at University of Cardiff, Wales.

  ISOURC = 31   BEAM CHARACTERIZATION MODEL,
  ****
     IQIN,ISOURC,CMSOU
              IQIN   charge of incident beam (not used)
              ISOURC   = 31
              CMSOU   particles start at the front surface of this CM

  Next record (If ISOURC=31)
  ****
     SPCNAM   FILENAME (with EXT) contains information on beam model
                     (maximum of 80 characters)

  Next Record (IF ISOURC <21)   SOURCE ENERGY INPUT
  ***********   ******************

     MONOEN (I8)
                     = 0 if monoenergetic beam (the default)
                     = 1 if energy spectrum to be used
                         Note: BEAMnrc sets MONOEN=2 for phase space inputs

   Next Record (IF MONOEN = 0)
   ****

```
   EIN (F15.0)
               kinetic energy of the incident beam in MeV
               (defaults to 1.25), only for MONOEN=0


Next Record (IF MONOEN = 1)
***********
  FILNAM(80A1)  FILENAME(WITH EXT) contains spectrum information
                which must be in  NRC's ensrcV format.

           -------------------------------------------------------
           FILE FORMAT:
           SPEC_TITLE    (A80)
             NENSRC,ENMIN,IMODE (I10,F15.0,I5)
             (ENSRCD(IB),SRCPDF(IB),IB=1,NENSRC) (2F20.0)
                   NENSRC  # Energy bins in spectrum histogram
                   ENMIN   Lower energy of first bin
                   IMODE   =0 assuming counts/bin, =1 counts/MeV
                   ENSRCD(I),SRCPDF(I)  I=1,NENSRC
                           Top of energy bin and probability of
                           initial particle being in this bin.
                           Probability does not need to be normalized

           -------------------------------------------------------


Next Record (IF MONOEN = 1)
***********
  IOUTSP (I5):
           = 0 no spectrum data in output summary
           = 1 include spectrum data in output summary


------------------------------------------------------------------------------


                   TRANSPORT CONTROL INPUT
                   ***********************
Next Record
***********
  ESTEPIN,SMAX,ECUTIN,PCUTIN,IDORAY,IREJCT_GLOBAL,ESAVE_GLOBAL,IFLUOR

           ESTEPIN Dummy variable (used to be ESTEPE -- max. fractional
                   energy loss/electron step)
           SMAX    Dummy variable (used to be SMAX -- max. step length)
           ECUTIN  electron cut off in MeV - total energy
                   If this is > ECUT as input in the EGSnrc input section
                   (see below), then ECUT is set to ECUTIN.  Default
                   value for ECUT is AE.
           PCUTIN  photon cut off in MeV.  If this is > PCUT as input
                   in the EGSnrc input section (see below), then PCUT
                   is set to PCUTIN.  Default value for PCUT is AP.
           IDORAY  Dummy variable (used to turn Rayleigh scattering on/off)
     IREJCT_GLOBAL = 0 no electron range rejection
                   = 1 do electron range rejection--if residual
```

3   DESCRIPTION OF MAIN BEAMNRC INPUT FILE

```
                            range to ECUTRR(IRL) is < DNEAR and
                            electron energy is < ESAVE_GLOBAL, terminate
                            history.  ECUTRR(IRL) may vary from component
                            module to component module and is calculated based
                            on the particle making it to the bottom of the
                            accelerator with energy > ECUT
                      = 2 as in =1, but use a non-calculated ECUTRR = ECUT(IRL)
                            this should be used if interested in more
                            than phase-space data at base of simulation
                  =-1,-2 Same as above, but now Russian Roulette is played
                            with ALL electrons that can not escape the region
                            and are not fat. Only applicable with DBS.
        ESAVE_GLOBAL  energy below which an electron will be discarded
                            If E<ESAVE_GLOBAL & the electron cannot escape from the
                            current region with E>ECUTRR(IRL). This ignores brem
                            losses.
              IFLUOR  Dummy variable (used to turn X-ray fluorescence on/off)

        The dummy inputs are retained for compatibility with EGS4/BEAM input
        files.
--------------------------------------------------------------------------------


  Next Record        Photon Forcing Controls
  ***********        ***********************
      IFORCE,NFMIN,NFMAX,NFCMIN,NFCMAX
            IFORCE = 0 normal photon transport (the default)
                    = 1 force photon interaction in the geometry
            NFMIN   number of photon interaction/history at which to start
                    photon-interaction-forcing (defaults to 1)--this
                    option has been deleted and NFMIN is now always treated
                    as 1.
            NFMAX   number of photon interaction/history after which to
                    stop forcing photon to interact (defaults to 1)
            NFCMIN  number of CM to start photon interaction forcing
                    ( default to 1 )
            NFCMAX  number of last CM in which photons forced to interact
                    ( default to max_cms )
                    If a particle passes thru NFCMIN to NFCMAX, it is
                    forced to interact there for the first NFMAX
                    interactions. The WEIGHT
                    of this photon is reduced and the remaining WEIGHT is
                    carried by another photon which will be transported
--------------------------------------------------------------------------------


  Next record              SCORING PLANE INPUT
  ***********              *******************

    NSC_PLANES, (IPLANE_to_CM(I), I=1,NSC_PLANES)
```

```
          NSC_PLANES      number of scoring planes >=0
          IPLANE_to_CM    CM numbers corresponding to the scoring planes
                          fluence is scored at the back of a component module
                          phase space data written from same planes


          (Note only IPLANE_to_CM(1) is used for beam model analysis);

   Next record                SCORING ZONE TYPE/DIMENSIONS
   ***********                ***************************


   Repeat the next pair of lines for ISCORE=1,...,NSC_PLANES


     NSC_ZONES(ISCORE), MZONE_TYPE(ISCORE)
          NSC_ZONES      number of scoring zones within each scoring plane
                         (= 0: maximum number available with equal zone area)
          MZONE_TYPE     0 annular zones (default)
                         1 square (ring) zones
                         2 grid

   Next record (for NSC_ZONES(ISCORE)>0 and MZONE_TYPE = 0 or 1)
   ***********
     (RSCORE_ZONE(ISCORE,I), I=1,NSC_ZONES)  (up to 10/line)
          RSCORE_ZONE    outer radius of each scoring zone in order of
                               increasing radius (MZONE_TYPE = 0)
                         half width from origin of each scoring zone
                            in order of increasing width (MZONE_TYPE = 1)

   Next record (for NSC_ZONES(ISCORE)>0 and MZONE_TYPE = 2)
   ***********
     XMIN_ZONE, XMAX_ZONE, YMIN_ZONE, YMAX_ZONE, NX_ZONE, NY_ZONE
           XMIN_ZONE     lower x bound of grid area (cm)
           XMAX_ZONE     upper x bound of grid area (cm)
           YMIN_ZONE     lower y bound of grid area (cm)
           YMAX_ZONE     upper y bound of grid area (cm)
           NX_ZONE       number of grid zones in x direction
           NY_ZONE       number of grid zones in y direction
----------------------------------------------------------------------------


   Next record              DOSE COMPONENTS CALCULATION INPUT
   ***********              ********************************
     ITDOSE_ON
         ITDOSE_ON  = 0 (DEFAULT) only total dose is calculated
                    = 1 total dose and dose components may be calculated

     There are 2 classes of components. First is selected as dose from
     particles or descendents of particular charge as they cross a
     specified boundary.  Second is based on bit selections in the
     variable LATCH, either inclusive or exclusive sets - i.e. depends
     on where particle has been or interacted.
```

3 DESCRIPTION OF MAIN BEAMNRC INPUT FILE

```
  Next record (if ITDOSE_ON=1)
  ***********
     ICM_CONTAM, IQ_CONTAM (2I5)
           All particles of type IQ_CONTAM (0=photons, 1=charged
           particles) are identified as contaminants when they enter
           the front of CM number ICM_CONTAM and their dose is scored as
           contaminant dose in all dose zones.
           If ICM_CONTAM = 0, no contaminant dose is scored.
           LATCH_OPTION = 1 is not allowed with ICM_CONTAM non-zero


  Next record (if ITDOSE_ON=1)
  ***********
     LNEXC (I5)
            LNEXC: # of dose components scored which exclude dose from
                   particles with certain LATCH bits set - i.e. which have
                   not been in certain regions.
                   LNEXC = 0 is allowed. LNEXC <= $MAXIT - 3


  Next records (if LNEXC > 0)
  ************
     (L_N_EXC(I,J), J=1, 31 ) (31I5) (repeat LNEXC times, line by line)
           L_N_EXC(I,J): Bit #s in LATCH for dose component I
                  (will exclude dose from component I if these bits set)


  Next record (if ITDOSE_ON=1)
  ***********
     LNINC (I5)
           LNINC: # of dose components scored for particles
                  from specified regions (with designated bit settings
                  in LATCH).  LNINC <= $MAXIT - LNEXC - 3


  Next records (if LNINC > 0)
  ************
     (L_N_INC(I,J), J=1, 31 ) (31I5) (repeat LNINC times, line by line)
           L_N_INC(I,J): Bit #s in LATCH for dose component I.
                          These are in two groups/line, separated by a zero.
                          Of the first group of bits, at least one must
                          be set to be in this dose component.
                          The second group need not be present, but if
                          it is, none of these bits can be set to be in
                          this dose component.

--------------------------------------------------------------------------------


  Next record              INPUT FRONT SURFACE (Z) FOR CM 1
  ***********              ********************************


     Z_min_CM(1)  Z-coordinate of front surface for component module 1
```

```
              This includes any air gap and defines front of model.
              A common value will be 0.0 except for sources 8 & 9.
              For most sources except for ISOURC=3, 21 & 31, this is
              also the source plane on which the particles are incident
         Note that the front of all CMs is given w.r.t. z = 0.0, not
         w.r.t. Z_min_CM(1).


Next record            Blank or dummy line indicating start of
***********            input for component modules



Next records (many)       COMPONENT MODULE INPUT
*************              **********************


Component module parameters are input in order of their appearance in the
code, that is, in the order they occur in $CM_LIST.  See the 'INPUT FROM
UNIT 5' section in each CM subroutine for the list of input parameters.

 There are two lines before the input of parameters for each CM:
 one is ************************************************ ,
 the other is RMAX_CM, the outer boundary(radius or 1/2 of square) of CM


-------------------------------------------------------------------------------


                   EGSnrc INPUTS
                   *************
  (modified from the description in $HEN_HOUSE/src/get_inputs.mortran)


The input for parameters associated with EGSnrc follows a format
used by all the standard EGSnrc user codes.  The rest of the BEAMnrc
input has not been changed because the GUI's make the details of the
format irrelevant.

All input associated with selection of EGSnrc transport parameters
is not crucial for the execution as there are default values set.
Therefore, if some of the input options in this section are
missing/misspelled, this will be ignored and defualt parameter assumed
As the transport parameter input routine uses get_inputs, a lot
of error/warning messages may be produced on UNIT 15, though.
If you don't have the intention of changing default settings,
simply ignore the error messages.

The delimeters are
            :start mc transport parameter:
            :stop mc transport parameter:

Currently, the following options are available (case does not matter and
            the internal variables are shown in [ ] brackets):
```

3   DESCRIPTION OF MAIN BEAMNRC INPUT FILE

```
Global ECUT=     Global (in all regions) electron transport cut
                 off energy (in MeV). If this imput is missing,
                 or is < ECUTIN from the main BEAMnrc inputs
                 (See above) then ECUTIN is used for Global ECUT.
                 Global ECUT defaults to AE(medium).
                 [ ECUT ]
Global PCUT=     Global (in all regions) photon transport cut
                 off energy (in MeV). If this imput is missing,
                 or is < PCUTIN from the main BEAMnrc inputs
                 (See above) then PCUTIN is used for Global PCUT.
                 Global PCUT defaults to AP(medium).
                 [ PCUT ]
Global SMAX=     Global (in all regions) maximum step-size
                 restriction for electron transport (in cm).
                 No SMAX restriction is necessary if the electron
                 step algorithm is PRESTA-II and the EXACT boundary
                 crossing algorithm (the default) is used.  In this
                 case, SMAX will default to 1e10.  However, if either
                  Electron-step algorithm= PRESTA-I
                  or
                  Boundary crossing algorithm= PRESTA-I,
                 then a step-size restriction is necessary, and
                 SMAX will default to 5 cm.
                 [ SMAXIR ]


ESTEPE=          Maximum fractional energy loss per step.
                 Note that this is a global option only, no
                 region-by-region setting is possible. If missing,
                 the defualt is 0.25 (25%).
                 [ ESTEPE ]
XImax=           Maximum first elastic scattering moment per step.
                 Default is 0.5, NEVER use value greater than 1 as
                 this is beyond the range of MS data available.
                 [ XIMAX ]
Boundary crossing algorithm=
                 There are two selections possible: EXACT and
                 PRESTA-I.  PRESTA-I means that boundaries will
                 be crossed a la PRESTA.  That is, with lateral
                 correlations turned off at a distance given by
                 'Skin depth for BCA' (see below) from the boundary
                 and MS forced at the boundary.  EXACT means
                 the algorithm will cross boundaries in a single
                 scattering (SS) mode, the distance from a boundary
                 at which the transition to SS mode is made is
                 determined by 'Skin depth for BCA' (see below).
                 Default is EXACT since PRESTA-I may result in
                 significant dose overestimates when CHAMBER is
                 used as a phantom, and EXACT will not significantly
                 increase CPU time in most accelerators.
```

```
                         [ bca_algorithm, exact_bca ]
        Skin depth for BCA=
                         If Boundary crossing algorithm= PRESTA-I
                         then this is the distance from the boundary (in
                         elastic MFP) at which lateral correlations will be
                         switched off.  The default in this case is to
                         calculate a value based on the scattering power at
                         ECUT (same as PRESTA with EGS4).  If
                         Boundary crossing algorithm= EXACT (default) then
                         this is the distance from the boundary (in elastic
                         MFP) at which the algorithm will go into single
                         scattering mode and defaults to 3 mfp.
                         Note that if you choose EXACT boundary crossing and
                         set Skin depth for BCA to a very large number (e.g.
                         1e10), the entire calculation will be in SS mode.
                         If you choose PRESTA-I boundary crossing and make
                         Skin depth for BCA large, you will get default EGS4
                         behaviour (no PRESTA).
                         [ skindepth_for_bca ]


        The new transport mechanics of EGSnrc are maintained away from
        boundaries.


    Electron-step algorithm=
                         PRESTA-II (the default), the name is
                         used for historical reasons
                         or PRESTA-I
                         Determines the algorithm used to take into account
                         lateral and longitudinal correlations in a
                         condensed history step.
                         [ transport_algorithm ]
    Spin effects=    Off, On, (default is On)
                         Turns off/on spin effects for electron elastic
                         scattering. Spin On is ABSOLUTELY necessary for
                         good backscattering calculations. Will make a
                         difference even in 'well conditioned' situations
                         (e.g. depth dose curves for RTP energy range
                         electrons).
                         [ spin_effects ]
    Brems angular sampling= Simple, KM, (default is Simple)
                         If Simple, use only the leading term of the Koch-Motz
                         distribution to determine the emission angle of
                         bremsstrahlung photons. If KM, complete
                         modified Koch-Motz 2BS is used (modifications
                         concern proper handling of kinematics at low
                         energies, makes 2BS almost the same as 2BN at low
                         energies).
                         [ IBRDST ]
    Brems cross sections= BH, NIST, NRC, default is BH
```

3   DESCRIPTION OF MAIN BEAMNRC INPUT FILE

                       If BH is selected, the Bethe-Heitler bremsstrahlung
                       cross sections (Coulomb corrected above 50 MeV)
                       will be used. If NIST is selected, the NIST brems
                       cross section data base (which is the basis for
                       the ICRU radiative stopping powers) will be employed.
                       Differences are negligible for E > ,say, 10 MeV,
                       but signifficant in the keV energy range. If NRC is
                       selected, NIST data including corrections for
                       electron-electron brems will be used (typically only
                       significant for low values of the atomic number Z
                       and for k/T < 0.005).
   Bound Compton scattering= On, Off or Norej (Default is Off)
                       If Off, Compton scattering will be treated with
                       Klein-Nishina, with On Compton scattering is
                       treated in the Impulse approximation.
                       Make sure to turn on for low energy applications,
                       not necessary above, say, 1 MeV. Option Norej
                       uses full bound Compton cross section data
                       supplied in input below and does not reject
                       interactions.
                       [IBCMP]
   Compton cross sections= Bound Compton cross-section data. User-
                       supplied bound Compton cross-sections in the file
                       $HEN_HOUSE/data/comp_xsections_compton.data, where
                       comp_xsections is the name supplied for this input.
                       This is only used if Bound Compton scattering= Simple
                       and is not available on a region-by-region basis
                       (see below). The default file (ie in the absence
                       of any user-supplied data) is compton_sigma.data.
                       [comp_xsections]
   Radiative Compton corrections= On or Off (default). If on, then
                       include radiative corrections for Compton scattering.
                       Equations are based on original Brown & Feynman
                       equations (Phys. Rev. 85, p 231--1952). Requires
                       a change to the user codes Makefile to include
                       $(EGS_SOURCEDIR)rad_compton1.mortran in the
                       SOURCES (just before $(EGS_SOURCEDIR)egsnrc.mortran).
                       [radc_flag]

   Pair angular sampling= Off, Simple or KM (Default is Simple)
                       If off, pairs are set in motion at an angle m/E
                       relative to the photon direction (m is electron rest
                       energy, E the photon energy). Simple turns on
                       the leading term of the angular distribution
                       (this is sufficient for most applications),
                       KM (comes from Koch and Motz) turns on using 2BS
                       from the article by Koch and Motz.
                       Always use Simple or KM.
                       [IPRDST]

Pair cross sections= BH (default) or NRC. If set to BH, then use
                    Bethe-Heitler pair production cross-sections. If set
                    to NRC, then use NRC pair production cross-sections
                    (in file $HEN_HOUSE/data/pair_nrc1.data). Only
                    of interest at low energies, where the NRC cross-
                    sections take into account the assymmetry in the
                    positron-electron energy distribution.
                    [pair_nrc]
Photoelectron angular sampling= Off or On (Default is Off)
                    If Off, photo-electrons get the direction of the
                    'mother' photon, with On, Sauter's furmula is
                    used (which is, striktly speaking, valid only for
                    K-shell photo-absorption).
                    If the user has a better approach, replace the macro
                        $SELECT-PHOTOELECTRON-DIRECTION;
                    The only application that
                    Only situation encountered where this made a
                    small difference was a big ion chamber (cavity size
                    comparable with electron range) with high-Z walls
                    in a low energy photon beam.
                    [IPHTER]
Rayleigh scattering= Off, On, custom
                    If On, turned on coherent (Rayleigh) scattering.
                    Default is Off. Should be turned on for low energy
                    applications. Not set to On by default because
                    On requires a special PEGS4 data set. If set to
                    custom, then media for which custom form factors
                    are to be specified are listed in the input:
                    ff media names=
                    and the corresponding files containing custom data
                    are listed in:
                    ff file names=
                    [IRAYLR]
Atomic relaxations= Off, On (Default is Off)
                    The effect of using On is twofold:
                    - In photo-electric absorption events, the element
                      (if material is mixture) and the shell the photon
                      is interacting with are sampled from the
                      appropriate cross seections
                    - Shell vacancies created in photo-absorption events
                      are relaxed via emission of fluorescent X-Rays,
                      Auger and Koster-Cronig electrons.
                     Make sure to turn this option on for low energy
                     applications.
                     [IEDGFL]

Electron impact ionization= Off, On, Casnati, Kolbenstvedt, Gryzinski
                           (Default is Off)
                     Determines which, if any, theory is used to model

```
                        electron impact ionization. If set to 'On' then the
                        theory of Kawrakow is used.  Other settings use the
                        theory associated with the name given.  See future
                        editions of the EGSnrc Manual (PIRS-701) for more
                        details.  This is only of interest in keV X-Ray
                        simulations.  Otherwise, leave it Off.
                        [ eii_flag ]
      Photon cross sections= epdl,xcom,custom (Default is Storm-Israel
                        cross-sections from PEGS4)
                        The name of the cross-section data for photon
                        interactions.  This input line must be left out
                        to access the default Storm-Israel cross-sections
                        from PEGS4.  'edpl' uses cross-sections from the
                        evaluated photon data library (EPDL) from Lawrence
                        Livermore.  'xcom' will use the XCOM cross-sections
                        from Burger and Hubbell.  The user also has the
                        option of using their own customized cross-section
                        data.  See the BEAMnrc manual for more details.
                        [ photon_xsections ]
      Photon cross-sections output= Off (default) or On.  If On, then
                        a file $EGS_HOME/user_code/inputfile.xsections is
                        output containing photon cross-section data used.
                        [ xsec_out ]


      Atomic relaxations, Rayleigh scattering, Photoelectron angular
      sampling and Bound Compton scattering can also be turned On/Off
      on a region-by-region basis. To do so, put e.g.
      Atomic relaxations= On in Regions    or
      Atomic relaxations= Off in regions
                        in your input file. Then use the relevant one of:
      Relaxations start region=
      Relaxations stop region=
              or
      Bound Compton start region=
      Bound Compton stop region=
              or
      Rayleigh start region=
      Rayleigh stop region=
              or
      PE sampling start region=
      PE sampling stop region=


          each followed by a list of one or more
          start and stop regions separated by commas.
       Example:
       Atomic relaxations= On in Regions
       Relaxations start region=  1, 40
       Relaxations stop region=  10, 99
                        will first turn off relaxations everywhere and
```

```
                          then turn on in regions 1-10 and 40-99.
                          Note that input is checked against min. and max.
                          region number and ignored if
                          start region < 1 or stop_region > $MXREG or
                          start region > stop region.
```

                          Rejection Plane Inputs
                          ****

Used to define a rejection plane for use in conjunction with directional
bremsstrahlung splitting (DBS, IBRSPL=2, see above).
Inputs can exist without IBRSPL=2, but they will not be used.

Inputs must appear between the delimiters:
:Start DBS rejection plane:
:Stop DBS rejection plane:

Inputs are:

    Use a rejection plane= Off, On (default is Off)
                         Set to On if you want to define a rejection
                         plane.
                         [USE_REJPLN]
    Z(cm) from zero reference plane= Z position of reference plane.
                         [Z_REJPLN]

    Fat photons and electrons will be discarded if they are about
    to interact at Z>=Z_REJPLN. Used to prevent correlated particles
    from being created close to a scoring plane, compromising
    statistics.

        Bremsstrahlung Cross Section Enhancement (BCSE) Inputs
        ****

Inputs for the BCSE variance reduction technique.

Inputs must appear between delimiters:
:Start BCSE:
:Stop BCSE:

Inputs are:

     Use BCSE= Off, On (default is Off)
              Set to On to use BCSE.

---

3 DESCRIPTION OF MAIN BEAMNRC INPUT FILE

```
                    [USE_BCSE]
        Media to enhance= A list of media in
                which to enhance the bremsstrahlung cross-section.  If
                none of the media is found in the accelerator, then no
                BCSE is done.
                [is_bcse_medium]
        Enhancement constant= Floating point factor by which bremsstrahlung
                cross-sections are enhanced. Typical values are in the range
                20 (megavoltage accelerators) -- 500 (x-ray tubes in
                mammography energy range).
                [BCSE_FACTOR_C]
        Enhancement power= Floating point number that us used for an energy
                dependent BCSE. If this input is <=0, then a constant
                BCSE factor is used that is given by BCSE_FACTOR_C. But if
                this input is >0, then the BCSE factor is computed on-th-fly
                using 1 + BCSE_FACTOR_C*(E(np)-rm)**BCSE_POWER_N.
                Typical values for BCSE_POWER_N are 2...4. Note that
                BCSE_FACTOR_C must be adjusted accordingly so that the above
                equations gives a factor of 20 (megavoltage accelerators) --
                500 (low energy x-ray tubes) for the maximum energy of the
                incident electron spectrum.
                If in doubt, just set to <=0 because the gain from an
                energy dependent BCSE is modest (~20%).
```

```
Note that if BCSE is used in conjunction with uniform bremsstrahlung
splitting (UBS) or selective bremsstrahlung splitting (SBS), then
Russian Roulette is automatically turned on (IRRLTT=2--see above).
```

```
BCSE is most efficient when used in conjunction with DBS or UBS
```

```
--------------------------------------------------------------------------

                        CUSTOMIZED USER INPUTS
                        **********************
```

```
This section contains inputs that are unique to the user.  In general
the user must modify beamnrc.mortran to read these inputs from the
.egsinp file.  Also, the GUI will not give access to these inputs.
If no custom inputs are required, then this section can be omitted
entirely.
```

```
Custom user inputs must appear in the .egsinp file between the delimiters:
     :Start user inputs:
     :Stop user inputs:
```

```
This section can appear either just before or just after the EGSnrc
inputs (see above).
```

```
   Currently, the only custom user input hard-coded in beamnrc.mortran
   is:

    PHSP OUTPUT DIRECTORY= /full directory path to where phase space files
                             are to be output, or blank
              This allows the user to specify a directory other than
              $EGS_HOME/BEAM_accelname in which to write phase space data.
              Useful if phase space files are large and, due to disk space
              limitations, must be written to a /temp area.  If left blank
              or omitted entirely, then phase space files are output to the
              default $EGS_HOME/BEAM_accelname directory.
              [ PHSP_OUTDIR ]


   ****************************************************************************
```

3.1   Sample input files

Complete input files for simulation of a 10 MeV electron beam and a 16 MV photon beam can be found in the appropriate subdirectories of:
`$OMEGA_HOME/beamnrc/BEAMnrc_examples` in files `EX10MeVe.egsinp` and `EX16MVp.egsinp` respectively. The file `EXphantom.egsinp` gives an example of using BEAMnrc to do an efficient depth-dose calculation (but only for the central axis). These examples are described in separate reports[21, 22, 23] which are somewhat dated since one should now use the GUIs to run these examples.

The auxiliary script `test_BEAMnrc` (found on `$HEN_HOUSE/scripts`) automatically create the needed subdirectories on the user's area (`BEAM_EX10MeVe` *etc.*) and copy the relevant files to them so that the user may execute BEAMnrc immediately (but the script only works on Linux/Unix and there is no corresponding replacement for Windows).

4   Source Routines

In general, the incident particles move in the direction of the z-axis. With the exception of ISOURC=3 (internal isotropic source), =10 and 13 (x-ray tube sources) or =21 and 31 (phase space inputs), the particles start being transported on the `Z_min_CM(1)` plane. Conceptually, some of them originate at a point outside the accelerator model and are essentially transported through vacuum to the accelerator which starts at the `Z_min_CM(1)` plane.

One of the major features of BEAMnrc is that a phase space file can be used as a source file between any two `CMs` in the accelerator. By writing a suitable off-line routine to generate the appropriate phase space file (use the macros in `phsp_macros.mortran`), virtually any source can be simulated. This feature also allows the accelerator simulation to be broken up into components. For example, one might simulate the fixed components at the top of an accelerator and then feed the phase space file into a variety of calculations with different jaw or applicator settings.

For historic and compatibility reasons, the input to specify the source type and the charge of the incident beam are on a single line (as outlined in section 3). In particular, note that the first variable is the charge, `IQIN`, or a dummy in those cases where the charge is determined elsewhere (*eg.* for a phase space file input).

The following subsections describe the source routines available in BEAMnrc.

ISOURC=0 Parallel Circular Beam

ISOURC=1 Isotropic Point Source on Z-axis

ISOURC=3 Interior Isotropic Cylindrical Source

ISOURC=5 NRC Swept Beam

ISOURC=6 Parallel Rectangular Beam

ISOURC=7 Scanning Beam (sawtooth)

ISOURC=8 Scanning Point Source for MM50

ISOURC=9 Discrete Point Source for MM50

ISOURC=10 Parallel Circular Beam Incident from Side

ISOURC=13 Parallel Rectangular Beam Incident from Side

ISOURC=15 NRC Swept Beam with Radial Intensity Distribution and Radial Divergence

ISOURC=19 Elliptical Beam with Gaussian Distributions in X and Y, Parallel or with Radial Divergence

ISOURC=21 Phase Space Source

ISOURC=23 BEAMnrc Simulation Source

Last edited 2012/10/09 17:59:26

ISOURC=24 Phase Space Source Incident from User-specified Angle

ISOURC=31 Beam characterization Model

4.1   ISOURC=0: Parallel Circular Beam

<div align="center">IQIN, 0, RBEAM, UINC, VINC, WINC</div>

The parallel circular beam is always assumed to be incident on the center of the front of the first CM (*i.e.* at `Z_min_CM(1)`). The input parameters are:

IQIN The charge of the incident beam (defaults to 0=photons)

RBEAM Radius of the beam in cm (maximum is `RMAX_CM(1)` for a circular first CM or $\sqrt{2}$`RMAX_CM(1)` for a square first CM, it defaults to maximum if set > maximum)

UINC X-axis direction cosine (defaults to 0)

VINC Y-axis direction cosine (defaults to 0)

WINC Z-axis direction cosine (defaults to 1, *i.e.* parallel to the z-axis)

UINC, VINC, and WINC are automatically normalized by $(\text{UINC}^2+\text{VINC}^2+\text{WINC}^2)$. Note that it is possible to have `RBEAM` $= 0$; this is a pencil beam. Figure 6 below shows the parallel circular beam and its input parameters.
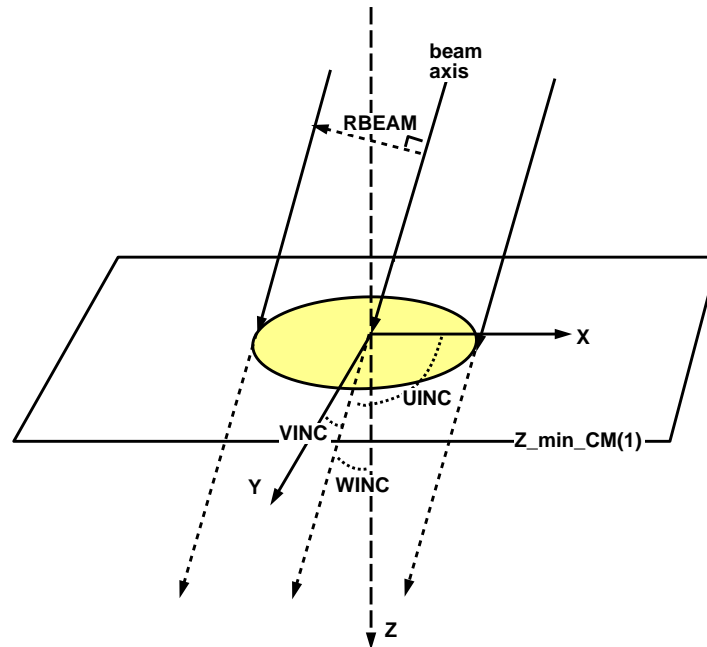
Figure 6: Parallel Circular Beam (ISOURC=0) showing the beam radius, `RBEAM`, and the direction cosines, UINC, VINC and WINC. Note that `RBEAM` is measured perpendicular to the beam central axis. The beam axis always intersects `Z_min_CM(1)` at (X=0,Y=0).

4.2 ISOURC=1: Isotropic Point Source on Z-axis

IQIN, 1, DISTZ, RBEAM, GAMMA, XINL, XINU, YINL, YINU

The isotropic point source is placed on the Z-axis. It is assumed to be incident on the first CM, but can be placed any distance above `Z_min_CM(1)`. The beam field can be circular or rectangular (see below). The circular beam is centred on the Z axis, while the rectangular beam can be placed anywhere on the top surface of CM 1.

DISTZ Distance of the source above `Z_min_CM(1)` in cm (defaults to 100 cm)

RBEAM Radius of circular field at `Z_min_CM(1)` (if > 0). If `RBEAM`=0, then `GAMMA` (see below) is used to define the circular field. If `RBEAM`<0, then `XINL, XINU, YINL, YINU` (see below) are used to define a rectangular field. Maximum value of `RBEAM` is `RMAX_CM(1)` (circular CM 1) or $\sqrt{2}$`RMAX_CM(1)` (square CM 1). `RBEAM` defaults to maximum if it is set > maximum or if `RBEAM`=0.0 and `GAMMA` = 0.0.

GAMMA Half-angle of circular field at `Z_min_CM(1)` relative to the point source (Note that this is simply another way of specifying the radius of a circular field, and is only used if `RBEAM` = 0.0; defaults to give maximum radius [see description of `RBEAM` above] if it is set so that the radius would be > maximum).

XINL, XINU, YINL, YINU Min. X, Max. X, Min. Y, Max Y dimensions of rectangular field (only used if `RBEAM`<0) in cm. The rectangular field is restricted to -`RMAX_CM(1)`≤X≤`RMAX_CM(1)` and -`RMAX_CM(1)`≤Y≤`RMAX_CM(1)`. The dimensions can be set to give a line or point "field" anywhere on the top of CM 1.

`DISTZ`, between the source and `Z_min_CM(1)` is assumed to be a vacuum. Figure 7 below illustrates the point source and its input variables.
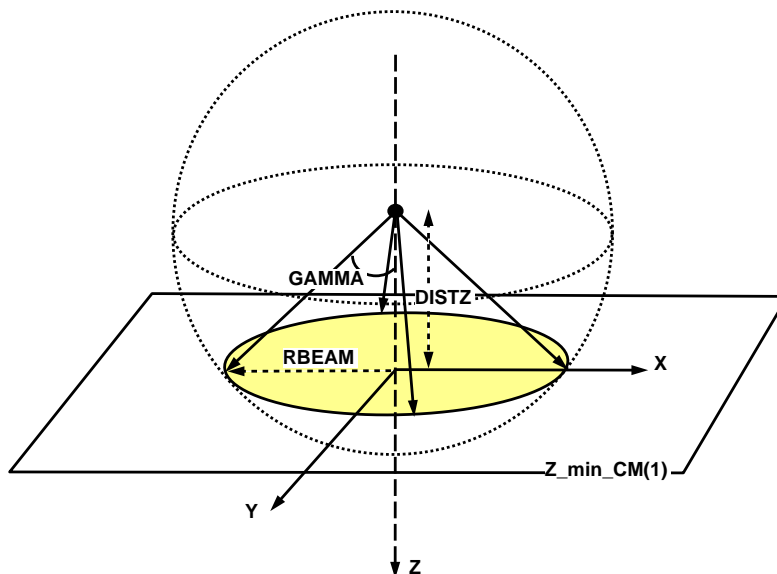
Figure 7: Point source (ISOURC=1) showing the distance of the source above `Z_min_CM(1)`, specified by `DISTZ`, and the two different ways of specifying a circular field at `Z_min_CM(1)`, `RBEAM` and `GAMMA`. A rectangular beam anywhere on the top surface of CM 1 can be specified by setting `RBEAM`<0 and using `XINL, XINU, YINL, YINU` to define the beam.

4.3   ISOURC=3: Interior Isotropic Cylindrical Source

IQIN, 3, RMINBM, RBEAM, ZSMIN, ZSMAX

The uniform radiating cylindrical isotropic source can be a ring centred on the Z-axis or a cylinder with its central axis parallel to the X-axis. The orientation is controlled by the sign of the input variable, RBEAM (see below). The source is contained within the geometry of one or more component modules (*i.e.* it is able to span adjacent CMs). Currently, this source can only be used inside CONESTAK, SIDETUBE or FLATFILT CMs because of the need to identify the initial regions.   The input parameters for this source are:

RMINBM Inner radius of vertical ring (if $RBEAM \geq 0$) *or* Z position of centre of horizontal cylinder (if $RBEAM < 0$)

RBEAM Outer radius of vertical ring (if ≥ 0) *or* negative radius of horizontal cylinder (if < 0) (for vertical ring, maximum value is RMAX_CM of largest CM containing the source; for horizontal cylinder, maximum value keeps the radius entirely contained between Z_min_CM(1) and the bottom of the geometry; defaults to maximum if absolute value is > maximum)

ZSMIN Z of top of vertical ring (if $RBEAM \geq 0$) *or* minimum X of horizontal cylinder (if $RBEAM < 0$) (for vertical ring, minimum value is Z_min_CM(1); for horizontal cylinder, minimum is -RMAX_CM of largest CM containing source; defaults to minimum if set < minimum)

ZSMAX Z of bottom of vertical ring (if $RBEAM \geq 0$) *or* maximum X of horizontal cylinder (if $RBEAM < 0$) in cm (for vertical ring, maximum is the bottom of geometry; for horizontal cylinder, maximum is RMAX_CM of largest CM containing source; defaults to maximum if set > maximum)
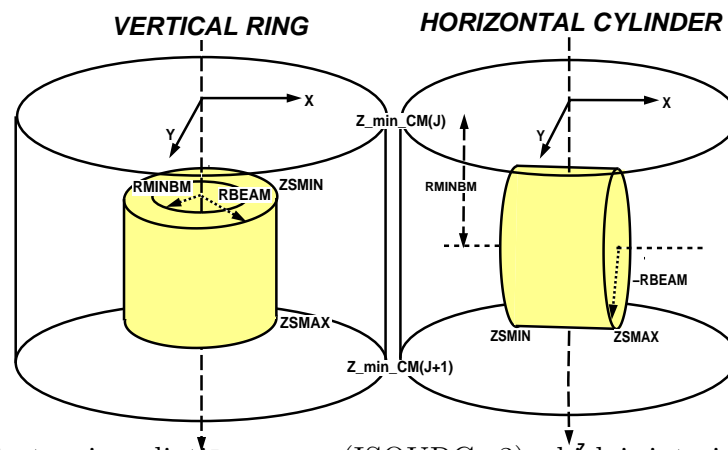
Figure 8: Uniform isotropic radiating source (ISOURC=3) which is interior to the accelerator model. For vertical ring configuration shown on the left, set $RBEAM \geq 0$. For horizontal cylinder configuration on right set $RBEAM < 0$. The meaning of the input variables depends on the configuration. The source is shown within a single CM, however it can span any number of adjacent CMs.

4.4   ISOURC=5: NRC Swept BEAM

<div align="center">IQIN, 5, GAMMA, RBEAM</div>

The NRC swept beam is a parallel circular beam swept around the outside of an imaginary cone. The beam is assumed incident on the front of CM 1. The apex of the imaginary cone is always at $(X=0, Y=0, Z=$`Z_min_CM(1)`$)$, and the cone angle is variable. Input parameters for this source are:

IQIN Charge of the incident beam (defaults to 0 = photons)

GAMMA Half-angle of the cone in degrees ($0^o \le$ `GAMMA` $< 90^o$)

RBEAM Radius of the beam in cm (maximum value `RMAX_CM(1)` if the first CM is circular or $\sqrt{2}$`RMAX_CM(1)` if the first CM is square, defaults to maximum if set $>$ maximum)

Figure 9 below shows the swept beam source and its input parameters.
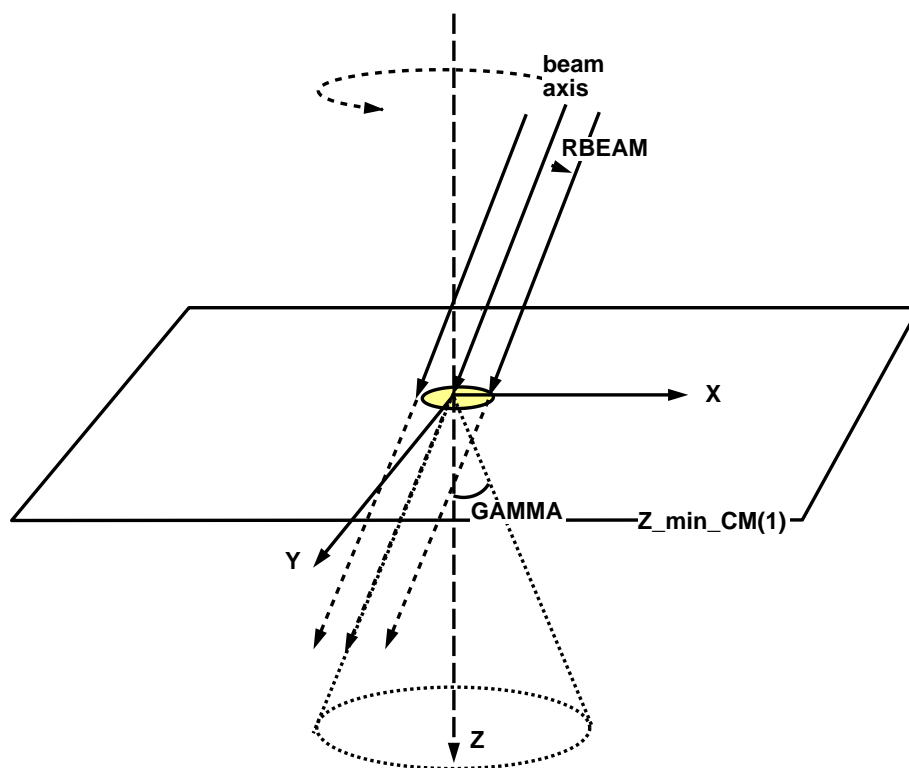
Figure 9: NRC swept beam source (ISOURC=5) showing the incident beam, with radius `RBEAM`, sweeping the surface of an imaginary cone with apex half-angle `GAMMA`. Note that it is the beam axis which actually sweeps the surface of the cone.

4.5   ISOURC=6: Parallel Rectangular Beam

IQIN, 6, XBEAM0, YBEAM0, XBEAM, YBEAM

The parallel rectangular beam is the rectangular equivalent of the parallel circular beam (ISOURC=0) with the exception that the beam cannot be oblique, it is always perpendicular to the front surface of the first CM and this beam can be offset from the Z-axis. Input parameters for the parallel rectangular beam are:

IQIN Charge of beam (defaults to 0 = photons)

XBEAM0 X position of beam centre in cm (defaults to 0 if $\texttt{XBEAM0}^2 + \texttt{YBEAM0}^2 > \texttt{RMAX\_CM(1)}^2$ when the first CM is circular or if $|\texttt{XBEAM0}|$ or $|\texttt{YBEAM0}| > \texttt{RMAX\_CM(1)}$ when the first CM is a square)

YBEAM0 Y position of beam centre in cm (defaults to 0 under same conditions as XBEAM0)

XBEAM Half-width of beam in X-direction in cm (defaults to keep $|\texttt{XBEAM0}| + |\texttt{XBEAM}| \leq \texttt{RMAX\_CM(1)}$)

YBEAM Half-width of beam in Y-direction in cm (defaults to keep $|\texttt{YBEAM0}| + |\texttt{YBEAM}| \leq \texttt{RMAX\_CM(1)}$)
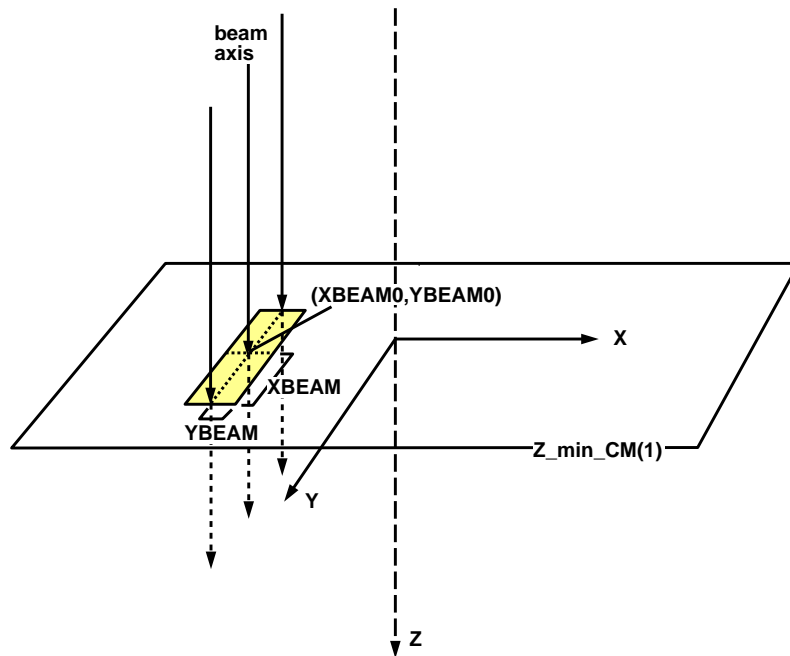
Figure 10:   Parallel rectangular beam (ISOURC=6) showing the beam centre (XBEAM0,YBEAM0) and the X and Y half-widths (XBEAM,YBEAM).

4.6   ISOURC=7: Scanning Sawtooth Beam

IQIN, 7, FD_AT100, IRATIO_YXF, RBEAM

In the scanning sawtooth beam source, a circular parallel beam, incident on the front of the first CM, is scanned in a zig-zag pattern on the X-Y plane. The angle for the scan is selected as the particles leave the `Z_min_CM(1)` plane. This is an approximation of the real case where the beam is bent below this point usually. The source randomly selects points in the scan. The input parameters are:

IQIN Charge of the incident beam (defaults to 0 = photons)

FD_AT100 Scanning field size at SSD=100cm in cm (field is `FD_AT100` x `FD_AT100`). This is often much greater than the actual field size.

IRATIO_YXF The number of Y scans per X scan (defaults to 6.5 if set ≤ 0; also, 2x`IRATIO_YXF` is always rounded up to the nearest odd integer)

RBEAM Radius of the incident beam in cm (defaults to 0.01 cm if set ≤ 0; defaults to maximum of `RMAX_CM(1)` [circular CM 1] or $\sqrt{2}$`RMAX_CM(1)` [square CM 1] if set > maximum)

Note that the field is always scanned twice in the X direction; thus, the number of Y scans is 2x`IRATIO_YXF`. It is also important to note that `FD_AT100` defines the field size covered by the beam central axis; a beam with finite `RBEAM` will actually go outside `FD_AT100`. Figure 11 below shows the scanning beam and its various input parameters.
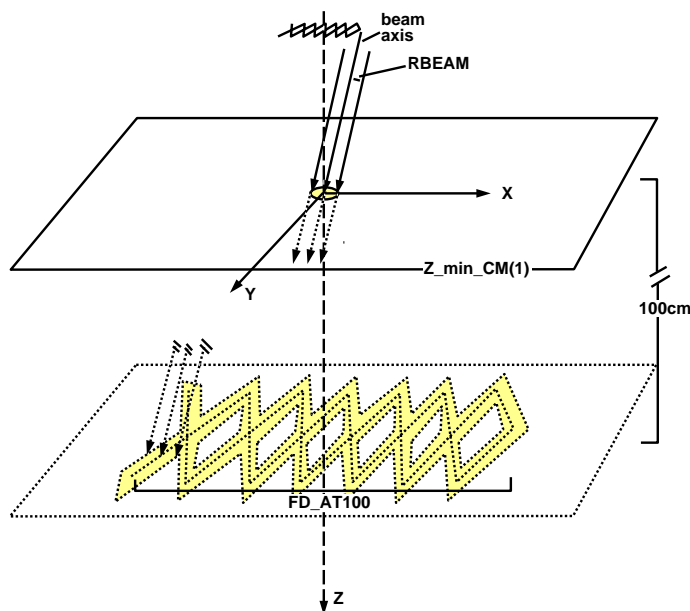
Figure 11: Scanning beam (ISOURC=7). The parallel circular beam of radius `RBEAM` is scanned in a zig-zag pattern in the X-Y plane to produce the zig-zag coverage at SSD=100cm. In the case shown here, `IRATIO_YXF` = 6.5; hence there are a total of 13 Y scans.

4.7   ISOURC=8: Scanned Point Source for MM50–Uniform Field

<div align="center">IQIN, 8, DISTZ, RBEAM, RBEAM0</div>

This source is designed to simulate the source used in the MM50 linear accelerator for electron beams. The source behaves like a point source located at Z=0 scanned to produce a uniform distribution of particles at a user-specified distance from the source (DISTZ) over a field of user-specified radius (RBEAM). Uncertainty in the X-Y position of the source at Z=0 can be simulated using the beam spot radius, RBEAM0. If RBEAM0 > 0, the particle distribution at DISTZ will be uniform out to RBEAM and tails off between RBEAM and RBEAM + RBEAM0. Note that Z_min_CM(1) must be ≥ 0 for this source. Input parameters are:

IQIN Charge of the incident beam (defaults to 0 = photon)

DISTZ The source-to-surface distance (SSD) at which uniform particle distribution is desired in cm (defaults to 100 cm if it is set ≤ 0)

RBEAM Radius of the field at DISTZ in cm (defaults to maximum value of RMAX_CM(1)*DISTZ/Z_min_CM(1) for circular CM 1 or $\sqrt{2}$RMAX_CM(1)*DISTZ/Z_min_CM(1) for square CM 1)

RBEAM0 Radius of the beam spot at Z=0 in cm (defaults to 0 if set < 0 and gets reset to max. value of RBEAM - RBEAM if RBEAM+RBEAM0 > max. value of RBEAM)
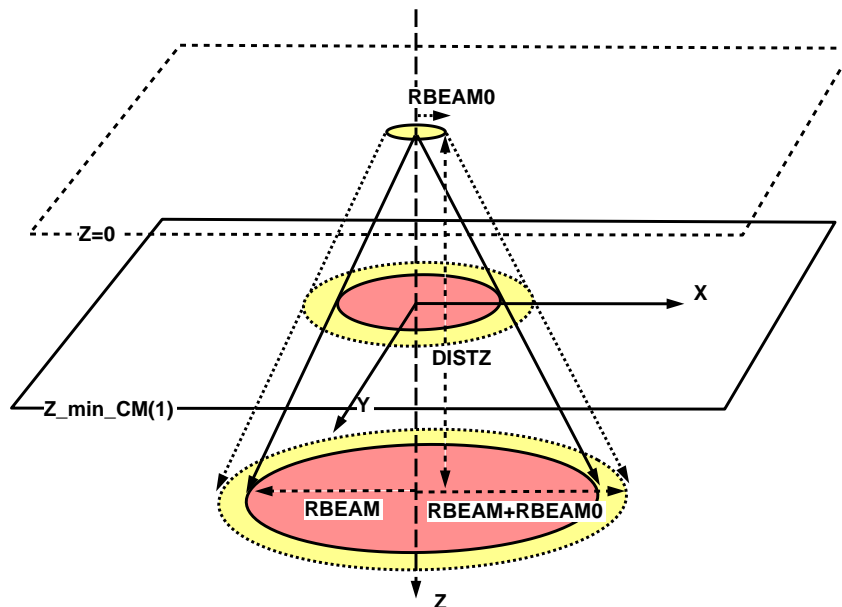
Figure 12: Scanned point source producing uniform field coverage (ISOURC=8). The figure shows a point source at Z=0 scanned to give uniform particle distribution in a field of radius RBEAM at a distance DISTZ below the source. The uncertainty of the position of the source at Z=0, is simulated using RBEAM0. Particles can originate anywhere within the circle defined by RBEAM0 which results in a particle distribution at DISTZ which is uniform out to RBEAM and falls off between RBEAM and RBEAM+RBEAM0.

4.8   ISOURC=9: Scanned Point Source for MM50–Discrete Field

IQIN, 9, DISTZ, NPTS_SRC9
X_SRC9, Y_SRC9, PROB_SRC9

This source is designed to simulate the source used in the MM50 linear accelerator to produce photon beams. It behaves like a point source emanating pencil beams to a finite number of dwell points, providing discrete field coverage. The user specifies by the X,Y coordinates of the dwell points on a plane perpendicular to the Z-axis at a user-specified source-to-surface distance (SSD) and the probability of a particle being initially directed towards each point. The source is always considered to be at Z=0. Input parameters are:

IQIN Charge of the incident beam (defaults to 0 = photon)

DISTZ The source-to-surface distance (SSD) in cm (defaults to 100 cm if it is set ≤ 0)

NPTS_SRC9 The number of points used to cover the field (defaults to 1 if set ≤ 0)

X_SRC9(I) (I=1,...,NPTS_SRC9) The X coordinate of point I on plane $\perp$ Z at `DISTZ` in cm

Y_SRC9(I) (I=1,...,NPTS_SRC9) The Y coordinate of point I on plane $\perp$ Z at `DISTZ` in cm

PROB_SRC9(I) (I=1,...,NPTS_SRC9) The probability of a particle being incident on point I. Probabilities are automatically normalized.
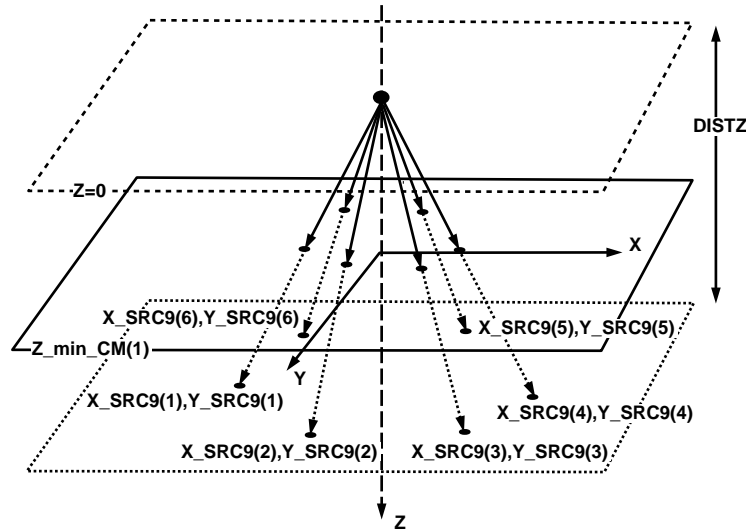
Figure 13: Scanned point source for discrete field coverage (ISOURC=9). The figure shows a point source at Z=0 producing 6 points scattered over the field (`NPTS_SRC9`=6). The user specifies the source-to-surface distance, `DISTZ`, the X and Y coordinates of the points at `DISTZ`, (`X_SRC9(I)`,`Y_SRC9(I)`), and the probability of a particle being incident at each point, `PROB_SRC9(I)` (not shown).

4.9 ISOURC=10: Parallel Circular Beam Incident from Side

IQIN, 10, RBEAM, UINC, VINC, WINC

This parallel circular beam enters the first CM from the side and can only be used as the source for an XTUBE. The input parameters for this source are:

IQIN Charge of incident beam (defaults to 0 = photons)

RBEAM Radius of beam in cm (defaults to half the thickness of the first CM if it is greater than this and to 0 if it is set < 0)

UINC Incident X-axis direction cosine (should be < 0 so the beam faces the Z-axis; reset to -UINC if it is > 0)

VINC Incident Y-axis direction cosine

WINC Incident Z-axis direction cosine
(UINC, VINC, WINC default to -1,0,0 if $UINC^2+VINC^2+WINC^2 = 0$)

The direction cosines UINC, VINC and WINC are each automatically normalized by $(UINC^2+VINC^2+WINC^2)$.
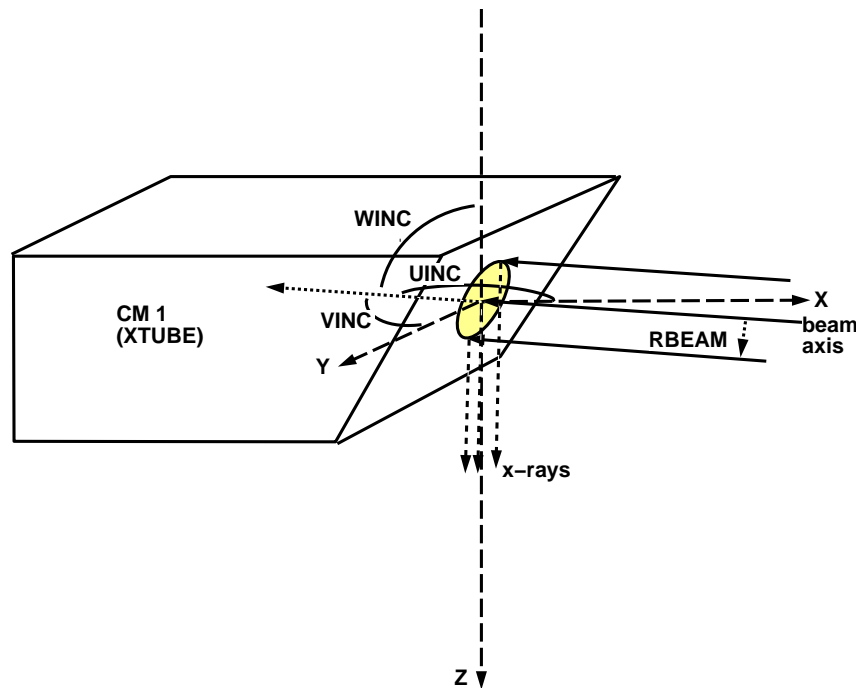
Figure 14: Parallel circular beam incident from the side (ISOURC=10). This source can only be used with an XTUBE CM as the first CM position.

4.10   ISOURC=13: Parallel Rectangular Beam Incident from Side

<div align="center">IQIN, 13, YBEAM, ZBEAM, UINC, VINC</div>

This beam is the rectangular equivalent of ISOURC=10 (see previous source). The input parameters for this beam are:

IQIN Charge of incident beam (defaults to 0 = photons)

YBEAM Half-width of beam in cm (defaults to 0.2 cm if set < 0; defaults to half the thickness of the first CM if set > than this)

ZBEAM Half-height of beam in cm (defaults to 0.2 cm if set < 0; defaults to half the thickness of the first CM if set > than this)

UINC Incident X-axis direction cosine (should be <0 so that beam faces Z-axis; reset to -UINC if it is set > 0)

VINC Incident Y-axis direction cosine
(UINC, VINC default to -1,0 if $UINC^2 + VINC^2 = 0$)

UINC and VINC are automatically normalized by $(UINC^2 + VINC^2)$. Note that the Z direction cosine, WINC, is always assumed to be 0 for this beam.
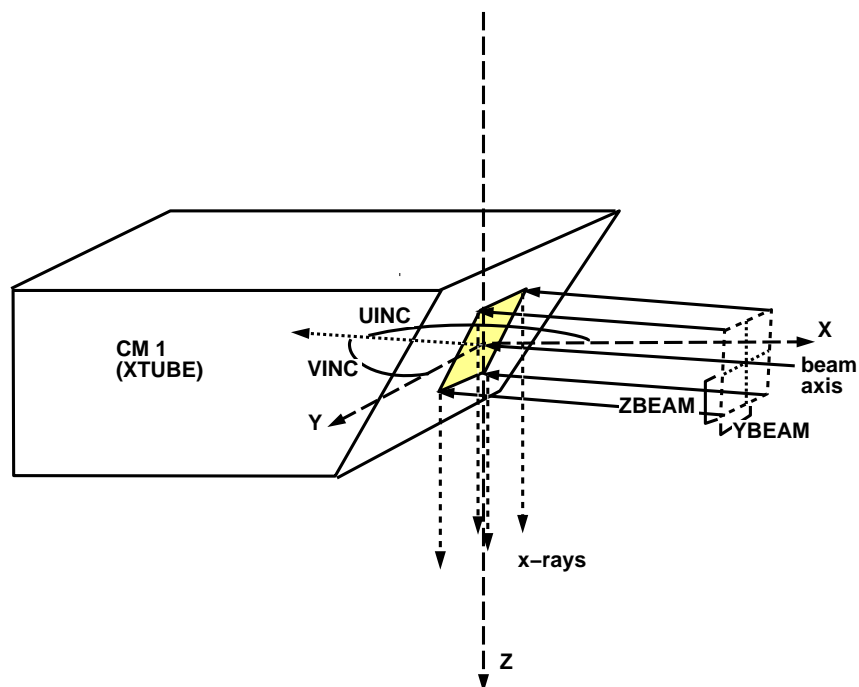
Figure 15: Parallel rectangular beam incident from the side (ISOURC=13).

4.11   ISOURC=15: NRC Swept Beam (Radial Variation, Divergence)

<div align="center">
IQIN, 15, GAMMA, ZFOCUS, RTHETAIN, THETAIN
SPCNAM
</div>

This source is similar to the NRC Swept Beam (`ISOURC = 5`) with the difference that the beam itself has a radially varying intensity distribution and an angular divergence, both specified by the user. Also, the cone swept by the beam has its apex at a user specified Z position, given by (X=0,Y=0,Z=ZFOCUS), rather than the apex always being at (X=0,Y=0,Z=`Z_min_CM(1)`) as it is in source 5. As with most other sources, particles enter from the top of CM 1.

Beam divergence means the beam is diverging or spreading out about the axis of each element of the swept beam. At any given instant each elemental beam can be thought of diverging from a point on the elemental beam's axis. This point is specified by the angle (`THETAIN`) created by the elemental beam's axis and a line from the apex to a point at a specified radius (`RTHETAIN`) in the `ZFOCUS` plane. This angle's only role is to define the distance to the apex.

Input parameters for this source are:

IQIN Charge of the incident beam (defaults to 0 = photons)

GAMMA Half-angle of the cone swept by the beam in degrees ($0^o \leq$ `GAMMA` $< 90^o$)

ZFOCUS Z position of the apex of the swept cone in cm (Note: ZFOCUS can be greater than, equal or less than `Z_min_CM(1)`).

RTHETAIN The beam radius (in cm) at which the beam divergence angle, THETAIN, is defined. `RTHETAIN` must be > 0.

THETAIN The beam divergence angle (in degrees) at RTHETAIN. If `GAMMA` $\neq 0$, then `THETAIN` can be set to 0; otherwise, it must be > 0 (Note, however, that it can still be set to such a small angle that divergence is practically zero).

SPCNAM The name of file containing beam's radial intensity distribution about the axis of the elemental beam in a plane at `Z = ZFOCUS`. The file `SPCNAM` has a specific format:

```
NRDIST
(RDISTF(I),RPDF(I),I=1,NRDIST)
```
where:

NRDIST = # of radial bins in the distribution

RDISTF(I) = upper radius of bin I in cm

RPDF(I) = probability (not necessarily normalized) of finding a particle in radial bin I. These probabilities are multiplied by bin area before sampling.

Some sample radial intensity distribution files are found in `$OMEGA_HOME/beamnrc/radial_source_dis`

The radial position of an incident particle, RIN, is chosen based on the radial intensity distribution. The divergence angle, THETAI, of the particle is then calculated from TAN(THETAI) = RIN/(RTHETAIN/TAN(THETAIN)). Note that the radial distribution is defined at Z = ZFOCUS (the apex of the swept cone) in a plane perpendicular to the Z axis. This is an approximation since, strictly speaking, the distribution should be defined in a plane perpendicular to the beam direction. However, for GAMMA of a few degrees, this approximation is valid.

In addition to simulating beams more realistically, this source can be applied to several useful cases. With GAMMA = 0 and THETAIN ≈0, it can be used to model an arbitrary radial intensity distribution from a parallel beam incident normally on the front face (*eg.*, the distribution could be ring shaped). With GAMMA = 0 one could mimic a point source at a distance from ZFOCUS of RTHETAIN/tan(THETAIN) along with a uniform intensity profile out to the beam radius. One should use source 1 for the uniform case, but could use source 15 to get an arbitrary radial distribution from a point source.

Figure 16 below shows the swept beam source with radial distribution and divergence.
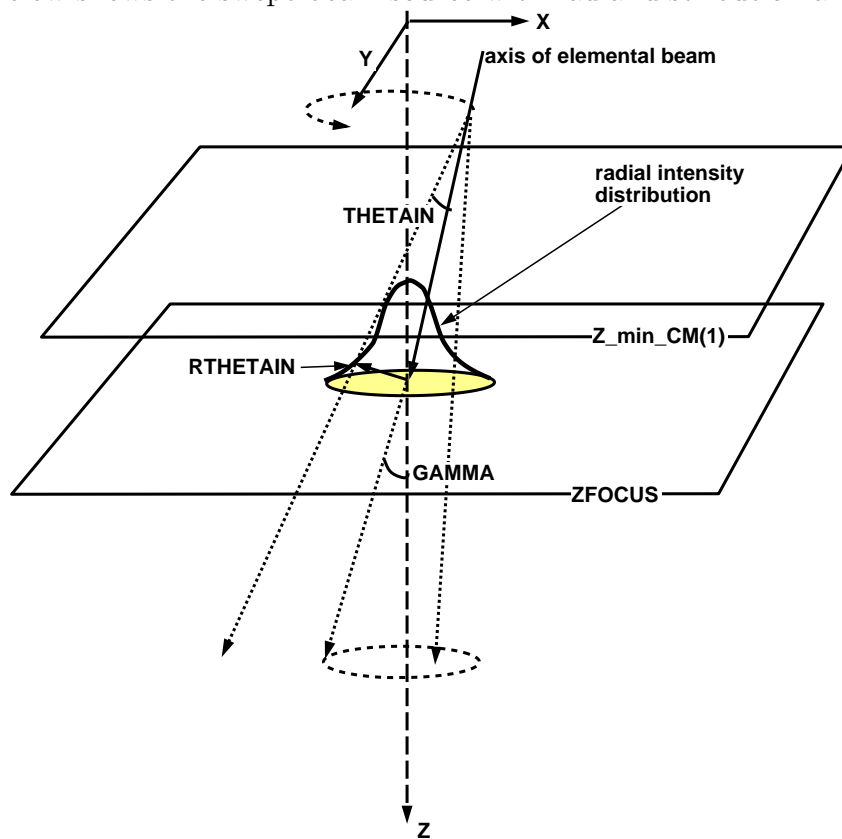
Figure 16: The beam sweeps on the surface of a cone with half-angle GAMMA and with the apex defined at ZFOCUS. The beam itself has a user-defined radial intensity distribution (in the case shown, a gaussian), defined at ZFOCUS and in a plane perpendicular to the Z axis. The beam then diverges from this original distribution as if it originated at an imaginary point back along the axis of each elemental beam. The position of this imaginary point is defined by the divergence angle, THETAIN, and the radius (at ZFOCUS) at which the divergence angle is defined, RTHETAIN.

4.12 ISOURC=19: Elliptical Beam with Gaussian Distributions in X and Y, Parallel or with Angular Spread

This is an elliptical beam where the ellipse is defined by gaussian intensity distributions in X and Y. The beam can either be parallel, with direction cosines specified by the user, or it can have an angular spread from the Z-axis, specified by a mean angular spread.

IQIN The charge of the incident beam (defaults to 0=photons)

RBEAM Standard deviation (σ) in the X-direction (if set > 0) in cm, or -FWHM (FWHM:full-width half-maximum) of the gaussian distribution in the X-direction (if set < 0) in cm. σ is automatically limited to RMAX_CM(1) for circular CM 1 or $\sqrt{2}$RMAX_CM(1) for square CM 1. If the user enters RBEAM=0, the source collapses to a pencil beam.

UINC X-axis direction cosine (defaults to 0)

VINC Y-axis direction cosine (defaults to 0)

WINC Z-axis direction cosine (defaults to 1, *i.e.* parallel to the z-axis)

sigma_src19 the mean angular spread about the Z-axis. If sigma_src19>0, then UINC, VINC, WINC automatically take their default values.

RBEAMY σ (if set > 0) or -FWHM of the gaussian distribution in the Y-direction in cm. If set $= 0$, then RBEAMY is set $=$ RBEAM, resulting in a circular beam with a gaussian radial distribution.

Note that in the case of a circular beam (RBEAMY=RBEAM) the radial distribution of particles is also gaussian with σ equal to $\sqrt{2}$ times the σ of the X and Y gaussian distribution.
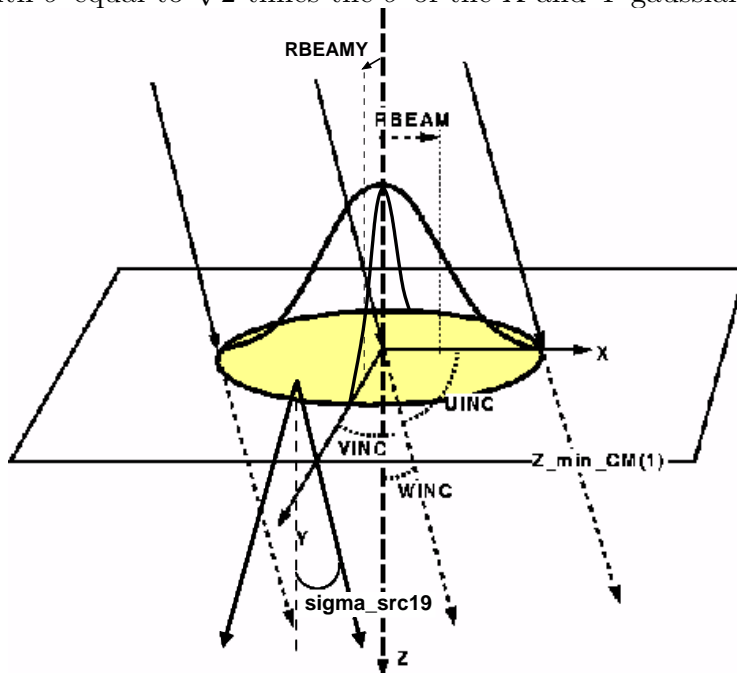
Figure 17: Elliptical Beam with Gaussian Distributions in X and Y (ISOURC=19). The shape of the ellipse is defined by RBEAM and RBEAMY which define the the σ (if set >0) or -FWHM of the gaussian intensity distributions in the X- and Y-directions respectively. The beam either has a mean angular spread about the Z-axis, sigma_src19, or, if sigma_src19≤ 0, it is a parallel beam with incident direction cosines specified by UINC, VINC, WINC.

4.13   ISOURC=21: Phase Space Source

IDUMMY, 21, INIT_ICM, NRCYCL, IPARALLEL, PARNUM, ISRC_DBS, RSRC_DBS,
SSDSRC_DBS, ZSRC_DBS
SPCNAM

This source routine allows a phase space file generated at any scoring plane to be used as a source in its own right. Unlike the sources described thus far, which are assumed to be incident on the first CM, phase space sources can be incident on any CM. They are particularly useful for doing repeated simulations of lower portions of an accelerator model (*i.e.* when the configuration of the lower part of the accelerator is changing, but repeated simulation of the top part is unnecessary), or simulating beams made up of several different particle types. Note that the LATCH values are passed on when using this source routine and it is necessary to number LATCH bits consistently between the two simulations (*i.e.* that generating the input file and the current simulation). Also, dose and fluence resulting from a phase space source are normalized by the number of initial particles in the original, non-phase space source (*i.e.* the source that generated the phase space source) and not by the number of particles incident from the phase space source itself. More about this normalization and how it is accomplished appears in section 7 below.

This source can also be used to simulate any incident beam shapes not covered by the sources described above. This is done by creating the appropriate phase space file for the general source by using a stand alone user-written program and the `phsp_macros.mortran` macros.

Input parameters for a phase space source are:

INIT_ICM Component module on which the phase space source is incident (defaults to 1 if it is set to < 1 or > the number of CMs in the model)

NRCYCL The number of times to recycle each incident particles before moving on to the next particle in the phase space source. Thus, each particle will be used a total of NRCYCL + 1 times before moving on to the next one. If set ≤ 0, then BEAMnrc will automatically calculate a value of NRCYCL based on the number of incident histories and the number of particles in the phase space file that should prevent restarting of the phase space source. If set > 0, the user-input value is used. See below for details.

IPARALLEL No longer necessary if using BEAMnrc's built-in parallel processing functionality for submitting parallel jobs. In previous versions of BEAMnrc, IPARALLEL = the number of parallel jobs. See below for more details.

PARNUM No longer necessary if using BEAMnrc's built-in parallel processing functionality for submitting parallel jobs. In previous versions of BEAMnrc, PARNUM was set to an integer value in the range 1≤PARNUM≤ IPARALLEL, with a different value of PARNUM for each of the parallel jobs. See below for more details.

ISRC_DBS Set to 1 if directional bremsstrahlung splitting (DBS) was used in the BEAMnrc simulation to generate this source **and** you wish to reject photons directed outside the splitting field (which are fat). Set to 0 otherwise.

RSRC_DBS Radius of the DBS splitting field (in cm) used in the BEAMnrc simulation to generate this source. Only needed if `ISRC_DBS` = 1.

SSDSRC_DBS SSD at which `RSRC_DBS` was defined in the BEAMnrc simulation that generated this source (in cm). Only needed if `ISRC_DBS` = 1.

ZSRC_DBS Z value where this phase space source was scored in the BEAMnrc simulation (cm). This will be at the back of a component module (CM). Note the restriction that `ZSRC_DBS` is $\leq$ `SSDSRC_DBS`.

SPCNAM Filename (with extension) of the phase space source. The full directory path to the file must be specified. In the case of an IAEA-format phase space source, you must supply the full name of the file containing the phase space data (*i.e.* the `.IAEAphsp` file). The `.IAEAheader` file will be assumed to be in the same directory. For more information on IAEA-format phase space data, see Section 7.3.

Value of `NRCYCL`

`NRCYCL` is an essential input for phase space sources because phase space data are often sparse, making it necessary to re-use particles in order to obtain adequate statistics. However, `NRCYCL` can only help reduce the inherent uncertainty in the dose calculation. The overall uncertainty will also be governed by the latent variance of the phase space file (see the report on history by history statistics in BEAMnrc and DOSXYZnrc[17]). In addition to recycling, particles are also re-used whenever a phase space source is restarted (which happens automatically when the simulation has used all particles in the source). However, restarting may result in an underestimate of uncertainties [17], making it important to choose a value of `NRCYCL` that both ensures adequate sampling of the phase space source and also prevents restarting.

If you are unsure of the value of `NRCYCL` to use, set `NRCYCL`$\leq$ 0 and BEAMnrc will automatically calculate its value based on the number of histories and the total number of particles in the phase space file. Even with an automatically-calculated value of `NRCYCL`, the phase space source may still restart, either because the algorithm used to calculate `NRCYCL` has determined that setting `NRCYCL`> 0 will cause the phase space source to be undersampled, or because some incident particles were multiple-passers (i.e., had crossed the phase space scoring plane more than once) and were rejected from the simulation. If the phase space source has only been restarted once and only a small fraction of it has been re-used on the second pass, the effect on uncertainty will not be significant. However, if a significant portion of the source has been re-used on the second pass, or if the source has been restarted more than once, we recommend re-running the simulation with a new value of `NRCYCL` given by:

$$\frac{\texttt{NCASE}}{[\texttt{NNPHSP} - \texttt{NNPHSP} * (\texttt{NPASS\_ph\_sp} + \texttt{NFAT\_ph\_sp})/(\texttt{NTOT\_ph\_sp} + \texttt{NPASS\_ph\_sp} + \texttt{NFAT\_ph\_sp})]} - 1$$

where `NCASE` is the number of histories, `NNPHSP` is the total number of particles in the phase space source, `NTOT_ph_sp` is the total number of particles used from the phase space source in the previous simulation (not including recycling), `NPASS_ph_sp` is the number of particles rejected from the previous simulation because they were multiple-passers (not including recycling), and `NFAT_ph_sp` is the number of photons (not including recycling) rejected because

they fall outside the directional bremsstrahlung splitting field radius at the SSD (only if
ISRC_DBS=1–see below for more details). NNPHSP, NTOT_ph_sp, NPASS_ph_sp and NFAT_ph_sp
can be found in the .egslst file from the previous simulation.

Note that, even with NRCYCL> 0, the input variable NCASE (see section 10.7) still controls
the number of histories simulated. The simulation will stop as soon as NCASE histories have
been run, regardless of whether the current particle has been recycled the full NRCYCL times
or not.

Inputs IPARALLEL and PARNUM
Note: the input variables IPARALLEL and PARNUM are no longer necessary if you are using
the built-in parallel processing functionality in BEAMnrc. In previous versions of BEAMnrc,
these inputs were essential for dividing a phase space source into IPARALLEL equal partitions.
Each job used a different partition given by:

$$(\text{PARNUM} - 1) * \left(\frac{\text{NNPHSP}}{\text{IPARALLEL}} \right) < \text{INPHSP} \leq \text{PARNUM} * \left(\frac{\text{NNPHSP}}{\text{IPARALLEL}} \right)$$

where NNPHSP is the total number of particles in the phase space source and INPHSP is the
particle number used. The Unix script, pprocess, used for parallel job submission with
previous versions of BEAMnrc, set the values of IPARALLEL and PARNUM automatically. You
only need to be concerned with these inputs if you are submitting parallel jobs one by one,
manually creating an input file for each of them.

Inputs re DBS
If directional bremsstrahlung splitting (DBS) was used in the BEAM simulation used to
generate this source, then it is recommended that you use the inputs ISRC_DBS, RSRC_DBS,
SSDSRC_DBS and ZSRC_DBS to prevent fat photons from compromising dose or fluence statistics
in the current simulation. RSRC_DBS and SSDSRC_DBS, the radius and SSD of the DBS splitting
field respectively, are available from the DBS inputs for the BEAMnrc simulation used to
generate the phase space source. ZSRC_DBS will be equal to the Z position of the back of the
component module where this source was scored (this information is available in the .egslst
file from the BEAMnrc simulation that generated the source). When ISRC_DBS is set to 1,
photons in the source are projected along their trajectory from ZSRC_DBS to SSDSRC_DBS. If
they fall outside RSRC_DBS at SSDSRC_DBS then they are not used in the simulation. In the
context of the BEAMnrc simulation that generated this source, these photons are fat (but
this information is not stored in the phase space file). Note that

Charged particles are never rejected with this technique, which means that if you do not
want fat charged particles to compromise dose statistics (especially near the surface of a
phantom) then you must use the electron splitting option in DBS.

For more information about DBS, see section 6.3.4 of this manual.

For more information related to phase space sources, see the section 7 on full phase space
files.

4.14 ISOURC=24: Phase Space Source Incident from User-specified Angle

IDUMMY, 24, INIT_ICM, NRCYCL, IPARALLEL, PARNUM, ISRC_DBS, RSRC_DBS,
SSDSRC_DBS, ZSRC_DBS
SPCNAM
ALPHA24, BETA24, DIST24

This source is similar to the phase space source (21) with the exception that the user can specify a point of rotation on the Z-axis above `INIT_ICM` and rotation angles about the X- and Y-axes. Input parameters are the same as `ISOURC=21` with additional inputs:

ALPHA24 Angle of rotation of source (phase-space) plane about an axis parallel to the X-axis (degrees). Positive angle is clockwise rotation. -90 degrees < `ALPHA24` < 90 degrees.

BETA24 Angle of rotation of source plane about an axis parallel to the Y-axis (degrees). Positive angle is counter-clockwise rotation. -90 degrees < `BETA24` < 90 degrees.

DIST24 Distance of point of rotation above `INIT_ICM` on the Z-axis (cm).

Note that, in general, `INIT_ICM` must be > 1, since any rotation of the source will result in particles incident from within `INIT_ICM`-1. Also, the following CMs currently do not handle the case of particles incident from within them: APPLICAT, ARCCHM, CHAMBER, CIRCAPP, CONESTAK, DYNJAWS, JAWS, MESH, PYRAMIDS, SIDETUBE. Thus, if `INIT_ICM` or `INIT_ICM`-1 is one of these, then `negative ustep` errors may result.

The initial idea and much of the coding of `ISOURC=24` is courtesy of Patrick Downes at University of Cardiff, Wales.
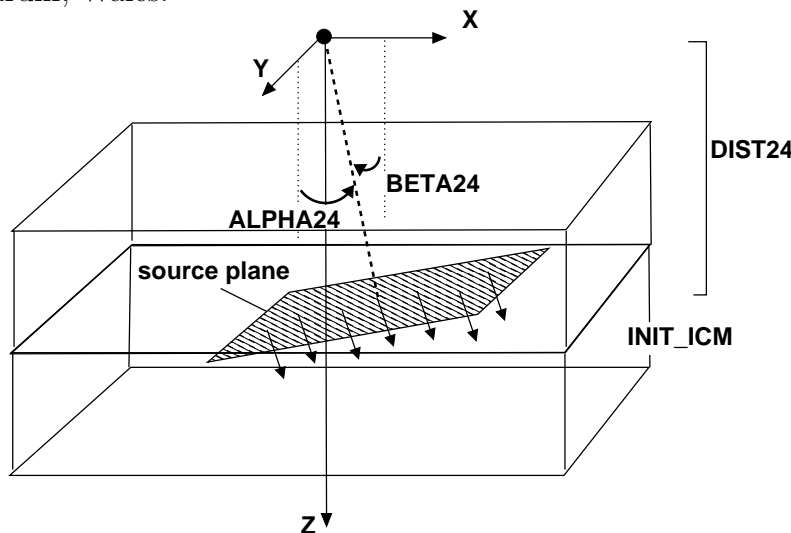
Figure 18: Phase-space source incident from user-specified angle (ISOURC=24). Similar to the standard phase-space source (ISOURC=21) except that the user can specify a point of rotation on the Z-axis above `INIT_ICM`, `DIST24`, and angles of rotation about axes parallel to the X-axis, `ALPHA24`, and Y-axis, `BETA24`.

4.15   ISOURC=23: BEAM Simulation Source Incident from User-specified Angle

<div align="center">
IDUMMY, 23, INIT_ICM, ISRC_DBS, ALPHA24, BETA24, DIST24

the_beam_code, the_pegs_file, the_input_file
</div>

This source allows an accelerator that has been compiled as a shared library to be used as a source in a BEAM simulation. In function, it is similar to the phase-space source incident from a user-specified angle, `ISOURC=24`, but it eliminates the need to store an intermediate phase space file since the BEAM simulation reads phase-space data at the scoring plane in the simulation source directly. The simulation source runs simultaneously with and under the control of the BEAM simulation using it, so primary histories in the simulation source are run each time new phase-space data is required by the BEAM simulation using it. Note that this also means that any need to recycle phase space data is eliminated, another important advantage of the simulation source.

For information about compiling an accelerator as a shared library source, the user is urged to refer to Section 4.10.1 in the DOSXYZnrc Users Manual[15].

Input parameters for source 23 are:

INIT_ICM CM no. at the top of which the source is incident.

ISRC_DBS Set to 1 if directional bremsstrahlung splitting (DBS) is used in the simulation source and you wish to reject fat photons outside the splitting field. This is recommended since these photons can compromise statistics in the downstream BEAM simulation.

ALPHA24 Angle of rotation of the source plane about an axis parallel to the X-axis (degrees). Positive angle is clockwise rotation.

BETA24 Angle of rotation of the source plane about an axis parallel to the Y-axis (degrees). Positive angle is counter-clockwise rotation.

DIST24 Distance of point of rotation on the Z-axis above `INIT_ICM` (cm).

the_beam_code The name of the accelerator to be used as a simulation source. This must include the `BEAM_` prefix (*i.e.* `BEAM_accelname`). The accelerator must have been compiled as a shared library on the architecture that the BEAM simulation is running on.

the_pegs_file The name of the PEGS4 data set used by the simulation source (*i.e.* may be different than that used by the BEAM simulation using the simulation source).

the_input_file The input file for the simulation source. This must exist in the directory `BEAM_accelname`, where `accelname` is the name of the accelerator being used as the simulation source. Also, this input file must specify one phase space scoring plane, and the input parameter `IO_OPT` must be set to 1 to output phase-space data at this scoring plane (see Section 10.4. No phase-space file will be output, but the phase-space data at this scoring plane will be read directly by the BEAM simulation using the source.

Note that `the_input_file` will specify a value for `NCASE` (no. of histories) but this will be ignored since the number of histories run is dictated by the number of incident particles required by the simulation using this source.

The definitions of the parameters specifying the incident direction, `ALPHA24`, `BETA24` and `DIST24`, are the same as those for `ISOURC`=24 and shown in Figure 18. Note that if `ALPHA24` and `BETA24` are not both 0, then `INIT_ICM` must be >1 since some particles will be incident from within `INIT_ICM`-1. Also, if `ALPHA24` and `BETA24` are not both 0, then `INIT_ICM` and `INIT_ICM`-1 cannot be any of APPLICAT, ARCCHM, CHAMBER, CIRCAPP, CONESTAK, DYNJAWS, JAWS, MESH, PYRAMIDS, or SIDETUBE since these CMs currently do not properly handle the case of a particle incident from within them.

Unlike the phase-space sources, `ISOURC`=21 and `ISOURC`=24, the only parameter that needs to be set to reject fat photons from a simulation source using directional bremsstrahlung splitting (DBS) is `ISRC_DBS`=1. This is because information about whether a photon is fat (directed outside the splitting field) is available directly from the simulation source, so there is no need to reconstruct the splitting field and the photon's path relative to it.

In the case of parallel BEAM runs using a simulation source, each parallel job starts its simulation source with different random number seeds, ensuring that incident particles are not identical. This is in contrast to phase-space sources, where each parallel job uses a different partition, or "chunk", of the phase-space data (see Section 13.2). For more information about parallel runs see Section 13.

4.16   ISOURC=31: Phase Space Reconstructed Using Beam Models

<div align="center">IDUMMY, 31, CMSOU, SPCNAM</div>

This source routine reconstructs the phase space parameters using the beam data derived from a beam characterization model. The reconstructed phase space sources can be incident on any CM. Use of beam characterization models can save CPU time for the accelerator head simulation and also result in significant reduction in disk space requirement for phase-space storage. For more information related to beam reconstruction using beam characterization models, see NRCC report "Beam Characterization: a Multiple-Source Model" by Ma and Rogers (1995).

Input parameters for a beam model source are:

CMSOU Component module on which the phase space source is incident (defaults to 1 if it is set to < 1 or $>$ the number of CMs in the model)

SPCNAM Filename (with extension) of the beam model parameters.

By default, this source is not included in the accelerator models. If you want to include them you must do several things.

- Copy the files `beammodel_macros.mortran` and `beammodel_routines.mortran` from `$OMEGA_HOME/progs/beamdp` to the directory with you accelerator.

- Update the `sources.make` file on that same area to include the above two files. If you are using a standard accelerator model, you may just copy `$OMEGA_HOME/beamnrc/sources.make.MS` to your accelerator subdirectory and rename it as `sources.make`.

- `make` the accelerator

5   Monoenergetic vs Energy Spectrum Sources

In any of the above sources, with the exception of the phase space source (ISOURC=21), incident beam energy can be either monoenergetic or described by an energy spectrum. Inputs associated with incident beam energy are:

MONOEN Set to 0 if the beam is monoenergetic (default); set to 1 if the beam energy is described by a spectrum

EIN (required if MONOEN = 0) The kinetic energy of a monoenergetic beam in MeV; defaults to 1.25 MeV if EIN is set ≤ 0

FILNAM (required if MONOEN = 1) The filename (with extension) containing information describing the incident beam's energy spectrum (more on spectrum file format below)

IOUTSP (required if MONOEN = 1) Set to 0 for no summary of the incident beam energy spectrum in the .egslst output file (default); set to 1 if a summary is desired

The energy spectrum file, FILNAM, has a specific format:

```
SPEC_TITLE
NENSRC, ENMIN, IMODE
ENSRCD(I), SRCPDF(I) (I = 1 to NENSRC)
```

where:

SPEC_TITLE is an 80-character spectrum title

NENSRC = # of energy energy bins in the spectrum histogram

ENMIN = lower energy of first bin in MeV

IMODE Set to 0 for histogram counts/bin; set to 1 for counts/MeV

ENSRCD(I) = upper energy of bin I in MeV

SRCPDF(I) = probability of finding a particle in bin I (SRCPDF need not be normalized)

The code randomly distributes incident particle energies equally across each energy bin.

On $HEN_HOUSE/ensrc_spectra there are a collection of spectra developed at NRC over the years in various situations. We intend to augment this collection with time, and will happily add any other spectra sent to us in the correct format, and with some semi-adequate documentation.

6   Variance Reduction in BEAMnrc

There is an extensive discussion of variance reduction in BEAM in the original BEAM paper[3].

6.1   Range Rejection

This section contains a brief description of how range rejection is performed and describes the input variables used to control range rejection.

Range rejection is used to save computing time during simulations. The basic method is to calculate the range of a charged particle and terminate its history (depositing all of its energy at that point) if it cannot leave the current region with energy > ECUTRR. ECUTRR is the range rejection cutoff energy which may vary from region to region depending on the type of range rejection used (see below).   To determine the range to ECUTRR, BEAMnrc calculates the range from ECUTRR to AE (using EGSnrc macros and EGSnrc-calculated tables of range to AE as a function of electron energy) for each region at the beginning of the simulation (zero if ECUTRR = AE). Then, before each charged particle step during the simulation, this value is subtracted from the particle's range to AE (which is always calculated by EGSnrc). Ranges are calculated using restricted stopping powers and, thus, represent the longest possible ranges to ECUTRR.

When the range rejection control variable IREJCT_GLOBAL is set to 2, range rejection is performed on a region-by-region basis. In this case ECUTRR in each region is simply equal to the value of ECUT in the region. If the range to ECUTRR is less than the perpendicular distance to the nearest region boundary, the history is terminated and energy is deposited in the current region. On the other hand, when IREJCT_GLOBAL is set to 1, ECUTRR is the minimum energy that a charged charged particle can have as it leaves the current region and still reach the bottom of the accelerator with an energy greater than the global ECUT. ECUTRR is automatically calculated for each region at the beginning of the simulation. Similar to IREJCT_GLOBAL=2, if the range to ECUTRR is less than the perpendicular distance to the nearest region boundary, the particle is terminated and energy deposited in the current region. IREJCT_GLOBAL=1 can save more time than IREJCT_GLOBAL=2 but can only be used if there is only 1 scoring plane and it is at the very bottom of the accelerator. One can approximate IREJCT_GLOBAL=1 for other situations by using IREJCT_GLOBAL=2 and carefully selecting ECUT for different regions throughout the accelerator.

Range rejection introduces an approximation because, in terminating a charged particle's history and depositing all of its energy in the current region, it is assumed that any bremsstrahlung photons that would have been created by the particle, do not leave the current region. The user can minimise inaccuracies resulting from this approximation using the input variable ESAVE_GLOBAL defining the maximum charged particle energy (in MeV) at which range rejection is considered. The choice of ESAVE_GLOBAL depends on the incident beam energy and the materials that it is passing through. ESAVE is treated internally on a region by region basis, but only in the CM SLABS does the user currently have the ability to assign individual values to each region (via ESAVEIN, this is because this CM is used for bremsstrahlung targets and we thought we might need more control).

The actual rejection of particles based on range to `ECUTRR` is performed in a BEAMnrc macro and does not make use of the internal range rejection macro available in EGSnrc. This is because the EGSnrc macro only performs range rejection based on range to `AE`.

6.2   Photon Forcing

BEAMnrc offers an option whereby the user can force photons to interact in specified CMs within a simulation. This option is useful for improving statistics of scattered photons when photon interactions are sparse (*eg.* in thin slabs of material or in material with low density). One of the main purposes of implementing this option was to study the generation of contaminant electrons in a photon beam.

Briefly, a photon forced to interact in a CM is "split" into a scattered photon whose weight is equal to the probability of interaction and an unscattered photon carrying the remaining weight. The unscattered photon proceeds as if an interaction did not take place, and it cannot be forced to interact any more within the specified forcing zone, which can consist of one or several component modules. However, once the unscattered photon gets out of the forcing zone, it may interact again depending on the sampled pathlength. The scattered photon can be forced again in the forcing zone depending on how many interactions are allowed to be forced.

The input variables used to control photon forcing are:

IFORCE Set to 0 for no forcing (the default); set to 1 for photon forcing

NFMIN Photon interaction # at which to begin forcing (defaults to 1). Currently, the option to start photon forcing at any interaction other than the first one in the forcing CMs has been disabled. Thus, `NFMIN` is effectively 1 regardless of what is input.

NFMAX Photon interaction # after which to stop forcing (defaults to 1)

NFCMIN CM # at which to start forcing (defaults to 1)

NFCMAX CM # beyond which to stop forcing (defaults to the # of `CMs` in the accelerator)

Photon forcing parameters are passed onto secondary photons, so that if the parent particle has not yet been forced to interact `NFMAX` times, each secondary photon is forced to interact the remaining number of times (*i.e.* `NFMAX` - # of times parent particle forced) as long as it is within the forcing CMs. Forcing of secondary photons does not affect the number of times the parent particle is forced to interact if it is a photon as well. The feature of passing forcing parameters to secondary photons is particularly useful to get good statistics for bremsstrahlung photon interactions. The incident electron creating the photons will not be forced at all, so each bremsstrahlung photon will be forced to interact `NFMAX` times. This feature makes Photon Forcing a powerful tool for improving statistics when used in conjunction with bremsstrahlung splitting (see next section).

Note that forcing in restricted regions with low mass can lead to a few electrons created outside the forcing region having much larger weights than those created inside the forcing region and thereby distorting results which are sensitive to these weight variations.

6.3   Bremsstrahlung Photon Splitting and Russian Roulette

Bremsstrahlung photon splitting offers the user another variance reduction technique which improves the statistics of bremsstrahlung photons resulting from electron interactions. The technique has been described for general use in EGS in references [24, 25]. BEAMnrc offers three bremsstrahlung splitting techniques, uniform bremsstrahlung splitting (UBS), selective bremsstrahlung splitting (SBS) and directional bremsstrahlung splitting (DBS). Both UBS and SBS have been optimised in BEAMnrc with the addition of the Russian Roulette feature (not required for DBS).

6.3.1   Uniform Bremsstrahlung Splitting

Input variables associated with uniform bremsstrahlung splitting (UBS) are:

IBRSPL Set to 1 for uniform bremsstrahlung splitting

NBRSPL The splitting number. Also applied to higher-order
       bremsstrahlung and annihilation photons if Russian Roulette is on (see section 6.3.3
       below).

Each bremsstrahlung event produces NBRSPL photons, each having a weight equal to $\frac{1}{\text{NBRSPL}}$ times the weight of the electron that underwent the bremsstrahlung event. The energies and directions of each photon are sampled individually according to the relevant probability distributions. The energy of the primary electron is decremented by the energy of just one of the photons. This must be done in order to preserve the effects on energy straggling but it does mean that energy is not conserved on a given history (the energy would have to be decremented by the average energy of the photons created) but it is conserved "on average" over many histories.

The splitting number, NBRSPL, is not applied to higher-order higher-order bremsstrahlung and annihilation photons unless Russian Roulette is turned on (see section 6.3.3 below). This prevents wasting simulation time by tracking many higher-order photons of vanishing weight.

Uniform bremsstrahlung splitting in BEAMnrc is now handled by the EGSnrc system using a more efficient internal bremsstrahlung splitting feature. So, other than input, there is no longer any coding in BEAMnrc related to this except for the feature which turns it off for higher orders.

6.3.2   Selective Bremsstrahlung Splitting

Selective bremsstrahlung splitting (SBS) is more efficient than UBS (by a factor of 3-4), but it has been superseded by the recently-implemented directional bremsstrahlung splitting (DBS) technique (see section 6.3.4 below), which offers an even greater improvement in efficiency. For historic and compatibility purposes, SBS is still enabled in BEAMnrc, and a short description of it is given here.

Input variables associated with SBS are:

`IBRSPL` Set to 29 for selective bremsstrahlung splitting

`NBRSPL` Maximum splitting number We have found values between 200 and 1000 to be appropriate[18] but note that you must adjust `$MXSTACK` to avoid overflows.

`NMIN` Minimum splitting number. Also, the splitting number to be used for higher-order bremsstrahlung and annihilation photons if Russian Roulette is on (see section 6.3.3 below). This should be set to $0.1\times$`NBRSPL` to obtain the optimum gain in efficiency.

`FS` Bigger side of the treatment field plus 10 cm. E.g., 14 for a 4×4 or 50 for a 40×10 field. In conjunction with `SSD`, this is used to define the relevant angle for the selective bremsstrahlung splitting routine. To be exact, the radius giving the same area as `FS`^{2} is used.

`SSD` The distance from the front of the bremsstrahlung target at which the above `FS` is defined. The approximation is made that all bremsstrahlung is created on the front surface.

SBS uses a variable splitting number, `NBR`, which reflects the probability that a brem photon will enter the field defined by `FS` and `SSD`. This probability is calculated using the energy/direction of the incident electron about to undergo bremsstrahlung. `NBR` is subject to the restriction `NMIN` $\leq$ `NBR` $\leq$ `NBRSPL`. Split photons are given a weight equal to $\frac{1}{NBR}$ times the weight of the electron that underwent the bremsstrahlung event.

In SBS, higher-order bremsstrahlung and annihilation photons are split with a fixed splitting number defined by `NMIN` provided that Russian Roulette is on (see section 6.3.3 below). Otherwise, higher-order bremsstrahlung and annihilation photons are not split. This prevents long simulation times tracking higher-order photons of diminishing weight.

Macros required to set `NBR`, initialize the selective bremsstrahlung splitting functions, input `NMIN`, `FS` and `SSD` and define variables required for selective bremsstrahlung splitting are contained in the file
`$OMEGA_HOME/beamnrc/sbsnrc_macros.mortran`.

When using either UBS or SBS, you must ensure that the size of the stack in BEAMnrc (*i.e.* `$MXSTACK`) is large enough to accommodate the split photons and their higher-order products. See section 2.11 on page 28 for more on `$MXSTACK`.

6.3.3   Charged Particle Russian Roulette

Note that directional bremsstrahlung splitting (DBS), described in section 6.3.4 below does not make use of this charged particle Russian Roulette input.

When using UBS or SBS, following all of the secondary charged particles created by the "split" photons increases the CPU time required for simulations. If the primary interest is in secondary electrons or their effects (*eg.* dose deposition), the extra computing time is obviously acceptable. But if, as is often the case, the main interest is in the bremsstrahlung photons themselves, one can reduce the CPU time while still preserving the variance reduction advantages of bremsstrahlung splitting by using a Russian Roulette technique with any charged particles generated by the split photons.

The possible settings of the Russian Roulette input, `IRRLTT`, are:

= 0 Russian Roulette off. No splitting of higher-order bremsstrahlung annihilation photons.

= 1 Option discontinued. This defaults to `IRRLTT`=2 (see below).

= 2 Russian Roulette on. Higher-order bremsstrahlung and annihilation photons are split with splitting number `NBRSPL` in UBS and `NMIN` in SBS.

Russian Roulette is implemented by giving secondary charged particles resulting from split photons a survival threshold. The survival threshold is always the inverse of the photon splitting number. Thus, in the case of UBS, the threshold is fixed and is equal to $\frac{1}{NBRSPL}$, while in the case of SBS, the survival threshold is $\frac{1}{NBR}$, where `NBR` is the variable splitting number. Then a random number is chosen for each charged particle. If the random number is less than the survival threshold, the charged particle survives, and its weight is increased by a factor of `NBRSPL` (for UBS) or `NBR` (for SBS). Otherwise, the charged particle is eliminated. Secondary charged particles subject to Russian Roulette are electrons resulting from Compton events and photoelectric events and electrons and positrons resulting from pair production.

Note that if Russian Roulette is turned on, then higher-order bremsstrahlung and annihilation photons are also split. This is because any charged particle surviving Russian Roulette has a weight higher than the photon that created it. If radiative products from this surviving charged particle are not split, then their high weight may interfere with the statistics of the original split bremsstrahlung photons. Also, splitting of higher-order bremsstrahlung and annihilation photons does not greatly increase computing time when Russian Roulette is on because most of the secondary charged particles have been eliminated.

For a more complete description of Uniform Bremsstrahlung Splitting and Russian Roulette, see the BEAM paper[3].

6.3.4 Directional Bremsstrahlung Splitting (DBS)

DBS inputs
Directional bremsstrahlung splitting was introduced into BEAMnrc in 2004 and results in even greater efficiency than SBS[18]. Our results have shown that using DBS in a photon beam can result in fluence efficiencies up to 8 times higher than with SBS (up to 20 times higher than with UBS) and dose efficiencies up 6 times higher than with SBS (up to 26 times higher than with UBS). The actual improvements will depend upon the energy and other details of the photon beam being simulated.

DBS began from the same philosophy as SBS, in which bremsstrahlung photons aimed into a field of interest (encompassing the treatment field) are split at the time of creation, while those aimed away from the field are not. Beyond that, however, the two algorithms are completely different.

The input parameters for DBS are:

IBRSPL Set to 2 for directional bremsstrahlung splitting

NBRSPL Splitting number. Set to $\approx$1000 if you are using charged particle splitting (see below) or $\approx$5000 if not. Note that $MXSTACK in the file beamnrc_user_macros must be increased to avoid overflows and to be safe, should probably be an order of magnitude larger than NBRSPL.

FS The radius of the DBS splitting field in cm. This must at least encompass the entire treatment field and should go sufficiently beyond the edges of the treatment field that the contribution of fat photons from outside the splitting field (see below for more details) to dose below SSD is negligible. For a 10×10 cm2 field, we recommend FS=10 cm.

SSD The Z value (in cm) at which the above FS is defined.

ICM_DBS The component module (CM) number in which electron (charged particle) splitting is to take place. Set this equal to the CM number modeling the flattening filter in your accelerator. Note that electron splitting is only enabled in the FLATFILT CM, so a FLATFILT must be used to model the flattening filter. If ICM_DBS is set to 0, then electron splitting is turned off.

ZPLANE_DBS The plane number in CM no. ICM_DBS at which electron splitting will take place. Planes correspond to layer boundaries within the CM. Set this equal to the back surface of the flattening filter (i.e., ZPLANE_DBS=no. of layers in flattening filter + 1).

IRAD_DBS Set to 1 to redistribute the NBRSPL split charged particles in a radially-symmetric manner about the Z axis. This can improve charged particle statistics if your beam is radially symmetric.

ZRR_DBS The Z position of the Russian Roulette plane used in conjunction with charged particle splitting (see below for more details). Place this several mm above the splitting plane, still within the flattening filter.

Use a rejection plane [USE_REJPLN] Options are ON or OFF. If Use a rejection plane= On then define a *rejection plane* which discards fat photons and fat charged particles if they interact below the rejection plane. This prevents correlated split photons from being created close to the scoring plane and degrading the statistics of the scored quantities. Our Monte Carlo studies show that eliminating these fat particles has a negligible effect on fluence scoring[26] and dose scoring[18] for typical field-sizes of interest. The rejection plane is meant to be placed in the air layer between the bottom of the last component in the accelerator geometry and the scoring plane. This input must appear between the delimiters :Start DBS rejection plane: and :Stop DBS rejection plane: at the end of the input file. See below for more details.

Z(cm) from zero reference plane [Z_REJPLN] The distance, in cm, between the rejection plane and the plane where Z=0. If the scoring plane is at an SSD = 100 cm, then a typical Z_REJPLN could be between 50 and 90 cm. This input must appear between the delimiters :Start DBS rejection plane: and :Stop DBS rejection plane: at the end of the input file. See below for more details.

Both the USE_REJPLN and Z_REJPLN differ from other DBS-related inputs in that they must appear between delimiters :Start DBS rejection plane: and :Stop DBS rejection plane: in the .egsinp file. For example, the following input defines a rejection plane at Z=22.5 cm:

```
:Start DBS rejection plane:

 Use a rejection plane= On
 Z(cm) from zero reference plane= 22.2

:Stop DBS rejection plane:
```

This section must appear in the .egsinp file somewhere below the input for the last component module. Note that the BEAM GUI automatically reads/writes the rejection plane inputs in the correct format, so in practice the user does not need to concern themselves with it.

Outline of the DBS algorithm

If a charged particle undergoes a bremsstrahlung or annihilation event, then DBS splits this event NBRSPL times. The resultant photons all have their weight multiplied by $NBRSPL^{-1}$. DBS then loops through these split photons and for each one, determines whether or not it is aimed into the splitting field defined by FS and SSD. If it is, then the photon is kept and is considered "non-fat" (low-weight). If not, then Russian Roulette is played on the photon by comparing a random number to a survival threshold of $NBRSPL^{-1}$. If the random number is less than this number, then the photon is kept and its weight is multiplied by NBRSPL and it is considered a "fat" (high-weight) photon. Splitting is not limited to bremsstrahlung and annihilation events, however. If one of these fat photons undergoes a Compton event, then it is also split NBRSPL times and the same Russian Roulette scheme as above is applied to photons not aimed into the splitting field. Moreover, the basic DBS algorithm eliminates all but a few charged particles by:

1. playing Russian Roulette (survival probability $NBRSPL^{-1}$) with all electrons resulting from a split Compton event

2. requiring "non-fat" photons (ie low-weight photons that are actually aimed into the splitting field) to survive Russian Roulette (survival probability $NBRSPL^{-1}$) before they can undergo pair production, photoelectric or Compton events. This condition is relaxed if the event is about to happen in a gas ($\rho < 1.2 \times 10^{-2}$ g/cm^3) to prevent fat photons (ie survivors of Russian Roulette) from being generated in the air just above the splitting field and then Compton scattering into the field, thus compromising the photon statistics.

It can be seen that, other than non-fat charged particles generated in the air just above the splitting field due to the exception in (2) above, those few charged particles that do survive will have a high weight (ie will be fat).

DBS results in many non-fat photons inside the splitting field and few fat photons outside the splitting field. All of the non-fat photons inside the splitting field will have the same low

weight ($\texttt{NBRSPL}^{-1}$ times the weight of the incident particles). This is desirable since a large variation of weights inside the field was one factor compromising the efficiency of SBS.

DBS as described thus far is very efficient for photon fluence/dose, however it will result in only a few fat charged particles reaching the SSD. If you are interested in the charged particle contribution to dose (as in most realistic cases), you must use the electron (really, charged particle) splitting function to "recover" the charged particles.

The inputs for electron splitting are `ICM_DBS`, `ZPLANE_DBS`, `IRAD_DBS` and `ZRR_DBS`. `ICM_DBS` and `ZPLANE_DBS` define a CM number and a plane number (which will correspond to a layer boundary) within that CM at which all fat charged particles are to be split `NBRSPL` times (with their weights multiplied by $\texttt{NBRSPL}^{-1}$). If you have set `IRAD_DBS`=1, then these split charged particles will be redistributed in a radially-symmetric manner about the beam axis. Radial redistribution can result in better statistics as long as the beam has radial symmetry above the electron splitting plane. `ZRR_DBS` defines a "Russian Roulette plane" which is always above the electron splitting plane. Below this Russian Roulette plane, DBS is carried out in the following manner:

1. electrons resulting from a split Compton event are not subject to Russian Roulette

2. non-fat photons undergo pair production, photoelectric and Compton events

3. If a fat photon (i.e., one not aimed into the splitting field) undergoes a pair production or photoelectric event, then the event is split `NBRSPL` times to create `NBRSPL` (photo-electric) or 2*`NBRSPL` (pair production) non-fat charged particles.

Together, use of the splitting and Russian Roulette planes ensures that all charged particles reaching the SSD are non-fat (and that there are many of them).

DBS parameter selection

We have found that, although optimal settings of the DBS splitting parameters depend on the details of the accelerator being simulated, some generalizations can be made. `NBRSPL`, the splitting number, should be set $\approx$1000 for peak efficiency if you are also using electron splitting. If electron splitting is off, then peak efficiency occurs at higher splitting numbers ($\approx$5000). The splitting field radius `FS`, must at least enclose the entire beam field. However, it should also go sufficiently beyond the edges of the beam field that the dose contribution below the SSD from fat photons (from outside the splitting field) is negligible (more about this below). The efficiency of DBS decreases with increasing `FS`, so it is desirable to select the smallest `FS` possible that still provides adequate coverage. We have found that for a 10×10 cm^2 field, a splitting field radius of 10 cm goes sufficiently beyond the edges without a significant sacrifice in efficiency. If you are in doubt about how far beyond the edges of the beam field the splitting field should go, it is okay to overestimate by up to 5 cm, and the gain in efficiency will still be significantly greater than UBS or SBS.

If you are using electron (charged particle) splitting, then you must select the locations of the splitting and Russian Roulette planes. We have found that the optimum location for the splitting plane is at the very back (ie downstream surface) of the flattening filter, with the Russian Roulette plane slightly above this. Because of the way electron splitting is coded, the splitting plane is specified by a CM number, `ICM_DBS`, and then a plane number, `ZPLANE_DBS`, within that CM. Thus, `ICM_DBS` should correspond to the CM modeling the

beam flattening filter. Currently, splitting is only enabled in the FLATFILT CM, so this means that your flattening filter must be modeled with FLATFILT. If you are using the BEAM GUI, input of ICM_DBS is made easier by only presenting the user with a list of the FLATFILT CMs in the accelerator. Planes within a CM correspond to layer boundaries in the geometry, so to place the splitting plane at the very back of the flattening filter, set ZPLANE_DBS=(no. of layers in flattening filter + 1). Again, GUI input makes selection of ZPLANE_DBS easier by showing the user a list of possible planes (along with their Z positions) within CM number ICM_DBS.

Finally, the user must select ZRR_DBS, the Z position (in cm) of the Russian Roulette plane. ZRR_DBS should be several millimeters above the splitting plane, but its exact value is not critical, since, with the eception of a sharp drop in efficiency when the ZRR_DBS is < 1 mm above the splitting plane, the overall variation in efficiency with distance between the Russian Roulette and splitting planes is small. In a simulated 6 MV beam (10×10 cm^2 field) from an SL25 accelerator, Russian Roulette was optimized with the splitting plane at Z=15.66 cm (the back of the flattening filter), and the ZRR_DBS=15.5 cm.

Augmented range rejection with DBS

When using DBS with electron splitting and charged particle range rejection in the simulation, the user has the option to use an augmented range rejection scheme which is more efficient. If you are using the BEAMnrc GUI, this is selected by clicking on the "Augmented range rejection" checkbox in the DBS input frame. If you are editing the .egsinp file directly, then this option is selected by changing the IREJCT_GLOBAL input (see Section 6.1) to -IREJCT_GLOBAL (*e.g.* if you had originally set IREJCT_GLOBAL=2, for region-by-region range rejection, then set IREJCT_GLOBAL=-2 for augmented region-by-region range rejection with DBS). Augmented range rejection is identical to the standard range rejection selected (see Section 6.1) with the exception that all non-fat charged particles (which will exist only if electron splitting is being used with DBS) that cannot make it to the nearest region boundary with energy > ECUT are subject to Russian Roulette (survival probability=1/NBRSPL) regardless of whether or not their energy is greater than the range rejection cutoff energy (ESAVE_GLOBAL). Recall that in the standard schemes, range rejection is not considered if the particle energy is > ESAVE_GLOBAL. The augmented scheme does not influence bremsstrahlung production, since particles that survive the Russian Roulette (with their weight increased by a factor of NBRSPL) still have a chance to undergo bremsstrahlung events. Preliminary studies in a simulated 6 MV photon beam have shown that augmented range rejection can reduce the CPU time by ∼20%. Note that if you have edited the .egsinp file and set IREJCT_GLOBAL<0 but are not using DBS, then standard range rejection corresponding to IREJCT_GLOBAL=|IREJCT_GLOBAL| is done.

When DBS is used in BEAM, an additional dose component (see Section 9) is output to the .egslst and .egsplot files which is the total dose minus the dose due to fat particles (or "total-fat"). The total-fat dose is included because the statistics of the total dose may be compromised by a few fat particles which do not significantly contribute to the dose. However, if the total dose is significantly higher than the total-fat dose over many dose zones, then the contribution from fat particles is significant, and you should consider re-running your simulation with the number of fat particles reduced. Fat charged particles are eliminated by turning charged particle splitting on, and the number of fat photons is reduced

by using a larger splitting field radius, `FS`.

DBS gives rise to additional considerations when it is used in a BEAM simulation to generate a phase space file that is then used as a source in a second, downstream simulation (using BEAM, DOSXYZ, or another EGSnrc user code). Since the fat or non-fat status of a particle is not explicitly stored in the phase space file, we cannot use this information to exclude dose (or other scored quantity such as fluence) from fat particles in the downstream simulation. To overcome this, phase space sources in BEAM (see section 4.13) and DOSXYZ (see the DOSXYZnrc User's Manual[15]) have additional inputs which allow the user to reject photons that are not aimed into the DBS splitting field at the SSD (ie fat photons). Note that there is no such option to reject charged particles, so you must have used electron splitting to generate the phase space source if you are concerned about fat charged particles compromising statistics.

For a full discussion of the DBS algorithm and optimization of splitting parameters, we urge you to read the recent paper submitted to Medical Physics[18]. In addition to the DBS algorithm outlined above, this paper describes the various "smart" subroutines that DBS uses to eliminate the CPU-intensive task of generating and playing Russian Roulette with split photons aimed outside of the splitting field (and these routines are very important to the efficiency gains).

6.4   Bremsstrahlung Cross Section Enhancement (BCSE)

The Bremsstrahlung Cross Section Enhancement (BCSE) variance reduction technique is introduced into BEAMnrc in the Fall of 2007 to improve the efficiency of simulations that involve x-ray production from bremsstrahlung targets (e.g. x-ray tubes and clinical linear accelerators), both in the kilovoltage and megavoltage range. The technique is discussed in detail in reference[26]. This section of the manual contains a brief description of the technique and the input variables used to control it.

BCSE is most valuable in low-energy applications where bremsstrahlung production is rarest, and in 4π geometries where DBS is not applicable. Except for the few restrictions listed in section 6.4.3 below, BCSE is compatible with all other variance reduction techniques already used in BEAMnrc (e.g. range rejection, UBS and DBS). BCSE can be used alone or in conjunction with UBS or DBS; however, the largest efficiency gains are achieved when BCSE is optimally combined with UBS or DBS. For typical simulations of interest in medical physics, the efficiency gains can be up to five orders of magnitude over those obtained with analog simulations, and up to a full order of magnitude over those obtained with optimized splitting alone.

6.4.1   BCSE inputs

The input paramters associated with BCSE are currently passed to the code using delimiters in the input file (the same way the EGSnrc `MC transport parameters` are entered). The following is a template the user should copy to the end of the input file to be able to use BCSE (make sure there is no space between the equal sign and the text preceding it).

`:Start BCSE:`

`Use BCSE= ON` (could be `OFF`, internal variable `USE_BCSE` is 0 or 1 for `OFF` or `ON`, respectively).

`Medium to enhance= med` (`med` is the name of the medium in which to enhance the bremsstrahlung cross section (internal variable `BCSE_MEDNAME`). The medium must exist in the accelerator, otherwise BCSE will not be done.

`Enhancement factor=` n_2 (n_2 is an integer number representing the factor by which the bremsstrahlung cross section is enhanced (internal variable `BCSE_FACTOR`))

`Russian Roulette= ON` (could be `OFF`. If the focus of the simulation is on the photons, it is strongly recommended to set Russian Roulette `ON`. (internal variable `IRRLTT` which is 0 if `OFF` and 2 if `ON`.))

`:Stop BCSE:`

6.4.2   Simulation optimization with BCSE

For maximum efficiency gains, BCSE should be used with either UBS or DBS. BCSE+UBS is suitable for 4π-geometry simulations (e.g. use of miniature x-ray sources in brachytherapy) whereas BCSE+DBS is suitable for directional geometry simulations (kilovoltage x-ray systems and megavoltage clinical linear accelerators). When BCSE is used with either UBS or DBS, the following two steps can be used to optimize the simulation for production runs.

Step 1: The user chooses an optimum `BCSE_FACTOR` depending on the simulation type. Recommended values of `BCSE_FACTOR` are listed in table 1. Getting the maximum efficiency gain is not very sensitive to the exact choice of `BCSE_FACTOR` because the complementary `NBRSPL` of the splitting technique should pick up any little difference and get the efficiency gain very close to its maximum. For justification of this argument, see reference[26].

Table 1: Recommended bremsstrahlung cross section enhancement factor (`BCSE_FACTOR`).

simulation type	incident electron energy range	recommended `BCSE_FACTOR`
4π geometry (brachytherapy)	kilovoltage range	$\sim$500
x-ray tubes	mammography range	$\sim$500
x-ray tubes	diagnostic range	$\sim$200
x-ray tubes	orthovoltage range	$\sim$100
clinical linear accelerators	megavoltage range	$\sim$20

Step 2: The user determines the complementary `NBRSPL` (this applies to both UBS and DBS) by applying Kawrakow's model from reference[27] as follows:

• Perform a few short runs with `BCSE_FACTOR` from step 1, each with a different `NBRSPL` value.

- Calculate the efficiency of each run, $\varepsilon_{N_i} = \frac{1}{T_i s_i^2}$, where N_i is the splitting number used for run i, T_i is the simulation CPU time for run i, and s_i^2 is the average statistical variance on the scored quantity of interest for run i.

- Fit N_i/ε_{N_i} versus $(N_i - 1)$ to the following quadratic equation:

$$\frac{N_i}{\varepsilon_{N_i}} \; = \; A_0 + A_1(N_i - 1) + A_2(N_i - 1)^2 \qquad (1)$$

where $A_i, i = 0, 1, 2$ are polynomial coefficients.

- Calculate the optimum splitting number NBRSPL using NBRSPL $= \sqrt{A_0/A_2}$.

Production runs then use BCSE_FACTOR from step 1, and NBRSPL from step 2 for maximum simulation efficiency.

To use BCSE alone (i.e. without UBS or DBS), table 1 should not be used. Instead, the user should perform a few short runs with different values of BCSE_FACTOR (typically much larger than those in table 1), then follow the remainder of step 2 above with BCSE_FACTOR replacing N_i.

6.4.3   Restrictions

The following are restrictions and cautionary remarks that should be considered when using BCSE.

- The user can enhance the bremsstrahlung cross section in one medium only (which may exist in single or multiple geometric regions). This is not an intrinsic restriction, but a restriction due to present coding.

- If more than one geometric region is made of the same medium (e.g. target and jaws made of tungsten), and the user wants to use BCSE for one or more of these regions (e.g. target only), the user needs to: (1) Duplicate the data of the particular medium of interest in the PEGS4 data file (copy and paste), (2) Assign the duplicate data set a different medium name, and (3) Use one medium name for the geometric region(s) where BCSE will be used and the other material name for the all other geometric regions where BCSE will not be used.

- When BCSE is used with UBS, Russian Roulette cannot be ON for one technique and OFF for the other. It must be either ON for both or OFF for both.

- BCSE cannot be used if the electron beam incident on the bremsstrahlung target has variable electron weights.

- BCSE cannot, at present, be combined with Directional Source Biasing (DSB).

- BCSE cannot be combined with the obsolete Selective Bremsstrahlung Splitting (SBS).

6.4.4   Outline of the BCSE algorithm

When BCSE is used *without* UBS or DBS, BEAMnrc implements the following algorithm[26]:

1. The bremsstrahlung cross section in the medium of interest, `BCSE_MEDNAME`, which is typically the bremsstrahlung target in an x-ray tube or a clinical linear accelerator, is scaled up by a user-supplied factor `BCSE_FACTOR`.

2. The weight of the resulting bremsstrahlung photons in the enhanced medium is reduced by a factor `1/BCSE_FACTOR`.

3. With a probability of `1/BCSE_FACTOR`, the energy of the charged particle is decremented by the amount given to the bremsstrahlung photon.

4. Higher generations of charged particles are created through photoelectric, Compton and pair production interactions of first-generation low-weight photons. If the user turns Russian Roulette `ON`, these higher-order charged particles are eliminated throughout the full geometry with a survival probability `1/BCSE_FACTOR`. The weight of the surviving charged particles is increased by a factor `BCSE_FACTOR`. If Russian Roulette is `OFF`, higher-order charged particles are tracked individually.

5. Fat photons would be created through relaxation events after electron impact ionization (both in the enhanced medium and elsewhere), through bremsstrahlung events outside the enhanced medium, and through positron annihilation events. In each of these events, photons are split, according to the UBS and DBS algorithms, into `BCSE_FACTOR` photons, each with a reduced weight of `1/BCSE_FACTOR`.

When BCSE is used *with* UBS or DBS, BEAMnrc implements an algorithm similar to the one above, except that the splitting number is made equal to `NBRSPL` only when a bremsstrahlung event is about to take place in the enhanced medium, and reset to (`BCSE_FACTOR` · `NBRSPL`) for all other aspects of the BCSE and splitting algorithms. See reference[26] for more on the implementation details.

7   Phase Space Files

This section describes the phase space files output by BEAMnrc and the utilities available for processing them.

7.1   Description of Phase Space Files

A phase space file contains data relating to particle position, direction, charge, *etc.* for every particle crossing a scoring plane. Phase space files can be output for each scoring plane in an accelerator. The input parameter `IO_OPT` controls whether or not phase space files are created. For more information on `IO_OPT` see section 10.4 dealing with this variable below.

   The phase space files are binary files and thus suffer from the problem of being being one of two types which depend on which machine they were written on (little-endians and big-endians). The utility program `readphsp` can be used to convert between these formats. Files from DEC alpha and PC Linux machines have the same byte order as each other and may be interchanged. Files from SUNs, SGIs, RS6000s, and HP9000s are the same and can be interchanged. We have also slightly compressed the files to save space.

The first record in a file is different from the others and contains the following information.

MODE_RW The file mode: it can be either 'MODE0' or 'MODE2' depending on whether `ZLAST` is included in the phase-space parameters.

NPPHSP The total number of particles in the file.

NPHOTPHSP The total number of photons in the file.

EKMAXPHSP The maximum kinetic energy of the particles stored in the file.

EKMINPHSPE The minimum electron kinetic energy (MeV).

NINCPHSP The number of particles incident from the original (non-phase space) source used to generate the phase space file.

Where a phase space file is generated by a simulation using a phase space file as the source (`ISOURC=21`), in the output phase space file, `NINCPHSP`, the equivalent number of particles incident from the original non-phase space source, is equal to

$$\text{NINCPHSP} = \frac{\text{IHSTRY} + (\text{NRCYCL} + 1)\,(\text{NPASS\_ph\_sp} + \text{NFAT\_ph\_sp})}{\text{NPPHSP}_{\text{source}}} * \text{NINCPHSP}_{\text{source}}$$

where `IHSTRY` is the number of histories run, `NPASS_ph_sp` is the number of particles rejected from the source because they were multiple passers, `NFAT_ph_sp` is the number of fat photons rejected from the source (only if directional bremsstrahlung is used–see section 6.3.4), and `NRCYCL` is the number of times each particle in the source is recycled (see section 4.13). The value of `NINCPHSP` is stored in any phase space files generated by the phase space source and is also used to normalize doses and fluences resulting from the phase space source. Thus, `NINCPHSP`, doses and fluences are traceable back to the original source. For example, consider a model of a ^{60}Co unit which is broken into 2 components. In the first part, the source capsule is modelled and a phase space file created with all the particles leaving the surface of the capsule. Here all outputs are normalized per photon from the ^{60}Co. In the second stage, the phase space file from the first part is used as input to a model of the collimator system. Here again the outputs are normalized per photon from the ^{60}Co, thus automatically maintaining the "natural" normalization.

Each record in a phase space file contains the following information about a particle (in this order):
`LATCH, E, X, Y, U, V, WT, (ZLAST)`
where:

LATCH contains the particle charge, IQ, the number of times the particle has crossed the scoring plane, `NPASS`, and information which allows the particle's history to be traced (see section 8 below). The value of `LATCH` in the phase space file is not the same as that internal to BEAMnrc or other analysis codes because of the compression used.

E is the particle total energy (kinetic and rest mass, single precision). This is set negative if this is the first particle scored from a new primary (ie from a non-phase space source) history.

X is the particle X-position (cm)

Y is the particle Y-position (cm)

U is the X-direction cosine

V is the Y-direction cosine

WT is the particle's weight; WT also carries the sign of W, the Z-direction cosine

ZLAST is the Z-position of last interaction for photons and is the Z-position of where an electron or its ancestor was set in motion by a photon (i.e. it does not flag the creation site of delta rays. This variable is in brackets because its inclusion in the phase space file depends upon the setting of the input variable `IZLAST` (see section 10.6 on `IZLAST`). If a particle does not interact, ZLAST is the value it had as it entered the simulation (`ZIN`).

The magnitude of the Z-direction cosine, W, is determined from $W = \sqrt{1 - (U^2 + V^2)}$. The particle's charge is stored in bits 29/30 of `LATCH` and recovered when read in.

We set `E` negative for the first particle scored by each new primary (non-phase space) history in order to be able to group scored quantities (energy deposited, fluence, etc) according to primary history when a phase space file is used as a source. This is necessary to account for correlations between incident particles and ensures a correct estimate of the uncertainty on the scored quantities [17]. The negative `E` marker is propagated to phase space files generated using a phase space source, so that, even in these second-generation files, particles can be grouped according to the original primary histories. When a negative `E` is read from a phase space source, the primary history counter is incremented and the particle's energy is set to |`E`|.

If you are using an old phase space file without negative `E` markers as a source, scored quantities will be grouped according to incident particle instead of primary history. BEAM-nrc will output a warning that uncertainties may be underestimated because correlations between incident particles cannot be accounted for. However, we have shown that in most cases the underestimates in uncertainty caused by not taking correlations into account are not significant [17].

When BEAMnrc writes a phase space file, it opens the file using
ACCESS = 'direct', FORM = 'unformatted', RECL = 'length'.
The `FORM='unformatted'` statement ensures that the file will be stored in a compressed format, requiring less disk space. The record length, 'length', depends on the machine being used to run BEAMnrc; on SUN stations and Linux PC's, 'length' is the number of bytes/record (28 or 32 [with `ZLAST`]); on SGI and DEC alpha machines, 'length' is the number of variables stored in a record (7 or 8 [with `ZLAST`]). Internally 'length' is determined as 7*$RECL_FACTOR or 8*$RECL_FACTOR (with `ZLAST`), where the value of the macro is 1 or 4 and is defined in `$HEN_HOUSE/lib/${my_machine}/machine.mortran`.

The format of phase space files opened with
ACCESS = 'direct', FORM = 'unformatted' is called 'MODE0' if `ZLAST` is not scored and 'MODE2' if `ZLAST` is scored. The 'MODE0' or 'MODE2' designation appears in the

first record of a phase space file along with the total number of particles contained in the file, the number of photons, maximum kinetic energy of any particle in the file, minimum kinetic energy of electrons, and the number of particles incident from the original source.

An older version of BEAM opened files in compressed format but with `ACCESS='sequential'`. These older files are designated 'MODE1' without `ZLAST` and 'MODE3' with `ZLAST`. The current version of BEAM requires conversion of files in the older format to access='direct' format before adding new phase space data to them. The `readphsp` program described below performs this conversion.

Phase space files have extension `.egsphsp#` where `#` is the number of the scoring plane in the accelerator. By default, the maximum number of scoring planes is 3, but this can be adjusted as described in section 2.11.

BEAMnrc makes use of macros stored in `$HEN_HOUSE/utils/phsp_macros.mortran` to read and write the phase space files. Currently, we have optimised the writing macros to store up and write 1000 particles at a time. This has been found to save network time. Certainly, there are other read/write optimisations that could be made, however, in a normal accelerator simulation, only a small fraction of the simulation time is taken up reading from and writing to phase space files. The source for these macros is kept separately so that users may utilise them in their own codes to read or write phase space files. The same macros are used uniformly throughout the BEAMnrc system (DOSXYZnrc, BEAMDP, BEAMnrc *etc.*).

7.2   Directory for Phase Space Output

By default, phase space files from an accelerator simulation are written to the directory `$EGS_HOME/BEAM_myaccel` (*i.e.* the accelerator directory). However, you do have the option to output phase space files to a different directory. This is particularly useful if you are writing large phase space files and, due to disk space limitations, you want to write them directly to a `/temp` storage directory on another machine (provided it is connected with NFS to the machine(s) you are running on).

BEAMnrc provides two different ways to change the phase space output directory:

In the `$OMEGA_HOME/beamnrc/beamnrc_user_macros.mortran` file, you can redefine the macro `$DIRECTORY-FOR-PHSP`, which is currently set to `$EGS_HOME/BEAM_myaccel`, to to be the directory you want to write phase space files to. Note that you must provide the full directory path in single quotes (*i.e.* `'/full path to directory'`). An example is given in the `beamnrc_user_macros.mortran` file. Once you have made this change, you must recompile your accelerator to put it into effect.

BEAMnrc also has a built-in custom user input that you can use to set the variable `PHSP_OUTDIR`, which redefines the output directory for phase space files. The input line is:
Phsp output directory= /full path to phase space output directory
This must appear in the accelerator `.egsinp` file in the custom input block delimited by `:Start user inputs:` and `:Stop user inputs:`. The custom input block can be

placed either just before or just after the EGSnrc transport parameters (see Section 12 for more about custom inputs). If the input is left blank or omitted entirely, then the phase space output directory defaults to that defined by the $DIRECTORY-FOR-PHSP macro in `beamnrc_user_macros.mortran` (see above). Note that the GUI does not give you access to this input. This method of changing the output directory for phase space files has the advantage that the accelerator does not need to be recompiled whenever the output directory is changed and that different input files for the same accelerator can specify different phase space output directories.

7.3   IAEA-format phase space data

Provided that the user's system has a working C++ compiler (determined automatically on BEAMnrc installation), then BEAMnrc has the capability to read/write phase space data in IAEA format. This allows the user to add to or make use of data from the IAEA's online accelerator phase space database at:
www-nds.iaea.org/phsp/phsp.htmlx

7.3.1   IAEA format

A complete description of the IAEA phase space format is given in IAEA's technical report INDC(NDS)-0484[28], which is available from the online phase space database indicated above. The format is quite flexible in terms of the amount of data that is stored. This section gives a brief description of the IAEA data that is output and read by BEAMnrc.

Phase space data in IAEA format is contained in 2 files: the data file (with extension `.IAEAphsp`) and the header file (with extension `.IAEAheader`). IAEA-format header and phase space data files output by BEAMnrc have the names `inputfile.#.IAEAheader` and `inputfile.#.IAEAphsp`, where # is the scoring plane number.

The IAEA header file output by BEAMnrc contains the following data:

$CHECKSUM: The size of the `.IAEAphsp` in bytes. This is used when opening an IAEA (to use it as a source or to add more data to it) to make sure that there are no errors in the file.

$RECORD_CONTENTS: A block indicating the data that is stored in the `.IAEAphsp` file. A "1" beside a variable indicates that this is stored in the `.IAEAphsp` file. This block also indicates how many extra long integers and extra floating point variables are stored. In the case of BEAMnrc, two extra long integers, LATCH and the number of primary histories between this particle and the last particle scored, are always stored. In addition, if the input IZLAST=1 (See Section 10.6) then ZLAST, the Z of the last interaction site for photons and the creation site for secondary charged particles, is stored as an extra floating point variable.

$RECORD_CONSTANT: Values which are set to a constant value. In the case of BEAMnrc, the Z of the scoring plane is stored here (note that this information is not available in a

standard BEAMnrc phase space file).

$RECORD_LENGTH: The length of a phase space record in bytes.

$BYTE_ORDER: "1234" for little endian machines, "4321" for big endian machines.

$ORIG_HISTORIES: No. of primary histories used to generate the phase space data.

$PARTICLES: Total no. of particles represented in phase space data.

$PHOTONS: No. of photons represented in phase space data.

$STATISTICAL_INFORMATION_PARTICLES: Total weight, min. weight, max. weight, average total energy, min. total energy and max. total energy for each particle type

$STATISTICAL_INFORMATION_GEOMETRY: min. and max. X and Y of all particles scored.

There are other fields in the header but they are not currently used by BEAMnrc.

The phase space data file (.IAEAphsp) consists of a record of data for each particle scored, where a record consists of:

type: particle type and sign of Z-direction cosine, W (CHAR*1)

E: total energy of the particle (REAL*4)

X: X position of the particle (REAL*4)

Y: Y position of the particle (REAL*4)

U: X-direction cosine (REAL*4)

V: Y-direction cosine (REAL*4)

WT: statistical weight of the particle (REAL*4)

n_stat: no. of primary histories since last particle scored (INTEGER*4)

LATCH: value of LATCH variable (INTEGER*4)

ZLAST: Z of last interaction site of photons or site of creation of secondary charged particles (REAL*4). This is only output if the input variable IZLAST=1 (See Section 10.6).

Thus, each record is 33 bytes long without ZLAST and 37 bytes long if ZLAST is included.

Note that, unlike standard BEAMnrc phase space files, LATCH is not modified to carry the particle charge and NPASS before being written to the phase space file. The charge is stored in type and if this particle has already crossed the scoring plane (NPASS=1) then it is simply not written to the file. On the other hand, the IAEA phase space file stores n_stat, the number of primary histories since the last scored event. This allows correlated particles to be grouped together for uncertainty analysis, similar to the negative energy marker used in standard BEAMnrc phase space files. Moreover, if the phase space file is used as a source

n_stat allows an exact calculation of the number of primary histories represented by the phase space data read. This potentially provides a more accurate normalization for scored quantities, such as dose and fluence, than the estimate of NINCPHSP (See Section 7.1 above) in the case of a standard BEAMnrc phase space source.

7.3.2   Writing IAEA phase space data

By setting the input variable IO_OPT=4 (See Section 10.4) you will output the IAEA phase space data and header files at each scoring plane.

7.3.3   Reading IAEA phase space data

If you are using IAEA-format space data as a source (*i.e.* ISOURC=21. See Section 4.13 above.) then you must specify the full name of the phase space data (.IAEAphsp) file. BEAMnrc will automatically detect the .IAEAphsp extension and read the data in IAEA-format (with an output message indicating this to the user). Note that the header file is assumed to be in the same directory as the phase space data file.

7.4   readphsp

There are several programs and routines in place for processing phase space files. One of the most basic of these is the program readphsp located on $OMEGA_HOME/progs/readphsp. readphsp is invoked by:

readphsp inputfile outputfile

where inputfile includes the .egsphsp# extension and is the name of the phase space file to be converted, and outputfile is the name of the file which will contain the converted data (and must also include any extension you want to add). Note that readphsp retains compatibility with phase space files having .egs4phsp# extensions generated by EGS4 versions of BEAM.

readphsp gives the user various file conversion options (*eg.* from direct access mode to sequential access mode). It can also convert phase space files into a format readable by the CERN program PAW provided that the necessary libraries are present (see below).

readphsp can be used as a byte swapping tool so that phase space files generated on another type of machine become compatible with the type of machine that readphsp is being run on.

The readphsp program also allows selection of a particular charge from a phase space file, or a subset of a file can be extracted.

Before converting phase space files readphsp prints a summary of the file contents. If this is nonsense, then very likely the file has the wrong binary format (*i.e.* was generated on a machine which uses a different binary format). The user should use the option of readphsp to swap the bytes of the phase space data to make it compatible with the current machine

before otherwise converting the phase space data to its new form. When using `readphsp` for byte swapping, the name of the output file can be the same as that of the input file however the swapped data will overwrite the original data.

Files related to `readphsp` are on `$OMEGA_HOME/progs/readphsp`.

Makefile Directs compilation of `readphsp`. Sources required files such as `$HEN_HOUSE/lib/config/machine.macros` (where `config` is the name of your configuration) for the record length factor (`$RECL-FACTOR`) for 4-byte phase space data on your particular configuration, and `$OMEGA_HOME/beamnrc/phsp_macros.mortran` for phase space reading/writing macros that `readphsp` makes use of. You must modify `Makefile` if you want to compile `readphsp` with PAW libraries (see below) so that phase space data can be converted into PAW format. Clear instructions for doing this are given in the `Makefile`.

readphsp.mortran the `MORTRAN` source file.

libpawlib.a, libpacklib.a (optional) Libraries required for the PAW format conversion. These are not included with the distribution because they are proprietary from CERN, but they are available for free from CERN for those working on medical research.

dummy.f which is a routine required to compile `readphsp` when there are no PAW libraries present.

Normally, `readphsp` is compiled as part of the BEAMnrc installation. However, it can be compiled separately by going into `$OMEGA_HOME/progs/readphsp` and typing `make`. This will leave behind the executable, `readphsp*`, in the `$HEN_HOUSE/bin/config` directory.

Note that `readphsp` does not work with IAEA-format phase space data.

7.5   BEAMDP

Another important program for processing phase space files is the `BEAMDP` program. For more details about running `BEAMDP` to derive energy, planar fluence, mean energy and angular distributions from a phase-space file please see the report "BEAMDP as a General-purpose Utility"[13].

8   Tracking a Particle's History using LATCH

The `LATCH` variable, associated with each particle in a simulation, is a 32-bit variable used to track the particle's history. In the input files there is an opportunity to define a mapping from geometric regions to bits (*i.e.* bit regions) using the `IREGION_to_BIT` variable. Thus, *eg.* it is possible that bit 5 corresponds to geometric region 3, and more importantly, one bit, say 3, can correspond to multiple geometric regions, *eg.* 1,5,8. Thus, although the JAWS may consist of 6 different geometric regions, they can all be associated with a single bit or bit region. All regions which are not associated with a bit/bit region by the user are associated with bit region 23 by default. Each bit is designated as follows:

bit 0 Set to 1 if a bremsstrahlung or positron annihilation event occurs in the history; 0 otherwise(not used for `LATCH_OPTION` = 1).

bit 1-23 Used to record the bit region where a particle has been and/or has interacted (Note that the bit set for a region is determined by `IREGION_TO_BIT` for that region)

bit 24-28 Stores the bit region number (as opposed to geometric region) in which a secondary particle is created; if these bits are all 0, the particle is a primary particle (not for `LATCH_OPTION` = 1).

bit 29-30 Store the charge of a particle when `LATCH` is output to a phase space file (see section 7 on phase space files). During a simulation, bit 30 is used to identify a contaminant particle but this information is not output to the phase space file. Set to 1 if the particle is a contaminant particle; 0 otherwise. Note that if `LATCH` is not inherited (*i.e.* when `LATCH_OPTION` = 1), bit 30 loses its meaning.

bit 31 Set to 1 if a particle has crossed a scoring plane more than once when `LATCH` is output to a phase space file (see section 7 on phase space files above)

For secondary particles, recording the region number in which they were created in bits 24-28 is equivalent to multiplying the region number by 2^{24}, or 16777216. Thus, to retrieve the region of origin of a secondary particles, the `LATCH` value of the particle must be divided by 16777216 (*i.e.* taking the value `INT(LATCH/16777216)`).

The user controls the protocol for setting `LATCH` using the `LATCH_OPTION` input variable. The possible settings of `LATCH_OPTION` are:

LATCH_OPTION = 1 (Non-Inherited LATCH Setting): secondaries do not inherit `LATCH` values from the primaries that created them; bits 1-23 of a secondary particle carry no information about the regions its primary parent(s) has(ve) been. This option must NOT be used if `ICM_CONTAM` is non-zero since the `ICM_CONTAM` option needs bit 30.

LATCH_OPTION = 2 (Comprehensive LATCH Setting –default): `LATCH` values are passed on to secondary particles from the primaries that created them; bits 1-23 for a secondary particle include all regions in which the secondary particle has been plus those in which its ancestors have been up to the point where the secondary was created; uses bits 24-28 to record the bit region where secondary particles are created and bit 0 to record whether or not a bremsstrahlung photon was involved in a particle's history.

LATCH_OPTION = 3 (Comprehensive LATCH Setting 2): similar to 2, but for photons bits 1-23 record the regions in which the particles have interacted, rather than simply the regions in which they have been. After a Compton, pair or photo-electric event, the charged particles, and in the latter case, also the fluorescent photons, have the bits 1–23 set for the region in which they are created to treat the case in which they are created below cutoff in a manner similar to being created above cutoff (where the bits would be set on the first step).

To clarify further the setting of `LATCH` bits under various `LATCH_OPTION` values, fig 19 and table 2 summarise the situation for a simple photon accelerator. Two electrons enter the simulation and produce bremsstrahlung photons in the target, bit (and geometric) region 1. One photon goes all the way to the scoring plane without interacting and the second undergoes a pair production event in the flattening filter (bit region 3). The electron escapes from the flattening filter an gets to the scoring plane and the positron annihilates in the flattening filter and one of the resulting 511 keV photons gets to the scoring plane. The user has also defined a contamination scoring plane just above the JAWS (*i.e.* `ITDOSE_ON=1`, `ICM_CONTAM=4` (just above the JAWS) and `IQ_CONTAM=-1`).

LATCH
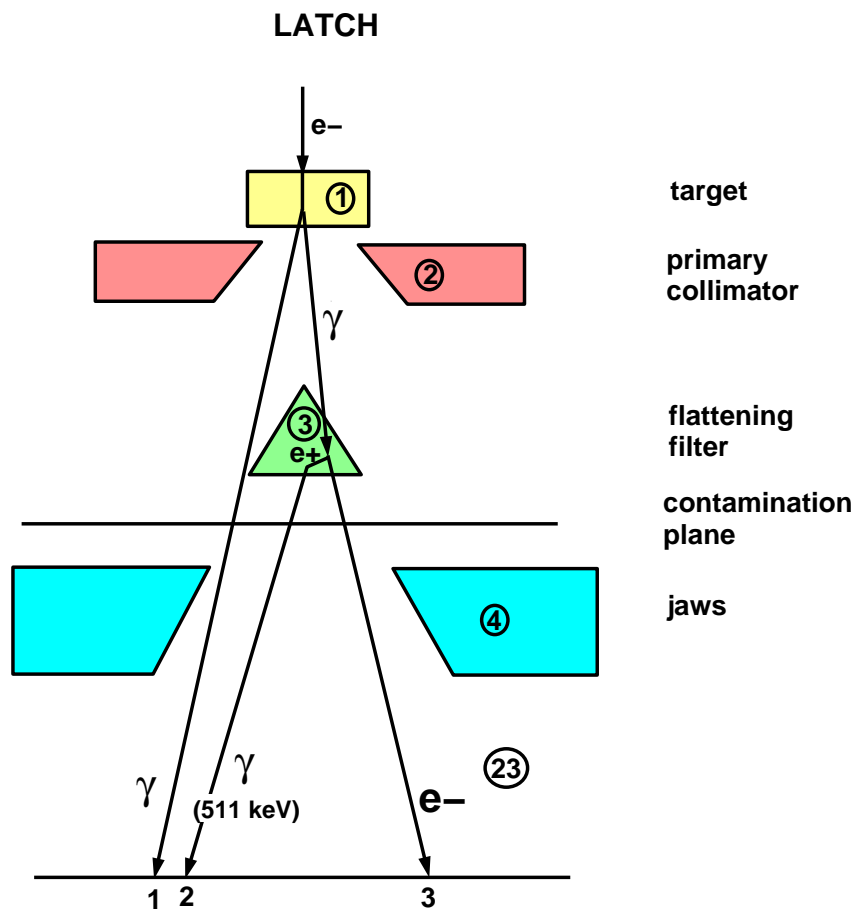
Figure 19: Simple photon accelerator model showing 3 particles reaching the phase space file after 2 electrons are incident. Contamination is defined as charged particles crossing the contamination plane. Table 2 shows the bit settings in LATCH as the particles reach the scoring plane. The bit region numbers are shown in circles.

Table 2: Bit settings in LATCH for the simple example shown in fig 19.

Latch option	Particle	Bit 0	1	2	3	4	23	24-28	30
1 non-inherited	1	0	1	0	0	0	1	0	0
	2	0	0	0	1	0	1	0	0
	3	0	0	0	1	0	1	0	0
2 comprehensive	1	1	1	0	0	0	1	"1"	0
inherited	2	1	1	0	1	0	1	"3"	0
where been	3	1	1	0	1	0	1	"3"	1
3 comprehensive	1	1	1	0	0	0	0	"1"	0
inherited	2	1	1	0	1	0	0	"3"	0
where interacted	3	1	1	0	1	0	1	"3"	1

9   Calculating Dose Components

The ability to trace a particle's history using `LATCH` also allows doses to be broken down into their components. In any dose zone, BEAMnrc is able to break dose down in 2 ways: dose from contaminant particles (identified on the basis of their charge only); or dose including only and/or excluding only contributions arising from particles with certain user-specified `LATCH` bit settings (this is called "bit filtering").

The input variables associated with dose component calculations are:

ITDOSE_ON 0 to calculate total dose only (default); 1 to calculate dose components

The following variables are associated with contaminant dose calculation and are only required if `ITDOSE_ON = 1`.

ICM_CONTAM contaminant particles are identified upon entering the top of `CM` number `ICM_CONTAM` if $1 \leq$ `ICM_CONTAM` $\leq$ total number of `CM`s (previously, this range was restricted to $1 <$ `ICM_CONTAM` $\leq$ total number of `CM`s); if it is set to 0, no contaminant dose will be calculated.

IQ_CONTAM The charge of the contaminant particles (0 for photons and 1 for charged particles); all particles with this charge will be marked as contaminant particles (by setting `LATCH` bit 30 of the particles to 1) upon entering the front of `CM` number `ICM_CONTAM`.

Contaminant dose is scored in every dose zone. However it is traced back only to those particles identified as having contaminant charge upon entering CM number `ICM_CONTAM`. For example, if `IQ_CONTAM` $= 1$, all the charged particles entering CM number `ICM_CONTAM` will be marked as contaminant particles and this mark will be passed onto their descendants via `LATCH` bit 30. The dose contributed by the contaminant particles and their descendants is then scored as the contaminant dose. Note that if `LATCH_OPTION = 1` `LATCH` values are not transferred to descendants (secondaries), and contaminant dose calculations will be meaningless. Thus, the contaminant dose option is automatically turned off if `LATCH_OPTION = 1`. Note also that prior to Sept 2002, contamination was defined as it entered the CM from the front or the back.

The following variables are associated with bit filtering of dose and are only required if `ITDOSE_ON = 1`:

LNEXC number of dose components that exclude contributions from particles with user-specified `LATCH` bits set (bit filters). `LNEXC = 0` is allowed. `LNEXC <= $MAXIT - 3` where `$MAXIT` can be changed from its default value of 12 (as described in section 2.11).

L_N_EXC(I,31) (I = 1,2,...,LNEXC) Bit filter for determining which particles to exclude from dose component I; each of the LNEXC components requires input of a separate bit filter; a bit filter consists of a sequence of up to 31 `LATCH` bit numbers on a single record (range 1-31), separated by commas; particles with any of the listed bits set are excluded from dose component I.

LNINC number of dose components with contributions only from particles with user specified bit filters which are a combination of inclusive and possibly exclusive patterns. LNINC = 0 is allowed. `LNINC <= $MAXIT - LNEXC - 3`.

L_N_INC(I,31) (I = 1,2,...,LNINC) Bit filter for determining which particles to include in dose component I; each of the LNINC components requires input of a separate bit filter which consists of 2 groups of `LATCH` bit numbers (range 1-31) separated by a 0; contributions from the particles with any of the `LATCH` bits in the first group of `L_N_INC(I,31)` set and none of those in the second group set are included in dose component I.

For example, if the inputs for `L_N_INC(I,31)` are 2,3,5,0,1,4,0,..., then dose component I will include the contributions from the particles with **any** of the `LATCH` bits 2, 3 or 5 set (=1) **and** both `LATCH` bits 1 and 4 not set (=0); the status of other `LATCH` bits will have no effect on this dose component. If the inputs are 1,2,3,0,0,...,(equivalently 1,2,3,) the dose component will include the contributions from the particles with any of the `LATCH` bits 1, 2 or 3 set (=1). Note that no dose will be scored if one inputs 0,2,3,5,0,...

Note that bit 0 of `LATCH` can not be used for the above dose component calculations.

Bit filtering of dose provides a particularly powerful tool for determining dose contributions. In view of the information stored in `LATCH` (see section above), dose contributions can be separated according to what regions particles have passed through/interacted in, whether the particle is a primary or secondary, if the particle is a secondary then where it was created, whether or not the particle is a contaminant, and any combination of these.

It should be noted that if directional bremsstrahlung splitting (DBS) is used in a BEAM simulation then an additional dose component, the total dose minus the dose due to fat particles, is automatically output to the `.egslst` and `.egsplot` files. see section 6.3.4 for more details.

10   Other Input Variables

This section provides descriptions of main BEAMnrc input variables not covered above.

10.1   IWATCH

`IWATCH` controls output to the screen (interactive run) or to the `.egslog` file (batch run) during beam execution. The possible settings are:

= 0 (default) On completion of each batch, outputs information about the batch (eg elapsed time, CPU time, elapsed/CPU time, random number used to begin batch, # of random seeds used in the batch, total number of histories run up to and including the current batch, the total number of particles scored in the first phase space file)

= 1 Outputs batch information plus, after every particle interaction, outputs complete information about the particle(s) involved (eg interaction type, particle type, position of particle on stack, particle energy, X-Y-Z position of particle, U-V-W direction cosines, `LATCH` value of particle, region # of particle); also informs user when a particle is being discarded, when a particle is passing from one CM to another

= 2 Similar to 1 only complete particle information is output at every step; also outputs total dose in a region whenever energy is deposited there

= 3 Similar to 2 only particle and dose information are output whenever dose is deposited.

= 4 Outputs same information as 0 plus `.egsgph` and `egsgeom` files for graphical representation of the accelerator and particle paths using `EGS_Windows`[16].

10.2 ISTORE

ISTORE controls how BEAMnrc reads and writes the state of the random number generator (RNG) used in simulations. The possible settings of ISTORE are:

= 0 (default) BEAMnrc stores the state of the RNG for the 1st history in each batch in the file `.egsrns`

= 1 BEAMnrc stores the state of the RNG for every history in the .egsrns file. The random number state of the current history overwrites that of the previous history.

= -1 BEAMnrc reads the state of the RNG for the 1st history from the `.egsrns` file. If there was an error during a history that caused the history to stop, then you can use `ISTORE = -1` to rerun the problem history, provided that you first stored the state of the RNG for each history using `ISTORE = 1`.

10.3 IRESTART

IRESTART controls how BEAMnrc treats the current run - either as new or as a continuation of a previous run. The possible settings of IRESTART are:

= 0 (default) BEAMnrc initiates a new run, deleting all of the output files from previous runs of the same name if present (`.egslog`, `.egslst`, `.egsrns`, `.egsphsp1`, *etc.*).

= 1 Restart of a previous run; BEAMnrc reads fluence and dose data from the previous run, the number of histories already run, the time taken by the previous run, and the state of the random number generator from the last batch of the previous run from the `.egsdat` file; fluence and dose and their statistics are averaged with those from the previous run; phase space output is appended to that from the previous run; note that the number of histories to run in the present run is given by NCASE but the reported total CPU time and number of histories analysed in the output include the histories and CPU time from the previous run.

= **2** BEAMnrc creates the `.egsinp` file and then exits without running the simulation.

= **3** BEAMnrc reads dose and fluence data from a previous run in the `.egsdat` file, performs statistical analysis on the data and outputs results to `.egslst`; no simulation is run.

= **4** For analysis of dose and fluence in a simulation that has been split up into parallel jobs. See section 13 page 123 below on parallel processing with BEAMnrc for more details.

10.4   IO_OPT

`IO_OPT` allows user control of phase space output. Some aspects of this variable have already been covered in section 7 on phase space files. However, it also controls data analysis of simplified source models. The possible settings for `IO_OPT` are:

= **0** (default) Phase space output at each scoring plane

= **1** No phase space output at scoring planes

= **2** No phase space output, but perform data analysis for simplified source models (see BEAMDP manual).

= **3** Phase space output for the first 100,000 particles across each scoring plane and perform data analysis for simplified source models

= **4** IAEA-format phase space output at each scoring plane

10.5   IDAT

`IDAT` controls output of the `.egsdat` file, which contains data from a previous run so that BEAMnrc is able to restart this run if required. The `.egsdat` files from parallel jobs are required for recombining to obtain the final results (see section 13). Possible settings of `IDAT` are:

= **0** (default) Output energy deposited and fluence data, number of histories completed, elapsed time and the current state of the random number generator to the `.egsdat` file upon completion of every batch

= **1** Do not output a `.egsdat` file

See section 2.10 for a more complete description of the `.egsdat` file.

10.6   IZLAST

`IZLAST` controls whether or not `ZLAST` is recorded in phase space files. `ZLAST` is the Z-position of last interaction for photons and is the Z-position of where an electron or its ancestor was set in motion by a photon (i.e. it does not flag the creation site of delta rays). The possible settings of `IZLAST` are:

= **0** (default) BEAMnrc does not score `ZLAST` in phase space files; this means that phase space files are read and written in compressed "mode0"

= **1** BEAMnrc scores the Z position of last interaction in phase space files; phase space files are read and written in compressed "mode2"; "mode2" files are approximately 14% larger than "mode0" files; when plotting phase space data with paw, `ZLAST` replaces the radial position of a photon (r)

= **2** Same as option 1 and in Addison, BEAMnrc writes the X-Y-Z positions of the last site of interaction for photons into the `.egsgph` file. These last sites of interaction can be viewed in 3-D using `EGS_Windows`[16]. `IWATCH`=4 must not be turned on at the same time since it also writes to the `.egsgph` file.

Note that the utility BEAMDP has an option to present graphs of `ZLAST` in 2-D.

10.7   NCASE

`NCASE` is the number of histories for this run. Minimum value is 100. Default is 100 if `NCASE` is set < 100. The number of histories per batch is equal to `NCASE`/(`$NBATCH`), where `$NBATCH` is the number of output batches. `$NBATCH` is set in the `beamnrc_user_macros.mortran` file and is currently equal to 10. The total number of histories after this run will be the sum of `NCASE` and the number of histories from the previous run.

Starting with BEAMnrc, the breaking of the run into batches is for saving intermediate data and diagnostics. It is no longer associated with statistical uncertainties.

Starting with BEAM99, `NCASE` and other variables related to the number of histories, such as `IHSTRY`, were made `INTEGER*8` variables. This allows $>2\mathrm{x}10^9$ total histories, and is particularly useful for restarts and parallel runs, where a large number of total histories can be accrued quite easily. However, `INTEGER*8` may cause problems for some Fortran compilers. For a more detailed description of this problem and how to fix it if it does occur, see section 18 on known bugs/restrictions in BEAMnrc.

10.8   IXXIN, JXXIN

The input values of `IXXIN` and `JXXIN` are seeds used to initialize the RANLUX[29, 30] or RANMAR[31, 32] random number generator. Within BEAMnrc, `IXXIN` is limited to the range 0<`IXXIN`≤31328 (defaults to 1802), and `JXXIN` has the range 0<`JXXIN`≤30081 (defaults to 9373). These ranges/defaults were originally designed for RANMAR, however,

if you are using RANLUX (the new default) then `IXXIN` is the "luxury level" of the random number generator and must be in the range $0 < \text{IXXIN} \leq 4$, otherwise, it will automatically be set to the default luxury level of 1 (Note: this means that the BEAM default value for `IXXIN` of 1802 will ultimately get reset to 1 by RANLUX). Also, with RANLUX, the input `JXXIN` can have a maximum value of 1073741824, however you will have to change the BEAMnrc code to allow this maxium.

Note that `IXXIN` and `JXXIN` are only used to initialize the random number generator, and, during a simulation, they no longer reflect the values of the seeds that are actually used to generate the random numbers. During restarts (`IRESTART` = 1), the state of the random number generator at the end of the previous run is read from the `.egsdat` file and is used at the beginning of the restart. Thus, a restarted run with a total of, eg, 10000+10000 histories should generate results identical to a single run of the same simulation with 20000 histories. Also note that when running parallel simulations with BEAMnrc (see section 13), `JXXIN` must have a different value for each of the individual jobs that make up the simulation. A property of the random number generators being used is that this simple change guarantees random number sequences which are independent.

To switch from the default RANMAR to RANLUX in your BEAM code, either go into `$HEN_HOUSE/specs/beamnrc.spec` and change the line:

`RANDOM = $(EGS_SOURCEDIR)ranmar`

to:

`RANDOM = $(EGS_SOURCEDIR)ranlux`

(recommended if you plan on using RANLUX as your default random number generator for all accelerators), or compile an individual accelerator using the command

`make RANDOM=$HEN_HOUSE/src/ranlux`

10.9   TIMMAX

`TIMMAX` is the maximum CPU time in hours allowed for this run. `TIMMAX` defaults to 0.99 hours if it is set = 0. At the end of each batch, an estimate is made of how much time is needed to complete another batch. If the next batch cannot be completed within `TIMMAX` for the current run, then BEAMnrc terminates the run and analyses the results for the shortened run. The restart feature (see section 10.3 on `IRESTART`) can be used to continue the run if needed and if `IDAT` (section 10.5) is set to 0 (*i.e.* the `.egsdat` files are created). Note that the total CPU time reported for a simulation is the sum of the CPU time so far for this run and the CPU time from the previous run, but `TIMMAX` refers to the present run's time only.

10.10   ECUTIN

`ECUTIN` is used together with the `Global ECUT` input in the EGSnrc input parameters (see section 11) to define the global electron cutoff energy in MeV. If `ECUTIN` > `Global ECUT` in the EGSnrc inputs, or if the `Global ECUT` input is missing from the EGSnrc inputs, then

ECUTIN is used as global cutoff energy.

As soon as an electron's total energy falls below the cutoff energy, its history is terminated and its energy deposited in the current region. The time required for a given calculation is strongly dependent on the value of ECUT and thus it is important to use as high a value as possible.

The user can override the Global ECUT with the ECUT's defined for individual regions within CMs (see CM descriptions below). However, if the ECUT for an individual region is <Global ECUT, then it is set equal to the Global ECUT.

Note that AE for the PEGS4 data set used is the lower limit on the value of ECUT used in a given region. The selection of AE also requires some care and is discussed in section 16.2.

Selection of ECUT is complex in general and is dependent on what is being calculated[33, 25]. For therapy beams, ECUT can be quite high since low-energy electrons contribute little to dose in phantom. For what we consider detailed work, we have used ECUT = 0.700 MeV but much higher may be possible. However, if the dose in the monitor chamber is an important part of the calculation, lower values of ECUT may be required.

As a general rule of thumb for calculations of dose distributions, ECUT should be chosen so that the electron's range at ECUT is less than about 1/3 of the smallest dimension in a dose scoring region. This ensures energy is transported and deposited in the correct region although for electrons which are moving isotropically, this can be a very conservative requirement.

10.11   PCUTIN

PCUTIN is used together with the Global PCUT input in the EGSnrc input section to define the global cutoff energy for photon transport in MeV. It is the photon equivalent of ECUTIN. Similar to ECUTIN, if PCUTIN is > the value input for Global PCUT in the EGSnrc input section, or if Global PCUT is omitted from the EGSnrc inputs, then PCUTIN is used as the Global PCUT. Also the user can override Global PCUT with PCUTs defined for individual regions within CMs.

The exact value of the Global PCUT is not critical in the sense that low values do not take much more time. A value of 0.01 MeV should generally be used.

10.12   ESTEPIN, SMAX, IDORAY, IFLUOR

These are dummy inputs which used to define the maximum fractional energy loss per electron step (ESTEPIN), the maximum step length (SMAX), a switch for turning on Rayleigh scattering (IDORAY), and a switch to turn on K-shell X-ray fluorescence (IFLUOR). Now all of these transport parameters are handled in the EGSnrc inputs (see section 11), however, the dummy inputs are retained for compatibility with older EGS4 BEAM input files.

10.13   ICM_SPLIT, NSPLIT_PHOT, NSPLIT_ELEC

ICM_SPLIT is used to split photons and electrons at an arbitrary plane within an accelerator. To use this option, the user sets ICM_SPLIT equal to the CM # at the top of which the photons and electrons are to be split. When ICM_SPLIT>0, then the user must input NSPLIT_PHOT and NSPLIT_ELEC, the splitting number for photons and electrons, respectively, upon entering the CM. If NSPLIT_PHOT or NSPLIT_ELEC is ≤1, then the relevant particle type is not split at all. Once a particle has been split, the resultant particles carry weight 1/NSPLIT_PHOT or 1/NSPLIT_ELEC. A particle is only split upon entering the user-specified CM from the top (*i.e.* W(NP) > 0) and if it has not been split by this option before.

Splitting at an arbitrary plane was designed primarily for improving statistics in dose calculations in a phantom, in which case particles (usually photons) would be split upon entering a phantom at the bottom of the accelerator. Use of this option near the top of an accelerator may produce undesirable correlations, without much gain in efficiency.

The ICM_SPLIT option operates in conjunction with any kind of bremsstrahlung splitting and photon forcing. In the case where ICM_SPLIT=NFCMIN (*i.e.* the arbitrary splitting plane corresponds to the plane at which photon forcing is to begin), the photons are split BEFORE they are forced to interact.

Currently, if ICM_SPLIT=1, then the option is turned on, but no splitting will occur. We hope to remove this restriction, but if you wish to split particles in CM 1, you can get around the problem, by inserting a dummy CM 1 and setting ICM_SPLIT=2.

11   EGSnrc inputs

The use of EGSnrc to simulate charged particle and photon transport in BEAMnrc allows the user a greater degree of control over the transport physics than was previously available in EGS4 versions of BEAM. For most accelerator applications, the default settings in the BEAMnrc code for the EGSnrc parameters should be adequate (these are not the same as the EGSnrc standard defaults). However, there are some cases, such as low energy applications, in which the user will want to vary the EGSnrc transport parameters using the EGSnrc inputs.

EGSnrc inputs appear at the end of a BEAMnrc input file using the new format used with the general purpose user EGSnrc user codes. The input occurs between the delimiters :start mc transport parameter: and :stop mc transport parameter:.

In general, EGSnrc inputs must appear in the input file in the format:
PARAMETER NAME= parameter value
Note that there is a space between the "=" sign and the parameter value but not before the "=" sign. If you are using the BEAMnrc GUI to set the EGSnrc inputs, then the above format is written to the input file automatically when you save the input parameters.

If any or all of the EGSnrc input parameters is missing, then the default setting will be used. This feature allows BEAM input files to be used directly with BEAMnrc. A better approach is to read the old BEAM input file into the beamnrc_gui and then save it since

this will explicitly add the required EGSnrc inputs to the file.

The following sections describe the EGSnrc inputs required in BEAMnrc. For more information, see the EGSnrc manual[1]. The actual internal variable name associated with each input appears in brackets.

11.1   `Global ECUT` (`ECUT`)

`Global ECUT` defines the global electron cutoff energy (`ECUT`) in MeV. This is one of the two EGSnrc input parameters that is also accessible through the main BEAMnrc input section of the input file (the other is `Global PCUT` described below). Specifically, if `ECUTIN` in the main BEAMnrc inputs is > `Global ECUT`, or if `Global ECUT` is missing from the EGSnrc input section, then the global value of `ECUT` is set to `ECUTIN`. See section 10.10 for a more detailed discussion of `ECUT`.

11.2   `Global PCUT` (`PCUT`)

`Global PCUT` defines the global photon cutoff energy (`PCUT`) in MeV. Similar to `Global ECUT`, this EGSnrc input parameter is also accessible through the main BEAMnrc input section of the input file. If `PCUTIN` in the main BEAMnrc inputs is > `Global PCUT`, or if `Global PCUT` is missing from the EGSnrc input section, then the global value of `PCUT` is set to `PCUTIN`. See section 10.11 for a more detailed discussion of `PCUT`.

11.3   `Global SMAX` (`SMAXIR`)

`Global SMAX` defines the maximum electron step length in cm. If the default EGSnrc electron step electron algorithm (see section 11.8) and the exact boundary crossing algorithm are used, then no restriction on maximum step length is needed. However, if using PRESTA-I (the EGS4 standard) as the electron step algorithm or the boundary crossing algorithm, then `Global SMAX` must be set to a reasonable value (eg 5 cm) to ensure proper electron transport in low density materials (air). `Global SMAX` defaults to 5 cm when the PRESTA-I boundary crossing algorithm (BCA) or electron step algorithm is used is used and 1.E10 cm when the EXACT BCA and PRESTA-II electron step algorithm are used.

11.4   `ESTEPE` (`ESTEPE`)

`ESTEPE` is the maximum fractional energy loss per electron step. For accurate electron transport with default EGSnrc electron step algorithm, `ESTEPE` should not exceed 0.25 (the default). The value of `ESTEPE` should not be changed unless PRESTA-I is being used as the electron transport algorithm.

11.5 XImax (XIMAX)

`XIMAX` is the maximum first multiple elastic scattering moment per electron step. It is equal to roughly half the average multiple scattering angle squared. Make sure you do not set `XIMAX` > 1, since this is beyond the range of available multiple scattering data. The default value of 0.5 should be sufficient for most applications.

11.6 Boundary crossing algorithm (BCA) (bca_algorithm)

This controls the algorithm used to transport electrons across region boundaries. There are two possible settings of `Boundary crossing algorithm`: `EXACT` and `PRESTA-I`. The default for BEAMnrc is `EXACT` (same as in EGSnrc itself). The `EXACT` boundary crossing algorithm was introduced in EGSnrc to eliminate a known fluence singularity caused by forcing a multiple scattering event at a boundary [34]. In the `EXACT` case, electrons are transported in single elastic scattering mode as soon as they are within a distance from the boundary given by the EGSnrc input `Skin depth for BCA` (see section 11.7 below). If the `PRESTA-I` BCA is used boundary crossing is carried out in a manner similar to EGS4/PRESTA. Specifically, the lateral correlation algorithm is turned off if the perpendicular distance from the electron to the boundary is less than `Skin depth for BCA` (see section 11.7 below) and then, once the electron reaches the boundary, a multiple scattering event is forced.

Until 08/06, the default BCA for BEAMnrc was `PRESTA-I`. However, this has been switched to the slower `EXACT` BCA because it was found that the use of the `PRESTA-I` can result in significant overestimates of dose (up to 2.5%) when the CHAMBER component module is used as a depth-dose phantom (see Section 15.3.6). The use of the `EXACT` BCA is not expected to significantly increase the CPU time in most accelerator simulations. See the technical note by Walters and Kawrakow[12] for more information.

11.7 Skin depth for BCA (skindepth_for_bca)

If `Boundary crossing algorithm`= `PRESTA-I`, then `Skin depth for BCA` is the perpendicular distance (in elastic mean free paths) from the boundary at which lateral pathlength corrections are turned off and the particle is transported in a straight line until it reaches the boundary. By default the distance at which to switch off lateral corrections is a fixed value calculated by EGSnrc to be the same as that used in the original implementation of PRESTA in EGS4 and depends on the value of `ECUT`.

If `Boundary crossing algorithm`= `EXACT`, then `Skin depth for BCA` determines the perpendicular distance (in elastic mean free paths) to the region boundary at which electron transport will go into single elastic scattering mode. A skin depth of 3 elastic mean free paths has been found to give peak efficiency in this case and is the default.

If `Boundary crossing algorithm`= `EXACT` and `Skin depth for BCA` is set to a very large number (eg 1e10), then the entire simulation will be done in single scattering mode.

11.8   Electron-step algorithm (`transport_algorithm`)

This input determines the algorithm used to calculate lateral and longitudinal corrections to account for elastic scattering in a condensed history electron step. There are 2 possible settings: `PRESTA-II` (the default) and `PRESTA-I`. `PRESTA-II` is the new, more accurate, algorithm developed for use with EGSnrc[1]. `PRESTA-I` is the original PRESTA algorithm with some modifications[35]. The original `PRESTA-I` is known to underestimate lateral deflections, to underestimate longitudinal straggling and to produce a singularity in the distribution describing the lateral spread of electrons in a single condensed history. While `PRESTA-I` may be accurate enough for high energies (since elastic scattering is weak), it is not recommended for low energy applications.

11.9   Spin effects (`spin_effects`)

If `Spin effects= on` (the default), then elastic scattering cross-sections that take into account relativistic spin effects are used in electron transport. If `Spin effects= off`, then screened Rutherford cross-sections (similar to EGS4) are used for elastic scattering. It should be noted that using `Spin effects= on` does increase calculation time, however, results are more accurate and it is ABSOLUTELY necessary for good backscatter calculations.

Including spin effects has a small but distinct effect on calculated depth-dose curves. In low-Z materials such as water, the value of R_{50} for a given energy is higher than with EGS4/PRESTA. For high-Z materials it is the reverse and backscatter also increases.

11.10   Brems angular sampling (IBRDST)

This input determines the type of angular sampling that is done when a bremsstrahlung photon is created. If `Brems angular sampling= Simple` (the default) then bremsstrahlung angles are sampled using only the leading term of modified equation 2BS of Koch and Motz[24, 36]. If `Brems angular sampling= KM`, then the bremsstrahlung angles are sampled using the entire modified equation. `Brems angular sampling= Simple` is adequate at high energies, however, there is little increase in simulation time associated with using the entire modified 2BS equation and the entire equation is recommended at low energies. Note that `Brems angular sampling= KM` is similar to the bremsstrahlung angular sampling scheme used by the latest version of EGS4/BEAM, with some modifications.

11.11   Brems cross sections (IBR_NIST)

This input determines the differential cross-section used for bremsstrahlung interactions. If `Brems cross sections= BH` (the default), then Bethe-Heitler cross-sections (Coulomb corrected above 50 MeV)[36] are used. These cross-sections are those used by EGS4/BEAM. If `Brems cross sections= NIST`, then differential cross-sections from the NIST bremsstrahlung cross-section data base[37, 38] are used. The NIST cross-sections are the basis for radiative stopping powers recommended by the ICRU[39]. The difference between `BH` and `NIST` is

negligible for energies > 10MeV, but becomes significant in the keV energy range where the NIST data base is preferred. In either case, the total bremsstrahlung cross sections are the same. There is also a `Brems cross sections= NRC` option. The NRC cross-sections are the NIST cross-sections including corrections for electron-electron bremsstrahlung (typically only significant for low values of the atomic number Z and for k/T ¡ 0.005).

11.12   Bound Compton scattering (IBCMP)

The `Bound Compton scattering` input is used to determine whether binding effects and Doppler broadening are simulated in Compton (incoherent) scattering events. If this input is set to `Off` (the default), then the Klein-Nishina formula[40] is used to determine differential cross-sections for Compton scattering. This is similar to the treatment of Compton scattering in EGS4/BEAM. If `Bound Compton scattering= On`, then the original Klein-Nishina formula is augmented with the impulse approximation[41] to simulate binding effects and Doppler broadening. Simulation of binding effects and Doppler broadening takes extra time and is only important below 1 MeV and/or if Rayleigh scattering is being simulated (see section 11.18). A third option, `Bound Compton scattering= Norej`, is provided which uses the total bound Compton cross sections (em i.e. no impulse approximation) and does not reject any Compton interactions at run time.

   Bound Compton scattering may also be turned on in selected regions (off everywhere else) using `Bound Compton scattering= On in regions` together with the inputs `Bound Compton start region` and `Bound Compton stop region` to define the region ranges for which bound Compton is to be turned on. Conversely, bound Compton can be turned off in selected regions (on everywhere else) by inputting `Bound Compton scattering= Off in regions` with `Bound Compton start region` and `Bound Compton stop region` used to define the region ranges where bound Compton is to be turned off. Of course, turning bound Compton on/off in regions is accomplished much more easily in the BEAMnrc GUI. Note that the `Norej` option cannot be used on a region-by-region basis.

11.13   Compton cross sections (comp_xsections)

If the `Bound Compton scattering= Norej` option is selected (see above), then the user also has the option of specifying their own Compton cross section data using the `Compton cross sections` input. Cross section data must exist in the `$HEN_HOUSE/data` directory and the file name must have the form `x_compton.data`, where `x` is a name specified by the user. All values of `x` will appear in the GUI menu where Compton cross section data can be selected. Alternatively, if editing the `.egsinp` file directly, the form of this input is:

`Compton cross sections= x`

Default Compton cross section data is contained in the file `compton_sigma.data` and is included with the EGSnrc system.

11.14 Radiative Compton corrections (radc_flag)

If set to `Radiative Compton corrections= On`, then radiative corrections for Compton scattering based on the equations of Brown and Feynman (Phys. Rev. 85, p 231–1952) are used. If set to `Off` (the default) no corrections are done. Note that if set to `On` then the variable `SOURCES` in the `sources.make` file for the accelerator (See Section 2.12.4 above) must be modified to include `$(EGS_SOURCEDIR)rad_compton1.mortran` just before `$(EGS_SOURCEDIR)egsnrc.mortran`.

11.15 Pair angular sampling (IPRDST)

This input determines the method used to sample the positron/electron emission angles (relative to the incoming photon) in a pair production event. There are three possible settings of this input: `Off`, `Simple` and `KM`. If it is set to `Off`, then the positron and electron created by pair production have fixed polar angles, $\theta_\pm$, given by $\theta_\pm = \frac{m}{E_\gamma}$, where m is the electron rest energy and E_γ is the energy of the original photon. This is similar to method used to determine positron/electron emission angles in the original version of EGS4. If `Pair angular sampling= KM`, then equation 3D-2003 in an article by Motz et al[42] is used to determine the positron/electron emission angles. This option is similar to the sampling technique used by the current version of EGS4/BEAM. Finally if `Pair angular sampling= Simple` (the default), then only the first term in the the Motz et al equation 3D-2003 is used. The `KM` option becomes less efficient with increasing accelerator energies and, moreover, involves assumptions that are questionable at low energy. For these reasons, the default setting is `Simple`.

11.16 Pair cross sections (pair_nrc)

The `Pair cross sections` input determines the cross-sections to use for pair production events. If set to `BH` (the default), then Bethe-Heitler cross sections are used. If set to `NRC`, then the NRC cross sections found in `$HEN_HOUSE/data/pair_nrc1.data` are used. The `NRC` setting is only of interest at low energies, where these cross-sections take into account assymmetry in the positron-electron energy distribution.

11.17 Photoelectron angular sampling (IPHTER)

The `Photoelectron angular sampling` input determines the sampling method used by EGSnrc to determine the angle of emission of photoelectrons. If `Photoelectron angular sampling= Off` (the default), then photoelectrons inherit the direction of the incident photon. If `Photoelectron angular sampling= On`, then Sauter's formula [43] is used to determine the angle of the photoelectron. Note that, in most applications, we have not observed any difference between the "Off" and "On" settings of this parameters. Also note that, strictly speaking, Sauter's formula is only valid for K-shell photo-absorption and is also derived from extreme relativistic approximations. Thus, if the user has a

better approach, they can insert it in the `$SELECT-PHOTOELECTRON-DIRECTION;` macro in `$HEN_HOUSE/egsnrc.macros`.

Similar to bound Compton scattering, photoelectron angular sampling can be turned on or off in selected regions (with the opposite setting everywhere else) by setting `Photoelectron angular sampling= On in regions` or `Photoelectron angular sampling= Off in regions` together with the inputs `PE sampling start region` and `PE sampling stop region` to define the region ranges for which photoelectron angular sampling is to be turned on or off.

11.18   Rayleigh scattering (IRAYLR)

This input determines whether Rayleigh (coherent) scattering is simulated or not. Note that this replaces the `IDORAY` input in the BEAM main inputs (see section 10.12). If `Rayleigh scattering= On`, then Rayleigh events are simulated using the total coherent cross-sections from Storm and Israel[44] and atomic form factors from Hubbell and Øverbø[45]. This data must be included in the PEGS4 material data set. If `Rayleigh scattering= Off` (the default), then Rayleigh events are not simulated. Rayleigh scattering is only recommended for low energy (< 1 MeV) simulations. Also, for proper simulation with Rayleigh events included, bound Compton scattering (see section 11.12 above) should also be turned on.

Rayleigh scattering can be turned on or off in selected regions (with the opposite setting everywhere else) using `Rayleigh scattering= On in regions` or `Rayleigh scattering= Off in regions` and the inputs `Relaxations start region` and `Relaxations stop region` to define the region ranges for turning Rayleigh scattering on or off.

EGSnrc also allows the user to specify custom Rayleigh form factors for specified media. To do this, the user must set `Rayleigh scattering= custom` and then specify the list of PEGS4 media in additional input `ff media names=` and the list of files containing custom form factors for each specified medium in the additional input `ff file names=` .

11.19   Atomic Relaxations (IEDGFL)

This input determines whether or not the relaxation of atoms to their ground state after Compton and photoelectric events is simulated. If `Atomic Relaxations= On`, then relaxation after Compton and photoelectric events is simulated via the emission of any combination of K-, L-, M- and N-shell fluorescent photons, Auger electrons and Coster-Kronig electrons. The lower energy limit for relaxation processes is 1 keV. Thus, only relaxation in shells with binding energy > 1 keV is simulated. If `Atomic Relaxations= Off` (the default), then atomic relaxations are not simulated. In this case, when there is a photoelectric event, EGSnrc transfers all of the photon energy to the photoelectron. This is different from EGS4/BEAM, where the binding energy of the electron is subtracted and deposited on the spot. Both approaches are approximations, but the EGSnrc approach is more accurate. `Atomic Relaxations= On` is only recommended for low energy applications. Note that the `Atomic Relaxations` option supersedes the `IFLUOR` option in EGS4/BEAM (see section 10.12), which only simulates emission of K-shell fluorescent photons after photoelectric events.

Similar to bound Compton, photoelectric angular sampling and Rayleigh scattering, atomic relaxations can be turned on/off in selected regions (with the opposite setting everywhere else) using `Atomic Relaxations= On in regions` or `Atomic Relaxations= Off in regions` and the inputs `Relaxations start region` and `Relaxations stop region` to define the region ranges for which relaxations are to be turned on/off.

11.20   Electron impact ionization (eii_flag)

This input determines what, if any, theory is used to simulate electron impact ionization. The possible values are "off" (the default), "on", "Casnati", "Kolbenstvedt", and "Gryzinski". When "on" is selected, Kawrakow's electron impact ionization theory[46] is used. For the other selections, the theory associated with the name given is used. See the EGSnrc Manual[1] for more details.

Since the details of electron impact ionization are only relevant at keV X-Ray energies, the default "off" setting should be used in most linac simulations.

11.21   Photon cross sections (photon_xsections)

This selects the photon interaction cross-sections to use in the simulation. Cross-sections included with the BEAMnrc/DOSXYZnrc distribution (and, thus, the possible settings of `photon_xsections` immediately after installation are): "Storm-Israel" (the default), "epdl" and "xcom". The Storm-Israel cross-sections are the standard PEGS4 cross-sections. The "epdl" setting will use cross-sections from the evaluated photon data library (EPDL) from Lawrence Livermore[47]. The "xcom" setting will use the XCOM photon cross-sections from Burger and Hubbell[48]. Note that the EGSnrc transport parameter input routine is coded in such a way that, if you are editing the `.egsinp` file directly instead of using the BEAMnrc GUI, the default Storm-Israel cross-sections can only be specified by leaving out the `Photon cross sections` input line altogether. This is taken care of automatically if you are using the GUI to set this parameter.

You can also use your own customized photon cross-section data. To do this, you must create the files `x_pair.data`, `x_photo.data`, `x_rayleigh.data` and `x_triplet.data` (where "x" is the name of your cross-section data) which contain cross-sections for pair production, photoelectric events, rayleigh scattering and triplet production, respectively. These files must be in your `$HEN_HOUSE/data` directory. Once these files are in place, then "x" will appear in the pull-down menu in the GUI where photon cross-sections are specified. Alternatively, if you are editing the `.egsinp` file directly, you can enter the line:

`Photon cross sections= x`

inside the block of EGSnrc transport parameter inputs.

11.22 Photon cross-sections output (xsec_out)

The input `Photon cross-sections output` can be set to `On` to output the file
`$EGS_HOME/BEAM_accelname/inputfile.xsections` which contains the photon cross section
data used in the simulation. Default is `Off`.

12 Custom user inputs

If custom input parameters are required (for any modifications that you have made to
`beamnrc.mortran`), then it is recommended that you include them in the `.egsinp` file be-
tween the delimiters: `:start user inputs:` and `:stop user inputs:`. This section, if
required, can appear either immediately before or immediately after the EGSnrc transport
parameters (see Section 11 above). You will not have access to custom inputs through the
GUI, but by putting them between the delimiters you ensure that they will not be "wiped
out" if you modify/save the input file using the GUI at some point.

The format of custom inputs will be similar to that for the EGSnrc transport parameters:
`PARAMETER NAME= parameter value`
Where `PARAMETER NAME` can either be the actual name of the input variable or a descriptive
text string. Note the space between the parameter value and the "=" sign.

The current release of BEAMnrc includes a a custom input for redefining the output
directory for phase space files, `PHSP_OUTDIR` (see Section phspoutdirsect for more information
about this). In general, however, it is up to the user to modify `beamnrc.mortran` to read in
their custom input parameters.

13 Parallel Processing with BEAMnrc

A BEAMnrc simulation can be split into smaller jobs which can then be run on different
processors in parallel to reduce the elapsed time required for a simulation. Parallel processing
requires that you be running in Unix/Linux and that you have a network queuing system
such as PBS or NQS.

In previous versions of BEAMnrc, the Unix script `pprocess` was used to automatically
create the individual input files for parallel jobs and submit them (also automatically setting
the random number seeds to a different values in each input file and the phase space
source inputs, `IPARALLEL` and `PARNUM`). This method of parallel processing had a major
limitation, though, in that each job consisted of the same number of histories, making the
total simulation time dependent on the slowest CPU.

In the current version of BEAMnrc, parallel processing is accomplished much more effi-
ciently using a built-in parallel processing functionality.

To submit a parallel job, use the batch submission command, `exb` (discussed in detail in
section 2.7.1). The command syntax for a parallel job is:

```
exb BEAM_myaccel inputfile pegsdata [short|medium|long] [batch=batch_system] p=N
```

where `N` is the number of parallel jobs to submit (see Section 2.7.1 for a detailed discussion of the other inputs).

Once this command is entered, the script `$HEN_HOUSE/scripts/run_user_code_batch` (to which `exb` is aliased) enters a loop which submits BEAM_myaccel to the batch queue `N` times. Each submission has the form:

```
BEAM_myaccel -i inputfile -p pegsdata -P n_parallel -j i_parallel
```

where `n_parallel=N` and `i_parallel` takes on values 1,2,..., `n_parallel`, depending on which job is being submitted. For each parallel job, BEAMnrc will create a temporary working directory (see section 2.9 for more on temporary working directories), and the output files from the `i_parallel`th run will have the naming scheme `inputfile_w[i_parallel].egslog`, `inputfile_w[i_parallel].egslst`, `inputfile_w[i_parallel].egsphsp1`, `inputfile_w[i_parallel].egsdat`, etc. Note that `N` different input files are not created, and that all parallel runs make use of the original input file.

On submission of the first job (`i_parallel=1`) BEAMnrc will create a file, `inputfile.lock`, in the `BEAM_myaccel` directory. The `.lock` file, or job control file, is accessed and updated by all parallel jobs and contains the number of histories remaining to be run, `n_left`, and the total number of jobs running, `n_job`, among other quantities.

Before beginning a run, a parallel job opens the `.lock` file and reads `n_left` (the file cannot be read by more than one job at a time). If `n_left>0`, then there are still histories to run, and the job begins execution. Rather than each job running NCASE/`n_parallel` histories (as in the old parallel processing scheme), each job runs only a fraction, or chunk, of this number at a time. Thus, the maximum number of histories that a job runs at a time is NCASE/(`n_parallel`*`$N_CHUNK`), where `$N_CHUNK` is defined in `$HEN_HOUSE/src/egsnrc.macros` and is set to a default value of 10. If `n_left` happens to be less than NCASE/(`n_parallel`*`$N_CHUNK`), then the job will run `n_left` histories. Before beginning the run, the job updates the `.lock` file, decrementing `n_left` by the number of histories it is about to run, and incrementing `n_job` if this is the first run for this job. After the job has finished its chunk of histories, it will loop back to the beginning of the simulation, read the contents of the `.lock` file again, and determine whether another chunk of histories needs to be run.

Doing parallel simulations in chunks like this prevents jobs that are running on slower CPU's from tying up large portions of the simulation and, hence, dominating the total elapsed time required.

If, on opening the `.lock` file, a parallel job finds that `n_left=0` (ie no more histories to run), then it immediately exits the simulation loop, analyzes the data from its runs for output to the `inputfile_w[i_parallel].egslst` file, moves all output files out of its temporary working directory, deletes this directory, and decrements `n_job`. If, after decrementing `n_job`, `n_job=0`, then this is the last job to stop running, and it automatically calls the EGSnrcMP subroutine `egs_combine_runs` to combine the results from all parallel jobs (see section 13.3 below).

13.1 Random Number Seeds with Parallel Jobs

It is important that each parallel job start with a different state of the random number generator, otherwise you will end up combining identical results at the end of a parallel run, compromising both the results and the uncertainty estimate on them. BEAMnrc avoids this problem by automatically incrementing the second random number seed, JXXIN (see section 10.8) for each parallel job. So for job number i_parallel:

$$JXXIN = JXXIN_{input} - 1 + i\_parallel \tag{2}$$

where $JXXIN_{input}$ is the value of JXXIN in inputfile.egsinp.

13.2 Phase Space Sources with Parallel Jobs

When running parallel jobs that use a single phase space source (see section 4.13), it is important that the entire source be evenly sampled over all jobs. In order to facilitate this, BEAMnrc divides the source into n_parallel*\$N_CHUNKS equal partitions. Each chunk of the run then uses a different partition of the source.

The number of particles in each source partition, p_per_phsp_chunk, is given by:

$$\texttt{p\_per\_phsp\_chunk} = \frac{\texttt{NNPHSP}}{(\texttt{n\_parallel} * \$\texttt{N\_CHUNKS})} \tag{3}$$

where NNPHSP is the total number of particles in the phase space source. Before a parallel job starts a chunk of histories, it calculates which chunk of the total histories it is about to run, n_run_chunk, using the equation:

$$\texttt{n\_run\_chunk} = \frac{(\texttt{NCASE} - \texttt{n\_left})}{\texttt{NCASE}} * (\texttt{n\_parallel} * \$\texttt{N\_CHUNKS}) \tag{4}$$

Recall that n_left is the total number of histories remaining AFTER this run begins. The partition of the phase space source used by this chunk of histories is then given by:

$$(\texttt{n\_run\_chunk} - 1) * \texttt{p\_per\_phsp\_chunk} + 1 \leq \texttt{INPHSP} \leq \texttt{n\_run\_chunk} * \texttt{p\_per\_phsp\_chunk} \tag{5}$$

where INPHSP is the particle number used.

Note that particles may be recycled within a partition, depending on the setting of NRCYCL (see section 4.13). Also, if all the particles in a partition are used up during a run, the partition is restarted at its first particle (ie INPHSP = (n_run_chunk-1)*p_per_phsp_chunk+1).

13.3 Combining Results from Parallel Jobs

The last parallel job to finish running automatically combines the results of all parallel runs by calling the EGSnrcMP subroutine egs_combine_runs. This subroutine loops through i_parallel=1,...,n_parallel and, for each value of i_parallel, calls the BEAMnrc subroutine combine_results. combine_results opens the file inputfile_w[i_parallel].egsdat,

reads the values of $\sum_{i=1}^{nhist}(\text{energy deposited})_i$, $\sum_{i=1}^{nhist}(\text{energy deposited})_i^2$, $\sum_{i=1}^{nhist}(\text{fluence})_i$ and $\sum_{i=1}^{nhist}(\text{fluence})_i^2$ (where `nhist` is the number of primary histories) in each dose or fluence scoring zone, and adds these values to their respective totals over all parallel jobs. Once the results from all parallel jobs have been summed, BEAMnrc then calculates the doses, fluence values and their uncertainties [17] for the combined run in the same way it would calculate the results of a single run.

The results of the analysis will be output to `inputfile.egslst`. It indicates which `inputfile_w[i_parallel].egsdat` have been added together followed by full output of the dose and fluence results.

Note that the `.egsdat` files are essential for combining parallel runs. Thus, if you are running in parallel, you must have the input variable `IDAT=0` so that the `.egsdat` files are output (see section 10.5, page 111 for more information about `IDAT`).

Parallel runs can be combined manually by re-running your accelerator with the input variable `IRESTART=4` (in the GUI, this is equivalent to setting the Run option to "analyze parallel jobs") after all parallel jobs have finished. Use of `IRESTART=4` is generally not necessary now that the last job automatically combines parallel results, however, it may be useful if, for some reason, all of the `.egsdat` files were not moved out of their temporary working directories or if you wish to add more `.egsdat` files from a separate group of parallel runs. For more information on `IRESTART` options, see section 10.3.

The process of combining parallel runs described above does not combine phase space files generated by the parallel jobs. These must be combined separately using the MORTRAN `addphsp` utility described in the next section.

13.4   Combining Phase Space Files from Parallel Runs using `addphsp`

Phase space files from parallel runs (with a naming scheme: `inputfile_w[i_parallel].egsphsp[scoringplanenumber]`) are not automatically combined at the end of the run. However, the MORTRAN utility, `addphsp`, can be used to combine phase space files after all parallel jobs are finished.

To use `addphsp`, simply go to the accelerator directory (ie where the phase space files are located) and type:

```
addphsp inputfile outputfile ipar [istart] [iscore] [i_iaea]
```

where `inputfile` is the name of the original input file (no `.egsinp` extension), `outputfile` is the name of the output file for the concatenated phase space data (a `.egsphsp[iscore]` extension is added automatically), `ipar` is the number of parallel jobs being added, `istart` is the job number at which adding begins (ie the first value of `i_parallel`)–defaults to 1, `iscore` is the scoring plane number for which you are combining phase space files (*i.e.* are you adding `.egsphsp1`, `.egsphsp2` or `.egsphsp3` files?)–defaults to 1, and `i_iaea` is set to 1 if you are adding IAEA-format phase space files–default is 0.

`addphsp` then concatenates the phase space files `inputfile_w[istart].egsphsp[iscore]`, `inputfile_w[istart+1].egsphsp[iscore]`, ...,`inputfile_w[istart+ipar-1].egsphsp[iscore]`. If you are missing some of the phase space files in the series `addphsp` will automatically skip

13.4 Combining Phase Space Files from Parallel Runs using `addphsp`

over those and just concatenate those that are available.

In the case of IAEA-format phase space files, the naming scheme is different, and `addphsp` concatenates the files `inputfile_w[istart].iscore.IAEAphsp`, `inputfile_w[istart+1].iscore.IAEAphsp`, ...,`inputfile_w[istart+ipar-1].iscore.IAEAphsp`. The corresponding `.IAEAheader` files (em i.e. `inputfile_w[istart].iscore.IAEAheader`, etc) are assumed to be in the same directory. Output will be to the files `outputfile.iscore.IAEAphsp` and `outputfile.iscore.IAEAheader`. IAEA functionality in `addphsp` requires that EGSnrc/BEAMnrc be installed on a machine with a working C++ compiler. See Section 7.3 for more information on the IAEA phase space format.

Note that `addphsp` requires twice the disk space of the total size of all phase space files being combined, since files are not automatically deleted after they are added.

`addphsp` makes use of the record length factor (`$RECL-FACTOR`) for 4-byte phase space data found in `$HEN_HOUSE/lib/config/machine.macros`, some of the phase space reading/writing macros contained in `$HEN_HOUSE/utils/phsp_macros.mortran` (see section 7, page 97), some of the IAEA-format reading/writing macros in `$HEN_HOUSE/utils/iaea_phsp_macros.mortran` (only if installed with a working C++ compiler), and some of the macros found in `$HEN_HOUSE/src/egsnrc.macros`.

Normally, `addphsp` is compiled as part of the BEAMnrc installation. However, you can compile it separately by going into `$OMEGA_HOME/progs/addphsp` and typing `make`. The executable, `addphsp*`, will be put into directory `$HEN_HOUSE/bin/config`.

13.5   Parallel Jobs Run from the GUI

You can start parallel jobs from the BEAMnrc GUI. Select "Run" from the "Execute" menu and the running window that opens up will allow you to select "run in parallel", together with the number of parallel jobs. Note that on selecting "run in parallel", the GUI automatically switches to "batch" mode if this has not already been selected. You can then select the queue ("short", "medium" or "long") that you would like to submit the parallel jobs to. The queuing system used defaults to PBS, unless specified otherwise by the `$EGS_BATCH_SYSTEM` environmental variable.

13.6   Restarting Parallel Jobs

Parallel runs can be restarted (by setting `IRESTART=1` in `inputfile.egsinp`) provided that you have the `.egsdat` files from all of the previous individual parallel jobs available in your `$EGS_HOME/BEAM_myaccel` directory. If you are using a phase space source, however, restarting presents a problem in that a particular partition of the source may get used by a different job the second time around. At the end of the run, results from the second use of this partition will be recombined with those from its first use with no attention paid to the correlation between the two results. This will result in the uncertainties being underestimated. We recommend that you do not restart a parallel run if you are using a phase space source.

14 Statistics in BEAMnrc

Starting with the February 2002 release of BEAMnrc, a history-by-history method of estimating uncertainties was implemented in BEAMnrc[17]. This method offers considerable improvements over the method of statistical batches used up until then.

The history-by-history method involves grouping scored quantities (eg fluence, energy deposited) according to primary history during a run and then determining the root mean square standard deviation on the mean of the groupings. For most sources, there is no difference between a primary history and an incident particle. However, for phase space sources, where more than one incident particle can be traced back to a single primary history, it is important to group according to primary history in order to account for correlations between the incident particles.

For more information, see the report on history by history statistics in BEAMnrc and DOSXYZnrc[17].

15 Component Modules

15.1 Introduction

One of the design features of BEAMnrc is that each part of the accelerator or source unit is considered to be a single component module which takes up an horizontal slab portion of the accelerator. These component modules are re-usable and are all completely independent. They communicate with the rest of the system in certain well specified ways. The purpose of this section is to list all component modules currently used at NRC, describe what each component module does and how it is used in BEAMnrc.

A component module can be considered as a block which has a 'front' surface and a 'back' surface. An accelerator is built with many such blocks. Very often there is gap between two blocks. This gap is automatically filled with AIR by the BEAMnrc main routine, which is consistent with the case of a real accelerator. The air gap which is in front of a module, and after the 'back' plane of the previous module, is considered as a part of this component module. However, we still define the 'front' surface of a module as the front plane of NON-AIR medium in the module.

15.2 What Each Component Module Does

SLABS
SLABS is used for multiple slabs of arbitrary thickness and material which are perpendicular to the z axis. The outer boundary is a square.

CONS3R
CONS3R is designed to simulate cylindrical structures that can be described using a series of (Z,R) points rotated about the Z axis. Examples of such structures are rings, cylindrical

slabs and cones (primary collimators). This CM has only 3 regions: the interior of the cylindrical structure, the outside of the structure, and (possibly) the air gap at the front). The current version can only allow convex shapes in the Z direction [i.e., Z(i+1) must be greater or equal to Z(i), see section 15.3.3 for more details].

CONESTAK
CONESTAK is used to simulate a stack of truncated cones. A primary collimator is a special case for CONESTAK using only one cone. Each layer is of user-defined thickness, and, within each layer, the user defines the radius of the cone at the top of the layer and at the bottom of the layer, as well as the medium inside the cone and outside the cone. In addition, the entire, multi-layered, conical structure may be surrounded by a cylindrical wall of user-defined medium. CONESTAK is rotationally symmetric about the beam axis.

FLATFILT
This is an even more general purpose CM to simulate a set of truncated stacked cones. It is necessary for some very complex flattening filter designs. Similar to CONS3R and CONESTAK, FLATFILT is rotationally symmetric about the beam axis.

CHAMBER
CHAMBER is used for parallel-plate ion chamber in the container with top and bottom planes of arbitrary thickness and material. CHAMBER can also be used to score the central axis dose in a water phantom outside an accelerator (see section 15.3.6 for more details). This is a cylindrical CM, rotationally symmetric about the beam axis.

JAWS
JAWS is used for sets of paired bars or jaws, which can be in the collimator or applicator. The user defines the angle of the inner faces of the jaws with respect to the Z (beam) axis. The jaws can open in either the X or Y direction. The bars are of arbitrary thickness and material. The outer boundary of JAWS is a square centered on the beam axis.

DYNJAWS
DYNJAWS is a version of JAWS in which the jaw settings (Z positions and opening coordinates) can be specified as changing over the course of a BEAMnrc simulation. The user can run DYNJAWS in "step-and-shoot" mode, which simulates jaw settings changing while the beam is off, and "dynamic" mode, which simulates jaw settings changing while the beam is on. When used in these modes, the jaw settings must be specified in a separate file, where each complete group of settings (ie Z positions and opening coordinates for all jaws) is defined as a "field", and the probability, or index, of each field is specified by the user. DYNJAWS is particularly useful for simulating dynamic wedges.

APPLICAT
APPLICAT is used for a set of rectangular scrapers. Each scraper is defined by the outer region of two concentric rectangles, the inner region being air. The scrapers are of arbitrary thickness, width and position relative to the reference plane ($Z = 0$). The scrapers can be of different materials. The outer boundary of APPLICAT has square symmetry centered on the beam axis. This is an extension of the CM APPSQ described in the BEAMnrc paper[3].

CIRCAPP
CIRCAPP is similar to APPLICAT, only the openings in the scrapers are circular. Each

scraper is defined by its rectangular outer boundary and circular inner boundary. The outer
boundary of CIRCAPP has square symmetry centered on the beam axis.

PYRAMIDS
PYRAMIDS is used to model a stacked set of truncated pyramids, often a rectangular
collimator or applicator. For each layer, the user defines the medium of the pyramid and
also of the layer out of which the pyramid is "carved". Layers must have a minimum air
gap in between them but need not extend all the way to the outer boundary. Similar to
APPLICAT and JAWS, the outer boundary of PYRAMIDS is a square centered on the beam
axis.

BLOCK
The component module BLOCK is used to model beam treatment blocks having non-
rectangular and/or multiple openings. BLOCK consists of a single layer of block material.
Openings in the material are comprised of subregions specified by the user. The inner sides
of the opening(s) all angle towards a single, user-specified "focus point" on the beam axis.
The outer boundary of BLOCK is a square centered on the beam axis.

MLC
MLC is used to model a double focused multi-leaf collimator with a flat face. The collimator
has a single layer composed of a user-specified number of leaves all opening in either the
X or Y direction. The collimator opening is composed of the openings of individual leaves
as specified by the user. MLC has two materials: the material in the collimator opening
(usually air), and the material of the leaves and collimator body. The outer boundary of
this CM is a square centered on the beam axis.

MLCQ
MLCQ is similar to MLC, however it has rounded leaf ends. The user specifies the radius
of the leaf ends. In addition, the leaf ends can be angled down or up by specifying the Z
position where the radius of the leaf ends originates.

VARMLC
VARMLC is similar to MLC and MLCQ (it can have either rounded or angled leaf ends).
The main difference between it and these earlier component modules is that VARMLC also
simulates the air gaps between leaves perpendicular to the leaf direction, the tongue-in-
groove which connects adjacent leaves, and the screws at the top and bottom which are used
to open and close leaves.

MLCE
MLCE was designed as a variation of VARMLC specifically for modeling Elekta MLC's.
The tongue-and-groove in VARMLC has been replaced by interlocking steps. Also, unlike
VARMLC, all leaves are identical and the sides of the leaves are focused (always to Z=0) by
tilting each leaf about an axis that runs down the centre of its top surface. The entire leaf
bank can also be tilted for off-axis focusing.

DYNVMLC
DYNVMLC, also based on VARMLC, was specifically designed to model the Varian Mille-
nium multi-leaf collimator. The user specifies the cross-sections of the 3 different leaf types
(FULL, TARGET and ISOCENTER leaves) in the collimator. These leaf cross-sections are
more complex than those in VARMLC. Then, the user assigns a leaf type to each leaf in the

leaf bank as well as the opening dimensions of the leaves. Instead of tongue-in-groove, the leaves fit together with interlocking steps. DYNVMLC allows the user to simulate different leaf opening coordinates during a single simulation, provided that the user supplies a file containing the leaf opening data. Leaf openings can be simulated to change while the beam is on (dynamic) or while the beam is off (step-and-shoot).

MESH

The MESH component module is used to model a single-layer wire mesh placed perpendicular to the beam direction in the path of the beam. The user specifies the X and Y dimensions of the rectangular "holes" in the mesh (all holes have the same dimensions) and the width of the wire between the holes. There is also the possibility of a region of air between the edge of the mesh and `RMAX_CM`. Air holes are placed in a regular pattern in the mesh, and the mesh has rectangular symmetry about the beam axis. The outer boundary of MESH is a square centered on the beam axis.

MIRROR

MIRROR is used for a mirror in the accelerator. It can have arbitrary angle with respect to the Z axis. The number of layers and their thicknesses, materials in the mirror can be arbitrary. The mirror is surrounded by air. The MIRROR outer boundary is a square centered on the beam axis.

XTUBE

XTUBE is used to simulate an x-ray source. It must be the first CM in a simulation. The target may consist of several flat layers of different materials backed by a target holder (backing material). The target angle is defined by the target surface and the z-axis. The incident circular or rectangular beam is from the side normal to the z-axis. XTUBE should always be used as the first CM and the second CM may be any of the available `CMs` with a central opening serving as the exit window of the x-ray tube. The square outer boundary of XTUBE is centered on the z-axis.

SIDETUBE

SIDETUBE models concentric cylinders parallel to the X-axis. The user specifies the X positions of the cylinder ends, the number of concentric cylinders, the radii and media of the cylinders, and the medium surrounding the cylinders. This is one of the 3 CM's in which an isotropic source (`ISOURC=3`) can be placed (CONESTAK and FLATFILT are the other ones) and is excellent for modelling isotropic radiating cylinders that are perpendicular to the beam (collimated) axis. Note that all cylinders are necessarily the same length (*i.e.* have the same X limits). The outer boundary of this CM is a square centred on the Z-axis.

ARCCHM

ARCCHM is designed to model an arc-shaped array of ion chambers such as those used in the prototype tomotherapy unit at the University of Wisconsin. However, this CM can be used to model any arc-shaped structure in the path of the beam. The user specifies the Z position and radius of the front of the arc. The arc is always concave up and curves in the Y direction (*i.e.* there is no concavity in the X direction), with it's lowest point at Y=0. The angle subtended by the arc is determined by the user-specified number of chambers, widths of the chambers and of the septa separating the chambers, and the thickness of the chambers/septa. The user also specifies the extent of the arc in the X direction.

15.3   Geometry and Input Parameters of Component Modules

15.3.1   Overview

For each of the component modules, a picture is provided here (*ZX* or *ZY* projection), showing its geometry and the input parameters needed to run BEAMnrc with this module. When creating input files using the BEAMnrc GUI, you may preview each CM with the geometry parameters that you have input. You may also refer to the in-line documentation in the source code of each component module for a clear input format, particularly for those input parameters other than geometrical parameters. The in-line documentation has been included in the next pages.

Each component module has a maximum radius or half-width, `RMAX_CM` beyond which the particles will not be followed during a simulation. This parameter is read in by the MAIN routine of BEAMnrc. In the input file to run BEAMnrc, dummy lines (filled with ********** in the beginning) are used to separate the different component modules.

If there is a gap between two component modules, as in the case of a real accelerator, this gap is filled automatically with air by the BEAMnrc main routine[1]. Therefore for each component module there is a possible air gap in front of it. In general, we consider the air gap is a part of the component module. The air gap, from the back plane of previous module to the front (non-air medium) plane of this module, is shown in the picture of a component module in this documentation.

The local region numbers (`IR`) are indicated in the picture of each component module. The code makes use of both the local region numbers and the global region numbers. The code has mappings from one to the other. For further information on these details see Appendix A.1 "Specifications for Component Modules for BEAMnrc".

The following subsections discuss the geometry, input parameters, and input format for each component module.

Except for the very first component module, usually an air gap exists in front of a module. We consider the air gap, which is in front of a module and after the previous module, is a part of this module. However we define the 'front' surface of a module is the front plane of non-air-gap medium in the module.

---

[1]Strictly it is with the material specified in the second record of the input file and nominally referred to as air

15.3.2 SLABS

SLABS is used for multiple planes of arbitrary thickness and material, i.e., a set of semi-infinite slabs. One single slab is a special case for SLABS. SLABS has square symmetry about the beam axis. Note that `ESAVE` can be set for each region in SLABS, unlike other CMs where `ESAVE_GLOBAL` is used. This is because SLABS is often used to model the bremsstrahlung target in photon accelerators (see section 6.1).
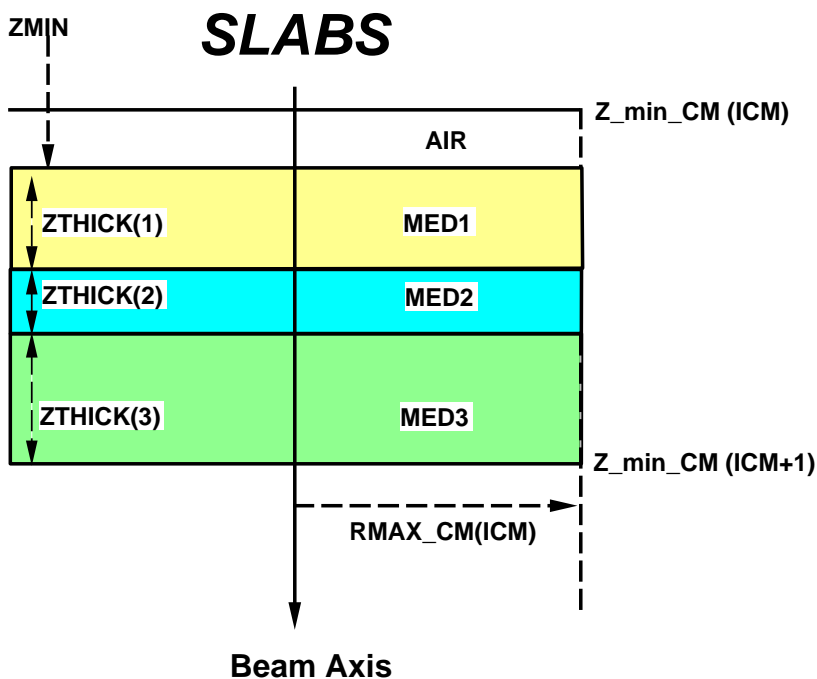
Figure 20: Schematic of CM SLABS, which in the example consists of three layers of different materials with thicknesses `ZTHICK(1)`, `ZTHICK(2)`, and `ZTHICK(3)`, respectively. The gap between the back of the previous CM and the front face of this CM is automatically filled with air. The outer boundary of this CM is a square of half-width `RMAX_CM`.

The input format for SLABS, and an example of the input file are given as follows.

```
CARDS CM_$SLABS   (SLABS: Rev 1.6)
**************

-1  dummy line (filled with ****)  read in main

 0  RMAX_CM(ICM_$SLABS)          outer boundary for CM - 1/2 side of
                                 square(read in main)

 1  TITLE_$SLABS (60A1):  Title of CM.

 2  N_$SLABS (I5):  Number of planar slabs in CM = # regions in CM,
                    excludes any air gap needed.

 3  ZMIN_$SLABS (F15.0):  Distance from front of first slab to reference
                          plane  (Z=0).

 4  Parameters of each slab from front to back (increasing Z).  One pair of
    cards (4a and 4b) for each of the slabs.

    4a   ZTHICK_$SLABS, ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT, ESAVEIN
                       (3F15.0,2I5,F15.0):

             ZTHICK_$SLABS:   slab thickness.
             ECUT, PCUT:      Cutoff energies-defaults are ECUTIN,PCUTIN
             DOSE_ZONE:       Dose zone to score dose - 0 if not scored
             IREGION_TO_BIT:  map this region to this bit number in LATCH
             ESAVEIN:         Value of ESAVE for this region if range
                              rejection on. Default is ESAVE_GLOBAL.

    4b   MED_IN (24A1):  Medium of the planar slab, used to set MED_INDEX.

 Example
 *******

The following set of cards defines a 1 cm thick slab of air sandwiched
between two 0.1 cm thick slabs of tungsten.  The front slab is at
Z=7.32 cm. Electrons will be followed in the slabs down to kinetic energies
of 10 keV (total energies of 0.521 MeV) and photons will be followed down
to energies of 1 keV.  The dose deposited in the air will be scored and
added to the dose from the other regions in dose zone 1, and the
dose deposited in both tungsten slabs will be scored and added to the dose
from the other regions in dose zone 2.  Particles interacting in
the first slab will be associated with BIT 1 in LATCH.  In all slabs,
ESAVEIN=0, thus ESAVE in each slab will default to ESAVE_GLOBAL.

10.0,                           RMAX_CM
Multiple slabs: 0.1cm W-1cm air-0.1cm W, ECUT=0.521, PCUT=0.001
```

```
3,                              N_SLABS
7.32,                           ZMIN_SLABS
0.1, 0.521, 0.001, 2,1,0.0,     ZTHICK_SLABS etc
W521ICRU
1., 0.521, 0.001, 1,0,0.0
AIR521ICRU
0.1, 0.521, 0.001, 2,0,0.0
W521ICRU
```

15.3.3   CONS3R

CONS3R is a stack of truncated cones coded as three regions. It can be used for any case if there is cylindrical symmetry and if there are only two radial regions. Examples are slab, ring, stack rings, cone (primary collimator),cone stack. The current version only allows convex shapes in the Z direction, not concave shapes (*i.e.* Z(I1)+ is greater than Z(i)). CONS3R is computationally more efficient than the other conical CMs when there is more than one layer (*i.e.* more than 3 node points). This efficiency comes because there are only 2 Z boundaries for particles to cross in CONS3R, no matter how many node points there are.
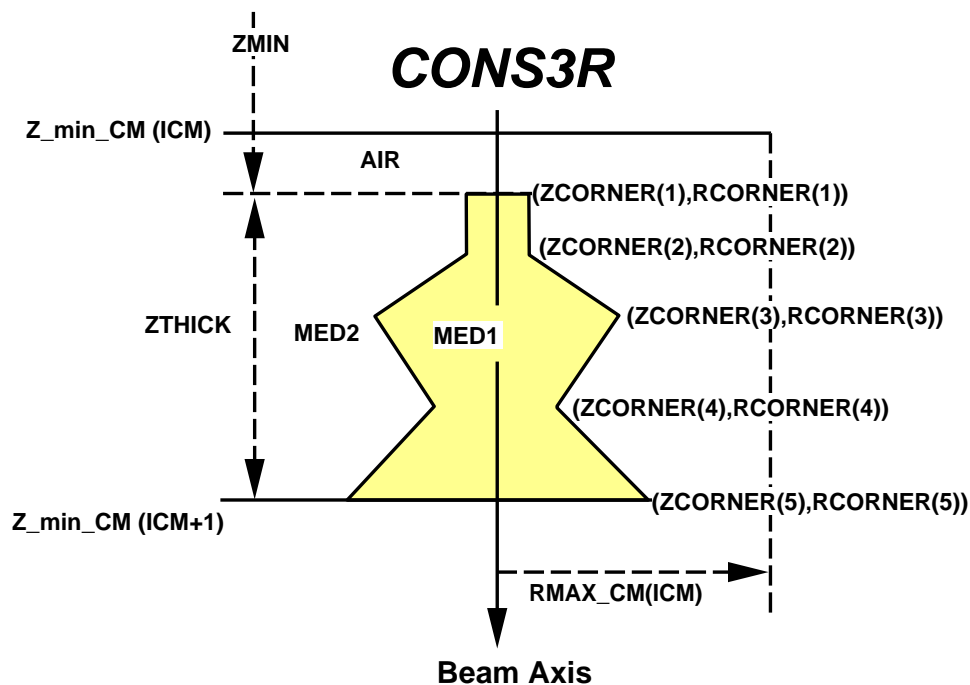
Figure 21: Component module CONS3R defined by 5 node points, where RCORNER values can be zero and RCORNER(i+1) can be smaller than RCORNER(i), but ZCORNER(i+1) must be greater than or equal to ZCORNER(i). Z_{MIN} is the distance from the reference plane (Z = 0) to the front plane of the CM (excluding the air gap). The beam axis is an axis of rotation and the outer boundary is a cylinder.

The input format for CONS3R, and an example of input file are given as follows.

```
CARDS CM_$CONS3R (CONS3R: Rev 1.10)
**************

 -1 Dummy line to indicate start of CM

 0  RMAX_CM(ICM_$CONS3R) (F10.0):  Outer radial boundary of CM (cm).

 1  TITLE_$CONS3R (60A1):  Title of CM.

 2  ZMIN_$CONS3R (F15.0):  Dist from front of cones to
                              reference plane (Z=0).

 3  ZTHICK_$CONS3R (F15.0): The thickness of cones (excludes front air).

 4  NUM_NODE_$CONS3R (I5):  The # of points to be used <$NPOINT_CONS3R.

 Repeat 5 for I=1,NUM_NODE_$CONS3R

 5  ZCORNER_$CONS3R(I), RCORNER_$CONS3R(I) (2F15.0):
       Positions (Z, R) for node I.  First & last must match
       ZMIN_$CONS3R and ZMIN_$CONS3R+ZTHICK_$CONS3R.  Note also the
       restriction Z(I+1) >= Z(I).

 Repeat 6, 7 for inner (ie inside cons3r), then outer region

 6  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT, IREJCTIN  (2F15.0,3I5):

        ECUT, PCUT:  Cutoff energies for electrons and photons.
        DOSE_ZONE:   If non-zero, dose in this zone is scored in this
                     dose zone
        IREGION_TO_BIT:   region to LATCH bit correspondence for particles
                     interacting in this region
        IREJCTIN:    If IREJCT_GLOBAL is on, then by setting IREJCT=-1
                     here, range rejection is turned off in this region
                     If left blank or zero, the global value is used.

 7  MED_IN (24A1):  Medium of region
                     used to set MED_INDEX.


    Example
    *******

The following input example describes a 1cm thick flat-top cone, having
a radius at the top of 0.8cm and a radius at the bottom of 1.2cm
sitting on a 0.3cm thick cylinder of radius 1.5cm which, in turn,
is sitting atop a flat-top cone of thickness 1.0cm with top radius
```

0.5cm and bottom radius=0.8cm. The two cones and cylinder are
made of H2O--note that all of these structures MUST be of the same
medium--and they are surrounded by AIR. Dose in the surrounding
AIR is stored in dose zone 1, and dose in the cones/cylinder
structure is stored in zone 2.

```
5.0
example cons3r
0.0
2.3
6
0.0, 0.8
1.0, 1.2
1.0, 1.5
1.3, 1.5
1.3, 0.5
2.3, 0.8
0.521, 0.01, 2, 0, 0
H2O
0.521, 0.01, 1, 0 ,0
AIR
```

15.3.4   CONESTAK

CONESTAK consists of a stack of coaxial truncated cones surrounded by a cylindrical wall. The vertices of the cones in each layer do not have to meet but the radii must not decrease as the depth increases. This CM can model scattering foils, primary collimators and many other components. CONESTAK has cylindrical symmetry.

CONESTAK

ZMIN

Z_min_CM (ICM)

AIR

RMIN(1)

ZTHICK(1)   MED1   MED2

RMAX(1)
RMIN(2)

ZTHICK(2)   MED4   MED5

RMAX(2)

RMIN(3)   MED3

MED7   MED8

ZTHICK(3)

RMAX(3)
RMIN(4)

MED10   MED11

ZTHICK(4)

RMAX(4)   Z_min_CM (ICM+1)

RBN

RMAX_CM(ICM)

Beam Axis
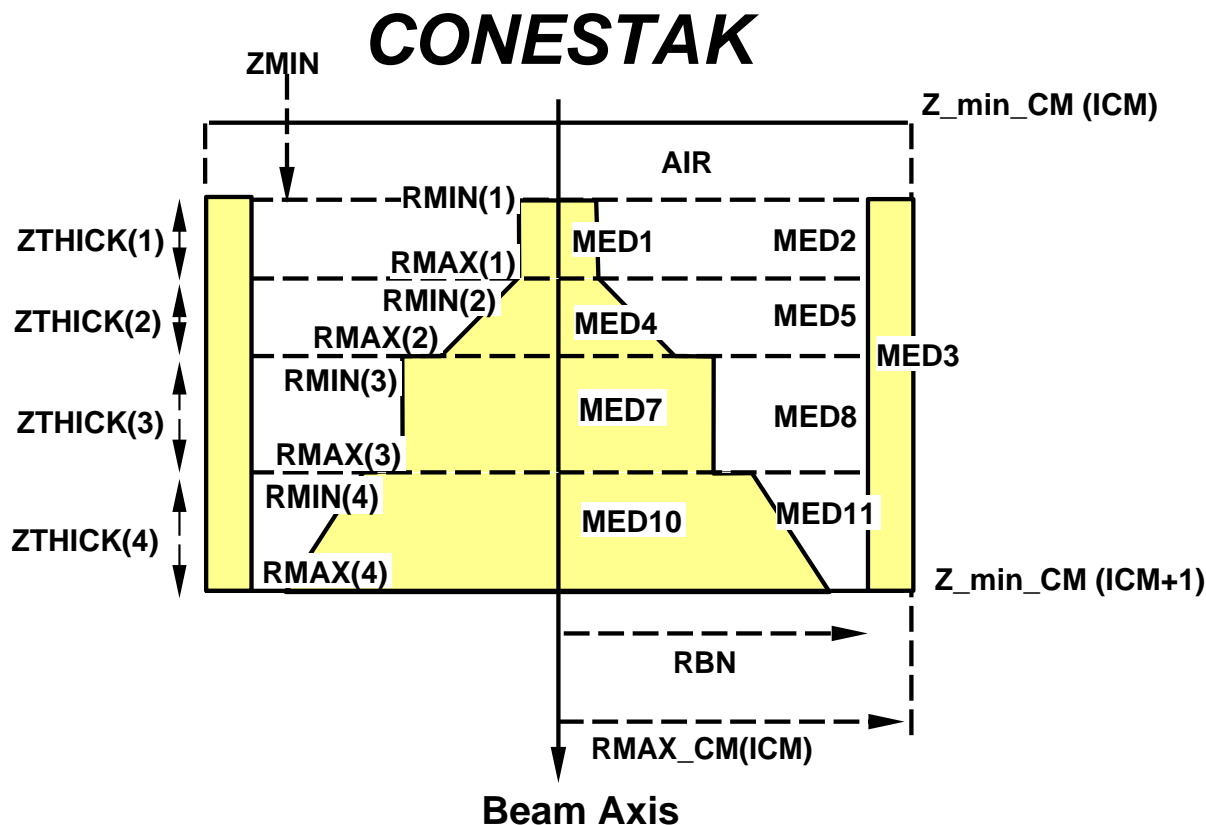
Figure 22: CONESTAK with 5 layers (`ISCM_MAX`=5). The outer wall has an inner radius defined by `RBN` and an outer radius equal to the outer boundary of the CM (`RMAX_CM`). Within each layer i, there is a cone with top and bottom radii defined by `RMIN(i)` and `RMAX(i)` respectively. Within each layer, there are three regions: the inner cone, the region between the cone and the outer wall (called the "outer cone"), and the outer wall. Regions 1,4,7,10,etc specify inner cones; regions 2,5,8,11,etc specify outer cones; and regions 3,6,9,etc specify the outer wall. If the user chooses to have an outer wall (by setting `RBN` > 0) then the medium of region 3 , `MED3`, is input first, even before `MED1` and `MED2`, and `MED3` is automatically applied to regions 6,9,etc, so that the outer wall has the same composition in every layer. If the user chooses not to have an outer wall (by setting `RBN` = 0), regions 3,6,9,etc shrink to zero. In specifying radii of cones, the following restrictions apply: `RMAX(i)` $\geq$ `RMIN(i)`, and `RMIN(i+1)` > `RMAX(i)`.

The input format for CONESTAK, and an example of input file are given as follows.

```
CARDS CM_$CONESTAK (CONESTAK Rev 1.8)
*********************************

-1  dummy line to indicate start of CM

 0  RMAX_CM(ICM_$CONESTAK) (F10.0): Outer radial boundary (cm).

 1  TITLE_$CONESTAK (60A1):  Title of CM.

 2  ZMIN_$CONESTAK, RBN_$CONESTAK (2F15.0):

        ZMIN_$CONESTAK:  Distance from front of first cone to
                         reference plane (Z=0),
        RBN_$CONESTAK:   Inner radius of outer wall
                         (Set to 0 if you do not want an outer wall)

 3  ISCM_MAX_$CONESTAK (I5): Number of conical layers

Repeat 4 once for I=1,ISCM_MAX_$CONESTAK

 4  ZTHICK_$CONESTAK(I), RMIN_$CONESTAK(I), RMAX_$CONESTAK(I) (3F15.0):

        ZTHICK_$CONESTAK(I):  Thickness of conical layer.
        RMIN_$CONESTAK(I):    Front radius of conical layer.
        RMAX_$CONESTAK(I):    Back radius of conical layer.
                         Note restrictions:
                         RMAX_$CONESTAK(I)>=RMIN_$CONESTAK(I)
                         RMIN_$CONESTAK(I+1)>=RMAX_$CONESTAK(I)

 5 and 6 are only required if there is an outer wall (ie RBN_$CONESTAK~=0)

 5  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in local region 3 (outer wall):
                                              (2F15.0,2I5)

        ECUT, PCUT:  Cutoff energies for electrons and photons.
        DOSE_ZONE:   Dose scoring flag, non-zero to score dose
                     deposited in it
        IREGION_TO_BIT:   Bit setting number for the region

 6  MED_IN (24A1):  Medium of local region 3
                    used to set MED_INDEX.

Repeat 7-10 for each conical layer

 7  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in local region 1 (inside cone):
                                              (2F15.0,2I5)
```

```
        ECUT, PCUT:  Cutoff energies for electrons and photons.
        DOSE_ZONE:   Dose scoring flag, 0 to score dose deposited in it
        IREGION_TO_BIT:   Bit setting number for the region

 8  MED_IN (24A1):  Medium of local region 1 (inside cone),
                    used to set MED_INDEX

 9  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in local region 2 (outside cone):
                                          (2F15.0,2I5)

        ECUT, PCUT:  Cutoff energies for electrons and photons.
        DOSE_ZONE:   Dose scoring flag, 0 to score dose deposited in it
        IREGION_TO_BIT:   Bit setting number for the region

 10  MED_IN (24A1):  Medium of local region 2 (outside of cone),
                     used to set MED_INDEX
 Example
 *******
```

The following input example describes two conical layers. The first
layer is a flat-top cone 1.0cm thick, with a radius at the top of 0.8cm
and a radius at the base of 1.2cm. The second layer is a cylinder, also
1.0cm thick, of radius 1.2cm. The top cone is made of Cu and the bottom
cylinder is made of Pb. The entire structure is encircled by a Pb wall
with inner radius 4cm and outer radius 5cm.
In both layers, the medium between the cone and the outer wall is H2O.

Dose in the Cu cone will be scored in dose zone 1. Dose in the PB cylinder
will appear in dose zone 2. The dose to the encircling PB wall will be in
zone 3. And the dose to the water in both layers will be scored in zone 4.
ECUT and PCUT in all cases is 0.521MeV and 0.01MeV respectively.

```
5.0                      RMAX_CM
cone and cylinder surrounded by PB wall
0.0, 4.0
2
1.0, 0.8, 1.2
1.0, 1.2, 1.2
0.521, 0.01, 3, 0
PB
0.521, 0.01, 1, 0
CU
0.521, 0.01, 4, 0
H2O
0.521, 0.01, 2, 0
PB
0.521, 0.01, 4, 0
H2O
```

15.3.5 FLATFILT

FLATFILT consists of a stacked set of truncated coaxial cones with an arbitrary number of cones on each level. The material in the cones need not be the same. Both the number of cones and the radii of cones in each layer can be specified independently. This CM can be used to model very complex beam flattening filters for photon beam simulations, including those interior to a conical collimator.
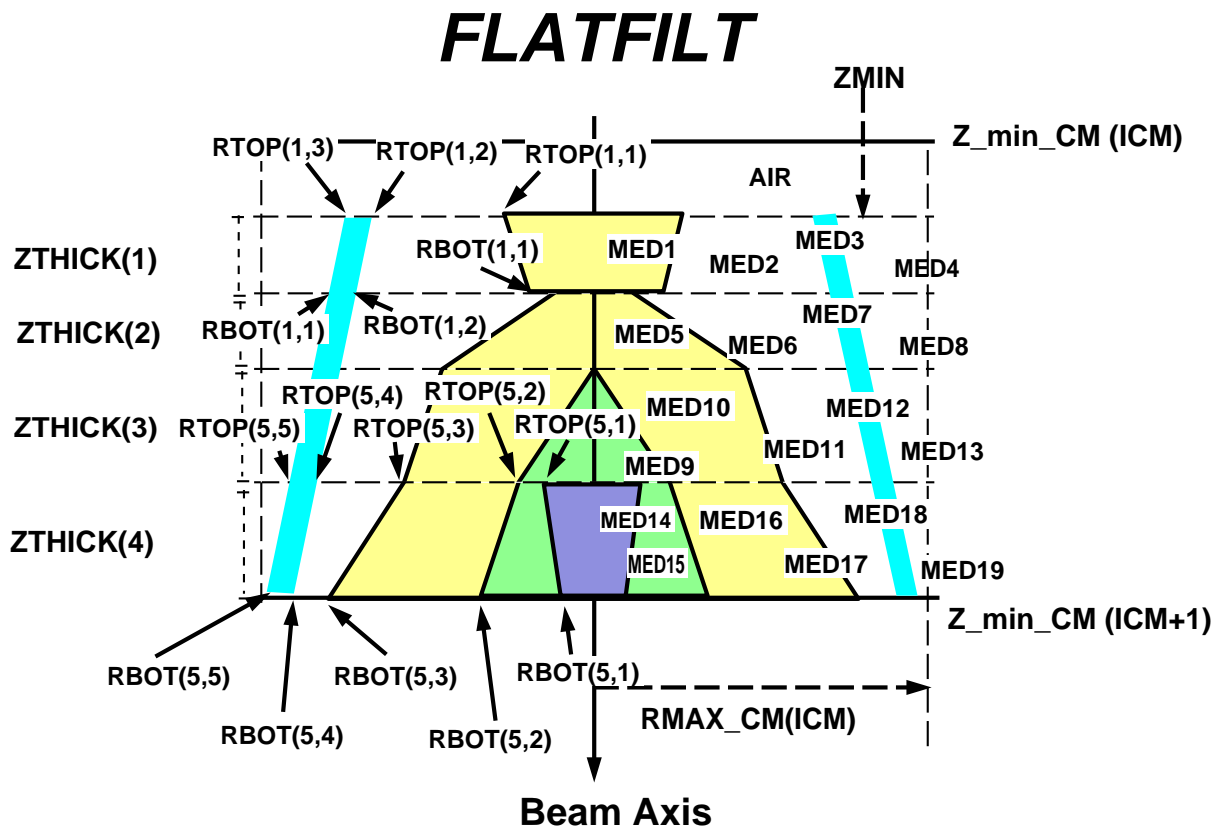
Figure 23: FLATFILT with 4 layers (ISCM_NO=4). Unlike CONESTAK, each layer can have an arbitrary number of cones. The number of cones in layer i is specified by ISSCM_NO(i). Within a layer i, each cone j is specified by its top and bottom radius, RTOP(i,j) and RBOT(i,j) respectively. Within a layer, cones cannot cross; so RTOP(i,j) < RTOP(i,j+1) and RBOT(i,j) < RBOT(i,j+1). For each layer i, ISSCM_NO(i) + 1 media must be specified. ISSCM_NO(i) media are required for the cones, and the extra medium is required for the region between the outermost cone and RMAX_CM. In the figure above, the 4 regions between outermost cones and RMAX_CM are composed of MED4, MED8, MED13 and MED19.

```
CARDS CM_$FLATFILT
******************
```

-1 Dummy line to indicate start of CM.

 0 RMAX_CM(ICM_$FLATFILT) (F10.0): Radius of outer boundary of CM (cm).

 1 TITLE_$FLATFILT (60A1): Title of CM.

 2 ZMIN_$FLATFILT (F10.0): Distance from front of CM (front of
     the first layer) to reference plane (Z=0),
     not including air gap.

 3 ISCM_NO_$FLATFILT (I5): Number of layers.

Repeat 4-6 for I=1,ISCM_NO_$FLATFILT

 4 ISSCM_NO_$FLATFILT(I), ZTHICK_$FLATFILT(I) (I5,F15.0):

           ISSCM_NO_$FLATFILT(I): # cones in layer I(ex outer region).
           ZTHICK_$FLATFILT(I):   Thickness of layer I.

Repeat 5 for J=1,ISSCM_NO_$FLATFILT(I) all on one line in order
of increasing cone radius.

 5 RTOP_$FLATFILT(I,J) (F15.0):
           Top radius of cone J in layer I.
           Note restriction: RTOP_$FLATFILT(I,J+1)>RTOP_$FLATFILT(I,J)

Repeat 6 for J=1,ISSCM_NO_$FLATFILT(I) all on one line in order
of increasing cone radius.

 6 RBOT_$FLATFILT(I,J) (F15.0):
           Bottom radius of cone J in layer I.
           Note restriction: RBOT_$FLATFILT(I,J+1)>RBOT_$FLATFILT(I,J)

Repeat 7 and 8 for J=1,ISSCM_NO_$FLATFILT(I)+1 for every layer I.
When J=ISSCM_NO_$FLATFILT(I)+1, you are specifying ECUT, PCUT,
MED_IN, etc. for the region between the outermost cone and RMAX_CM
in layer I.

 7 ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5):

        ECUT, PCUT: Cutoff energies for electrons and photons
                      in cone J.
         DOSE_ZONE: Dose scoring flag for cone J.
     IREGION_TO_BIT: Bit setting for region defined by cone J.

 8 MED_IN (24A1): Medium of cone J

```
                          used to set MED_INDEX.

  Example
  *******

The following example describes a FLATFILT with 2 layers.  The
RMAX of this FLATFILT is 2cm.  There is no air gap between the
first layer and the top of the CM.

The first layer is 0.3 cm thick and comprises a convex cone made
of H2O within a concave cone made of PB.  The convex H2O cone has
a top radius of 0.0cm and a bottom radius of 1.0cm.  The concave PB
cone has a top radius of 1.5cm and a bottom radius of 1.1cm.
The region between the outer, concave cone and RMAX_CM is AIR.
The second layer is also 0.3 cm thick and comprises a single
cylinder of H2O having radius 1cm (ie top radius=bottom
radius=1cm).  The region between the outer boundary of this
cylinder and RMAX_CM (an annular region) is AIR.

The dose to the AIR regions is scored in dose zone 1.  The dose
to H2O regions (the convex cone in the first layer and the cylinder
in the second layer) is scored in zone 2.  And the dose to PB
(the concave cone in the first layer) is scored in zone 3.
ECUT and PCUT for all regions are 0.521 MeV and 0.01 MeV
respectively.

2.00000,                RMAX_CM
flatfilt
0.0,                    ZMIN of first layer
2,                      no. of layers
2,  0.30,               no. of cones in first layer and thickness
0.0,1.5,                top radii of cones
1.0,1.1,                bottom radii of cones
1, 0.3                  no. of cones in second layer and thickness
1.0                     top radius of cone
1.0                     bottom radius of cone
0.521, 0.01, 2, 0,      ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT
H2O                     MEDIUM -- for layer 1, cone 1
0.521, 0.01, 3, 0
PB                             -- for layer 1, cone 2
0.521, 0.01, 1, 0
AIR                            -- for region between cone 2 and RMAX
0.521, 0.01, 2, 0
H2O                            -- for layer 2, cone 1
0.521, 0.01, 1, 0
AIR                            -- for region between cone 1 and RMAX
```

15.3.6 CHAMBER

CHAMBER models a parallel-plate ion chamber in a container with top and bottom planes of arbitrary thickness and material. CHAMBER also models a combination of scattering foils, cylindrical collimators and an ion chamber, *etc.*. This CM is also useful for central-axis depth-dose calculations and for analysis of dose components due to particles coming from different parts of an accelerator. CHAMBER has cylindrical symmetry.
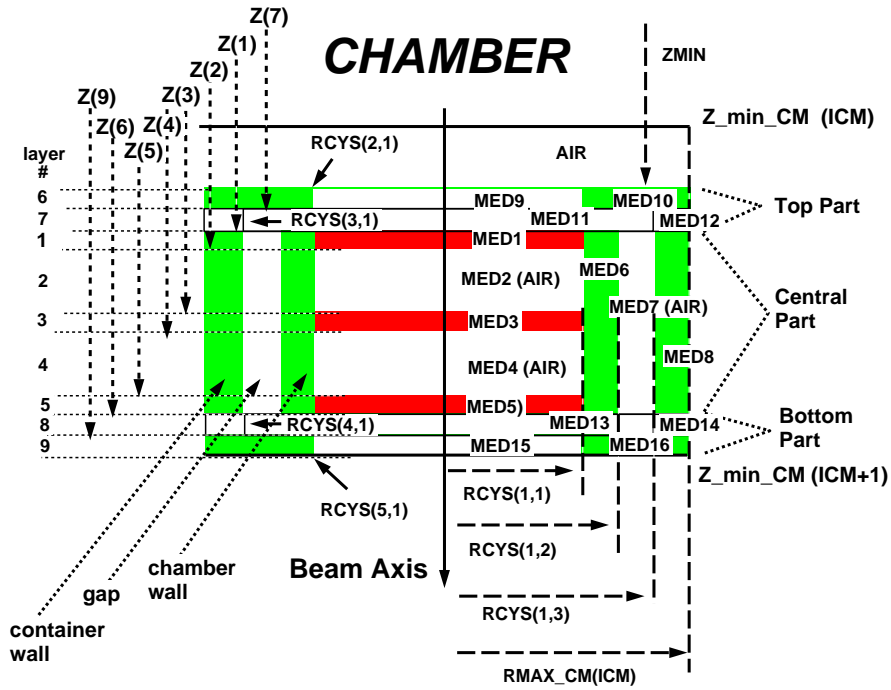
Figure 24: CHAMBER component module for an ionization chamber or any symmetric cylindrical-planar geometries. The CM shown consists of three parts: top, central(chamber), and bottom. The top and bottom parts each have 2 layers (N_TOP = N_BOT=2), and the chamber part has 5 layers (N_CHM=5). In general, the central part is required while the top and bottom parts are not. Each layer in both top and bottom parts is divided into an inner cylinder and an outer annulus. For a layer i in the top part, the outer radius of the cylinder (inner radius of the annulus) is given by RCYS(i+1,1), and the outer radius of the annulus is RMAX_CM. For a layer j in the bottom part, the outer radius of the cylinder is given by RCYS(j+N_TOP+1,1), and outer radius of the annulus is RMAX_CM. Within the top or bottom part, layers can have different RCYS and media; however, input for the top or bottom part is simplified if all layers within that part have the same RCYS, the same medium in the inner cylinders, and the same medium in the outer annuli. In the central part, all layers have the same outer radius, RCYS(1,1), which also defines the inner radius of the chamber wall. The outer radius of the chamber wall is given by RCYS(1,2), which, in turn, is the inner radius of the gap between the chamber wall and the container wall. The gap has outer radius RCYS(1,3), which is also the inner radius of the container wall. The container wall has outer radius RMAX_CM.
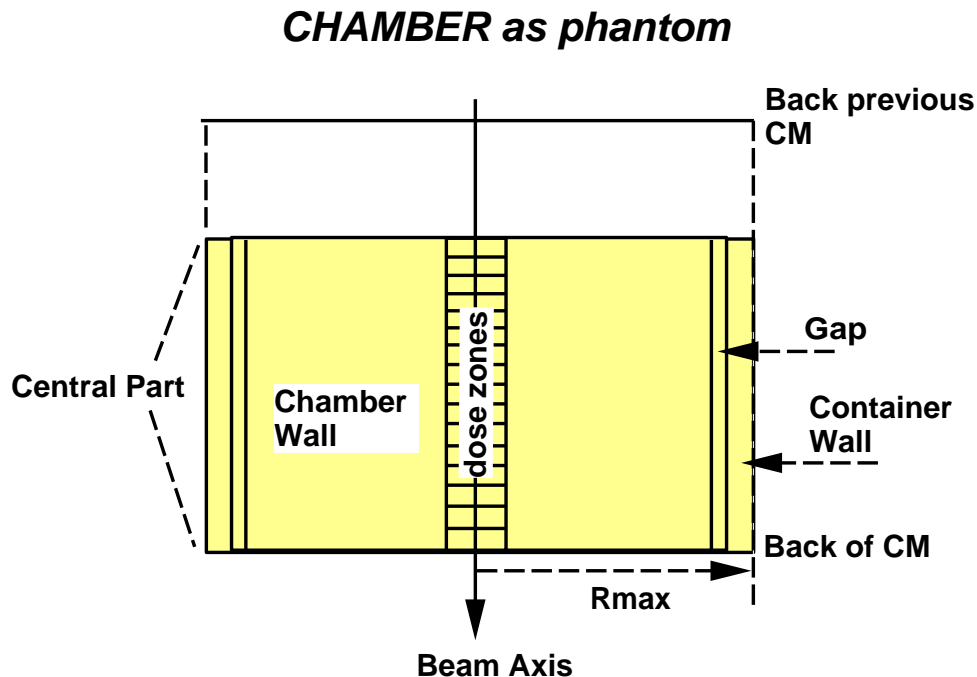
CHAMBER as phantom

Figure 25: CHAMBER as a depth-dose phantom. Only the central (chamber) part of the CM is used. The layers within this part become the depth-dose zones, and, hence, all layers are composed of the same material (the phantom material). Depth resolution is determined by the number of layers. If all layers have the same thickness, then the input can be made more efficient by inputting the number of layers on the same line as the layer thickness, ZTHICK. Then you only have to input ECUT, PCUT, DOSE_ZONE and MED once for all layers. If DOSE_ZONE >0, the input value will be used for the first layer and then DOSE_ZONE will be incremented by 1 automatically for each subsequent layer. Alternatively, if there are N separate groups of layers, where the layers in each group have equal thickness, you can set N_CHM_$CHAMBER = -N. For each of the N groups of layers, you must specify the layer thickness and the number of layers in the group. Following this, you only have to specify ECUT, PCUT, DOSE_ZONE and MED once for all layers, similar to the case where ALL layers are of equal thickness. Note that all depth-dose zones have the same radius, determined by RCYS(1,1). A small RCYS(1,1) determines the depth-dose on the beam axis, while a larger RCYS(1,1) includes dose deposited further away from the beam axis. If the phantom extends beyond the radius of the dose zones, then the chamber wall, gap, and container wall can be composed of the same phantom material. This set-up can be very efficient for central axis depth-dose calculations when used in conjunction with range rejection because the large volume in the chamber wall leads to effective range rejection. However, it has been observed for electron beams that the lack of z boundaries in the wall region can lead to small step size effects. This effect can be minimised by decreasing ESTEPE (see section 11.4).

```
   CARDS CM_$CHAMBER
   ****************

   -1  Dummy line to indicate start of CM.

    0  RMAX_CM(ICM_$CHAMBER):( F10.0):   Maximum radius of component module

    1  TITLE_$CHAMBER (60A1):  Title of CM.

    2  ZMIN_$CHAMBER (F15.0): Distance from front surface of 1st cylinder to
                              reference plane (Z=0). Excludes any air gap.

    3  N_TOP_$CHAMBER, N_CHM_$CHAMBER, N_BOT_$CHAMBER (3I5)

          N_TOP_$CHAMBER:    Number of layers in top part (>= 0).
          N_CHM_$CHAMBER:    Number of layers in chamber itself (> 0 to input
                             chamber layers individually or if ALL layers have
                             the same thickness and medium; < 0 to
                             input -N_CHM_$CHAMBER groups of layers where
                             layers in each group have the same thickness and
                             ALL layers have the same MED).
          N_BOT_$CHAMBER:    Number of layers in bottom part (>= 0).
   =========================================================================
    4  Inputs for the top part (If N_TOP_$CHAMBER >0):
   =========================================================================

    If all layers in this part are identical, then in line (a) include
    NFLAG=N_TOP_$CHAMBER, otherwise repeat (a) to (e) for each of the
    layers.

     a) ZTHICK, RCYS_$CHAMBER , NFLAG (2F15.0,I5)
           ZTHICK (F15.0):         Thickness of each layer in top part
           RCYS_$CHAMBER (F15.0): Radius of inner cylinders in each layer
           N_TOP_$CHAMBER (I5):   Number of layers in top part

     b) ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT for inner cylinders
                                       (2F15.0,2I5,1-line):
        ECUT, PCUT: Cutoff energies for electrons and photons.
        DOSE_ZONE:  Dose scoring region for this region, 0=>no dose scored.
        IREGION_TO_BIT :  Bit # in LATCH designated to this region

     c) MED_IN (24A1):  Medium of inner cylinder (used for MED_INDEX)

     d) ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT for outer annuli
                                       (2F15.0,2I5,1-line):

     e) MED_IN (24A1):  Medium for outer annuli (used for MED_INDEX)


     =========================================================================
```

```
5  Inputs for the chamber/phantom part:
=========================================================================


 The chamber/phantom part has a central part of potentially many layers
 which may have different media and dimensions.  Outside this there
 are 3 cylindrical shells, called the chamber wall, gap, and container
 wall.  Each is a single material running the entire Z-span of the
 central part.


5.1) RCYS_$CHAMBER(1,1), RCYS_$CHAMBER(1,2), RCYS_$CHAMBER(1,3) (3F15.0)
        RCYS_$CHAMBER (1,1): Inner r of chamber wall=outer r central region
        RCYS_$CHAMBER (1,2): Outer r of chamber wall=inner r of gap
        RCYS_$CHAMBER (1,3): Inner r of container wall=outer r of gap



5.2) If N_CHM_$CHAMBER>0: If all layers in this part are identical, then
          in line (a) include NFLAG=N_CHM_$CHAMBER and input (b) once for
          all layers, otherwise repeat (a) to (c) for each of the layers.
       If N_CHM_$CHAMBER<0: Repeat (a) once for each of the
          -N_CHM_$CHAMBER groups of layers of equal thickness. In this case,
          NFLAG is the number of layers in the group.   Then input (b) once
          for all layers.

  a) ZTHICK, NFLAG  (F15.0,I5)
        ZTHICK:    Thickness of each layer in chamber part
                   (N_CHM_$CHAMBER>0) or of each layer in this particular
                   group of layers (N_CHM_$CHAMBER<0)
        NFLAG:     Number of layers in chamber IF all same
                   (N_CHM_$CHAMBER>0) or number of layers in the group
                   (N_CHM_$CHAMBER<0)


  b) ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT for chamber layers
                                        (2F15.0,2I5,one line):
        ECUT, PCUT: Cutoff energies for electrons and photons.
        DOSE_ZONE:  if all layers are of equal thickness or there are
                    groups of layers of equal thickness (ie
                    NFLAG=N_CHM_$CHAMBER or N_CHM_$CHAMBER<0) then, if
                    DOSE_ZONE>0 the dose is scored in regions
                    DOSE_ZONE, DOSE_ZONE+1,.., DOSE_ZONE+N_CHM_$CHAMBER-1
                    ie, a sequence of dose scoring zones are set up
                    automatically for all layers.
                    For single region at a time
                    Dose scoring region for this region,0=>no dose scored.
        IREGION_TO_BIT :  Bit # in LATCH designated to this region

  c) MED_IN (24A1):  Medium of chamber layers (used to set MED_INDEX)


  5.3) Inputs for the chamber wall:
```

```
      ---------------------------------

   a) ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT, (2F15.0,2I5):
        ECUT, PCUT: Cutoff energies for electrons and photons.
        DOSE_ZONE:  Dose scoring region for this region,0=>no dose scored.
        IREGION_TO_BIT:  Bit # in LATCH designated to this region

   b) MED_IN (24A1):  Medium of chamber wall (used to set MED_INDEX)

   5.4) Inputs for the gap between chamber wall and container wall:
   --------------------------------------------------------------

   a) ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT, (2F15.0,2I5):

   b) MED_IN (24A1):  Medium of gap (used to set MED_INDEX)


   5.5) Inputs for the container wall:
   -------------------------------------------

   a) ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT, (2F15.0,2I5):

   b) MED_IN (24A1):  Medium of container wall (used to set MED_INDEX)
=======================================================================
6  Inputs for the bottom part (If N_BOT_$CHAMBER >0):
=======================================================================

 5.6) If all layers in this part are identical, then in line (a) include
      NFLAG=N_BOT_$CHAMBER, otherwise repeat (a) to (e) for each of the
      layers.

   a) ZTHICK, RCYS_$CHAMBER , NFLAG (2F15.0,I5)
        ZTHICK:            Thickness of each layer in bottom part
        RCYS_$CHAMBER:     Radius of inner cylinders in bottom part

   b) ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT for inner cylinders
                                     (2F15.0,2I5,1-line):
        ECUT, PCUT:  Cutoff energies for electrons and photons.
        DOSE_ZONE:   Dose scoring flag, 0=>do not score dose.
        IREGION_TO_BIT :  Bit # in LATCH designated to this region

   c) MED_IN (24A1):  Medium of inner cylinders (used for MED_INDEX)

   d) ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT for outer annuli
                                     (2F15.0,2I5,1-line):
   e) MED_IN (24A1):  Medium of outer annuli (used for MED_INDEX)


   =====================================================================
7  Inputs for range rejection options:
```

```
    ===================================================================

        MRNGE (I5)          0 or 1

            MRNGE      : = 1 to estimate thickness of the CHAMBER for
                            ECUTRR calculations in automated range rejection
                            (IREJCT_GLOBAL=1) (crude approx for 5 layers)
                         = 0 no ECUTRR calculation--range rejection will
                            still be done on a region-by-region basis

                Note that MRNGE only has an effect if automated range
                rejection is on (IREJCT_GLOBAL=1).

  Example
  *******


The following set of cards defines a chamber with 2 top layers, 3 chamber
layers, and 2 bottom layers.
The chamber wall is AL & the chamber container is CU. The detecting
material is air.
The air cavities are assigned as dose region 1 and the rest as region 2.


10.5; radius of CM
Chamber with 2 top layers, 3 chamber layers, 2 bottom layers
10.0; distance from front surface of the CM to the reference plane (z=0)
2,3,2;                      2 top layers, 3 chamber layers, 2 bottom layers
0.1,5.0,0;         first layer in the top part, 0.1cm thick, IR=5cm
0.521,0.010,2,2;  dose region # = 2
CU              ;  medium
0.521,0.010,2,2;
CU
0.2,5.0,0;          second layer is 0.2 cm thick, radius = 5.0 cm
0.521,0.010,2,2;  for inner cylinder (dose region # = 2)
AL
0.521,0.010,2,2;  for outer annulus
AL
5.0,5.2,10.0;     IR & OR of chamber wall, IR of container
0.2;              thickness of the first layer (air) in chamber part
0.521,0.010,1,2;  dose region # = 1
AIR
0.1;              thickness of the second layer (AL) in chamber part
0.521,0.010,2,2;  dose region # = 2
AL
0.2;              thickness of the third layer (air) in chamber part
0.521,0.010,1,2;  dose region # = 1
AIR
0.521,0.010,2,2;  chamber wall (dose region # = 2)
AL
0.521,0.010,2,2;  air gap betweem chamber wall and container wall
```

```
AIR
0.521,0.010,2,2;  chamber container
CU
0.1,5.0,2;        2 layers in bot. part have = thickness, IR
0.521,0.010,2,2;  for inner cylinders
AL
0.521,0.010,2,2;  for outer annuli
AL
0;                do not calculate ECUTRR
```

15.3.7   JAWS

JAWS is used for a set of paired bars, which can be bars in the collimator or applicator. The bars are of arbitrary thickness and material, and X or Y orientation. The outer boundary of JAWS has square symmetry but the jaws themselves can be very asymmetric. The input for JAWS can be tedious but the GUI has a feature which will automatically set the X and Y values to give an arbitrary on-axis rectangular photon beam field size at a given SSD based on tracing back to an arbitrary focus located on the central axis.
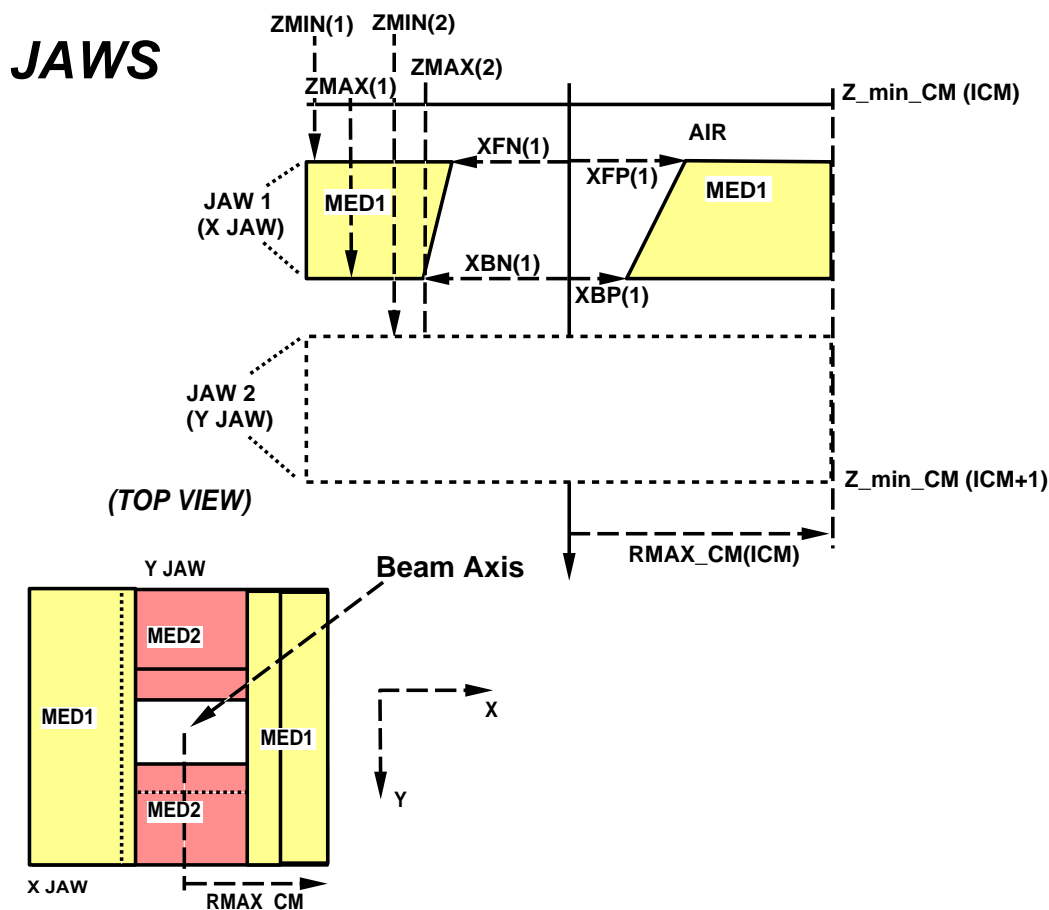
Figure 26: JAWS component module with paired X and Y jaws (`ISCM_MAX=2`). The cutaway view cuts right through the opening in the Y jaws, hence the representation of the Y jaws as a dashed rectangle. For a set of jaws i, the front opening of the jaws is specified by `XFP(i)` and `XFN(i)`. The back opening is specified by `XBP(i)` and `XBN(i)`. These coordinates are not shown for the Y jaws, however they are evident in the top view. The inside jaw surfaces are 2 planes, one connecting `XFP(i)` to `XBP(i)` and the other connecting `XFN(i)` to `XBN(i)` X jaws and openings between X jaws extend to $\pm$ `RMAX_CM` in the Y-direction; Y jaws and openings extend to $\pm$ `RMAX_CM` in the X direction. There must be a gap of at least 0.01 cm between the first set of jaws and the top of the CM (*i.e.* `ZMIN(1)-ZMIN_CM` $\geq$ 0.01) and between sets of jaws (*i.e.* `ZMIN(i)-ZMAX(i-1)` $\geq$ 0.01). Materials in each layer can be different but opposite jaws must be the same material.

The input format for JAWS, and an example of input file are given as follows.

```
CARDS CM_$JAWS(JAWS: Rev 1.8)
**************

-1  dummy line read in main used to separate input for CMs

 0  RMAX_CM(ICM_JAWS) (F10.0):
                        Perpendicular distance from Z-axis to boundary
                        surrounding component module.  This component
                        module has a square boundary.

 1  TITLE_$JAWS (60A1):  Title of CM.

 2  ISCM_MAX_$JAWS (I5):  Number of paired bars or jaws in CM.

Repeat 3 and 4 for I=1,ISCM_MAX_$JAWS

 3  XY_CHOICE (A1):  indicate orientation of the paired bars/jaws
                      X means bars/jaws perpendicular to x axis
                      i.e. separation and movement is along x-axis

 4  ZMIN_$JAWS(I), ZMAX_$JAWS(I), XFP_$JAWS(I), XBP_$JAWS(I),
    XFN_$JAWS(I), XBN_$JAWS(I) (6F15.0)

      ZMIN_$JAWS(I): Distance front of bars/jaws to reference plane.
      ZMAX_$JAWS(I): Distance back of bars/jaws to reference plane.
      XFP_$JAWS(I):  positive bar/jaw x or y coodinate at front.
      XBP_$JAWS(I):  positive bar/jaw x or y coodinate at back.
      XFN_$JAWS(I):  negative bar/jaw x or y coodinate at front.
      XBN_$JAWS(I):  negative bar/jaw x or y coodinate at back.

 5   ECUT, PCUT, DOSE_ZONE, IREGION_to_BIT (2F15.0,2I5): for interior
                                                   (assumed to be AIR)

        ECUT, PCUT:     Cutoff energies for electrons and photons.
        DOSE_ZONE:      Dose scoring zone of air surrounding bars.
        IREGION_TO_BIT: This region associated with this bit in LATCH

Repeat 6 and 7 for I=1,ISCM_MAX_$JAWS

 6   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5)

        ECUT, PCUT:     Cutoff energies for electrons and photons in
                         jaw I.
        DOSE_ZONE:      Dose scoring zone for jaw I.
        IREGION_TO_BIT: Both bars of jaw I associated with this bit.

 7   MED_IN (24A1):  Medium of jaw I, used to set MED_INDEX.
```

```
  Example
  *******
```

The following set of cards defines a pair of 5 cm-thick Al jaws.
The first set of bars open along the X axis. The inside faces of
this jaw are vertical at |X|=3cm. The second set of bars open along
the Y axis. The inside
faces of this jaw are angled out slightly, beginning at |Y|=3cm at the top
of the jaw and ending at |Y|=3.05cm at the bottom of the jaw.
The first jaw starts at Z=30.0 cm. Note the 0.01 cm airgap between
the two jaws.

Electrons will be followed in the CM down to kinetic energies of
10 keV (total energies of 0.521 MeV) and photons will be followed down
to energies of 1 keV. The dose deposited in the air will be scored and
added to the dose deposited in the bars in dose zone 1.

```
15.0    RMAX_CM
JAWS: 2 Al jaws, 5cm thick
2
X
30.0, 35.0, 3.0, 3.0, -3.0, -3.0
Y
35.01, 40.01, 3.0, 3.05, -3.0, -3.05
0.0, 0.0, 1, 0
0.0, 0.0, 1, 0
AL
0.0, 0.0, 1, 0
AL
```

15.3.8 DYNJAWS

DYNJAWS is a version of JAWS (see Section 15.3.7 above) which allows the user to specify changing jaw settings (Z positions and opening coordinates) over the course of a simulation. There are two dynamic modes: "step-and-shoot" (MODE_$DYNJAWS=2) and "dynamic" (MODE_$DYNJAWS=1). If being run in one of these modes, then the user must specify the jaw settings in a separate file. The format of this file is:

```
TITLE
NFIELD
FOR I=1,NFIELD[
INDEX(I)
FOR J=1,ISCM_MAX[
ZMIN(J,I),ZMAX(J,I),XFP(J,I),XBP(J,I),XFN(J,I),XBN(J,I)
]
]
```

where:

TITLE is a title line for the file.

NFIELD = the number of "sets" of jaw settings, or fields.

INDEX(I) = the index of field I, where 0≤INDEX(I)≤1 and INDEX(I)>INDEX(I-1).

ISCM_MAX = the number of jaws.

ZMIN(J,I) = the Z position of the front of jaw J for field I.

ZMAX(J,I) = the Z position of the back of jaw J for field I.

XFP(J,I) = the front X or Y coordinate (depending on jaw orientation) of the positive bar of jaw J for field I.

XBP(J,I) = the back X or Y coordinate of the positive bar of jaw J for field I.

XFN(J,I) = the front X or Y coordinate of the negative bar of jaw J for field I.

XBN(J,I) = the back X or Y coordinate of the negative bar of jaw J for field I.

The parameter, INDEX(I), is used to determine which field is used for a given primary history simulated. At the beginning of each primary history a random number, RND, on [0,1] is chosen and compared to the INDEX(I). The field no. used is the lowest value of I for which RND≤INDEX(I). If the user has selected "step-and-shoot" mode, then the jaw settings for field I are simply used. If the user has selected "dynamic" mode, then the jaw settings are a linear interpolation between field I-1 and I based on the value of RND. For example, the Z position of the front of jaw J, ZMIN(J), would be given by:

$$\mathtt{ZMIN}\,(\mathtt{J}) = \mathtt{ZMIN}\,(\mathtt{J},\mathtt{I}-1) + \frac{[\mathtt{ZMIN}\,(\mathtt{J},\mathtt{I}) - \mathtt{ZMIN}\,(\mathtt{J},\mathtt{I}-1)]}{[\mathtt{INDEX}\,(\mathtt{I}) - \mathtt{INDEX}\,(\mathtt{I}-1)]} \times [\mathtt{RND} - \mathtt{INDEX}\,(\mathtt{I}-1)] \quad (6)$$

In this way, the "dynamic" mode simulates jaw motion while the beam is on.

Note that the user can also select "static" mode, in which the inputs are identical to JAWS and jaw settings are fixed.

An example file containing the jaw settings for 3 fields for a single jaw are contained in the file `$OMEGA_HOME/beamnrc/CMs/dynjaw.opening.file`.

The input format for DYNJAWS, and an example of an input are given as follows.

```
CARDS CM_$DYNJAWS(JAWS: Rev 1.8)
**************

-1  dummy line read in main used to separate input for CMs

 0  RMAX_CM(ICM_JAWS) (F10.0):
                       Perpendicular distance from Z-axis to boundary
                       surrounding component module.  This component
                       module has a square boundary.

 1  TITLE_$DYNJAWS (60A1):  Title of CM.

 2  ISCM_MAX_$DYNJAWS, MODE_$DYNJAWS (2I5)

        ISCM_MAX_$DYNJAWS = Number of paired bars or jaws in CM.
           MODE_$DYNJAWS = 0 for static settings of jaw openings
                           1 for dynamic settings with simulated
                             jaw movement while beam is on
                           2 for step-and-shoot jaw movement--beam
                             off while jaw settings are changed

 Repeat 3 (if MODE_$DYNJAWS=1,2) or 3 and 4 (if MODE_$DYNJAWS=0)
            for I=1,ISCM_MAX_$DYNJAWS

 3  XY_CHOICE (A1):  indicate orientation of the paired bars/jaws
                      X means bars/jaws perpendicular to x axis
                      i.e. separation and movement is along x-axis

 Next input is only required if MODE_$DYNJAWS=0 (static)

 4  ZMIN_$DYNJAWS(I), ZMAX_$DYNJAWS(I), (XFP_$DYNJAWS(I), XBP_$DYNJAWS(I),
    XFN_$DYNJAWS(I), XBN_$DYNJAWS(I)) (6F15.0)

     ZMIN_$DYNJAWS(I): Distance front of bars/jaws to reference plane.
     ZMAX_$DYNJAWS(I): Distance back of bars/jaws to reference plane.
     XFP_$DYNJAWS(I):  positive bar/jaw x or y coodinate at front.
     XBP_$DYNJAWS(I):  positive bar/jaw x or y coodinate at back.
     XFN_$DYNJAWS(I):  negative bar/jaw x or y coodinate at front.
     XBN_$DYNJAWS(I):  negative bar/jaw x or y coodinate at back.
```

Next input is only required if MODE_$DYNJAWS=1 or 2

4a jaws_file (A80)

    jaws_file: The full name of a file containing jaw opening data
              in the following format:

```
NFIELDS_$DYNJAWS (I10)
FOR J=1,NFIELDS_$DYNJAWS[
 INDEX_$DYNJAWS(J) (F15.0)
 (ZMIN_$DYNJAWS(I),ZMAX_$DYNJAWS(I),XFP_$DYNJAWS(I),XBP_$DYNJAWS(I),
  XFN_$DYNJAWS(I),XBN_$DYNJAWS(I), I=1,ISCM_MAX_$DYNJAWS)
]
```

where:

NFIELDS_$DYNJAWS: Total number of jaw settings.
INDEX_$DYNJAWS(J): Index of setting J. 0 <= INDEX_$DYNVMLC(J) <= 1
                     and INDEX_$DYNVMLC(J) > INDEX_$DYNVMLC(J-1). This
                     number is compared to a random number on [0,1] at
                     the start of each history; if the random number is
                     <= INDEX_$DYNVMLC(J), then, if MODE_$DYNJAWS=2,
                     settings J are used. If MODE_$DYNJAWS=1, then
                     the settings used are a linear interpolation
                     between fields J-1 and J based on the random
                     number selected. See manual for more details.
ZMIN_$DYNJAWS(I),ZMAX_$DYNJAWS(I),XFP_$DYNJAWS(I),XBP_$DYNJAWS(I),
XFN_$DYNJAWS(I),XBN_$DYNJAWS(I):
                     See description of input 4. These are defined
                     for each field, J.

5   ECUT, PCUT, DOSE_ZONE, IREGION_to_BIT (2F15.0,2I5): for interior
                                          (assumed to be AIR)

      ECUT, PCUT: Cutoff energies for electrons and photons.
      DOSE_ZONE: Dose scoring zone of air surrounding bars.
      IREGION_TO_BIT: This region associated with this bit in LATCH

Repeat 6 and 7 for I=1,ISCM_MAX_$DYNJAWS

6   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5)

      ECUT, PCUT: Cutoff energies for electrons and photons in
                   jaw I.
      DOSE_ZONE: Dose scoring zone for jaw I.
      IREGION_TO_BIT: Both bars of jaw I associated with this bit.

7   MED_IN (24A1): Medium of jaw I, used to set MED_INDEX.

```
  Example
  *******


The following input defines a single tungsten jaw oriented in the Y
direction operating in step-and-shoot (MODE_$DYNJAWS=2) mode.
The jaw settings (Z position, opening coordinates) are defined
in the file $OMEGA_HOME/beamnrc/CMs/dynjaw.opening.file
(included with the distribution).


40.0, RMAX
CM #7: jaws set for a 10x10cm field at SSD=100 cm
1,2, # PAIRED BARS OR JAWS, MODE
Y
$OMEGA_HOME/beamnrc/CMs/dynjaw.opening.file
0.7, 0.01, 0, 21,
0.7, 0.01, 0, 13,
W700ICRU
```

15.3.9   APPLICAT

APPLICAT is used for a set of rectangular scrapers. Each scraper is defined by the outer region of two concentric rectangles, the inner region being air. The scrapers are of arbitrary thickness, width in both X and Y directions, and position relative to the reference plane ($Z = 0$). The scrapers can be of different materials. This CM may be used for modelling the square applicator found in electron beams although it does not allow for a bevelled edge. The outer boundary of APPLICAT is a square centered on the beam axis.
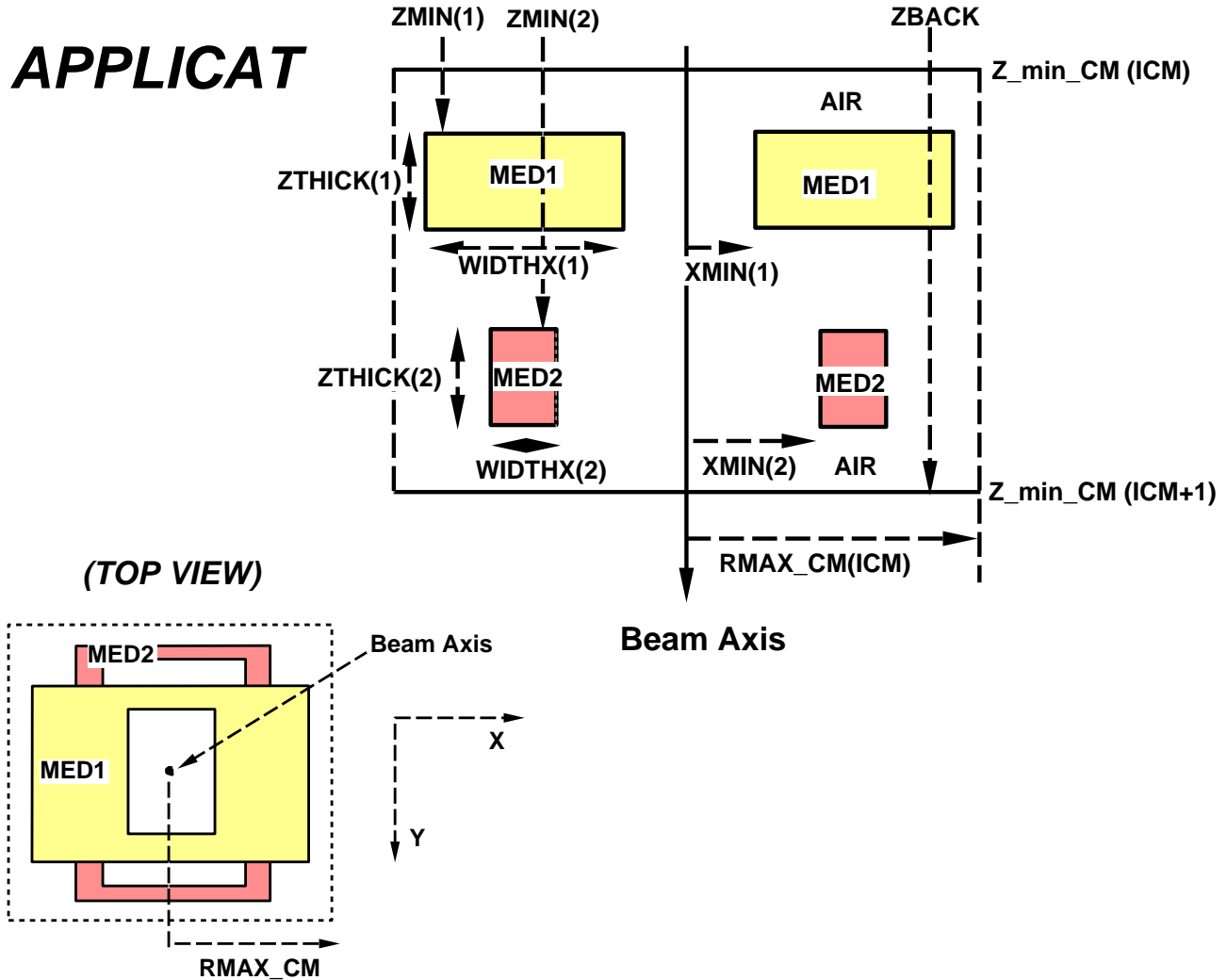
Figure 27: An APPLICAT with 2 scrapers (N=2). Each scraper i has a rectangular opening in the centre with half-width in the X-direction XMIN(i) and half-width in the Y-direction YMIN(i) . The width of scraper i in the X-direction is given by WIDTHX(i) and in the Y-direction by WIDTHY(i). The cross section does not show the Y dimensions of the scrapers, however the Y dimensions are evident in the top view. The minimum space between two scraper is 0.01 cm of air. There is also a minimum 0.01 cm air gap at the front (ie between Z_min_CM(1) and ZMIN(1)) and back (*i.e.* between ZBACK and the back of the last scraper) of the CM.

The input format for APPLICAT, and an example of input file are given as follows.

```
CARDS CM_APPLICAT
*****************

-1  Dummy line to indicate start of CM.

 0  RMAX_CM(ICM_$APPLICAT) (F10.0): Half-width of outer boundary of CM (cm).

 1  TITLE_$APPLICAT (60A1):  Title of CM.

 2  ZBACK_$APPLICAT (F15.0): Z of back face of the CM
                             (air will be added if necessary below the
                             last applicator)

    Note that there is always an air gap (thickness =
    AIRGAPMIN) in the front and the back of this CM.
    Therefore ZBACK_$APPLICAT should be >= Z of the back face of the
    last scraper + AIRGAPMIN.

 3  N_$APPLICAT, ISHAPE (2I5):
      N_$APPLICAT:  Number of scrapers in the CM.
      ISHAPE:       Index of applicators' shape, default to square, 1 for
                    rectangle.

Repeat 4 for I=1,N_$APPLICAT.

 4  ZMIN_$APPLICAT(I), ZTHICK_$APPLICAT(I), XMIN_$APPLICAT(I),
    YMIN_$APPLICAT(I),
    WIDTHX_$APPLICAT(I), WIDTHY_$APPLICAT(I),
    DOSE_ZONE,IREGION_TO_BIT (6F15.0,2I4):

      ZMIN_$APPLICAT(I):   Z of front face of scraper I.
                           Note that ZMIN_$APPLICAT(1)-
                           Z_min_CM must be >= AIRGAPMIN.
      ZTHICK_$APPLICAT(I): Thickness of scraper I.  Note that
                           ZMIN_$APPLICAT(I+1)-(ZMIN_$APPLICAT(I)+
                           ZTHICK_$APPLICAT(I))
                           must be >= AIRGAPMIN.
      XMIN_$APPLICAT(I):   (ISHAPE=1)  Half-width of inner opening in
                                       x in scraper I.
                           (ISHAPE~=1) Half-width of inner opening in x
                                       and y in scraper I.
      YMIN_$APPLICAT(I):   (ISHAPE=1)  Half-width of inner opening in y in
                                       scraper I.
                           (ISHAPE~=1) Not required.
      WIDTHX_$APPLICAT(I): (ISHAPE=1)  Width of bar in x (ie material
                                       surrounding inner opening) of
                                       scraper I.
```

```
                              (ISHAPE~=1) Width of bar in x and y for
                                          scraper I.
        WIDTHY_$APPLICAT(I): (ISHAPE=1)  Width of bar in y (ie material
                                         surrounding inner opening) of
                                         scraper I.
                              (ISHAPE~=1) Not required.
        DOSE_ZONE:            Dose scoring zone for the scraper bar.
        IREGION_TO_BIT:       Bit setting number for the scraper bar.

    Note restrictions to allow air gaps between scrapers and
    before the first scraper:
     ZMIN_$APPLICAT(1)-Z_min_CM >= AIRGAPMIN
     ZMIN_$APPLICAT(I+1)-(ZMIN_$APPLICAT(I)+ZTHICK_$APPLICAT(I)) >=
     AIRGAPMIN

 6   ECUT, PCUT, DOSE_ZONE_AIR,IREGION_TO_BIT_AIR (2F15.0,2I5):

       ECUT, PCUT:      Cutoff energies for electrons and photons for
                        both the bars and the surrounding (air) region
       DOSE_ZONE_AIR:   Dose scoring zone for the surrounding region
   IREGION_TO_BIT_AIR:  Bit set number for the surrounding (air) region

 Repeat 7 for I=1,N_$APPLICAT.

 7   MED_IN (24A1):  Medium of the bar of scraper I, used to
                     set MED_INDEX.

 Example
 *******

The following set of cards defines an applicator consisting of 2
0.2cm-thick Al scrapers.  The scrapers are separated by 8cm of air.
Scrapers can be thought of as made of 4 bars arranged in a rectangle
orthogonal to the Z axis.  For both scrapers in this example, the half-
width of the openings created by the bars is 2cm in x, 4cm in y, and the
width of the bars themselves is 1cm in x, 1.5cm in y.  The front scraper
starts at Z=60.5 cm.

Electrons will be followed in the CM down to kinetic energies of
10 keV (total energies of 0.521 MeV) and photons will be followed down
to energies of 1 keV.  The dose deposited in the air will be scored and
added to the dose from the other regions in dose scoring zone 1, and the
dose deposited in both scrapers will be scored and added to the dose
from the other regions in dose scoring zone 2.  There is a minimum 0.1 cm
air gap at the front and back of the scrapers CM so that the applicator
bars are completely surrounded by air.

10.0,    RMAX_CM
Applicators: 0.2cm Al at 60.5cm and 68.7cm, ECUT=0.521, PCUT=0.01
```

```
100.0,    extended air to Z=100 cm
2, 1,     two rectangular applicators
60.5, 0.2, 2.0, 4.0, 1.0, 1.5, 2,3
68.7, 0.2, 2.0, 4.0, 1.0, 1.5, 2,2
0.521, 0.01, 1, 0
AL521ICRU
AL521ICRU
```

15.3.10   CIRCAPP

CIRCAPP is similar to APPLICAT, however, it is used to model scrapers that have circular openings. The scrapers retain their rectangular outer edges, but the opening is now defined as a circle concentric with the rectangle defining the outer edges. Similar to APPLICAT, CIRCAPP does not allow bevelled edges. The outer boundary of CIRCAPP is a square centered on the beam axis.
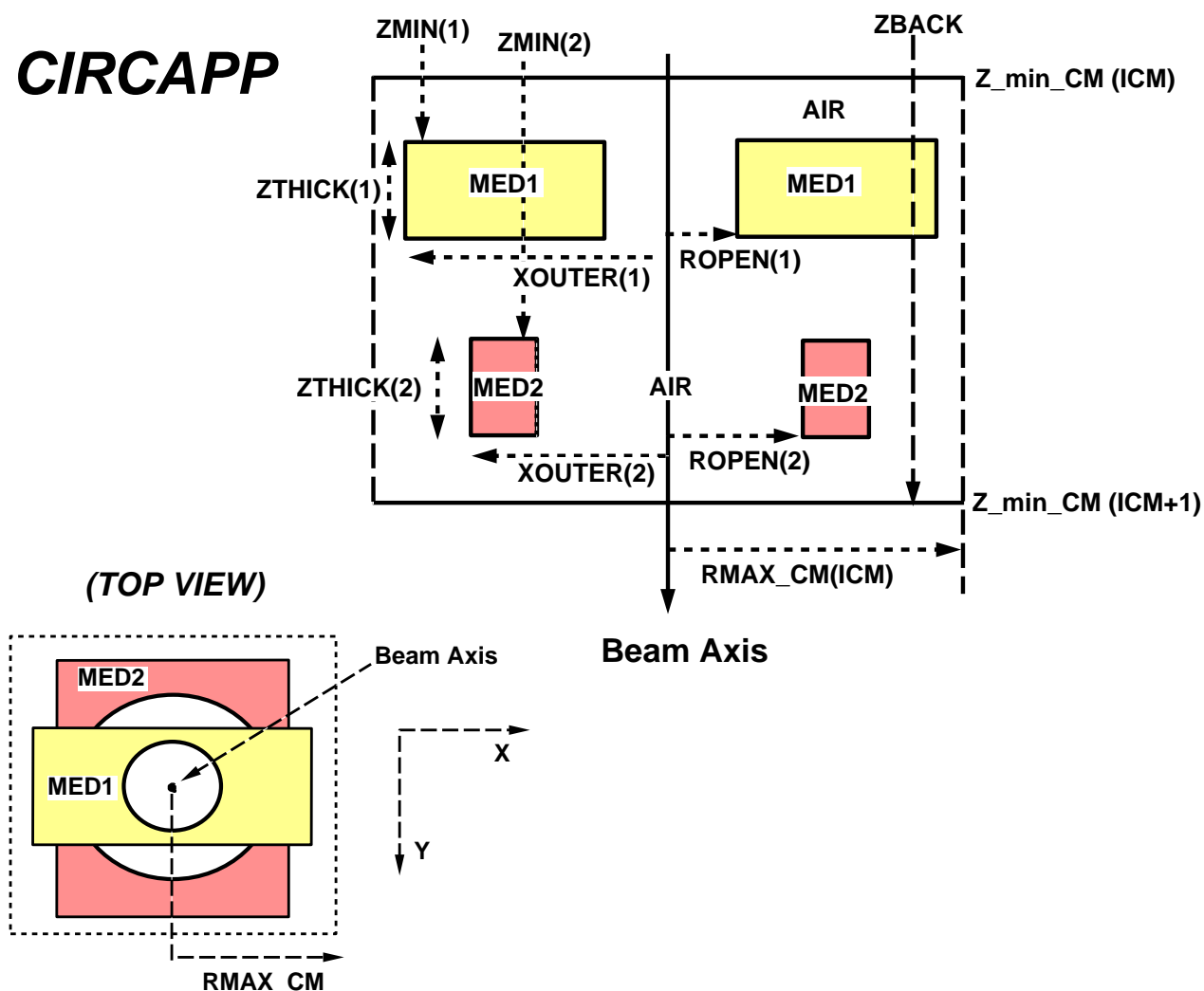
Figure 28: A CIRCAPP with 2 scrapers (N=2). Each scraper i has half-widths in the X- and Y-directions given by `XOUTER(i)` and `YOUTER(i)` and a circular opening in the centre with radius `ROPEN(i)`. The Y dimensions are not shown in the cross section, however, they are apparent in the top view. As in APPLICAT, there are minimum air gaps of 0.01 cm between scrapers, between `Z_min_CM(1)` and the first scraper and between `ZBACK` and the last scraper.

The input format for CIRCAPP, and an example of input file are given as follows.

```
CARDS CM_CIRCAPP
****************

-1  Dummy line to indicate start of CM.

 0  RMAX_CM(ICM_$CIRCAPP) (F10.0): Half-width of outer boundary of CM (cm).

 1  TITLE_$CIRCAPP (60A1):  Title of CM.

 2  ZBACK_$CIRCAPP (F15.0): Z of back face of the CM
                            (air will be added if necessary below the
                            last scraper)

     Note that there is always an air gap (thickness =
     AIRGAPMIN) in the front and the back of this CM.
     Therefore ZBACK_$CIRCAPP should be >= Z of the back face of the
     last scraper + AIRGAPMIN.

 3  N_$CIRCAPP:
       N_$CIRCAPP:  Number of scrapers in the CM.

  Repeat 4 for I=1,N_$CIRCAPP.

 4  ZMIN_$CIRCAPP(I), ZTHICK_$CIRCAPP(I), ROPEN_$CIRCAPP(I),
    XOUTER_$CIRCAPP(I), YOUTER_$CIRCAPP(I),
    DOSE_ZONE,IREGION_TO_BIT (6F15.0,2I4):

       ZMIN_$CIRCAPP(I):   Z of front face of scraper I. Note that
                           ZMIN_$CIRCAPP(1)-Z_min_CM must be >= AIRGAPMIN.
       ZTHICK_$CIRCAPP(I): Thickness of scraper I.  Note that
                            ZMIN_$CIRCAPP(I+1)-(ZMIN_$CIRCAPP(I)+
                            ZTHICK_$CIRCAPP(I)) must be >= AIRGAPMIN.
       ROPEN_$CIRCAPP(I):   Radius of inner opening in scraper I.
       XOUTER_$CIRCAPP(I):  X half-width of outer edge of scraper I.
       YOUTER_$CIRCAPP(I):  Y half-width of outer edge of scraper I.
       DOSE_ZONE:           Dose scoring zone for the scraper bar.
       IREGION_TO_BIT:      Bit setting number for the scraper bar.

       Note restrictions to allow air gaps between scrapers and
       before the first scraper:
        ZMIN_$CIRCAPP(1)-Z_min_CM >= AIRGAPMIN
        ZMIN_$CIRCAPP(I+1)-(ZMIN_$CIRCAPP(I)+ZTHICK_$CIRCAPP(I)) >=
        AIRGAPMIN

 6   ECUT, PCUT, DOSE_ZONE_AIR,IREGION_TO_BIT_AIR (2F15.0,2I5):

        ECUT, PCUT:    Cutoff energies for electrons and photons for
```

```
                               both the bars and the surrounding (air) region
            DOSE_ZONE_AIR:   Dose scoring zone for the surrounding region
     IREGION_TO_BIT_AIR:   Bit set number for the surrounding (air) region


   Repeat 7 for I=1,N_$CIRCAPP.


   7   MED_IN (24A1):   Medium of the bar of scraper I, used to
                        set MED_INDEX.


   Example
   *******


The input defines a circular applicator with 2 scrapers, one of Pb with
radius of opening = 0.6cm, X and Y half-widths of 1.0cm, and thickness
0.5cm. The other scraper, consisting of Al, has an opening with radius
2.4cm, X half-width = 4.6cm, Y half-width = 3.2cm, and thickness 1.0cm.
The scrapers are separated by 0.1cm of air. The front scraper starts at
Z=1.1cm and the second at 1.6 cm.


Electrons will be followed down to kinetic energies of 10 keV (total
energies of 0.521 MeV) and photons will be followed down to 1 keV.  The
dose deposited in the air will be scored in dose zone 1, and the dose in
the 2 scrapers will be scored in dose zones 2 and 3. There is a minimum
0.1 cm air gap at the front and back of the CIRCAPP CM.


14.0,    RMAX_CM
circular applicator
44.0,      extended air to Z=44 cm
2,      two scrapers
1.10,  0.50,  0.600, 1.0, 1.0, 2, 6,
1.60,  1.00,  2.40,  4.6, 3.2, 3, 7,
0.521, 0.01, 1, 0,
PB521ICRU
AL521ICRU
```

15.3.11   PYRAMIDS

The PYRAMIDS CM is used to model pyramid-shaped structures comprising one or more layers in the path of the beam. Each layer has three distinct regions: the central region (the pyramid), the surrounding region and the outer region (beyond the outer edges of the layer). The central and outer regions default to air but can also be filled with a user-specified medium (assumed the same for the central and outer regions within a layer). PYRAMIDS is useful for modelling rectangular collimators and beam blocks. This CM has a square outer boundary centered on the beam axis.
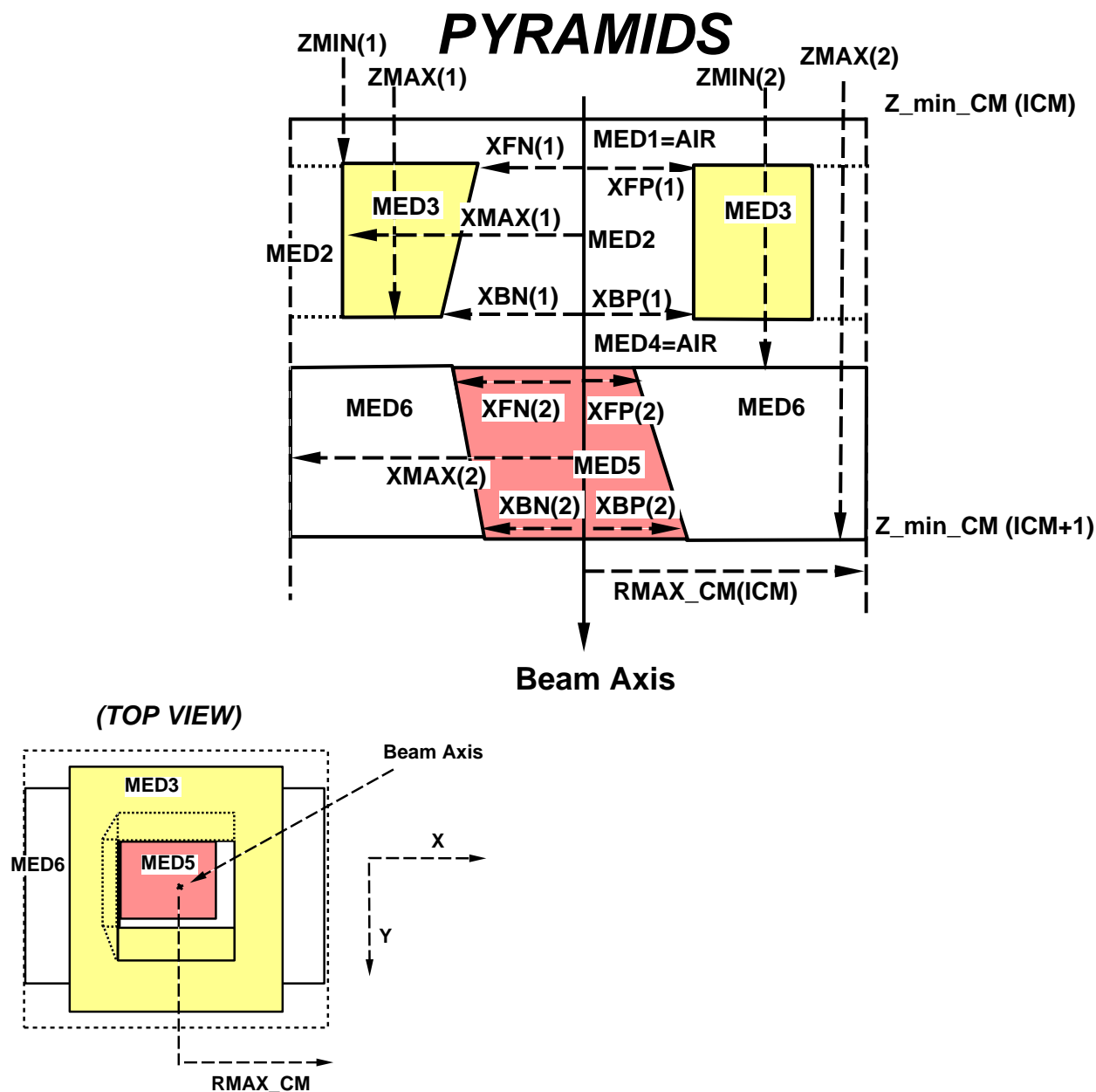
Figure 29: A PYRAMIDS component module with 2 layers (ISCM_MAX=2). Within a layer i, the front of the central region is specified by XFP(i), XFN(i), YFP(i), YFN(i), and the back is defined by XBP(i), XBN(i), YBP(i), YBN(i). The central region's inner faces are the planes connecting rectangle defining the front of the region with that defining the back of the region. The X and Y outer edges of layer i are defined by XMAX(i) and YMAX(i) respectively. The figure does not show the Y dimensions (*i.e.* YFN(i), YFP(i), YMAX(i), *etc.*), however the Y dimensions are evident in the top view. Layers must be separated by at least 0.01cm (ZMIN(i+1)-ZMAX(i) $\geq$ 0.01cm). By setting IFILL=1, the user can set the medium filling the central and outer regions in each layer (MED2 and MED5). The gaps between layers are assumed to be filled with air.

The input format for PYRAMIDS and an example input are given below.

```
CARDS CM_PYRAMIDS(Rev 1.5)
*****************

-1  dummy line filled with ***  read in main

 0  RMAX_CM(ICM_$PYRAMIDS):  Perpendicular distance from Z-axis
                             to boundary surrounding component
                             This component module has a square
                             boundary.

 1  TITLE_$PYRAMIDS (60A1):  Title of CM.

 2  ISCM_MAX_$PYRAMIDS, IFILL_$PYRAMIDS (2I5):

       ISCM_MAX_$PYRAMIDS: Number of layers in CM.
          IFILL_$PYRAMIDS: Set to 0 (default) if all central
                           and outer regions contain air, or 1
                           if central and outer regions contain
                           user-specified media.

    Repeat 3 for I=1,ISCM_MAX_$PYRAMIDS

 3  ZMIN_$PYRAMIDS(I), ZMAX_$PYRAMIDS(I), XFP_$PYRAMIDS(I),
    XBP_$PYRAMIDS(I), XFN_$PYRAMIDS(I), XBN_$PYRAMIDS(I),
    YFP_$PYRAMIDS(I), YBP_$PYRAMIDS(I), YFN_$PYRAMIDS(I),
    YBN_$PYRAMIDS(I),XMAX_$PYRAMIDS(I), YMAX_$PYRAMIDS(I) (12F15.0):

       ZMIN_$PYRAMIDS(I): Distance from front of layer I to reference plane.
       ZMAX_$PYRAMIDS(I): Distance from back of layer I to reference plane.

       XFP_$PYRAMIDS(I):  positive x dimension of central region at front
       XBP_$PYRAMIDS(I):  positive x dimension of central region at back
       XFN_$PYRAMIDS(I):  negative x dimension of central region at front
       XBN_$PYRAMIDS(I):  negative x dimension of central region at back

       YFP_$PYRAMIDS(I):  positive y dimension of central region at front
       YBP_$PYRAMIDS(I):  positive y dimension of central region at back
       YFN_$PYRAMIDS(I):  negative y dimension of central region at front
       YBN_$PYRAMIDS(I):  negative y dimension of central region at back

       XMAX_$PYRAMIDS(I): outer x edge of layer (absolute value)
       YMAX_$PYRAMIDS(I): outer y edge of layer (absolute value)

       Note restriction to leave airgap between layers:
             ZMIN_$PYRAMIDS(I+1)-ZMAX_$PYRAMIDS(I) >= AIRGAPMIN

 4   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5): for AIR
```

```
                                              between layers, and,
                                              if IFILL_$PYRAMIDS=0,
                                              in all central and outer regions.
        ECUT, PCUT:    Cutoff energies for electrons and photons.
         DOSE_ZONE:    Dose scoring zone of air.
    IREGION_TO_BIT:    mapping of region to bit for LATCH
```

```
 Repeat 7-8 (IFILL_$PYRAMIDS=0) or 5-8 (IFILL_$PYRAMIDS=1)
 for I=1,ISCM_MAX_$PYRAMIDS
```

```
 5 and 6 are required only if IFILL_$PYRAMIDS=1
```

```
 5  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5):
        ECUT, PCUT:    Cutoff energies for electrons and photons
                       in central region and outer region, layer I
         DOSE_ZONE:    Dose scoring zone for central region and outer region
    IREGION_TO_BIT:    mapping of central region and outer region to bit
                       for LATCH
```

```
 6  MED_IN (24A1):    Medium of central region and outer region in layer I,
                      used to set MED_INDEX.
```

```
 7   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5):
        ECUT, PCUT:    Cutoff energies for electrons and photons
                       in surrounding region, layer I
         DOSE_ZONE:    Dose scoring zone of surrounding region.
    IREGION_TO_BIT:    mapping of surrounding region to bit for LATCH
```

```
 8  MED_IN (24A1):    Medium of surrounding region in layer I,
                      used to set MED_INDEX.
```

```
 Example
 *******
The following input file describes two pyramidal openings,
both 0.3cm thick, and both filled with air (ie IFILL_$PYRAMIDS=0).
The first opening begins at Z=0.0 cm and is cut out of Pb.  Its positive
X face begins at XFP=0.8cm and angles back to XBP=1.2cm.  The negative X
face is a mirror image, beginning at XFN=-0.8cm and going to XBN=-1.2cm.
The outer x edge is at XMAX=5.0cm. The Y faces of the first opening are
perpendicular to the beam, with YFP=YBP=0.8cm and YFN=YBN=-0.8cm. The
outer y edge is at YMAX=4.6cm.
The second opening begins at Z=0.31 and is cut out of aluminum.  Its
positive and negative X faces describe a parallelepiped, with XFP=1.2cm,
XBP=0.8cm, XFN=-0.8cm, XBN=-1.2cm.
The positive Y face angles from YFP=2.0cm to
YBP=0.5cm and the negative Y face is a mirror image of this, going from
XFN=-2.0cm to XBN=-0.5cm. The outer x edge and y edge are at XMAX=5.1cm,
YMAX=5.2cm respectively. Note that an air gap >= AIRGAPMIN (=0.01cm) must
be left between pyramids and the top of the CM and the first pyramid.
```

```
In this particular input file, there is no gap at the top of the
CM, so Z_min_CM will be automatically reset to -0.01cm to
provide the required gap.

Dose in the air gaps and openings will be scored in dose zone 1.
Dose in the Pb will be scored in zone 2, and dose in the Al will be
scored in zone 3.  ECUT and PCUT in all regions are set to 0.521MeV and
0.01 MeV respectively.


10.0000,
PYR
2,0
0.0, 0.3, 0.8, 1.2, -0.8, -1.2, 0.8, 0.8, -0.8, -0.8, 5.0, 4.6
0.31, 0.61, 1.2, 0.8, -0.8, -1.2, 2.0, 0.5, -2.0, -0.5, 5.1, 5.2
0.521, 0.01, 1, 0
0.521, 0.01, 2, 0
PB521ICRU
0.521, 0.01, 3, 0
AL521ICRU
```

15.3.12   BLOCK

The BLOCK CM is used to model a treatment block having non-rectangular and/or multiple openings. The user specifies openings in the block material using up to 20 "subregions". For each subregion the user specifies the X-Y coordinates of its vertices at the top surface of the block material (either clockwise or counter-clockwise around the perimeter). The inner planes of all subregions are angled with respect to the beam (Z) axis towards a single user-specified "focus point" on the beam axis. Note that no subregion can have an inner angle > 180 degrees. Openings can consist of a single subregion or may require several adjoining subregions in order to avoid the restriction on the inner angles. The user also specifies the X and Y coordinates of the 4 outer edges of the block material, so the material need not extend to the square outer boundary of this CM. Due to its generality, BLOCK may require up to 2 times the CPU time of PYRAMIDS to simulate simple rectangular geometries; thus, PYRAMIDS is recommended when there is a single rectangular opening.

The current version of BLOCK has been significantly changed from BLOCK in BEAM99 (see `CHANGES_from_BEAM99.for.BEAM00`). This is because John Antolak of the M.D. Anderson Cancer Center found and corrected some serious errors in this CM.

BLOCK

Top View

MED2

XHI_POINT (1,1)
YHI_POINT (1,1)

X

XHI_POINT (5,1)
YHI_POINT (5,1)

MED2

XHI_POINT (2,1)
YHI_POINT (2,1)

XPMAX

Y

XNMAX

MED3

XHI_POINT (3,1)
YHI_POINT (3,1)

XHI_POINT (4,1)
YHI_POINT (4,1)

YNMAX   YPMAX

RMAX_CM(ICM)

X–Section through Opening

Y

ZMIN   ZFOCUS

ZMAX

Z_min_CM (ICM)

MED1=AIR

MED2   MED3   MED2

Z_min_CM (ICM+1)

YNMAX   YPMAX

RMAX_CM(ICM)

**beam axis
(Z)**
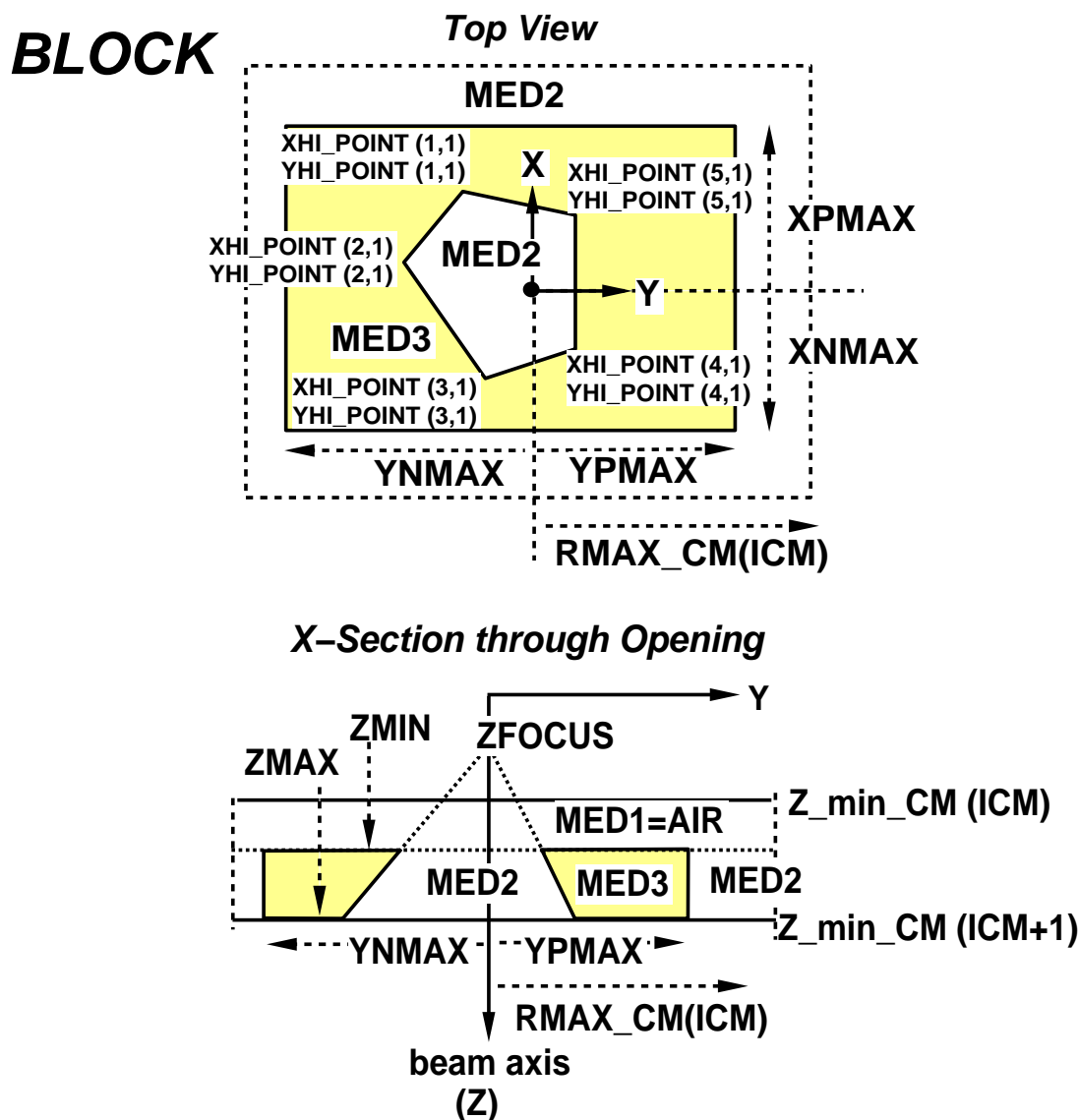
Figure 30: A BLOCK component module with a single opening composed of 1 sub-region (ISUB_MAX=1). As shown in the top view, the subregion has 5 vertices (NSUB(1)=5). The X-Y coordinates of the vertices at the top surface of the block material are specified counter-clockwise around the perimeter of the subregion, beginning with (XHI_POINT(1,1),YHI_POINT(1,1)). Also shown in the top view are the coordinates defining the outer boundary of the block, XPMAX, YPMAX, XNMAX and YNMAX. The cross-section through the opening shows that the planes defining the inner surfaces of the subregion/opening are all angled with respect to the Z-axis towards the single focus point on the Z-axis. Note that there must be an air gap between ZMIN and Z_min_CM(ICM) of ≥ 0.01 cm. Note that the user specifies MED2, the material filling the opening(s) and the region beyond the block edges. The top gap is assumed to be air. More complicated opening shapes may require several adjoining subregions.

The input format for BLOCK and an example input are given below.

```
CARDS CM_BLOCK(Rev 1.4)
*****************

-1  dummy line filled with ***  read in main

 0  RMAX_CM(ICM_$BLOCK):  Perpendicular distance from Z-axis
                          to boundary surrounding component
                          This component module has a square
                          boundary.

 1  TITLE_$BLOCK (60A1):  Title of CM.

 2  ZMIN_$BLOCK, ZMAX_$BLOCK, ZFOCUS_$BLOCK (3F15.0):

       ZMIN_$BLOCK:  Z of front of CM (not including airgap) (cm).
       ZMAX_$BLOCK:  Z of back of CM (cm).
     ZFOCUS_$BLOCK:  Z at which the inner sides of the opening(s)
                     in the block are focused (cm).

       Note restrictions: ZMAX < ZFOCUS or ZFOCUS < ZMIN, ie not in between
                          ZMIN - ZFOCUS >= 0.01 if ZFOCUS < ZMIN

 3  ISUB_MAX_$BLOCK (I5): Number of subregions.  Each opening is made
                          up of one or more subregions.

 Repeat 4 - 4a for J = 1, ISUB_MAX_$BLOCK

 4  NSUB_$BLOCK(J)   (I5)

      NSUB_$BLOCK(J):  number of points defining subregion J

 Repeat 4a for I = 1, NSUB_$BLOCK(J)

4a  XHI_POINT_$BLOCK(I,J),YHI_POINT_$BLOCK(I,J) (2F15.0):

    XHI_POINT_$BLOCK(I,J): X coordinate of point I at upper surface (cm)
    YHI_POINT_$BLOCK(I,J): Y coordinate of point I at upper surface (cm)

    NOTE:  Input the points clockwise or counter-clockwise around
           the perimeter of each subregion.  A subregion may not
           have an interior angle > 180 degrees.

 5  XPMAX_$BLOCK,YPMAX_$BLOCK,XNMAX_$BLOCK,YNMAX_$BLOCK (4F15.0):

      XPMAX_$BLOCK: X coordinate of block edge in +X direction (cm)
      YPMAX_$BLOCK: Y coordinate of block edge in +Y direction (cm)
      XNMAX_$BLOCK: X coordinate of block edge in -X direction (cm)
```

```
          YNMAX_$BLOCK: Y coordinate of block edge in -Y direction (cm)


  6    ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5) for air
                              in gap at top.


         ECUT, PCUT:    Cutoff energies for electrons and photons.
          DOSE_ZONE:    Dose scoring zone of air.
    IREGION_TO_BIT:     mapping of region to bit for LATCH


  7    ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5) in openings
                              and beyond edges of the block


         ECUT, PCUT:    Cutoff energies for electrons and photons in
                        openings and beyond edges of block
          DOSE_ZONE:    Dose scoring zone of openings and region beyond
                        edges of block
    IREGION_TO_BIT:     mapping of region comprising openings and region
                        beyond block edges to bit for LATCH


  8    MED_IN (24A1):   Medium in openings and beyond block edges,
                        used to set MED_INDEX.


  9    ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5) in block material


         ECUT, PCUT:    Cutoff energies for electrons and photons
                        in material surrounding openings
          DOSE_ZONE:    Dose scoring zone of material surrounding
                        openings.
    IREGION_TO_BIT:     mapping of region surrounding openings
                        to bit for LATCH


 10   MED_IN (24A1):   Medium of block,
                        used to set MED_INDEX.
```

 Example
 ****
The following input file describes a BLOCK of 4cm thick.
The block begins at Z=0.0 cm and is made of MILDSTEEL. The air-filled
opening(s) focus at Z=-10cm. Its positive X boundary begins at XPMAX=4.2cm,
positive Y boundary at YPMAX=4.8cm, and its negative X boundary at
XNMAX=-5.0cm, negative Y boundary at YNMAX=-3.0cm.
There are 2 sub-regions describing one opening shaped like an arrow.
In the first sub-region, there are 5 defining points; in the second one,
there are 4 defining points. The defining points should be input
clockwisely or counter-clockwisely. Each point is input as x, y in
a line.
In this particular input file, there is no gap at the top of the
CM, so Z_min_CM will be automatically reset to -0.01cm to
provide the required gap.

```
Dose in the air regions will be scored in dose zone 1.
Dose in the block material will be scored in zone 2.
ECUT and PCUT in all regions are set to 0.521MeV and
0.01 MeV respectively.

************************************************
10.0,            RMAX
arrow shaped cutoff
0.0, 4.0, -10.0,     ZMIN, ZMAX, ZFOCUS
2,               2 sub-regions
5,        5 defining points in sub 1
0., 3.,          x,y of point 1 in sub 1
-2., 1.,         x,y of point 2 in sub 1
-1., 0.,         x,y of point 3 in sub 1
1., 0.,          x,y of point 4 in sub 1
2., 1.,          x,y of point 5 in sub 1     end of sub 1
4,        4 defining points in sub 2
-1., 0.,         x,y of point 1 in sub 2
1., 0.,          x,y of point 2 in sub 2
1., -2.,         x,y of point 3 in sub 2
-1., -2.,        x,y of point 4 in sub 2.    end of sub 2.
4.2, 4.8, -5.0, -3.0,      xpmax,ypmax,xnmax,ynmax
0.0, 0.0, 1, 0,     ecut, pcut, dose-zone, ir-to-bit for air
0.0, 0.0, 1, 0,     ecut, pcut, dose-zone, ir-to-bit for openings
AIR521ICRU
0.0, 0.0, 2, 0,     ecut, pcut, dose-zone, ir-to-bit for materail
MILDSTEEL521
************************************************
```

15.3.13   MLC

The MLC CM is used to model a double-focusing multi-leaf collimator with flat faces. The
collimator has a single layer with a user-specified number of leaves all opening in either the X
or Y direction. The collimator opening is specified by the coordinates of the individual leaf
openings at the top of the collimator, the thickness of the leaves in the Z direction, and two
Z "foci" that determine the angles of the leaf side and end surfaces. The outer boundary of
the MLC CM is a square centred on the beam axis. Currently, the collimator body extends
to this outer boundary.
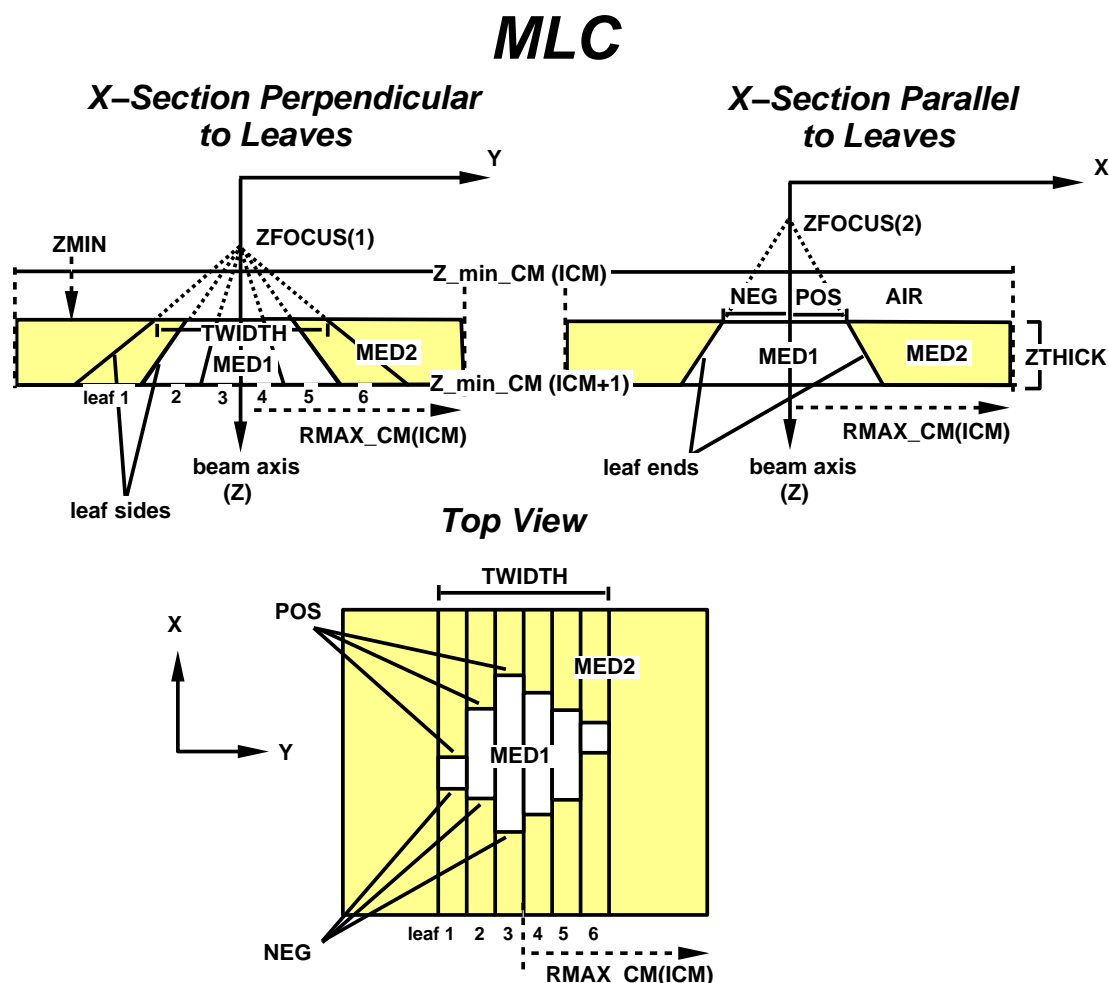
Figure 31: An MLC component module with 6 leaves (NUM_LEAF=6) opening in the X direction (IDMLFC=1). The cross-section perpendicular to the leaves (in the Y direction) shows the total width of the leaves at the top of the collimator, TWIDTH. The width of each leaf at the top is TWIDTH/NUM_LEAF. Note that NUM_LEAF must be an even number so that there is an equal number of leaves on either side of the X axis. The Y coordinates of the leaves at the bottom of the collimator are calculated automatically so that the tangents to the side surfaces of the leaves all cross the Z-axis at Z=ZFOCUS(1). The cross-section parallel to the leaves (in the X direction) shows the negative, NEG, and positive, POS, coordinates of the opening in one leaf. NEG and POS coordinates are input for every leaf. The X coordinates of the leaf openings at the bottom of the collimator are calculated automatically so that the tangents to the leaf ends all cross the Z-axis at Z=ZFOCUS(2). The top view of the collimator shows how the collimator opening is composed of the individual leaf openings. MED1 defines the material in the collimator opening and MED2 defines the material of the collimator leaves and body. This example would be applicable to leaves opening in the Y direction (IDMLFC=0) by reversing the roles of X and Y. Although this particular example shows ZFOCUS points that are < ZMIN, the ZFOCUS points can also be > ZMIN+ZTHICK, in which case leaf surfaces angle in towards the Z-axis with increasing Z.

The input format for MLC and an example input are given below.

```
CARDS CM_$MLC
**************
-1  Dummy line to indicate start of CM

 0  RMAX_CM(ICM_$MLC) (F10.0):  Half-width of CM boundary (cm).

 1  TITLE_$MLC (60A1):  Title of CM.

 2  IDMLFC_$MLC (I5) = 0 for leaves parallel to Y direction
                     = 1 for leaves parallel to X direction

 3  ZMIN_$MLC (F15.0): Z of top of collimator (excluding airgap)

 4  ZTHICK_$MLC (F15.0): Thickness of the leaves (cm)

 5  NUM_LEAF_$MLC, TWIDTH_$MLC (I5,F15.0)

        NUM_LEAF_$MLC: Number of leaves
        TWIDTH_$MLC:   Total width of leaves in X (IDMLFC_$MLC=0)
                       or Y (IDMLFC_$MLC=1) direction (cm)

     Note: width of each leaf = TWIDTH_$MLC/NUM_LEAF_$MLC

 6  ZFOCUS_$MLC(1) (F15.0): Focal point on Z-axis of leaf sides (ie.
                            imaginary lines drawn extending the slopes of
                            the leaf sides will all intersect the Z-axis
                            at this point)

     Note restriction: ZFOCUS_$MLC(1) < ZMIN_$MLC or
                                       > ZMIN_$MLC + ZTHICK_$MLC

 7  ZFOCUS_$MLC(2) (F15.0): Focal point on Z-axis of leaf ends (ie.
                            imaginary lines drawn extending the slopes of
                            the leaf ends will all intersect the Z-axis
                            at this point)

     Note restriction: ZFOCUS_$MLC(1) < ZMIN_$MLC or
                                       > ZMIN_$MLC + ZTHICK_$MLC

    Repeat 8 until coordinates of all leaves are defined once.  Leaves
    are numbered 1,2,...NUM_LEAF_$MLC, where numbering goes from left
    to right in the X-Y plane if IDMLFC_$MLC=0 and from top to bottom
    in the X-Y plane if IDMLFC_$MLC=1.

 8  NEG_$MLC, POS_$MLC, NUM_$MLC (2F15.0,I5)

        NEG_$MLC:    Min. Y (IDMLFC_$MLC=0) or X (IDMLFC_$MLC=1)
```

```
                         of front opening in leaf I (ie the opening at
                         ZMIN_$MLC)
        POS_$MLC:    Max. Y (IDMLFC_$MLC=0) or X (IDMLFC_$MLC=1)
                         of front opening in leaf I
        NUM_$MLC:    Apply NEG_$MLC(I) and POS_$MLC(I) to leaves
                         I,...,I+NUM_$MLC-1.  Defaults to 1 if set <= 0.
                         Defaults to NUM_LEAF_$MLC-I+1 if set >
                         NUM_LEAF_$MLC-I+1


 9  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in local region 1
                                      (inside collimator) (2F15.0,I5):


        ECUT, PCUT:  Cutoff energies for electrons and photons.
        DOSE_ZONE:   Dose scoring flag, 0 to not score dose
   IREGION_TO_BIT:   Bit number associated with this region


10  MED_IN (24A1):  Medium of in local region 1 (inside collimator)
                      used to set MED_INDEX.


11  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in local region 2
                                      (collimator leaves) (2F15.0,I5):


        ECUT, PCUT:  Cutoff energies for electrons and photons.
        DOSE_ZONE:   Dose scoring flag, 0 to note score dose
   IREGION_TO_BIT:   Bit number associated with this region


12  MED_IN (24A1):  Medium of local region 2 (collimator leaves),
                      used to set MED_INDEX.


Example
*******


The following example defines a multi-leaf collimator design based
loosely on that used with the MM50 Racetrack Microtron accelerator.
The collimator starts at Z=65 cm and has 64 tungsten leaves opening
in the X direction.  The leaves are each 0.8125cm wide and 7.5cm
thick.  The Z focus of the leaf sides is at Z=-1000 cm, resulting
in sides that are essentially straight up and down.  The Z focus
of the leaf ends is at Z=0, which is the position of the beam source.
In this example, leaf opening coordinates are chosen to create a
irregular off-center collimator opening.


Electrons and photons in both the collimator and the opening regions
will be followed down to kinetic energies of 10 keV (ECUT=0.521,
PCUT=0.01).  Dose deposited in the tungsten leaves will be stored
in dose zone 2, and dose deposited in the opening will be stored
in dose zone 1.


26.0,            RMAX_CM
```

```
Collimator based on MLC for MM50 accelerator
1,              Leaves open in X direction
65.0,           ZMIN
7.5,            ZTHICK
64, 52.0,       NUM_LEAF, TWIDTH
-1000.0,        ZFOCUS(1)
0.0             ZFOCUS(2)
0.0,0.0,15,     15 closed leaves
0.0,2.0,5,      5 leaves with opening 0.0 - 2.0
0.5,3.0,2       2 leaves with 0.5 - 3.0
1.0,4.0,3       3 leaves with 1.0 - 4.0
2.0,7.0,10,     10 leaves with opening 2.0 - 7.0
1.5,6.0
1.0,6.0
0.0,5.0,3,      3 leaves with 0.0 - 5.0
-1.0,4.0,5,     5 leaves with -1.0 - 4.0
-2.0,4.0,3,     3 leaves with -2.0 - 4.0
-4.0,4.0,5,     5 leaves with -4.0 - 4.0
-5.0,3.0
-6.0,1.0
-8.0,0.0
-10.0,-2.0,3,   3 leaves with -10.0 - -2.0
-12.0,-2.0,2,   2 leaves with -12.0 - -2.0
-15.0,-3.0,2,   2 leaves with -15.0 - -3.0
-15.0,-15.0
0.5210,  0.010,  1,  0
AIR700ICRU
0.5210,  0.010,  2,  0
W700ICRU
```

15.3.14   MLCQ

The MLCQ CM is used to model a focusing multi-leaf collimator with rounded leaf ends. The collimator is similar to MLC with the exception that, rather than specifying a Z focus for the leaf ends, the user specifies a radius for the leaf ends and the Z position of the origin of this radius (*i.e.* so the leaf ends can be angled up or down). The collimator opening is defined by specifying the X or Y (depending on leaf orientation) origin of the radius for the positive and negative portions of each leaf.

The first version of this CM, which was based on MLC, was coded by Hugo Palmans and Kristiaan De Vlamynck of the University of Gent, Belgium.

MLCQ

X–Section Perpendicular to Leaves

Y

ZMIN

ZFOCUS(1)

Z_min_CM (ICM)

TWIDTH

MED2

MED1

Z_min_CM (ICM+1)

leaf 1   2   3   4   5   6

RMAX_CM(ICM)

beam axis (Z)

leaf sides

X–Section Parallel to Leaves

X

Z0LEAF

Z_min_CM (ICM)

X0N — X0P

AIR

R0LEAF

MED2   MED1

ZTHICK

leaf ends

beam axis (Z)

RMAX_CM(ICM)

Top View

TWIDTH

X

X0P

MED2

MED1

Y

X0N

leaf 1   2   3   4   5   6

RMAX_CM(ICM)
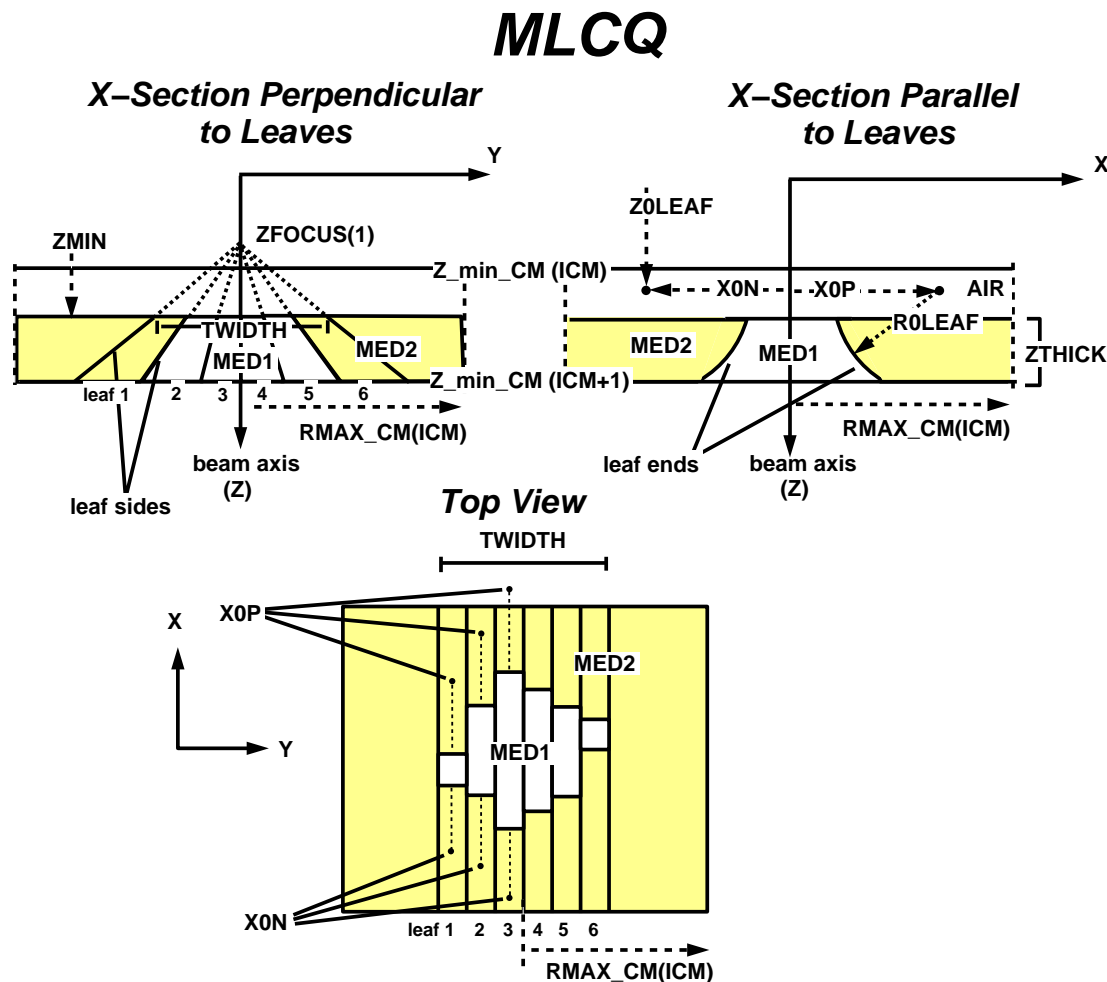
Figure 32: An MLCQ component module with 6 leaves (NUM_LEAF=6) opening in the X direction (IDMLFC=1). Parameters are similar to MLC with the exception of those for the leaf ends. The cross-section parallel to the leaves shows how the rounded leaf ends are defined. Basically, for each leaf, the user defines a circle with radius ROLEAF (the radius of the leaf ends), centred at (X0N,Z0LEAF) for the negative portion of the leaf and (X0P,Z0LEAF) for the positive portion of the leaf. Note that X0N,X0P define X coordinates if the leaves are parallel to the X direction and Y coordinates if the leaves are parallel to the Y direction. Also note that Z0LEAF applies to all leaves. The rounded leaf end is then the arc subtended by the intersection of this circle with ZMIN and ZMIN + ZTHICK. The user controls the angle of the leaf ends with respect to the Z axis using Z0LEAF and the dimensions of the leaf opening using X0N and X0P. The top view shows that ABS(X0N) and ABS(X0P) can be > RMAX_CM, however the points of intersection of the circle defining a leaf end with ZMIN and ZMIN + ZTHICK must be within the boundaries of the CM or else the volume/mass of the collimator will be determined incorrectly for dose calculations.

The input format for MLCQ and an example input are given below.

```
CARDS CM_$MLCQ
**************
-1  Dummy line to indicate start of CM

 0  RMAX_CM(ICM_$MLCQ) (F10.0):  Half-width of CM boundary (cm).

 1  TITLE_$MLCQ (60A1):  Title of CM.

 2  IDMLFC_$MLCQ (I5) = 0 for leaves parallel to Y direction
                      = 1 for leaves parallel to X direction

 3  ZMIN_$MLCQ (F15.0): Z of top of collimator (excluding airgap)

 4  ZTHICK_$MLCQ (F15.0): Thickness of the leaves (cm)

 5  NUM_LEAF_$MLCQ, TWIDTH_$MLCQ (I5,F15.0)

        NUM_LEAF_$MLCQ: Number of leaves
        TWIDTH_$MLCQ:   Total width of leaves in X (IDMLFC_$MLCQ=0)
                        or Y (IDMLFC_$MLCQ=1) direction (cm)

     Note: width of each leaf = TWIDTH_$MLCQ/NUM_LEAF_$MLCQ

 6  ZFOCUS_$MLCQ(1) (F15.0): Focal point on Z-axis of leaf sides (ie.
                             imaginary lines drawn extending the slopes of
                             the leaf sides will all intersect the Z-axis
                             at this point)

     Note restriction: ZFOCUS_$MLCQ(1) < ZMIN_$MLCQ or
                                        > ZMIN_$MLCQ + ZTHICK_$MLCQ

 7  ROLEAF_$MLCQ,ZOLEAF_$MLCQ (2F15.0)

        ROLEAF_$MLCQ: Radius of leaf ends in cm.
        ZOLEAF_$MLCQ: Z where radius of leaf ends originates in cm.

    Note restrictions: 1. ZMIN_$MLCQ < ZOLEAF_$MLCQ <
                                       ZMIN_$MLCQ + ZTHICK_$MLCQ
                       2. ROLEAF_$MLCQ >
                         MAX(ZMIN_$MLCQ+ZTHICK_$MLCQ-ZOLEAF_$MLCQ,
                             ZOLEAF_$MLCQ-ZMIN_$MLCQ)
    Repeat 8 until coordinates of all leaves are defined once.  Leaves
    are numbered 1,2,...NUM_LEAF_$MLCQ, where numbering goes from left
    to right in the X-Y plane if IDMLFC_$MLCQ=0 and from top to bottom
    in the X-Y plane if IDMLFC_$MLCQ=1.

 8  XON_$MLCQ, XOP_$MLCQ, NUM_$MLCQ (2F15.0,I5)
```

```
        XON_$MLCQ:    Y (IDMLFC_$MLCQ=0) or X (IDMLFC_$MLCQ=1)
                      of origin of radius of negative part of leaf I
        XOP_$MLCQ:    Y (IDMLFC_$MLCQ=0) or X (IDMLFC_$MLCQ=1)
                      of origin of radius of positive part of leaf I
        NUM_$MLCQ:    Apply XON_$MLCQ and XOP_$MLCQ to leaves
                      I,...,I+NUM_$MLCQ-1.  Defaults to 1 if set <= 0.
                      Defaults to NUM_LEAF_$MLCQ-I+1 if set >
                      NUM_LEAF_$MLCQ-I+1


 9  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in local region 1
                                      (inside collimator) (2F15.0,I5):


      ECUT, PCUT:   Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring flag, 0 to not score dose
   IREGION_TO_BIT:  Bit number associated with this region


10  MED_IN (24A1):  Medium of in local region 1 (inside collimator)
                    used to set MED_INDEX.


11  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in local region 2
                                      (collimator leaves) (2F15.0,I5):


      ECUT, PCUT:   Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring flag, 0 to note score dose
   IREGION_TO_BIT:  Bit number associated with this region


12  MED_IN (24A1):  Medium of local region 2 (collimator leaves),
                    used to set MED_INDEX.


Example
*******

The following example defines a multi-leaf collimator
that starts at Z=65 cm and has 64 tungsten leaves opening
in the X direction.  The leaves are each 0.8125cm wide and 7.5cm
thick.  The Z focus of the leaf sides is at Z=-1000 cm, resulting
in sides that are essentially straight up and down.  The radius
of the leaf ends is 10cm and has a Z origin of 65cm (ie ZMIN),
The unique position of the Z origin of the radius means that the X
dimensions of the opening in each leaf at the top of the collimator
(ZMIN) will be given by XON+10 - XOP-10 cm.
In this example, the X origins of the radii of the individual leaves
are chosen to create an irregular off-center collimator opening.

Electrons and photons in both the collimator and the opening regions
will be followed down to kinetic energies of 10 keV (ECUT=0.521,
PCUT=0.01).  Dose deposited in the tungsten leaves will be stored
in dose zone 2, and dose deposited in the opening will be stored
```

```
in dose zone 1.

26.0,            RMAX_CM
Collimator based on MLC for MM50 accelerator
1,               Leaves open in X direction
65.0,            ZMIN
7.5,             ZTHICK
64, 52.0,        NUM_LEAF, TWIDTH
-1000.0,         ZFOCUS(1)
10.0,65.0        ROLEAF,ZOLEAF
-10.0,10.0,15,   15 leaves closed at top
-10.0,12.0,5,    5 leaves with opening 0.0 - 2.0 at top
-9.5,13.0,2      2 leaves with 0.5 - 3.0 at top
-9.0,14.0,3      3 leaves with 1.0 - 4.0 at top
-8.0,17.0,10,    10 leaves with opening 2.0 - 7.0 at top
-8.5,16.0
-9.0,16.0
-10.0,15.0,3,    3 leaves with 0.0 - 5.0 at top
-11.0,14.0,5,    5 leaves with -1.0 - 4.0 at top
-12.0,14.0,3,    3 leaves with -2.0 - 4.0 at top
-14.0,14.0,5,    5 leaves with -4.0 - 4.0 at top
-15.0,13.0
-16.0,11.0
-18.0,10.0
-20.0,8.0,3,     3 leaves with -10.0 - -2.0 at top
-22.0,8.0,2,     2 leaves with -12.0 - -2.0 at top
-25.0,7.0,2,     2 leaves with -15.0 - -3.0 at top
-25.0,-5.0
0.5210,  0.010,  1,  0
AIR700ICRU
0.5210,  0.010,  2,  0
W700ICRU
```

15.3.15 VARMLC

The VARMLC CM is used to model a focusing multi-leaf collimator with either rounded leaf ends or straight leaf ends with a Z focus. The major difference between VARMLC and MLC or MLCQ is that VARMLC simulates the air gaps between leaves, the tongue-in-groove mechanism by which adjacent leaves slide against each other, and the driving screws at the top and bottom of each leaf used to open and close the leaves. VARMLC can also simulate leaves of different widths within the same collimator. In VARMLC the medium beyond the leaves in the direction perpendicular to the leaves defaults to be the same as the medium in the leaf opening(s) (ie AIR). This is different from MLC and MLCQ in which the medium in this region defaults to the leaf medium (ie solid). This default can be changed within the VARMLC_cm.mortran and VARMLC_macros.mortran codes.

The first version of this CM was coded by Ajay Kapur with Charlie Ma at Stanford University. The current version has significant modifications.
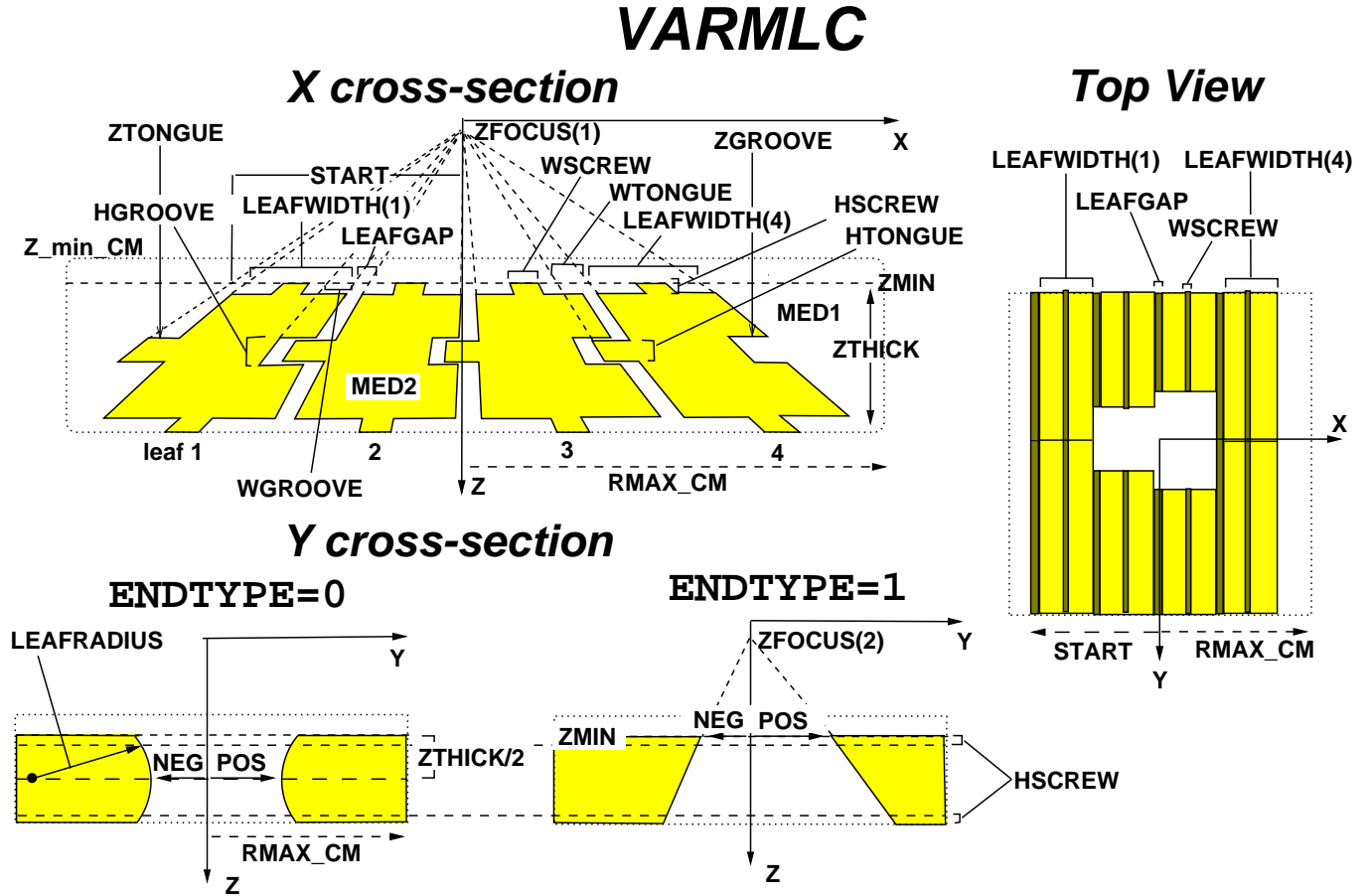
Figure 33: A VARMLC component module with 4 leaves (NUM_LEAF=4) opening in the Y direction (ORIENT=0). The X cross-section (perpendicular to the leaf direction) shows the widths of the leaves (LEAFWIDTH(1)...LEAFWIDTH(4)) and the dimensions of the leaf gap (LEAFGAP), tongue-in-groove mechanism (WGROOVE, HGROOVE, WTONGUE, HTONGUE) and the driving screws (WSCREW, HSCREW) as input by the user. The user also sets the position of the top of the tongue, ZTONGUE, and the top of the groove, ZGROOVE, with the restriction that the tongue and groove cannot overlap. If ZTONGUE=ZGROOVE=0 then the tongue and groove are centred at Z = ZMIN + ZTHICK/2. For restrictions on leaf gap and tongue-in-groove inputs, see the detailed description of input parameters below. Note that all leaf side surfaces are focused at ZFOCUS(1) and that horizontal dimensions (WGROOVE, WTONGUE, WSCREW, START) are all given at ZMIN. Due to LEAFGAP and varying LEAFWIDTH the leaves may not be symmetric about the Z axis, although START can be set to give approximate symmetry. Also note that the medium in regions beyond the leaves perpendicular to the leaf direction defaults to MED1, the medium of the openings and air gaps. Y cross-sections are shown for leaves with rounded ends (ENDTYPE=0) and straight ends (ENDTYPE=1). For rounded ends, the user inputs the radius of the ends (LEAFRADIUS) and, for each leaf, the minimum and maximum coordinates of the leaf opening at Z = ZMIN + ZTHICK/2 (NEG, POS). The end radius is always assumed to originate at Z = ZMIN + ZTHICK/2. For straight leaf ends, the user inputs the Z focus of the ends (ZFOCUS(2)) and, for each leaf, the minimum and maximum coordinates of the opening at Z = ZMIN (NEG, POS).

The input format for VARMLC and an example input are given below.

```
CARDS CM_$VARMLC
**************
-1  Dummy line to indicate start of CM

 0  RMAX_CM(ICM_$VARMLC) (F10.0):  Half-width of CM boundary (cm).

 1  TITLE_$VARMLC (60A1):  Title of CM.

 2  ORIENT_$VARMLC, NGROUP_$VARMLC (2I5)

         ORIENT_$VARMLC= 0 for leaves parallel to Y direction
                       = 1 for leaves parallel to X direction
         NGROUP_$VARMLC= number of groups of adjacent leaves where
                         all leaves in a group have the same width
                         (defaults to 1 if set <= 0)

 3  ZMIN_$VARMLC (F15.0): Z of top of MLC (excluding airgap)

 4  ZTHICK_$VARMLC (F15.0): Thickness of the leaves ( z-axis (cm))

Repeat 5 NGROUP_$VARMLC times

 5  NUM_LEAF_$VARMLC(I), LEAFWIDTH_$VARMLC(I)  (I5,F15.0)

         NUM_LEAF_$VARMLC(I):  Number of adjacent leaves in group I
         LEAFWIDTH_$VARMLC(I): Width of each leaf in group I
                       as projected at ZMIN_$VARMLC - the width does
                       not include the tongue (see figure).

    Note: total number of leaves is stored in TOT_LEAF_$VARMLC

 6  START_$VARMLC (F15.0) : the start position (cm) wrt the CAX of
                    leaf 1 tongue as projected to ZMIN_$VARMLC.

 7  WSCREW_$VARMLC, HSCREW_$VARMLC (2F15.0) : The width and height
          of the screw on the carriage railing. The width is
          as projected at ZMIN_$VARMLC and the height projected at
          the z-axis.

 8  WTONGUE_$VARMLC, HTONGUE_$VARMLC, ZTONGUE_$VARMLC (3F15.0) :
          The width and height of the tongue projected to ZMIN_$VARMLC
          and z-axis respectively and the Z starting position of the
          tongue.  ZTONGUE_$VARMLC=0 assumes that the tongue is centred
          at ZMIN_$VARMLC+ZTHICK_$VARMLC/2 (ie centre of leaf body).

 9  WGROOVE_$VARMLC, HGROOVE_$VARMLC, ZGROOVE_$VARMLC (3F15.0) : The
          width and height of the groove projected to ZMIN_$VARMLC
```

and z-axis respectively and the Z starting position of the
groove. ZGROOVE_$VARMLC=0 assumes that the groove is centred
at ZMIN_$VARMLC+ZTHICK_$VARMLC/2 (ie centre of leaf body).

Note restriction: ZTONGUE >= ZGROOVE
                  ZTONGUE+HTONGUE <= ZGROOVE+HGROOVE
                  WTONGUE <= WGROOVE

10 LEAFGAP_$VARMLC (F15.0) : The width of the interleaf air gap
         at ZMIN_$VARMLC.

Note restriction: LEAFGAP_$VARMLC <=WTONGUE_$VARMLC

11 ENDTYPE_$VARMLC (I5) : The type of leaf end :
         0 -- rounded leaf end and
         1 -- focused divergent leaf end.

12 ZFOCUS_$VARMLC (F15.0) : Focal point on Z-axis of leaf ends
                 (i.e. imaginary lines drawn extending the slopes
                 of leaf ends will all intersect the Z-axis
                 at this point) - chosen if ENDTYPE_$VARMLC = 1.

Note restriction: ZFOCUS_$VARMLC(1) < ZMIN_$VARMLC or
                              > ZMIN_$VARMLC + ZTHICK_$VARMLC

LEAFRADIUS_$VARMLC (F15.0) : Radius of the leaf end if
                  ENDTYPE_$VARMLC = 0. This must be greater
                  than or equal to half the leaf thickness.

13 ZFOCUS_$VARMLC(1) (F15.0): Focal point on Z-axis of leaf sides
                 imaginary lines drawn extending the slopes of
                 the leaf sides will all intersect the Z-axis
                 at this point)

Note restriction: ZFOCUS_$VARMLC(1) < ZMIN_$VARMLC or
                              > ZMIN_$VARMLC + ZTHICK_$VARMLC

For focused ends the leaf position is defined
at ZMIN_$VARMLC; for rounded at ZMIN_$VARMLC +
0.5*ZTHICK_$VARMLC (ie center of the leaf in z)

Repeat 14 until coordinates of all leaves are defined once. Leaves
are numbered 1,2,...TOT_LEAF_$VARMLC, where numbering goes from leaf
1 to leaf TOT_LEAF_$VARMLC. Convention is lower to upper or
left to right depending on ORIENT_$VARMLC i.e from negative to
positive.

14 NEG_$VARMLC, POS_$VARMLC, NUM_$VARMLC (2F15.0,I5)

```
    NEG_$VARMLC:    Min. Y (ORIENT_$VARMLC=0) or X (ORIENT_$VARMLC=1)
                    of front opening in leaf I (ie the opening at
                    ZMIN_$VARMLC) if ENDTYPE=1, or of rounded end
                    of leaf I if ENDTYPE=0.
    POS_$VARMLC:    Max. Y (ORIENT_$VARMLC=0) or X (ORIENT_$VARMLC=1)
                    of front opening in leaf I if ENDTYPE=1, or of
                    rounded end of leaf I if ENDTYPE=0.
    NUM_$VARMLC:    Apply NEG_$VARMLC and POS_$VARMLC to leaves
                    I,...,I+NUM_$VARMLC-1.  Defaults to 1 if set <=0.
                    Defaults to TOT_LEAF_$VARMLC-I+1 if set >
                    TOT_LEAF_$VARMLC-I+1.


 15    ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in opening(s) and
                                             air gaps (2F15.0,2I5)


       ECUT, PCUT:  Cutoff energies for electrons and photons.
        DOSE_ZONE:  Dose scoring flag, 0 to not score dose
    IREGION_TO_BIT: Bit number associated with this region


  16  MED_IN (24A1):  Medium in opening(s) and air gaps
                      used to set MED_INDEX.


  17  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in leaves, IGNOREGAPS_$VARMLC
                                                (2F15.0,3I5):


       ECUT, PCUT:  Cutoff energies for electrons and photons.
        DOSE_ZONE:  Dose scoring flag, 0 to note score dose
    IREGION_TO_BIT: Bit number associated with this region
       IGNOREGAPS:  If set to 1, ignore all air gaps when doing range
                    rejection in leaf material when the particle X
                    position
                    is < min X of all leaf openings (not including leaf
                    ends) or > max X of leaf openings (not including ends)
                    (ORIENT_$VARMLC=1) or if the particle Y position
                    is < min Y of all leaf openings (not including leaf
                    ends) or > max Y of leaf openings (not including ends)
                    (ORIENT_$VARMLC=0). This approximation is designed
                    to make range rejection more efficient deep in the
                    leaves, while still preserving accurate transport
                    in the leaf ends.  Note that if you have significant
                    air gaps between leaves, it is recommended that
                    you not use this option (ie run with the default
                    setting of 0).


  18  MED_IN (24A1):  Medium of leaves,
                      used to set MED_INDEX.


    Example
    *******
```

The following example defines a multi-leaf collimator design based
loosely on that used with the Varian Clinac 2100C 26 leaf pair.
Actual parameters are DIFFERENT - this serves just as a template.
Do not attempt to use these parameters for a simulation of the real
machine.

The collimator starts at Z=50 cm and has 26 tungsten leaves opening
in the X direction. The leaves are each 0.5cm wide and 6.0cm
thick. The Z focus of the leaf sides is at Z=-1000 cm, resulting
in sides that are essentially straight up and down. The Z focus
of the leaf ends is at Z=0, which is the position of the beam source.
In this example, leaf opening coordinates are chosen to create a
pattern of alternating open and closed leaves.

Electrons and photons in both the collimator and the opening regions
will be followed down to kinetic energies of 10 keV (ECUT=0.521,
PCUT=0.01). Dose deposited in the tungsten leaves will be stored
in dose zone 2, and dose deposited in the opening will be stored
in dose zone 1. Finally, the option to ignore air gaps when doing
range rejection deep in the leaves is off.

```
26.0,              RMAX_CM
MLC based on mock 26 leaf pair Varian 2100C type of accelerator
1,1,               Leaves open in X direction, all have same thickness
50.0,              ZMIN
6.0,               ZTHICK
26, 0.50,          NUM_LEAF, LEAFWIDTH
-6.5,              START POSITION
0,2, 0,4,          WIDTH AND HEIGHT OF SCREW
0.2, 2,0,          WIDTH, HEIGHT, Z OF TONGUE, tongue centred in leaf
0.21,2.6,0,        WIDTH, HEIGHT, Z OF GROOVE, groove centred in leaf
0.005,             LEAFGAP BETWEEN ADJACENT LEAVES
1,                 ENDTYPE IS FOCUSED AND DIVERGENT
0.0,               ZFOCUS(2)
-1000.0            ZFOCUS(1)
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
```

```
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.5210,  0.010,  1,  0
AIR700ICRU
0.5210,  0.010,  2,  0, 0
W700ICRU
```

Note that VARMLC has an additional input, IGNOREGAPS, appearing on the same line as ECUT, PCUT etc for the leaves. This input increases the efficiency of range rejection for particles in the leaves. If IGNOREGAPS is set to 1, then all air gaps within the leaves (ie gaps between leaves and the Z-directional gaps caused by the presence of carriage screws) are ignored when doing charged particle range rejection provided that the particle is in the leaves and satisfies:

particle X < minimum X of all leaf openings (excluding rounded or focused leaf end) or
particle X > maximum X of all leaf openings (excluding leaf end)

if the leaves are parallel to X (ORIENT=1) or:

particle Y < minimum Y of all leaf openings (excluding leaf end) or
particle Y > maximum Y of all leaf openings (excluding leaf end)

if the leaves are parallel to Y (ORIENT=0). Exact transport is preserved in the shaped leaf ends (rounded or focused) by excluding them from the volume in which air gaps are ignored. This is important, since the leaf ends define the field shape and particles interacting in the ends are more likely to reach the field at the SSD.

Use of the IGNOREGAPS option can reduce the CPU time spent in VARMLC by a factor of 2. However, if the multi-leaf collimator you are modeling has significant air gaps between leaves, we recommend that you run with this option turned off (ie IGNOREGAPS set to 0; the default) so that there is exact transport everywhere.

15.3.16 MLCE

The MLCE CM is used to model multi-leaf collimators specific for Elekta machines. Instead of specifying a tongue-and-groove, the user specifies the dimensions of interlocking steps typical of this class of MLC. All leaves have identical cross-sections, and leaf sides are focused (always to Z=0) by tilting each leaf about an axis that runs parallel to the leaf opening direction and along the centre of its top surface. The entire leaf bank can also be rotated in a plane perpendicular to the leaf opening direction by a user-specified angle. As in VARMLC, the medium beyond the leaves in the direction perpendicular to the leaves defaults to be the same as the medium in the leaf opening(s) (ie AIR).

This CM was mostly coded by Nick Reynaert at the University of Ghent. There have been some modifications of inputs and some small bugs were fixed.

Details of MLCE are shown in Figure 34. All leaves have identical cross-sections, which are defined using an imaginary "central leaf". For this leaf, the user specifies the 4 points defining the cross-section (which is symmetric about X=0 if the steps are ignored), X3, ZMIN, X4, ZMAX, the Z positions of the left and right steps, ZSTEPL,ZSTEPR, and the step width, TGW. Note that ZSTEPL must be >ZSTEPR for the steps to fit into one another once the leaves are generated. The central leaf is then duplicated NUM_LEAF times, and each leaf is rotated/translated in the X (ORIENT=0) or Y (ORIENT=1) direction. Rotation angle and translation distance are determined by user inputs which define the spacing between the centres of the leaf cross-sections, SPACE, as projected down to SSD and the requirement that the leaf sides focus to Z=0. Leaves are first rotated about the X=0 (ORIENT=0) or Y=0 (ORIENT=1), Z=ZMIN using:

$$
rotation = \begin{cases} ATAN\left[-(2I-1)\left(\frac{SPACE}{2}\right)\left(\frac{ZMIN}{SSD}\right)\right] & \text{for } 1\leq I \leq \frac{NUM\_LEAF}{2}, \\ ATAN\left[(2I-1)\left(\frac{SPACE}{2}\right)\left(\frac{ZMIN}{SSD}\right)\right] & \text{for } \frac{NUM\_LEAF}{2}+1 \leq I \leq NUM\_LEAF. \end{cases}
$$
(7)

Then they are translated in the X (ORIENT=0) or Y (ORIENT=1) direction using:

$$
translation = \begin{cases} -(2I-1)\left(\frac{SPACE}{2}\right)\left(\frac{ZMIN}{SSD}\right) & \text{for } 1\leq I \leq \frac{NUM\_LEAF}{2}, \\ (2I-1)\left(\frac{SPACE}{2}\right)\left(\frac{ZMIN}{SSD}\right) & \text{for } \frac{NUM\_LEAF}{2}+1 \leq I \leq NUM\_LEAF. \end{cases}
$$
(8)

Note the restriction that NUM_LEAF must be an even number. After the leaves have been rotated/translated according to the above equations, their cross-sections are symmetric about the Z axis.

The user also has the option to rotate the entire leaf bank about the axis X=0 (ORIENT=0) or Y=0 (ORIENT=1), Z=ZMIN by angle LBROT (which must be specified in radians). This rotates the central axis of the leaves (ie the axis along which they are focused) from Z to Z', as shown in Figure 34. Rotation of the leaves, both individually and of the entire leaf bank, requires automatic resetting of ZMIN and ZMAX, shown as (adjusted) values in the figure.

Figure 34 also shows the two types of leaf ends possible with MLCE. Cylindrical leaf ends (ENDTYPE=0) are defined by the radius, LEAFRADIUS, and Z position, ZCIL, of the cylinder describing the ends. Leaf openings are specified by the X (ORIENT=1) or Y (ORIENT=0) coordinates of the cylinder origins for the positive (POS) and negative (NEG) portions of the leaf. Straight leaf ends (ENDTYPE=1) are angled according to the user-input ZFOCUS, and leaf

Figure 34: Example MLCE with 8 leaves (NUM_LEAF=8) opening in the Y direction (ORIENT=0) and with a leaf bank rotation angle of LBROT. The figure also shows the cross-section of the imaginary "central leaf", used to create the leaves, and the two possible leaf end types: cylindrical (ENDTYPE=0) and straight (ENDTYPE=1). See text for more details.

openings are specified by the X (`ORIENT=1`) or Y (`ORIENT=0`) coordinates of the ends of the positive (`POS`) and negative (`NEG`) portions of the leaf at Z=`ZMIN`.

Note that MLCE input allows you to apply values of `POS` and `NEG` to groups of adjacent leaves, which can save tedious input in the case where many adjacent leaves have the same opening coordinates. See the input description below for more details.

The input format for MLCE and an example input are given below.

```
CARDS CM_$MLCE
**************
-1  Dummy line to indicate start of CM

 0  RMAX_CM(ICM_$MLCE) (F10.0):  Half-width of CM boundary (cm).

 1  TITLE_$MLCE (60A1):  Title of CM.

 2  ORIENT_$MLCE (I5) = 0 for leaves parallel to Y direction
                      = 1 for leaves parallel to X direction

 3  NUM_LEAF_$MLCE: Number of leaves.  Note: this must be even.

 4  ZMIN_$MLCE,ZMAX_$MLCE (2F15.0): upper and lower z coordinates
       of leafbank (before tilt, see below)

 5  ZSTEPL_$MLCE, ZSTEPR_$MLCE: Z-coordinates of left and right step
       in central leaf (an imaginary, unrotated leaf on the Z axis).

 6  TGW_$MLCE (F15.0): X (ORIENT_$MLCE=0) or Y (ORIENT_$MLCE=1)
       width of steps in central leaf (cm).

 7  X3_$MLCE, X4_$MLCE (2F15.0): X (ORIENT_$MLCE=0) or Y
        (ORIENT_$MLCE=1) coordinates of the upper right and
        lower right corners of central leaf, ignoring steps defined above.

 8  SPACE_$MLCE, SSD_$MLCE (2F15.0)

        SPACE_$MLCE: distance between centres of adjacent leaves in X
                    (ORIENT_$MLCE=0) or Y (ORIENT_$MLCE=1) direction
                    as projected to SSD_$MLCE (cm).
          SSD_$MLCE: distance from Z=0 at which SPACE_$MLCE is
                    defined (cm).

        Leaf numbers I= 1-NUM_LEAF_$MLCE/2 are created by rotating a
        duplicate of the central leaf about the axis X=0 (if
        ORIENT_$MLCE=0) or Y=0 (if ORIENT_$MLCE=1),
        Z=ZMIN_$MLCE by an angle:
        ARCTAN(-(2I-1)*SPACE_$MLCE/2.*ZMIN_$MLCE/SSD_$MLCE)
        and then translating it in the X (if ORIENT_$MLCE=0) or
        or Y (ORIENT_$MLCE=1) direction by a distance
```

```
      -(2I-1)*SPACE_$MLCE/2.*ZMIN_$MLCE/SSD_$MLCE
```

Leaf numbers I= NUM_LEAF_$MLCE/2+1 NUM_LEAF_$MLCE are created by
rotating a duplicate of the central leaf about the axis X=0
(if ORIENT_$MLCE=0) or Y=0 (if ORIENT_$MLCE=1), Z=ZMIN_$MLCE by:
ARCTAN((2I-1)*SPACE_$MLCE/2.*ZMIN_$MLCE/SSD_$MLCE)
and then translating it in the X (if ORIENT_$MLCE=0) or
or Y (ORIENT_$MLCE=1) direction by a distance
(2I-1)*SPACE_$MLCE/2.*ZMIN_$MLCE/SSD_$MLCE

9   LBROT_$MLCE (F15.0): Leaf bank rototian angle (tilt) about
     X=0 (ORIENT_$MLCE=0) or Y=0 (ORIENT_$MLCE=1) and Z=ZMIN_$MLCE
     (radians). This is applied to the leaves after they have been
     translated/rotated according to SPACE_$MLCE, SSD_$MLCE above.

10 ENDTYPE_$MLCE (I5) : The type of leaf end :
          0 -- rounded (cylindrical) leaf end and
          1 -- focused divergent leaf end.

IF ENDTYPE_$MLCE=0
11 LEAFRADIUS_$MLCE,CIL_$MLCE (2F15.0)

       LEAFRADIUS_$MLCE: Radius curvature leaf ends
              CIL_$MLCE: Z position from which LEAFRADIUS_$MLCE is
                          defined

IF ENDTYPE_$MLCE=1
11 ZFOCUS_$MLCE (F15.0): Z position of focal point of leaf ends

 Repeat 12 until coordinates of all leaves are defined once. Leaves
 are numbered 1,2,...NUM_LEAF_$MLCE, where numbering goes from leaf
 1 to leaf NUM_LEAF_$MLCE. Convention is lower to upper or
 left to right depending on ORIENT_$MLCE i.e from negative to
 positive.

12 NEG_$MLCE, POS_$MLCE, NUM_$MLCE (2F15.0,I5)

   NEG_$MLCE: Min. Y (ORIENT_$MLCE=0) or X (ORIENT_$MLCE=1)
                   of a) opening in leaf I at ZMIN_$MLCE (ENDTYPE=1)
                   or b) of origin of cylindrical leaf end (ENDTYPE=0)
   POS_$MLCE: Max. Y (ORIENT_$MLCE=0) or X (ORIENT_$MLCE=1)
                   of a) opening in leaf I at ZMIN_$MLCE (ENDTYPE=1)
                   or b) of origin of cylindrical leaf end (ENDTYPE=0)
   NUM_$MLCE: Apply NEG_$MLCE and POS_$MLCE to leaves
                   I,...,I+NUM_$MLCE-1. Defaults to 1 if set <=0.
                   Defaults to NUM_LEAF_$MLCE-I+1 if set >
                   NUM_LEAF_$MLCE-I+1.

13   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in opening(s) and
```

                                                              air gaps (2F15.0,I5)


        ECUT, PCUT:   Cutoff energies for electrons and photons.
        DOSE_ZONE:   Dose scoring flag, 0 to not score dose
   IREGION_TO_BIT:   Bit number associated with this region


 14  MED_IN (24A1):  Medium in opening(s) and air gaps
                     used to set MED_INDEX.


 15  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in leaves (2F15.0,I5):


        ECUT, PCUT:   Cutoff energies for electrons and photons.
        DOSE_ZONE:   Dose scoring flag, 0 to note score dose
   IREGION_TO_BIT:   Bit number associated with this region


 16  MED_IN (24A1):  Medium of leaves,
                     used to set MED_INDEX.


  Example
  *******


  The following example defines a multi-leaf collimator design based
  loosely on that used with the Elekta SLiplus 40 leaf pair.
  Actual parameters are DIFFERENT - this serves just as a template.
  Do not attempt to use these parameters for a simulation of the real
  machine.


  The collimator starts at Z=30 cm and has 40 tungsten leaves opening
  in the X direction.  The leaves are each ~0.4cm wide and 7.0 cm
  thick.  In this example, the leaf openings will form a barbel shape
  with its long axis parallel to Y.  It will be slightly off-centre
  due to the leaf bank rotation of -0.01 rads.


  Electrons and photons in both the collimator and the opening regions
  will be followed down to kinetic energies of 10 keV (ECUT=0.521,
  PCUT=0.01).  Dose deposited in the tungsten leaves will be stored
  in dose zone 2, and dose deposited in the opening will be stored
  in dose zone 1.


  In this example the numbers are only approximate, more details should
  be obtained from the vendor
  26.0,            RMAX_CM
  MLC based on mock 40 leaf pair Elekta SLiplus type of accelerator
  1,               Leaves open in X direction
  40,              40 leaf paires
  30.0,37.0,       ZMIN,ZMAX
  34.0, 33.5,      ZSTEPL,ZSTEPR
  0.04,            step width

```
 0.17,0.2,        X3, X4 of central leaf
 1.2, 100.0,      leaf centres spaced 1.2 cm apart projected to SSD=100cm
 -0.01,           leaf bank tilt angle (radians)
 0,               ENDTYPE IS CURVED
 15.0,33.5,       curvature radius, zposition cylinder axis curvature
-15.0, 15.0, 16
-17.0, 17.0, 2
-16.0, 16.0, 4
-17.0, 17.0, 2
-15.0, 15.0, 16
0.5210,  0.010,  1,  0
AIR700ICRU
0.5210,  0.010,  2,  0
W700ICRU
```

15.3.17 DYNVMLC

DYNVMLC is a CM specifically designed to model the Varian Millenium multi-leaf collimator. The code is based on VARMLC. The user specifies cross-sections perpendicular to the leaf opening direction for the 3 leaf types (FULL, TARGET and ISOCENTER leaves) found in the mlc. Each leaf in the leaf bank (usually comprising 120 leaves in all) is assigned a type, with TARGET/ISOCENTER leaves always occuring in pairs. The user also specifies a Z focal point for the leaf sides and either the radius (for cylindrical ends) or Z focal point (for straight ends) for the leaf ends. Note that DYNVMLC leaf cross sections are more complex than those in the VARMLC CM.

Unique among the CMs simulating multileaf collimators, DYNVMLC allows the simulation of multiple fields (where a field is defined by a complete set of leaf openings) during a single run. Leaf opening coordinates can be simulated changing while the beam is on (dynamic, or `MODE=1`, simulation) or while the beam is off (step-and-shoot, or `MODE=2`, simulation). For these simulations, the user must supply a file containing the leaf opening data for each field and the relative intensity of each field. Alternatively, DYNVMLC can be used to simulate a fixed set of leaf opening coordinates for the entire run (static, or `MODE=0`, simulation).

DYNVMLC was originally coded by Emily Heath at McGill University. There have been some modifications of inputs and some bugs were fixed.

Figure 35 shows an example DYNVMLC with 8 leaves (`NUM_LEAF`=8) opening in the Y direction (`ORIENT`=0). The X cross-sections of the 3 leaf types, along with the dimensions that you must specify, are shown in detail at the top. The meanings of the labels are shown in Table 3.

FULL and ISOCENTER leaves extend from `ztip` to `ZMIN+ZTHICK`, while TARGET leaves extend from `ZMIN` to `ztip`. For each leaf, you must also specify the Z dimensions of a driving screw hole (filled with `MED3`) that spans the entire width of the leaf cross-section. When specifying the cross-sections of each leaf type, the grid lines shown in Figure 35 must not change order, however they can overlap. This means, for example, that when specifying the FULL leaf dimensions `zt` $\leq$ `zg`, `wt` $\leq$ `wtip` $\leq$ `wt + wts - wbs`, etc. Also, Z positions of tongues and grooves must allow leaves to fit together in the following combinations: FULL/FULL (ie `zt` $\leq$ `zg`), FULL/TARGET (ie $zg_{FULL} \geq zt_{TARGET}$), TARGET/ISOCENTER (ie zg_{TARGET} $\leq zt_{ISOCENTER}$), ISOCENTER/TARGET (ie $zg_{ISOCENTER} \geq zt_{TARGET}$) and ISOCENTER/FULL (ie $zg_{ISOCENTER} \geq zt_{FULL}$). All of these requirements put fairly tight restrictions on cross-section dimensions.

Not shown in the leaf cross-sections is the distance, `HOLEPOS`, between the leaf end and the end of the driving screw hole in the leaf opening direction. This must also be specified for each leaf type.

Once you have specified the cross-sections for the 3 leaf types, then you must assign a type to each leaf in the leaf bank. Since TARGET/ISOCENTER leaves must always occur in pairs (with the TARGET leaf on the negative X side if `ORIENT`=0 or on the negative Y side if `ORIENT`=1), you really only have a choice of two leaf types: 1. FULL leaf or 2. TARGET/ISOCENTER pair. You can also specify the number of adjacent leaves that have the same type so that you do not have to input the type for each leaf or leaf pair separately.

DYNVMLC

Figure 35: Example DYNVMLC with 8 leaves (NUM_LEAF=8) opening in the Y direction (ORIENT=0). At the top of the figure are detailed X cross-sections of the 3 leaf types, FULL, TARGET and ISOCENTER, showing the dimensions that the user must input. The dimension labels are explained in the text below. The Y cross-section shows the two possible leaf end types: cylindrical (ENDTYPE=0) and straight (ENDTYPE=1).

Table 3: Meaning of labels in Figure 35.

Leaf Type

	FULL	TARGET	ISOCENTER
widths:			
`wl`	leaf (excl. tongue)	leaf (excl. tongue)	leaf (excl. tongue)
`wt`	tongue	tongue	tongue
`wg`	groove	groove	groove
`wtip`	tip at top of leaf	tip at bottom of leaf	tip at top of leaf
`wts`	top support rail	top support rail	top support rail
`wbs`	bottom support rail	bottom support rail	bottom support rail
Z positions:			
`ztip`	top of tip	bottom of tip	top of tip
`zl`	top of leaf	bottom of leaf	top of leaf
`zt`	bottom of tongue	bottom of tongue	top of tongue
`zg`	bottom of groove	top of groove	bottom of groove
`zth`	top of driving screw hole	top of driving screw hole	top of driving screw hole
`zbh`	bottom of driving screw hole	bottom of driving screw hole	bottom of driving screw hole
`zts`	top of support rail	top of support rail	top of support rail
`zbs`	bottom of support rail	bottom of support rail	bottom of support rail

In the example shown in Figure 35, leaves 1,2,7,8 are FULL (type 1) and leaves 3-6 are TARGET/ISOCENTER pairs (type 2).

The X cross-section of the entire leaf bank shows how the different leaf types fit together with interlocking steps and with an air gap between each leaf specified by the `LEAFGAP` input. The negative-most leaf begins at X=`START`. All leaf sides are focused on the Z axis at the user-specified focal point, `ZFOCUS(1)`. The focusing of leaf sides means that leaf widths are not uniform from leaf top to bottom (although in a realistic case, where the MLC is close to the SSD and `ZFOCUS(1)`=0, the variation in widths will be slight), and it is important to note that all X (`ORIENT`=0) or Y (`ORIENT`=1) dimensions that you input (whether they be widths for the individual leaf types or dimensions relevant to the entire leaf bank, such as `LEAFGAP` or `START`) are specified at `ZMIN`.

The Y cross-section in Figure 35 shows the two possible leaf end types: cylindrical (`ENDTYPE`=0) and straight focused (`ENDTYPE`=1). Cylindrical leaf ends have radius `LEAFRADIUS` with the origin of the radius at Z=`ZMIN`+`ZTHICK`/2 (ie at the mid-point of the leaf thickness). The negative and positive dimensions of the opening in a leaf with cylindrical ends are specified at Z=`ZMIN`+`ZTHICK`/2. Straight leaf ends are focused to Z=`ZFOCUS(2)`, and the negative and positive dimensions of the opening are specified at Z=`ZMIN`. The Y cross-section also shows how the distance from the end of the driving screw hole to the leaf end, `HOLEPOS`, is defined. In the case of cylindrical ends, it is the distance from the leaf end at Z=`ZMIN`+`ZTHICK`/2, while for straight ends, it is the distance from the end at Z=`ZMIN`. `HOLEPOS` remains the same for a given leaf type (FULL, TARGET or ISOCENTER) regardless of the leaf opening dimensions.

For static (`MODE=0`, the default) simulations, in which leaf openings are fixed for the entire simulation, the leaf openings coordinates are specified in the GUI or input file in a manner similar to MLC, MLCQ, VARMLC, etc. Note that the user can specify the number of adjacent leaves that have the same opening dimensions, thus potentially saving input lines. For dynamic (`MODE=1`) or step-and-shoot (`MODE=2`) simulations, where multiple treatment fields are simulated in a single run, however, the user must supply a file of leaf opening data. The format of this file is:

```
TITLE
NFIELDS
FOR I=1,NFIELDS[
  INDEX(I)
  NEG, POS, NUM -- repeat until NEG, POS have been defined for all leaves
]
```

where `TITLE` is a title line for the file, `NFIELDS` is the number of treatment fields, `INDEX(I)` is the index of treatment field I, `NEG` is the lower Y (`ORIENT=0`) or X (`ORIENT=1`) coordinate of the leaf opening, `POS` is the upper Y or X coordinate of the leaf opening, and `NUM` is the number of adjacent leaves for which `NEG` and `POS` apply (defaults to 1 if left blank). Note that the `NEG`, `POS`, `NUM` line must be repeated until the opening coordinates for all leaves have been defined for field I.

The treatment field indices, `INDEX(I)`'s, are numbers in the range [0,1] (with `INDEX(I)`>`INDEX(I-1)`) which determine which field is used in a particular history. At the beginning of each history a random number in the range [0,1], `RNDM1`, is compared to (`INDEX(I)`,I=1,`NFIELDS`). The lowest value of I for which `INDEX(I)`≥`RNDM1` is the field used in the history. Note that this means that `INDEX(I)`-`INDEX(I-1)` is a measure of the probability, or intensity, of field I. In the case of step-and-shoot (`MODE=2`) runs, the leaf opening coordinates are simply those defined for field I. In the case of dynamic (`MODE=1`) simulations, however, `RNDM1` is used to calculate intermediate opening coordinates between fields I and I-1 to simulate leaf coordinates changing while the beam is on. For example, if `NEG(J,I)` and `NEG(J,I-1)` are the lower coordinates of leaf J in fields I and I-1, respectively, then the intermediate lower coordinate used, `NEG_INT` is given by:

$$\texttt{NEG\_INT} = \texttt{NEG}(\texttt{J}, \texttt{I} - 1) + (\texttt{NEG}(\texttt{J}, \texttt{I}) - \texttt{NEG}(\texttt{J}, \texttt{I} - 1)) * \frac{(\texttt{RNDM1} - \texttt{INDEX}(\texttt{I} - 1))}{(\texttt{INDEX}(\texttt{I}) - \texttt{INDEX}(\texttt{I} - 1))} \quad (9)$$

A sample file containing leaf opening data for DYNVMLC is included with the distribution.

Similar to VARMLC, DYNVMLC has an `IGNOREGAPS` input (appearing on the same line as `ECUT`, `PCUT` etc for the leaves) which can be set to 1 to instruct the code to ignore all air gaps within the leaf bank (ie gaps between leaves and Z-directional gaps caused by the presence of leaf tips, support rails, etc) and the driving screw holes when doing charged particle range rejection provided that the particle is in the leaf medium and:

particle X < minimum X of all leaf openings (excluding rounded or focused leaf end) or particle X > maximum X of all leaf openings (excluding leaf end)

if the leaves are parallel to X (`ORIENT=1`) or:

particle Y < minimum Y of all leaf openings (excluding leaf end) or
particle Y > maximum Y of all leaf openings (excluding leaf end)

if the leaves are parallel to Y (`ORIENT=0`). Exact transport is preserved in the shaped leaf ends (rounded or focused) by excluding them from the volume in which air gaps and driving screw holes are ignored. Setting `IGNOREGAPS=1` greatly increases the efficiency of range rejection in DYNVMLC and can reduce CPU time spent in the CM by a factor of 2. However, if you are concerned that the air gaps and driving screw holes have a significant effect on the beam field, we recommend that you run with `IGNOREGAPS=0` to preserve exact transport everywhere. Note that in the case of dynamic or step-and-shoot simulations, the minimum and maximum opening coordinates must be recalculated at the beginning of each history.

The input format for DYNVMLC and an example input are given below.

```
CARDS CM_$DYNVMLC
*******************
-1  Dummy line to indicate start of CM

 0  RMAX_CM(ICM_$DYNVMLC) (F10.5):  Half-width of CM boundary (cm).

 1  TITLE_$DYNVMLC (60A1):  Title of CM.

 2  ORIENT_$DYNVMLC, NGROUP_$DYNVMLC, MODE_$DYNVMLC (3I5)

            ORIENT_$DYNVMLC = 0 for leaves parallel to Y direction
                            = 1 for leaves parallel to X direction
            NGROUP_$DYNVMLC = number of groups of adjacent leaves where
                              all leaves in a group are:
                              1. FULL leaves
                              2. TARGET/ISOCENTER pairs with TARGET leaf
                                 on the -X (ORIENT=0) or -Y (ORIENT=1) side
                              NGROUP_$DYNVMLC defaults to 3 if set <=0
            MODE_$DYNVMLC = 0 for single setting of leaf openings (static
                              field)
                          = 1 for dynamic mlc delivery--simulated leaf
                              movement while beam is on
                          = 2 for step-and-shoot delivery--beam off while
                              leaf positions change

 3  ZMIN_$DYNVMLC (F15.0): Z of top of MLC (excluding airgap)

 4  ZTHICK_$DYNVMLC (F15.0): Thickness of the leaves ( z-axis (cm))

 5  LEAFWIDTH_$DYNVMLC(1), WTONGUE_$DYNVMLC(1), WGROOVE_$DYNVMLC(1),
    WTIP_$DYNVMLC(1), WRAILTOP_$DYNVMLC(1), WRAILBOT_$DYNVMLC(1),
    ZTIP_$DYNVMLC(1), ZLEAF_$DYNVMLC(1), ZTONGUE_$DYNVMLC(1),
```

```
      ZGROOVE_$DYNVMLC(1), ZHOLETOP_$DYNVMLC(1), ZHOLEBOT_$DYNVMLC(1),
   HOLEPOS_FULL_$DYNVMLC, ZRAILTOP_$DYNVMLC(1), ZRAILBOT_$DYNVMLC(1)
      (15F15.0)
```

For a FULL type leaf (all dimensions in cm--all widths are
                      projected back to ZMIN_$DYNVMLC):

```
    LEAFWIDTH_$DYNVMLC(1): Width of leaf (not including tongue)
      WTONGUE_$DYNVMLC(1): Width of tongue
      WGROOVE_$DYNVMLC(1): Width of groove
         WTIP_$DYNVMLC(1): Width of tip at top of leaf
     WRAILTOP_$DYNVMLC(1): Width of top of support rail
     WRAILBOT_$DYNVMLC(1): Width of bottom of support rail
         ZTIP_$DYNVMLC(1): Z at which tip at top of leaf begins
        ZLEAF_$DYNVMLC(1): Z of top of leaf
      ZTONGUE_$DYNVMLC(1): Z of bottom of tongue
      ZGROOVE_$DYNVMLC(1): Z of bottom of groove
     ZHOLETOP_$DYNVMLC(1): Z of top of driving screw hole
     ZHOLEBOT_$DYNVMLC(1): Z of bottom of driving screw hole
    HOLEPOS_FULL_$DYNVMLC: Distance of hole from leaf tip
     ZRAILTOP_$DYNVMLC(1): Z of top of support rail
     ZRAILBOT_$DYNVMLC(1): Z of bottom of support rail
```

   Note: Z positions are input in order of increasing Z. Thus
         ZLEAF_$DYNVMLC(1)>=ZTIP_$DYNVMLC(1), etc. See the BEAM
         manual or GUI help for restrictions on widths.


```
6  LEAFWIDTH_$DYNVMLC(2), WTONGUE_$DYNVMLC(2), WGROOVE_$DYNVMLC(2),
   WTIP_$DYNVMLC(2), WRAILTOP_$DYNVMLC(2), WRAILBOT_$DYNVMLC(2),
   ZRAILTOP_$DYNVMLC(2), ZRAILBOT_$DYNVMLC(2), ZHOLETOP_$DYNVMLC(2),
   ZHOLEBOT_$DYNVMLC(2), HOLEPOS_TAR_$DYNVMLC, ZTONGUE_$DYNVMLC(2),
   ZGROOVE_$DYNVMLC(2), ZLEAF_$DYNVMLC(2), ZTIP_$DYNVMLC(2) (15F15.0)
```

   For a TARGET type leaf (all dimensions in cm--all widths are
                      projected back to ZMIN_$DYNVMLC):

```
    LEAFWIDTH_$DYNVMLC(2): Width of leaf (not including tongue)
      WTONGUE_$DYNVMLC(2): Width of tongue
      WGROOVE_$DYNVMLC(2): Width of groove
         WTIP_$DYNVMLC(2): Width of tip at bottom of leaf
     WRAILTOP_$DYNVMLC(2): Width of top of support rail
     WRAILBOT_$DYNVMLC(2): Width of bottom of support rail
     ZRAILTOP_$DYNVMLC(2): Z of top of support rail
     ZRAILBOT_$DYNVMLC(2): Z of bottom of support rail
     ZHOLETOP_$DYNVMLC(2): Z of top of driving screw hole
     ZHOLEBOT_$DYNVMLC(2): Z of bottom of driving screw hole
     HOLEPOS_TAR_$DYNVMLC: Distance of hole from leaf tip
      ZTONGUE_$DYNVMLC(2): Z of bottom of tongue
```

```
           ZGROOVE_$DYNVMLC(2): Z of top of groove
             ZLEAF_$DYNVMLC(2): Z of bottom of leaf
              ZTIP_$DYNVMLC(2): Z of bottom of tip at bottom of leaf


    Note: Z positions are input in order of increasing Z.  Thus
           ZLEAF_$DYNVMLC(1)>=ZTIP_$DYNVMLC(1), etc.  See the BEAM
           manual or GUI help for restrictions on widths.



   7  LEAFWIDTH_$DYNVMLC(3), WTONGUE_$DYNVMLC(3), WGROOVE_$DYNVMLC(3),
      WTIP_$DYNVMLC(3), WRAILTOP_$DYNVMLC(3), WRAILBOT_$DYNVMLC(3),
      ZTIP_$DYNVMLC(3), ZLEAF_$DYNVMLC(3), ZTONGUE_$DYNVMLC(3),
      ZGROOVE_$DYNVMLC(3), ZHOLETOP_$DYNVMLC(3), ZHOLEBOT_$DYNVMLC(3),
      HOLEPOS_ISO_$DYNVMLC, ZRAILTOP_$DYNVMLC(3), ZRAILBOT_$DYNVMLC(3)
                   (15F15.0)


      For a ISOCENTER type leaf (all dimensions in cm--all widths are
                         projected back to ZMIN_$DYNVMLC):

        LEAFWIDTH_$DYNVMLC(3): Width of leaf (not including tongue)
          WTONGUE_$DYNVMLC(3): Width of tongue
          WGROOVE_$DYNVMLC(3): Width of groove
             WTIP_$DYNVMLC(3): Width of tip at top of leaf
         WRAILTOP_$DYNVMLC(3): Width of top of support rail
         WRAILBOT_$DYNVMLC(3): Width of bottom of support rail
             ZTIP_$DYNVMLC(3): Z at which tip at top of leaf begins
            ZLEAF_$DYNVMLC(3): Z of top of leaf
          ZTONGUE_$DYNVMLC(3): Z of top of tongue
          ZGROOVE_$DYNVMLC(3): Z of bottom of groove
         ZHOLETOP_$DYNVMLC(3): Z of top of driving screw hole
         ZHOLEBOT_$DYNVMLC(3): Z of bottom of driving screw hole
         HOLEPOS_ISO_$DYNVMLC: Distance of hole from leaf tip
         ZRAILTOP_$DYNVMLC(3): Z of top of support rail
         ZRAILBOT_$DYNVMLC(3): Z of bottom of support rail

    Note: Z positions are input in order of increasing Z.  Thus
           ZLEAF_$DYNVMLC(1)>=ZTIP_$DYNVMLC(1), etc.  See the BEAM
           manual or GUI help for restrictions on widths.

    Note: 1. For TARGET and ISOCENTER leaves to fit together,
             ZTONGUE_$DYNVMLC(3)>=ZGROOVE_$DYNVMLC(2) and
             ZTONGUE_$DYNVMLC(2)<=ZGROOVE_$DYNVMLC(3).
          2. For TARGET and FULL leaves to fit together (FULL
             leaf on -X [ORIENT=0] or -Y [ORIENT=1] side of TARGET
             leaf only) ZTONGUE_$DYNVMLC(2)<=ZGROOVE_$DYNVMLC(1)
          3. For ISOCENTER and FULL leaves to fit together (FULL
             leaf on +X [ORIENT=0] or +Y [ORIENT=1] side of ISOCENTER
             leaf only) ZTONGUE_$DYNVMLC(1)<=ZGROOVE_$DYNVMLC(3)
```

```
   Repeat 8 NGROUP_$DYNVMLC times

 8  NUM_LEAF_$DYNVMLC(I), LEAFTYPE  (2I5)

       NUM_LEAF_$DYNVMLC(I): Number of adjacent leaves in group I
                  LEAFTYPE: Type of leaf in group I.
                            Set to: 1 for FULL leaves
                                    2 for TARGET/ISOCENTER pair with
                                      TARGET leaf on the -X (ORIENT=0)
                                      or -Y (ORIENT=1) side


       Note: If LEAFTYPE is 2, then you must have an even number
             of leaves in the group.

 9  START_$DYNVMLC (F15.0) : the start position (cm) wrt the CAX of
                     leaf 1 as projected to ZMIN_$DYNVMLC.

10  LEAFGAP_$DYNVMLC (F15.5) : The width of the interleaf air gap
           at ZMIN_$DYNVMLC.

          Note restriction: LEAFGAP_$DYNVMLC<=WTONGUE_$DYNVMLC(1,2,3),

11  ENDTYPE_$DYNVMLC (I5) : The  type of leaf end :
            0 -- rounded leaf end and
            1 -- focused divergent leaf end.

12  ZFOCUS_$DYNVMLC (F15.5) : Focal point on Z-axis of leaf ends
                   (i.e. imaginary lines drawn extending the slopes
                   of leaf ends will all intersect the Z-axis
                   at this point) - chosen if ENDTYPE_$DYNVMLC = 1.

      Note restriction: ZFOCUS_$DYNVMLC(1) < ZMIN_$DYNVMLC or
                                    > ZMIN_$DYNVMLC + ZTHICK_$DYNVMLC

    LEAFRADIUS_$DYNVMLC (F15.5) : Radius of the leaf end if
                        ENDTYPE_$DYNVMLC = 0. This must be greater
                        than or equal to half the leaf thickness.

13  ZFOCUS_$DYNVMLC(1) (F15.5): Focal point on Z-axis of leaf sides
                     imaginary lines drawn extending the slopes of
                     the leaf sides will all intersect the Z-axis
                     at this point)

      Note restriction: ZFOCUS_$DYNVMLC(1) < ZMIN_$DYNVMLC or
                                  > ZMIN_$DYNVMLC + ZTHICK_$DYNVMLC

                    For focused ends the leaf position is defined
                    at ZMIN_$DYNVMLC; for rounded at ZMIN_$DYNVMLC +
```

```
                         0.5*ZTHICK_$DYNVMLC (ie center of the leaf in z)
```

   If MODE_$DYNVMLC=0 (static field):

   Repeat 14a until opening coordinates of all leaves are defined once.
   Leaves are numbered 1,2,...TOT_LEAF_$DYNVMLC, where numbering goes from
   leaf 1 to leaf TOT_LEAF_$DYNVMLC. Convention is lower to upper or
   left to right depending on ORIENT_$DYNVMLC i.e from negative to
   positive. Note that for dynamic or step-and-shoot simulations, these
   are the default coordinates, used unless specified otherwise in the
   file of leaf opening data input in line 14a (see below).

14a NEG_$DYNVMLC, POS_$DYNVMLC, NUM_$DYNVMLC (2F15.5,I5)

   NEG_$DYNVMLC: Min. Y (ORIENT_$DYNVMLC=0) or X (ORIENT_$DYNVMLC=1)
                  of front opening in leaf I (ie the opening at
                  ZMIN_$DYNVMLC) if ENDTYPE=1, or of rounded end
                  of leaf I if ENDTYPE=0.
   POS_$DYNVMLC: Max. Y (ORIENT_$DYNVMLC=0) or X (ORIENT_$DYNVMLC=1)
                  of front opening in leaf I if ENDTYPE=1, or of
                  rounded end of leaf I if ENDTYPE=0.
   NUM_$DYNVMLC: Apply NEG_$DYNVMLC and POS_$DYNVMLC to leaves
                  I,...,I+NUM_$DYNVMLC-1. Defaults to 1 if set <=0.
                  Defaults to TOT_LEAF_$DYNVMLC-I+1 if set >
                  TOT_LEAF_$DYNVMLC-I+1.

 If MODE_$DYNVMLC=1 or 2 (dynamic delivery or step-and-shoot delivery):

14b mlc_file (A80)

       mlc_file: The full name of the file containing leaf opening
                  data. The format of the file contents is as follows:

   MLC_TITLE (A80)
   NFIELDS_$DYNVMLC (I10)
   FOR I=1,NFIELDS_$DYNVMLC[
    INDEX_$DYNVMLC(I) (F15.0)
    NEG_$DYNVMLC, POS_$DYNVMLC, NUM_$DYNVMLC (2F15.0,I5) -- repeat this
                                                            line until
                                                            coordinates
                                                            for all leaves
                                                            have been
                                                            defined for
                                                            field I.
   ]

   where:

         MLC_TITLE: A title line
```

```
       NFIELDS_$DYNVMLC:   Total number of fields
      INDEX_$DYNVMLC(I):   Index of field I. 0 <= INDEX_$DYNVMLC(I) <= 1 and
                           INDEX_$DYNVMLC(I) > INDEX_$DYNVMLC(I-1).  This
                           number is compared to a random number on (0,1) at
                           the start of each history; if the random number is
                           <= INDEX_$DYNVMLC(I), then field I is used.
          NEG_$DYNVMLC:   Min. Y (ORIENT_$DYNVMLC=0) or X (ORIENT_$DYNVMLC=1)
                           of front opening in leaf (ie the opening at
                           ZMIN_$DYNVMLC) if ENDTYPE=1, or of rounded end
                           of leaf if ENDTYPE=0 for leaf J in field I.
          POS_$DYNVMLC:   Max. Y (ORIENT_$DYNVMLC=0) or X (ORIENT_$DYNVMLC=1)
                           of front opening in leaf if ENDTYPE=1, or of
                           rounded end of leaf if ENDTYPE=0 for leaf J in
                           field I.
          NUM_$DYNVMLC:   Apply NEG_$DYNVMLC and POS_$DYNVMLC to leaves
                           J,...,J+NUM_$DYNVMLC-1.  Defaults to 1 if set <=0.
                           Defaults to TOT_LEAF_$DYNVMLC-J+1 if set >
                           TOT_LEAF_$DYNVMLC-J+1.
```

  Note that the inputs NEG_$DYNVMLC, POS_$DYNVMLC and NUM_$DYNVMLC have
  the same meanings as in 14a (static field inputs) but that they must
  now be repeated for every field I.

```
 15    ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in opening(s) and
                                            air gaps (2F15.5,2I5)


       ECUT, PCUT:   Cutoff energies for electrons and photons.
        DOSE_ZONE:    Dose scoring flag, 0 to not score dose
    IREGION_TO_BIT:    Bit number associated with this region


 16  MED_IN (24A1):   Medium in opening(s) and air gaps
                      used to set MED_INDEX.


 17  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in leaves, IGNOREGAPS_$DYNVMLC
                                               (2F15.0,3I5):


       ECUT, PCUT:   Cutoff energies for electrons and photons.
        DOSE_ZONE:    Dose scoring flag, 0 to note score dose
    IREGION_TO_BIT:    Bit number associated with this region
       IGNOREGAPS:    If set to 1, ignore all air gaps and driving screw
                      holes when doing range
                      rejection in leaf material when the particle X position
                      is < min X of all leaf openings (not including leaf
                      ends) or > max X of leaf openings (not including ends)
                      (ORIENT_$DYNVMLC=1) or if the particle Y position
                      is < min Y of all leaf openings (not including leaf
                      ends) or > max Y of leaf openings (not including ends)
                      (ORIENT_$DYNVMLC=0). This approximation is designed
                      to make range rejection more efficient deep in the
```

                    leaves, while still preserving accurate transport
                    in the leaf ends. Note that if you have significant
                    air gaps between leaves or are concerned with the
                    effects of the driving screw holes it is recommended
                    that you not use this option (ie run with the default
                    setting of 0).

  18 MED_IN (24A1): Medium of leaves,
                    used to set MED_INDEX.

  19 ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in driving screw holes
                                      (2F15.5,2I5):

      ECUT, PCUT: Cutoff energies for electrons and photons.
       DOSE_ZONE: Dose scoring flag, 0 to note score dose
  IREGION_TO_BIT: Bit number associated with this region

  20 MED_IN (24A1): Medium in driving screw holes,
                    used to set MED_INDEX.

  Example
  ****

  The following example defines a multi-leaf tungsten collimator design
  based loosely on that used with the Varian Millenium MLC.
  Actual parameters are DIFFERENT - this serves just as a template.
  Do not attempt to use these parameters for a simulation of the real
  machine.

  The collimator starts at Z=48.25 cm and has 60 tungsten leaves opening
  in the X direction. Leaves 1-10 and 51-60 are FULL and leaves 11-50 are
  TARGET/ISOCENTER pairs. The Z focus of the leaf sides is at Z=0 cm
  which is the position of the source. The leaf ends are rounded with a
   radius of 8 cm. In this example, leaf opening coordinates are chosen to
  create a square of width ~ 2cm centred on the beam axis.

  Electrons and photons in both the collimator and the opening regions
  will be followed down to kinetic energies of 189 keV (ECUT=0.7,
  PCUT=0.01). Dose deposited in the tungsten leaves will be stored
  in dose zone 2, and dose deposited in the opening will be stored
  in dose zone 1.

```
20.5, RMAX
CL21X - Millenium MLC
1, 3, ORIENT, NGROUP
48.25, ZMIN
6.7, ZTHICK
0.5, 0.04, 0.04, 0.1354, 0.3252, 0.1227, 48.25, 48.533, 51.524, 51.732,
   52.98, 53.28, 2, 54.5474, 54.812, FULL leaf
```

```
0.25, 0.04, 0.04, 0.0929, 0.1371, 0.1371, 48.345, 48.6096, 49.5277,
  49.8277, 2, 51.625, 51.627, 54.7, 54.746, TARGET leaf
0.25, 0.04, 0.04, 0.0354, 0.1285, 0.1235, 48.412, 48.531, 51.631, 51.732,
  53.3293, 53.6293, 2, 54.5474, 54.812, ISOCENTER leaf
10, 1, FULL leaves
40, 2, TARGET/ISOCENTER pairs
10, 1, FULL leaves
-10.2, START
0.006, LEAFGAP
0, ENDTYPE
8, ZFOCUS or RADIUS of leaf ends
0, ZFOCUS of leaf sides
0, 0, 26
-1.0, 1.0, 8
0,0,26
0.7, 0.01, 1, 0,
AIR521ICRU
0.7, 0.01, 2, 0, 0,
W521ICRU
0.7, 0.01, 3, 0,
AIR521ICRU
```

15.3.18   MESH

The MESH CM models a single-layer wire mesh placed perpendicular to the beam direction in the path of the beam. Details are shown in Figure 36 below.
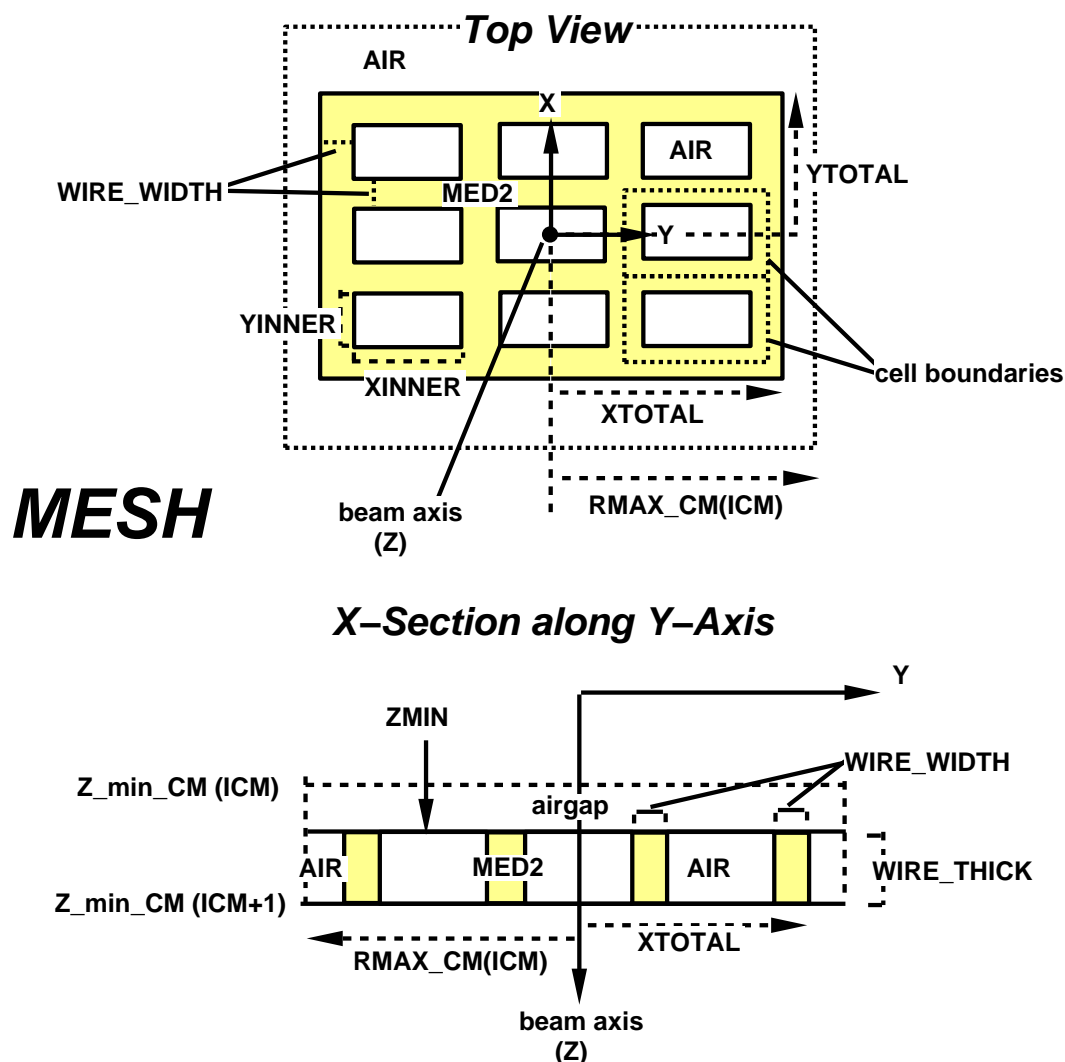
Figure 36: A MESH component module. The top view shows the identical rectangular air "holes", each with user-input dimensions XINNER and YINNER, separated from each other and the edge of the mesh by wire of user-input width, WIRE_WIDTH. The outer boundaries of the mesh are defined by half- widths XTOTAL and YTOTAL input by the user. Note that there is always one air hole centred on the beam axis. The top view also shows that a mesh "cell" is a single air hole plus half of the wire surrounding it. From a coding point of view, the mesh is made up of adjacent cells which behave identically with respect to particle transport. Thus the code actually only models one cell (the central one), and particles are translated from outer cells to their equivalent positions in this cell before transport. If the user-input values of XTOTAL and/or YTOTAL happen to cut across cells, the outer boundaries are automatically adjusted to accommodate these cells. The cross-section along the Y-axis shows the Z position of the front of the mesh, ZMIN, and the thickness of the mesh, WIRE_THICK, both input by the user. Note that it is possible to have an air gap above the mesh. For dose scoring and bit setting purposes, the wire is considered a single region, and the air in the holes and any air surrounding the mesh (excluding the air gap) are considered another region.

The input format for MESH and an example input are given below.

```
CARDS CM_$MESH   (MESH: Rev 1.4)
**************

-1  dummy line (filled with ****)  read in main

 0  RMAX_CM        outer boundary for CM - 1/2 side of square(read in main)

 1  TITLE_$MESH (60A1):  Title of CM.

 2  ZMIN_$MESH (F10.0):  Z position of front of MESH (excluding airgap).

 3  X_AIR_WIDTH_$MESH, Y_AIR_WIDTH_$MESH, WIRE_WIDTH_$MESH,
    WIRE_THICK_$MESH (4F15.0):

        X_AIR_WIDTH_$MESH: X width of each air region in mesh (cm)
        Y_AIR_WIDTH_$MESH: Y width of each air region in mesh (cm)
                           Y_AIR_WIDTH_$MESH defaults to
                           X_AIR_WIDTH_$MESH if it is set to 0.
       WIRE_WIDTH_$MESH:  Width of wire in the mesh (cm)
       WIRE_THICK_$MESH:  Thickness of wire in the mesh (cm)

      Note restrictions: X_AIR_WIDTH_$MESH,Y_AIR_WIDTH_$MESH,
                         WIRE_WIDTH_$MESH must all be >= 0.0001 cm

 4  XTOTAL_$MESH, YTOTAL_$MESH (2F15.0):

        XTOTAL_$MESH: Half-width of outer X dimension of mesh (cm)
        YTOTAL_$MESH: Half-width of outer Y dimension of mesh (cm)

      Note: XTOTAL_$MESH,YTOTAL_$MESH default to RMAX_CM if set to 0
            Also, if they fall within mesh cells (individual air regions +
               1/2 of the wire surrounding them) they are pushed out
               to include those cells.

 5  ECUT,PCUT,DOSE_ZONE,IR_TO_BIT for air inside and
                                  surrounding mesh (2F15.0,2I5):

       ECUT, PCUT:  Cutoff energies for electrons and photons.
        DOSE_ZONE:   Dose scoring flag, 0 to not score dose
    IREGION_TO_BIT:   Bit number associated with these regions

 6  ECUT,PCUT,DOSE_ZONE,IR_TO_BIT in local region 2 (wire region)
                                  (2F15.0,2I5):

       ECUT, PCUT:   Cutoff energies for electrons and photons.
        DOSE_ZONE:   Dose scoring flag, 0 to not score dose
    IREGION_TO_BIT:   Bit number associated with this region
```

```
7  MED_IN (24A1):    medium of local region 2 (wire), used to set
                      MED_INDEX
```

```
Example
*******
```

The following input example describes a lead mesh placed at Z=100 cm
in a beam. The mesh is 0.0299 cm thick, has square air regions
with dimensions 0.2159x0.2159cm, and the lead wire separating the air
regions is 0.0381cm wide. Although the RMAX_CM is 10cm, the outer
boundary of the mesh itself, as input, is a 6x6cm square. MESH
automatically adjusts this boundary to 6.2421x6.2421cm to accommodate
exactly 49 mesh cells (air regions + half of the wire width
surrounding them) in the X and Y directions plus the extra half-thickness
of wire surrounding the entire mesh and not belonging to any cell.

ECUT and PCUT in all regions are set to 0.521MeV and 0.01MeV
respectively. Air regions (cell air + air surrounding mesh) will
have dose scored in zone 1 and is associated with bit # 1. The
lead wire has dose scored in zone 2 and is associated with bit # 2.

Note that MESH cannot be the first CM in a simulation, and the
example given here must be preceeded by at least an airgap modelled
by another type of CM.

```
10.00000,                         Outer boundary
lead mesh: 0.0381cm with 0.2159cm air
100.0
0.2159,  0.0,  0.0381, 0.0299,  depth ensures equal mass as circle
6.0,  6.0,                       outer boundary fills entire beam
0.521,  0.01,  1,  1,            air is bit 1
0.521,  0.01,  2,  2,            wire is bit 2
PB700ICRU
```

15.3.19   MIRROR

MIRROR is used for a mirror in the accelerator. It can have arbitrary angle < 85 degrees
with respect to the Z axis (for angles between 85 and 90 degrees, approximate using SLABS).
The mirror portion itself can be made up of an arbitrary number of layers having different
thicknesses and media. MIRROR has square symmetry.
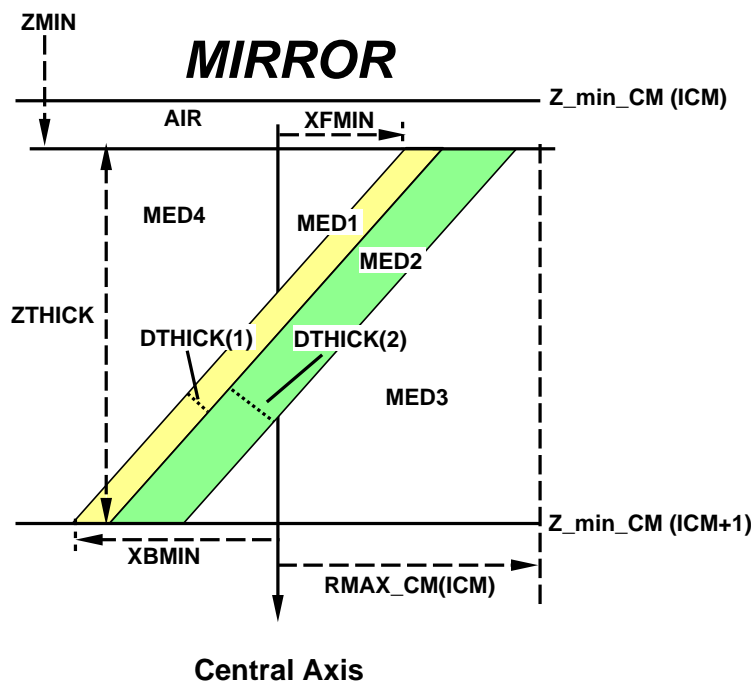
Figure 37: MIRROR example made up of 2 layers (N=2). The angle of the mirror with
respect to the Z-axis is determined by XFMIN and XBMIN, the positions of the front face
(layer 1) of the mirror, and ZTHICK thickness of the mirror in the Z-direction. This angle
must be > 5 degrees. The thickness of layer i of the mirror is given by DTHICK(i). Note
that layer i+1 of the mirror is flush with the right-most face of layer i. Thus, it is possible
to specify a mirror where some layers are out of the path of the beam. It is even possible to
specify XFMIN and XBMIN so that the front face of the mirror (the front face of layer 1) is out
of the path of the beam; BEAMnrc checks for this and gives a warning message. The mirror
extends to +−RMAX_CM in the Y-direction. The media in front of and behind the mirror,
MED4 and MED3 in the figure, will usually be air but need not be.

The input format for MIRROR, and an example of input file are given as follows.

```
CARDS CM_$MIRROR
**************

-1  Dummy line to indicate start of CM.

 0  RMAX_CM(ICM_$MIRROR) (F10.0): Half-width of CM boundary (cm).

 1  TITLE_$MIRROR (60A1): Title of CM.

 2  ZMIN_$MIRROR,ZTHICK_$MIRROR (2F15.0):

        ZMIN_$MIRROR:    Distance from front of CM(excluding air gap) to
                         ref plane(Z=0).
        ZTHICK_$MIRROR: Z-direction span.

 3  XFMIN_$MIRROR, XBMIN_$MIRROR (2F15.0):

        XFMIN_$MIRROR:  X value at which front face of mirror
                        intersects ZMIN_$MIRROR.
        XBMIN_$MIRROR:  X value at which front face of mirror
                        intersects ZMIN_$MIRROR + ZTHICK_$MIRROR.
   Note restriction:
  5 degrees<ATAN(ZTHICK_$MIRROR/(XFMIN_$MIRROR-XBMIN_$MIRROR)<90 degrees

 4  N_$MIRROR, (I10):    Number of layers

Repeat 5 for I=1,N_$MIRROR

 5  DTHICK_$MIRROR(I) (F15.0): Thickness of layer I in mirror,
                               in cm.  Layer 1 is the front face
                               of the mirror.

Repeat 6 and 7 for I=1,N_$MIRROR.

 6  ECUT, PCUT,DOSE_ZONE, IREGION_TO_BIT (2F15.0, 2I5): for each layer

        ECUT, PCUT:  Cutoff energies for electrons and photons in
                     layer I.
        DOSE_ZONE:   Dose scoring flag in layer I.
   IREGION_TO_BIT:   bit to associate local region of layer I with

 7  MED_IN (24A1):  Medium of layer I, used to set MED_INDEX.

 8  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in region behind mirror
                                           (2F15.0,2I5):

        ECUT, PCUT:  Cutoff energies for electrons and photons.
```

```
        DOSE_ZONE:    Dose scoring flag, 0 to score dose deposited in it
  IREGION_TO_BIT:    bit to associate local region with


 9  MED_IN (24A1):  Medium of local region behind the mirror,
                      used to set MED_INDEX.


10  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in region in front of
                                      mirror(2F15.0,2I5):


      ECUT, PCUT:    Cutoff energies for electrons and photons.
        DOSE_ZONE:    Dose scoring flag, 0 to score dose deposited in it
  IREGION_TO_BIT:    bit to associate local region with


11  MED_IN (24A1): Medium of region in front of mirror,
                      used to set MED_INDEX.



Example
*******

The following describes a mirror composed of 1 AL slab.
The front face of the mirror intersects the front of the
CM (excluding any airgap) at X=3cm and the back of the CM at X=-3cm.
The mirror starts at 19.7cm and is 2cm in the z-direction.
The thickness of the AL slab is 0.00508cm.
Note that thickness is measured perpendicular to the
face of the mirror.  AIR is both in front of and behind
the AL mirror.  No dose is scored in this example.


10.00   RMAX_CM
MIRROR : z=19.5074 downstream, original, dif overlap
19.70, 2.0
3.0, -3.0
1
0.00508
0.0, 0.0, 0, 0
AL
0.0, 0.0, 0, 0
AIR
0.0, 0.0, 0, 0
AIR
```

15.3.20   XTUBE

A CM, which must be the first CM, for simulating an x-ray tube. XTUBE is actually similar in geometry to MIRROR. However, unlike MIRROR, XTUBE is usually the first CM in a beam model, and the source beam is usually incident on XTUBE from the side. The second CM may be any of the available CMs with a central opening serving as the exit window of the x-ray tube. The target of the xtube may have an arbitrary number of layers, each with a different thickness and medium. XTUBE has square symmetry.
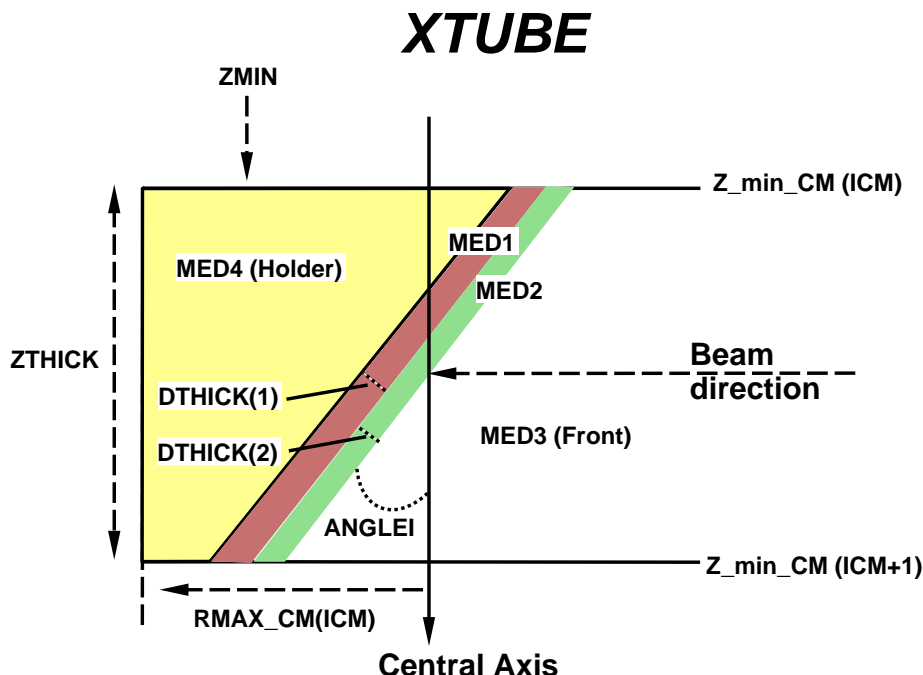
Figure 38: XTUBE component module with 2 layers in the target (N=2). The angle of the target relative to the Z-axis is given by ANGLEI. The thickness of a target layer i is given by DTHICK(i). The target is positioned so that the front of the target (layer 2 in the figure) intersects with the beam at the central (Z) axis. Target layers are arranged with the front of layer i-1 flush with the back of layer i. The target and target holder extend to ± RMAX_CM in the Y-direction. MED3 in the figure above would usually be air or vacuum, but could be any material. MED4 specifies the material of the target holder.

The input format for XTUBE and a sample input are given below.

```
CARDS CM_XTUBE   (XTUBE SID: 2.5)
**************

-1  Dummy line to indicate start of CM.

 0  RMAX_CM(ICM_$XTUBE) (F10.0): Half-width of CM boundary (cm).

 1  TITLE_$XTUBE (60A1): Title of CM.

 2  ZMIN_$XTUBE,ZTHICK_$XTUBE (2F15.0):

      ZMIN_$XTUBE: Distance from front of CM to reference plane(Z=0).
    ZTHICK_$XTUBE: Thickness of xtube in Z-direction.

    Note that there will usually be no air gap at the front of
    this CM.

 3  ANGLEI (F15.0):  Angle between the target surface
                     and Z-axis (in degrees >= 0. and <75.)

 4  N_$XTUBE, I_XTRA_$XTUBE (I5):

          N_$XTUBE:   Number of layers in the target

Repeat 5-7 for I=N_$XTUBE,N_$XTUBE-1,...,1
  (in XTUBE the first layer of the target is I=N_$XTUBE)

 5  DTHICK_$XTUBE(I), I_XTRA_$XTUBE (F15.0,I5)

  DTHICK_$XTUBE(I): Thickness of layer I (cm)
     I_XTRA_$XTUBE: Set to 1 if the layer has an extra central region
                    (only input for the outermost target layer)

 6  ECUT, PCUT,DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5):

      ECUT, PCUT:       Cutoff energies for electron and photon transport
                        of layer I.
       DOSE_ZONE:       Dose scoring flag of layer I.
       IREGION_TO_BIT:  Bit setting of region # corresponding to layer I.

 7  MED_IN (24A1):  Medium of layer I

Note that inputs 8-10 below are only required if I_XTRA_$XTUBE=1
(i.e. the outermost target layer has an extra central region)

 8  WXTRA_$XTUBE, HXTRA_$XTUBE (2F15.0):
```

```
        WXTRA_$XTUBE:   Width of extra central region in layer
                        N_$XTUBE (cm).
        HXTRA_$XTUBE:   Height of extra central region in layer
                        N_$XTUBE (cm).
```

9 ECUT, PCUT,DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5):

```
        ECUT, PCUT:        Cutoff energies for electron and photon transport
                           in extra central region of layer N_$XTUBE.
        DOSE_ZONE:         Dose scoring flag of extra central region.
        IREGION_TO_BIT:  Bit setting of extra central region.
```

10 MED_IN (24A1): Medium of extra central region.

11 ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5): for region in
                                                front of target

```
        ECUT, PCUT:        Cutoff energies for electron and photon transport
                           in front of target
        DOSE_ZONE:         Dose scoring flag for this region.
        IREGION_TO_BIT: Bit setting of region # corresponding to this one.
```

12 MED_IN (24A1): Medium in front of target

13 ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5): for target holder.

```
        ECUT, PCUT:        Cutoff energies for electron and photon transport
                           in target holder.
        DOSE_ZONE:         Dose scoring flag for target holder.
        IREGION_TO_BIT:  Bit setting of region for target holder.
```

14 MED_IN (24A1): Medium of target holder

Example
****

The following example describes a tungsten target, 1mm thick,
mounted on a copper holder. The target is angled at 22 degrees
with respect to the z-axis. It spans 2cm in the z direction.
The medium in front of the target is AIR.
ECUT and PCUT for all regions are set to 0.521, 0.01 respectively.
Note that the input file specifies the target to start at 10cm
from the reference plane; if Z_min_CM(1) is not 10cm and XTUBE
is the first CM in the beam, the code will autumatically reset the
target to start at Z_min_CM(1).

```
10.0        RMAX_CM
XTUBE: z=10 cm, 1mm Tungsten target(1 slab), copper holder, 22 degrees
10.0, 2.0; distance to reference plane = 10 cm, total thickness=2 cm
```

```
22.0;        angle = 22 degrees
1;           1 slab in the target
0.1;         thickness of the slab = 0.1 cm
0.521, 0.01, 0, 2; ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT for this slab
T;           medium is Tungsten
0.521, 0.01, 0, 2; ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT in front of target
AIR          medium is AIR
0.521, 0.01, 0, 2; ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT for the holder
CU;          medium for the holder is copper
```

15.3.21   SIDETUBE

SIDETUBE is a CM for modelling concentric cylinders parallel to the X-axis. As one of 3 CMs in which a radiating isotropic source (`ISOURC=3`) can be placed, it is designed for modelling any isotropic source that is perpendicular to the beam/collimation. The user can specify the number, radii and media of the concentric cylinders, the X positions of the cylinder ends, and the Z position of the axis of the cylinders. Note that all of the cylinders have the same end points, `XMIN` and `XMAX`. The user also specifies the medium surrounding the outer cylinder and filling up the rest of the cube in which the cylinders are contained. SIDETUBE has square symmetry about the beam axis.
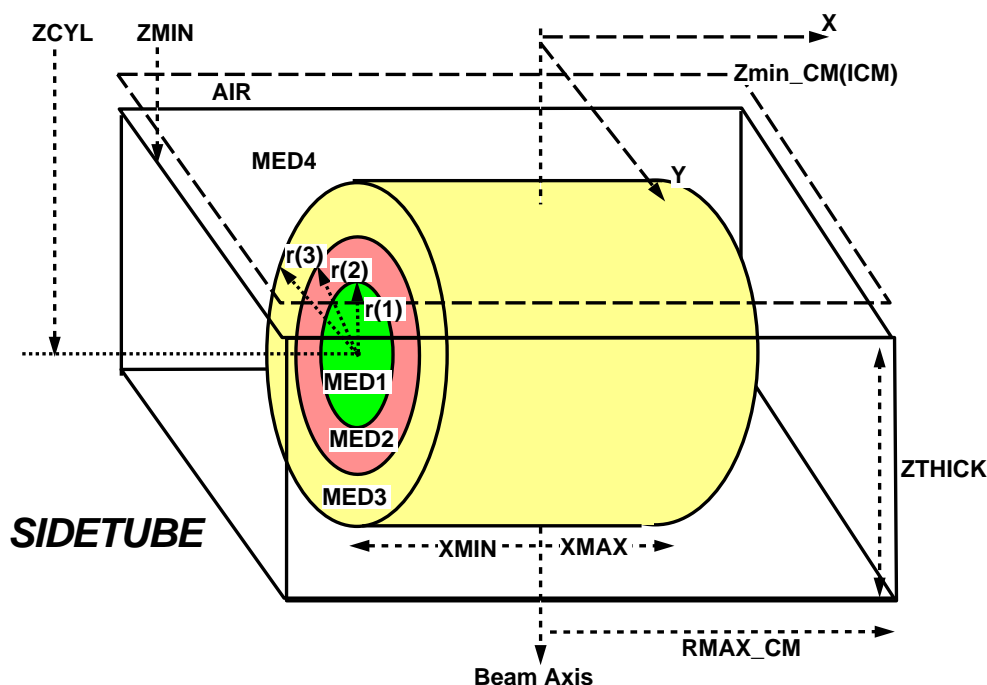
Figure 39: SIDETUBE component module with 3 concentric cylinders (`N=3`), their common axis parallel to the X-axis and located at Z=ZCYL. The outer radii of the cylinders are given by `r(1)`, `r(2)`, `r(3)`, and the cylinders media are `MED1`, `MED2`, `MED3`, respectively. The ends of the cylinders are specified by `XMIN` and `XMAX`. `MED4` is the user-specified medium surrounding the cylinders. Note that all of the cylinders must fit into the rectangular parallelepiped defined by thickness, `ZTHICK`, and half-width, `RMAX_CM`. ZMIN, the Z at which the cube containing the cylinders starts is not necessarily equal to `Z_min_CM(ICM)`, allowing an air gap at the top of this CM.

The input format for SIDETUBE and a sample input are given below.

```
CARDS CM_$SIDETUBE
******************

-1  Dummy line to indicate start of CM.

 0  RMAX_CM(ICM_$SIDETUBE) (F10.0): Half-width of outer boundary
                                    of CM (cm).

 1  TITLE_$SIDETUBE (60A1):  Title of CM.

 2  ZMIN_$SIDETUBE (F10.0):  Distance from front of CM to reference plane
                             (not including air gap) in cm.

 3  ZTHICK_$SIDETUBE (F10.0): Thickness of the CM in the Z-direction
                              (not including airgap) in cm

 4  ZCYL_$SIDETUBE (F10.0):  Z position of axis of coaxial cylinders (cm)

 5  XMIN_$SIDETUBE,XMAX_$SIDETUBE (2F10.0):

        XMIN_$SIDETUBE: Lower X edge of cylinders (cm).
        XMAX_$SIDETUBE: Upper X edge of cylinders (cm).

 6  N_$SIDETUBE (I5): Number of coaxial cylinders.

Repeat 7 for I=1,N_$SIDETUBE all on one line in order of increasing
radius.

 7  R_$SIDETUBE(I) (F15.0): Outer radius of cylinder I (cm).

Repeat 8-9 for I=1,N_$SIDETUBE+1.  When I=N_$SIDETUBE+1 you
are specifying parameters for the region containing the concentric
cylinders.

 8  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT  (2F15.0,2I5):

        ECUT, PCUT:  Cutoff energies for electrons and photons
         DOSE_ZONE:  Dose scoring flag
     IREGION_TO_BIT:  Bit setting for the region

 9  MED_IN (24A1):  Medium of region
                    used to set MED_INDEX.
Example
*******

The following example is a model of a cesium source parallel to
the X-axis.  The source has radius 0.9cm and is made of cesium
```

chloride, CSCL700. This source cylinder is contained within an
aluminum (AL700ICRU) cylinder of outer radius 1.25cm. The
aluminum-cesium chloride assembly is contained in a stainless steel
(STEEL700ICRU) cylinder of inner radius 2.25cm and outer radius 2.4cm.
Note that the airgap between the aluminum and stainless steel is
modelled as a cylinder of air having outside radius 2.25cm. The
cylindrical source assembly extends from X=-1.25cm to X=1.25cm. The
volume of air in which the source assembly is located starts at
ZMIN=-2.4cm and has a thickness, ZTHICK=4.8cm, allowing it to just
contain the source assembly in the Z-direction. Setting ZMIN=-2.4cm
allows the cylinders comprising the source assembly to be centered at
the reference plane (ZCYL=0).

In this example, ECUT, and PCUT in all regions are set to 0.7MeV
and 0.01MeV respectively. The cesium source has dose zone 1; the
aluminum surrounding it has dose zone 2; the air between the aluminum
and the stainless steel outer tube has dose zone 3; and the stainless
steel tube has dose zone 4.

```
10.0,          Outer boundary
100Ci Cs source in holder + crude model of SS tube
-2.4,                ZMIN
4.8,                 ZTHICK
0.,                  ZCYL
-1.125,1.125,        XMIN,XMAX
4,                   Number of cylinders
0.9,1.25,2.25,2.4,   Radii of cylinders
0.700,0.01,0,1,      ECUT, PCUT,...,MED of 1st cylinder
CSCL700
0.700,0.01,0,2,      ECUT, PCUT,...,MED of 2nd cylinder
AL700ICRU
0.700,0.01,0,3,      ECUT, PCUT,...,MED of 3rd cylinder
AIR700ICRU
0.700,0.01,0,4,      ECUT, PCUT,...,MED of 4th cylinder
STEEL700ICRU
0.700,0.01,0,0,      ECUT, PCUT,...,MED surrounding cylinders
AIR700ICRU
```

15.3.22   ARCCHM

ARCCHM is a CM designed specifically for modelling an arc-shaped array of chambers found in the prototype tomotherapy unit at the University of Wisconsin. However, it can be used to model any arc-shaped structure in the path of the beam. The arc is always concave up and curves in the Y direction with the lowest point of the arc (max. Z) always at Y=0. The arc itself consists of a single layer of chambers separated by septa sandwiched between a chamber front face and chamber back face. The user specifies the Z position of the chamber front face at Y=0, `ZSRC`, and the radius of the front face, `ZRAD1`. The angular extent of the arc and it's total span in the Z direction are determined by other user-specified parameters: the number of chambers, `NUMCHM` (which must be an even number), their width, `WIDTHCHM`, the width of the septa separating them, `WIDTHSEP`, the thickness of the chambers and septa, `ARCTHICK`, and the thicknesses of the chamber front and back, `FRONTHCK` and `BACKTHCK` respectively. The chamber Y ends are automatically added and the chamber back face extended so that they are flush with a plane at the minimum Z, `ZMIN` (calculated, not specified), of the arc. The user also specifies the minimum and maximum X limits of the arc (`XMIN1` and `XMAX2`, respectively), the thickness of the X ends of the chambers (`WIDXWALL`) and the maximum Z of the CM (`ZMAX`). The user can specify the media in all regions of the CM with the exception of the AIR above `ZMIN`. If the chambers all have the same `MED`, `ECUT`, `PCUT` and `IREGION_TO_BIT` and the septa do as well, then by setting `IREPEAT`=1, the user only has to input these parameters once for all chambers and once again for all septa. In this case, if the `DOSE_ZONE` input for the chambers is >0, then it is automatically incremented by 1 for each chamber, so that each chamber has its own dose zone.

This CM was originally coded by Marv Glass of the University of Wisconsin.
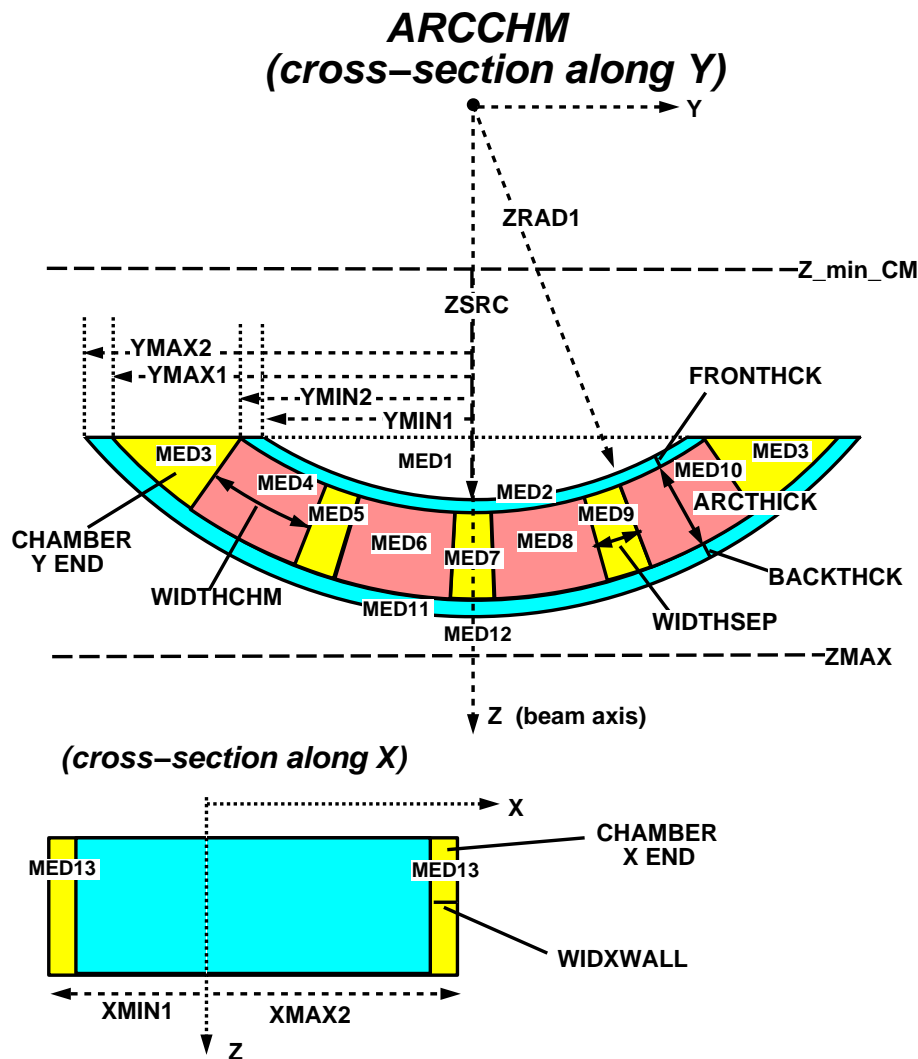
Figure 40: ARCCHM component module with 4 chambers (`NUMCHM=4`). The number of chambers must always be even so that they can be arrayed as shown in the Y cross-section, with a septum centred on the beam axis at Y=0. The number of septa is always 1 less than the number of chambers. The user also specifies the Z position of the front face of the chambers at Y=0, `ZSRC`, the radius of the arc at the front face, `ZRAD1` (`ZSRC=ZRAD1` for a perfectly isotropic source), the arc-width of the chambers, `WIDTHCHM`, and septa, `WIDTHSEP`, the thickness of the chambers/septa `ARCTHCK`, the thickness of the front face, `FRONTHCK`, and back face, `BACKTHCK` and the maximum Z of the CM, `ZMAX`. Other parameters such as the angular extent of the arc, the minimum and maximum Y dimensions, `YMIN1`, `YMIN2`, `YMAX1` and `YMAX2`, and the minimum Z of the arc, `ZMIN` are calculated based on the user-specified parameters above. As shown, the chamber Y ends and back face are automatically extended so that they are flush with `ZMIN`. Note that `ZMAX` must be chosen to accommodate the full Z range of the arc. The cross-section in the X-direction shows the user-specified minimum and maximum X of the arc (`XMIN1` and `XMAX2` respectively) and width of the chamber X walls (`WIDXWALL`). The user can specify the medium in all regions except above `ZMIN`. If all the chambers have the same `MED`, `ECUT`, etc, and the septa do as well, then the number of inputs can be reduced by using the `IREPEAT` option.

The input format for ARCCHM and a sample input are given below.

```
CARDS CM_$ARCCHM
*****************

-1  Dummy line to indicate start of CM.

 0  RMAX_CM(ICM_$ARCCHM) (F10.0): Half-width of outer boundary
                                     of CM (cm) Read in MAIN

 1  TITLE_$ARCCHM (60A1):  Title of CM.

 2  ZSRC_$ARCCHM (F15.0): Distance from front face of chamber at Y=0
                          (ie the lowest point in the arc) to reference
                          plane in cm.

 3  ZRAD1_$ARCCHM (F15.0): Radius of front face of chamber in cm.  For
                           a fully-divergent beam, this may equal
                           ZSRC_$ARCCHM.

 4  NUMCHM_$ARCCHM (I4):  Number of individual ion chambers

 5  WIDTHCHM_$ARCCHM (F15.0): Width of each ion chamber in cm.

 6  WIDTHSEP_$ARCCHM (F15.0): Width of each septum in cm.

 7  ARCTHICK_$ARCCHM (F15.0): Thickness of chambers & septa in cm.

 8  FRONTHCK_$ARCCHM (F15.0):  Thickness of front face of chamber in cm.

 9  BACKTHCK_$ARCCHM (F15.0):  Thickness of back face of chamber in cm.

10  WIDXWALL_$ARCCHM (F15.0):  Width of chamber wall along x in cm.

11  XMIN1_$ARCCHM,XMAX2_$ARCCHM (2F15.0):

        XMIN1_$ARCCHM: Min x dimension outside of x wall (cm).
        XMAX2_$ARCCHM: Max x dimension outside of x wall (cm).
     All YMIN and YMAX are calculated from ZRADs and PHI
     The min/max x dimensions inside of the x wall are calculated.
     ZMIN is calculated from ZRAD, ZSRC, and PHI

12  ZMAX_$ARCCHM (F15.0):  The max Z of the CM in cm.  Note that
                           ZMAX must be >= ZSRC+ARCTHICK+FRONTHCK+
                                     BACKTHCK

  Repeat 13-14 for the following regions:
        Region 1:  region before arc-shaped ion chamber.
        Region 2:  front face of arc-shaped ion chamber.
```

```
        Region 3:  ends of the ion chamber.
        Region 4 -- 2*NUMCHM+2: chamber or septa
              EVEN:  chamber   ODD: septa  Numbering of chambers and
              septa goes from -Y to +Y  (13-14 only have to be
              repeated twice for these regions if IREPEAT=1.  See
              below.)
        Region 2*NUMCHM+3:  back face of arc-shaped ion chamber.
        Region 2*NUMCHM+4:  region surrounding the arc.
        Region 2*NUMCHM+5:  x walls of the chamber


 13  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT, (IREPEAT) (2F15.0,2I5,(I5)):


        ECUT, PCUT:  Cutoff energies for electrons and photons
         DOSE_ZONE:  Dose scoring flag
    IREGION_TO_BIT:  Bit setting for the region
           IREPEAT:  Only input for Region 4.  Set to 1 to apply this
                     ECUT, PCUT, IREGION_TO_BIT and MED_IN to all
                     chambers (if DOSE_ZONE > 0, then it will be
                     incremented automatically for each chamber) and the
                     following ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT and
                     MED_IN to all septa.


 14  MED_IN (24A1):  Medium of region
                     used to set MED_INDEX.


  Example
  *******


  The following example is an arc-shaped ion chamber containing 10
  individual chambers.  Used for the tomotherapy prototype under
  development at the UW.


  In this example, ECUT, and PCUT in all regions are set to 0.7 MeV
  and 0.01 MeV respectively.  Note that IREPEAT is set to 1 for the
  first chamber so that ECUT, PCUT, etc only have to be input once
  for all chambers and once for all septa.  Also, DOSE_ZONE will
  be incremented automatically for the chambers, so that the chambers
  will have dose zones 1-10.


  *************dummy line to indicate new CM**********************
  60.0,             RMAX:  outer boundary
  Tomotherapy ion chamber
  133.0,                ZSRC: distance to front of ARC
  133.0,                ZRAD1: radius of front of arc
  10,                   NUMCHM: the number of individual ion chambers
  1.0,                  WIDTHCHM: width of each individual ion chamber
  0.1,                  WIDTHSEP: width of septa
  5.0,                  ARCTHICK: thickness of the arc
  0.1,                  FRONTHCK: thickness of front face of chamber
```

```
0.2,                     BACKTHCK: thickness of back face of chamber
0.2,                     WIDXWALL: thickness of x wall
-5.0, 5.0,               XMIN1, XMAX2: min/max x dimension outside of x wall
139.0,                   ZMAX: Z limit of CM
0.700,0.01,0,1,          ECUT, PCUT,...,MED of region 1
AIR521ICRU
0.700,0.01,0,2,          ECUT, PCUT,...,MED of front face
AL521ICRU
0.700,0.01,0,3,          ECUT, PCUT,...,MED of edges of chamber
AL521ICRU
0.700,0.01,1,1,1,        ECUT, PCUT,...,MED of all chambers
H2O521ICRU
0.700,0.01,0,1,          ECUT, PCUT,...,MED of all septa
PB521ICRU
0.700,0.01,0,2,          ECUT, PCUT,...,MED of back face
AL521ICRU
0.700,0.01,0,1,          ECUT, PCUT,...,MED of region surrounding arc
AIR521ICRU
0.700,0.01,0,2,          ECUT, PCUT,...,MED of x walls
AL521ICRU
```

15.3.23   SYNCJAWS

SYNCJAWS is a CM similar to DYNJAWS, but which allows synchronization between the motion of the jaws and other dynamic components of the model. In particular, it allows for modeling continuously varying beam configurations by synchronizing source and phantom motions when used in conjunction with sources 20 and 21 in DOSXYZnrc. This CM module was contributed by Lobo and Popescu. For the time being, please refer to their original article on these new sources for more details: *Phys. Med. Biol.* **55**, 4431–4443 (2010).

15.3.24   SYNCVMLC

SYNCVMLC is a CM similar to DYNVMLC, but which allows synchronization between the motion of the MLC and other dynamic components of the model. In particular, it allows for modeling continuously varying beam configurations by synchronizing source and phantom motions when used in conjunction with sources 20 and 21 in DOSXYZnrc. This CM module was contributed by Lobo and Popescu. For the time being, please refer to their original article on these new sources for more details: *Phys. Med. Biol.* **55**, 4431–4443 (2010).

15.3.25   SYNCMLCE

SYNCMLCE is a CM similar to MLCE, but which allows synchronization between the motion of the MLC and other dynamic components of the model. In particular, it allows for modeling continuously varying beam configurations by synchronizing source and phantom motions when used in conjunction with sources 20 and 21 in DOSXYZnrc. This CM module

was contributed by Lobo and Popescu. For the time being, please refer to their original article on these new sources for more details: *Phys. Med. Biol.* **55**, 4431–4443 (2010).

16   Cross-Section Data – PEGS4

Cross-section data for many commonly-used media are included in EGSnrc installation in the files `521icru.pegs4dat` and `700icru.pegs4dat`, both located in the `$HEN_HOUSE/pegs4/data` directory. The file `521icru.pegs4dat` consists of cross-section data from a lower electron energy, `AE`, of 0.521 MeV to an upper electron energy, `UE`, of 55 MeV, while `700icru.pegs4dat` contains data from `AE`=0.700 MeV to `UE`=55 MeV. In both files the lower photon energy, `AP`, is 0.01 MeV and the upper photon energy, `UP`, is 55 MeV. These data are based on the density effect corrections in ICRU Report 37 [39].

If you wish to determine exactly what the composition of a given material is, it is specified on the lines immediately following the `MEDIUM` label in the `.pegsdat` file. For an up-to-date listing of what materials are available in these files, do as follows (only for Linux/Unix):
```
cd $HEN_HOUSE/pegs4/data
grep MEDIUM 700icru.pegs4dat
```
or
```
grep MEDIUM 521icru.pegs4dat
```
To use these data you must specify the names **exactly** as they appear in the data file being used.

The NRC EGS user-code examin (found on `$HEN_HOUSE/user_codes/examin`) can be used to tabulate or plot the various cross sections for each material.

16.1   Creating additional cross section data

Additional cross section data for BEAMnrc is created by the code PEGS4 (located in `$HEN_HOUSE/pegs4`). A complete description of the PEGS4 code is included in the EGSnrc manual[1], and users wishing to create their own PEGS4 data are referred to that manual.

Briefly, PEGS4 can be invoked on the command line by typing:

```
pegs4.exe -i inputfile [-d densityfile]
```

where `inputfile` omits the `.pegs4inp` extention and `densityfile` omits the `.density` extension. The `.pegs4inp` file specifies whether the the medium is an element (`ELEM`), compound (`COMP`) or mixture (`MIXT`); the composition of the medium (by mass, `RHOZ`, in the case of `MIXT` and number of atoms, `PZ`, in the case of `COMP`); the density (`RHO`; whether Rayleigh cross-sections are to be included (`IRAYL`=1); the stopping powers calculated (`IUNRST`=0 for restricted stopping powers, `IUNRST`=1 for unrestricted collision stopping powers, etc.); and whether ICRU 37 density corrections are to be included (`EPSTFL`=1). The `.pegs4inp` file also specifies the values of `AE`, `AP`, `UE` and `UP` to use and the name by which the medium will be referred in all input files using it. There are other input options, but they are beyond the scope of this manual.

If the user has selected to use ICRU 37 density corrections (`EPSTFL=1`), then they must include the `-d densityfile` input when running `pegs4`. Density correction files are found in the directory `$HEN_HOUSE/pegs4/density_corrections`. It is important to note that if you are using a density correction file, the value of `RHO` specified in the `.pegs4inp` file must match the material density in the `densityfile.density` file, which is the third value on the second line of the file.

The EGSnrc distribution includes some sample input files in the directory `$HEN_HOUSE/pegs4/input`, and it is useful to use these as references if creating your own `.pegs4inp` file.

When running PEGS4 as described above PEGS4 data is automatically output to either `$HEN_HOUSE/pegs4/data/inputfile.pegs4dat` or `$EGS_HOME/pegs4/data/inputfile.pegs4dat`, depending on what directory it is being run from.

An much easier way to create PEGS4 data is to run PEGS4 from the `egs_gui` (invoked simply by typing "`egs_gui`"). A screen shot of the PEGS4 option in the `egs_gui` is shown in Figure 41. For more information on `egs_gui` see the EGSnrcMP Manual[2].

Note that `egs_gui` offers an option to append newly-created PEGS4 data to an existing `.pegs4dat` file.
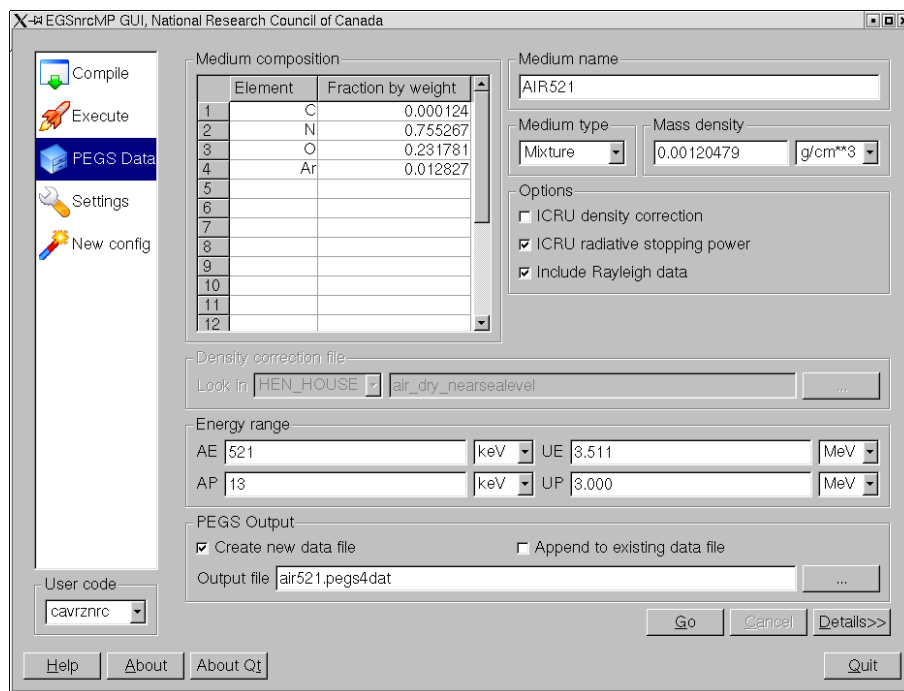
Figure 41: Screen shot of the `egs_gui` set to run PEGS4 to create an air data set. Filling in this form is much easier than creating an input file and access to the density effect corrections is easy. The GUI does not allow any value except `IUNRST = 0`. For details, see PIRS-877[2].

16.2   Choice of AE,AP

The parameters `AP` and `AE` are the low-energy thresholds for the production of secondary bremsstrahlung photons and knock-on electrons, respectively. The parameters `PCUT` and `ECUT` are required to be greater than or equal to `AP` and `AE` (see sections 10.11 and 10.10).

Selection of `AP` is simple since one can afford to use a very low value which ensures accurate photon transport. We recommend using `AP` = 0.010 MeV. Note that in practice this means all bremsstrahlung events are simulated as discrete events.

The choice of `AE` is more complex since there is some computing time associated with lower values of `AE` and in some cases, lower values lead to more accurate simulations. The value of `AE` controls the statistical fluctuations in the energy loss, can affect the electron step sizes and is a lower limit on `ECUT`. For a general discussion see, *eg.* [33, 25].

Although the choice of `AE` is complex in general, it is fairly easy to give general rules for. In practice, the value of `AE` has little effect on dose calculations, except in the sense that `AE` is the lower limit on `ECUT`. Thus for dose calculations, one should select `ECUT` as discussed in section 10.10 and then use `AE = ECUT`. For calculations in which one is looking at electron spectra directly in an electron beam, it is important to use a fairly low value of `AE` in order to avoid well known artifacts[33, 25]. Figure 42 gives an example of this artifact for the 9 MeV beam from a Clinac2100C. The spectrum calculated with AE=0.521 MeV is clearly more realistic than that calculated with AE=0.700 MeV. We find that `AE` = 0.521 MeV is adequate - but note, one can still use `ECUT` much higher.
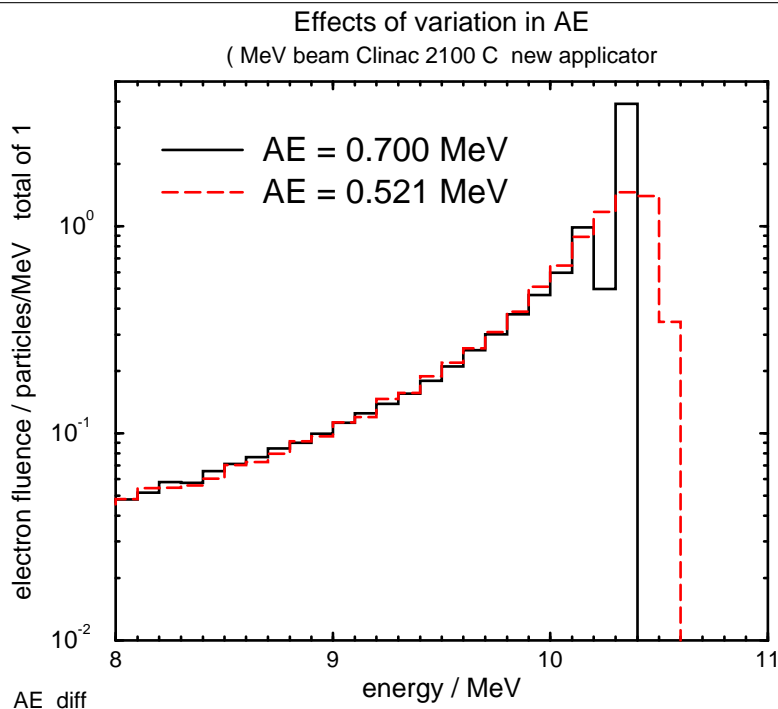
Figure 42: Example of the differences in electron spectra calculated with AE=0.521 MeV and AE=0.700 MeV. The spectra are normalized to one particle in the total spectrum. The specific example is for the 9 MeV beam from a Clinac2100C with a type III applicator. The artifact near the peak in the AE=0.700 MeV case is discussed in detail in references [33, 25].

17   Distribution / Installation

Installation of the OMEGA/BEAM codes requires you to have already installed and configured EGSnrcMP on your system. A brief description of the installation/configuration of EGSnrcMP will be given here, but for more detail you are urged to refer to the EGSnrcMP Users Manual[2].

Also, in order to install and run EGSnrcMP and OMEGA/BEAM, your system must have:

1. A working Fortran compiler. If you do not have one on your system, GNU provides a suite of compilers that can be downloaded for free at `http://www.gnu.org`

2. A working `make` utility. Most systems have this built in, although there is one available at the GNU site as well.

3. A working C or C++ compiler. This is a requirement on Unix/Linux systems if you want to take advantage of the built-in parallel processing capability of BEAMnrc and DOSXYZnrc (and other EGSnrc user codes) and is of no use if you do not have a batch submission capability. It is also required on Unix/Linux systems (but not on Windows) if you want to use a BEAM simulation as a source with DOSXYZnrc or with any other EGSnrc user code. The compiler is available at the GNU site.

4. A working C++ compiler. This is required if you want to be able to read/write IAEA-format phase space files.

5. Tcl/Tk. This is necessary for running the BEAMnrc, DOSXYZnrc and BEAMDP GUIs. It can be downloaded from `http://www.activestate.com/activetcl/downloads`. See the GUI User's Manual[4] for more information. Note that most Linux distributions already include Tcl/Tk.

6. A Qt library version 3 (revision 3.1 or newer). This is necessary for compiling and running the GUIs for the EGSnrcMP user codes and is, thus, not strictly necessary for the OMEGA/BEAM codes. You can download the Qt library for free from `ftp://ftp.qt.nokia.com/qt/source/qt-x11-free-3.3.8b.tar.gz`. Note that many current Linux distributions include Qt, since the popular Desktop Environment KDE is based on Qt.

7. The xmgrace plotting package which is available at `http://plasma-gate.weizmann.ac.il/Grace`. This is distributed with many Linux distributions and there is a Windows version.

Note that some of these system requirements are included as part of the EGSnrcMP distribution (depending upon which method you choose to install EGSnrcMP). More details about this will be given below.

17.1 Installing EGSnrcMP

The EGSnrcMP distribution is available online at `http://irs.inms.nrc.ca/software/egsnrc/`. Download the version appropriate for your system. There are several different methods for installing the system. The easiest of these is "Method 1", the self-extracting installation wizard. This includes compressed versions of all codes required for a complete EGSnrcMP distribution which are automatically uncompressed at installation time. In the case of Windows, the self extracting download also includes GNU Fortran, C and C++ compilers and the GNU `make` utility.

"Method 2" for installing EGSnrcMP involves downloading the installation wizard and the required, compressed codes separately. In addition to the installation wizard itself, the required downloads to install a complete EGSnrcMP system are:

`V4_EGSnrc.egszip` The EGSnrcMP system.

`V4_spinms.egszip` Spin and multiple-scattering data.

`V4_user_codes.egszip` The EGSnrcMP user codes. Not necessary if you are only planning to run the OMEGA/BEAM codes.

`V4_manuals.egszip` EGSnrcMP documentation.

Additional downloads for Windows are:

`V4_MinGW.egszip` GNU Fortran, C, C++ compilers and `make` utility. Only required if you
   do not have working versions of these on your system.

`V4_winstuff.egszip` GUIs for EGSnrcMP user codes and other Windows-specific codes.

For Unix/Linux, if you are planning to use the GUIs for the EGSnrcMP user codes (not
necessary if you are only going to run the OMEGA/BEAM codes), then you should also
download the GUI codes, `V4_EGSgui.egszip`. These downloads and their functions are made
clear on the distribution website. If you have only downloaded a subset of the above `.egszip`
files (*e.g.* the minimum required for eventually installing/running the OMEGA/BEAM
codes) then you should instruct the GUI to only uncompress those archives that you have
downloaded. Rather than physically dowloading the `.egszip` files yourself, you can also
instruct the GUI to automatically download the archives (any subset of them or all of them)
from the distribution website before installation. Note that "Method 1" and "Method 2" for
EGSnrc installation are not available for the Mac OSX.

   The third method, "Method 3", of installing EGSnrcMP is for Unix/Linux systems and
Mac OSX systems and uses the installation script `install_egs`. In addition to the script,
you must also download the `tar` files (which may be in compressed format)

`V4_EGSnrc.tar[ |.gz|.bz2]`

`V4_EGSgui.tar[ |.gz|.bz2]`

`V4_manuals.tar[ |.gz|.bz2]`

`V4_user_codes.tar[ |.gz|.bz2]`

`V4_spinms.tar[ |.gz|.bz2]`

where the base names of the files are the same as for the `.egszip` files described above. These
files must all be in the same directory before running the `install_egs` script. This method of
installation may be necessary if your Linux/Unix system cannot run the installation wizard
and is currently the only way of installing EGSnrcMP on Mac OSX. Note that when you use
this method of installation, you must run the script `$HEN_HOUSE/scripts/finalize_egs_foruser`
at the end (`install_egs` gives you an option to do this automatically) in order to de-
fine/create your `$EGS_HOME` area (see next paragraph).

   During installation of EGSnrcMP, you will be asked to define the location of `$HEN_HOUSE`,
the repository for all EGSnrcMP and OMEGA/BEAM codes (see Figure 2, page 15), and
your `$EGS_HOME` area, where each compiled user code (including accelerators modeled with
BEAMnrc) has a subdirectory (see Figure 3). You will also be asked to define the Fortran
compiler that you want to use (defaults are suggested if found on your system). . On
Unix/Linux/Mac OSX systems, if you wish to use functionality that requires a C compiler
such as the built-in parallel processing functionality (see Section 13) or BEAMnrc shared
library sources (see Section 2.3.1), then you must also supply the name of a working C com-
piler at this point (default C compilers are suggested if found on your system). On Windows
systems, the parallel processing is unavailable (no batch queuing system) and a precompiled
object file required for BEAM shared library sources is provided during installation, so the

reason to specify a C compiler here is to make use of the C interfaces which allow EGSnrc user codes to be written in C (see the EGSnrcMP Manual[49] for more details). You will be asked to give a name to your machine configuration and supply a prefix for the configuration file, *i.e.*, the `.conf` file. The name you give for your configuration will determine the names of the subdirectories off `$HEN_HOUSE/lib` `$HEN_HOUSE/bin` and `$EGS_HOME/bin`. See section 1.3, page 13, for more information about the directory structure of the `$HEN_HOUSE` and section 2.12.3, page 31 for more information about the `.conf` file. We recommend that the prefix of the `.conf` file be the same as the configuration name to avoid confusion. Finally, you will be given the option to input the name of a working C++ compiler (default C++ compilers are suggested if found on your system). This is required to compile the library necessary for reading/writing IAEA-format phase space data (see Section 7.3) on all systems and to allow compilation of the EGSnrc C++ class library, egspp (see the egspp Manual[50] for more details).

Note that, on Unix/Linux/Mac OSX machines, if you supply a working C compiler during EGSnrcMP installation and the C routines for built-in parallel processing and/or BEAM shared library sources are compiled successfully, then the installation automatically sets up your `config.conf` to enable these functions. Similarly, on all systems, if you supply a working C++ compiler during installation and the library of IAEA phase space handling routines is compiled successfully, the installation automatically configures `config.conf` so that user codes link this library (`$HEN_HOUSE/iaea_phsp/iaea_phsp.a`) during compilation and can read/write IAEA-format phase space data.

When installing on Windows, you will be given the option to automatically set the environment variables `HEN_HOUSE`, `EGS_HOME` and `EGS_CONFIG` (which is set to `$HEN_HOUSE/specs/config.conf`, where `config` is the prefix you have chosen for the `.conf` file). It is recommended that you do this to avoid having to add them manually. When installing on Unix/Linux/Mac OSX, on the other hand, you do not have the option to set these variables automatically. Instead, you will be directed to add the statements:

```
setenv EGS_HOME /full directory path to your EGS_HOME/
(or export EGS_HOME=/full directory path to your EGS_HOME/)
setenv EGS_CONFIG /full_path_to_your_HEN_HOUSE/specs/config.conf
(or export EGS_CONFIG=/full_path_to_your_HEN_HOUSE/specs/config.conf)
source /full_path_to_your_HEN_HOUSE/scripts/egsnrc_cshrc_additions
(or . /full_path_to_your_HEN_HOUSE/scripts/egsnrc_bashrc_additions)
```

to the `.cshrc` (or `.bashrc`) file in your `$HOME` area. These directions are reiterated in the file `$HEN_HOUSE/README_UNIX`. Note that the `egsnrc_cshrc_additions` (or `egsnrc_bashrc_additions`) script defines `$HEN_HOUSE`. Once you have added these statements, you will then have to `source` your `.cshrc` (or `.bashrc`) file, or else open a new window, to run EGSnrcMP.

If you are using a batch queuing system (Unix/Linux only) then the file `$HEN_HOUSE/scripts/batch_options.YourSystemName` must exist (the distribution includes files for `at`, `pbs`, `keg` and `nqs` but these may require modifications for your system) and you also must include:
```
setenv EGS_BATCH YourSystemName
```
in your `.cshrc` file or equivalent in your `.bashrc` file.

If you wish to use a second compiler or operating system, you can either run the instal-

lation wizard (self-extracting or stand-alone) and select the "Configure existing EGSnrc" option or, if on a Unix/Linux/Mac OSX system, you can run the script $HEN_HOUSE/scripts/configure from the $HEN_HOUSE/scripts directory. Alternatively, this task can be done from the egs_gui interactive GUI for general EGSnrc user codes[51]. Note that you must have write permissions on $HEN_HOUSE to configure a second compiler/operating system.

To see the status of your EGSnrcMP installation (especially important if something goes wrong) then:

- If you have used an installation wizard, look in the file $HEN_HOUSE/install_status/config_[unix|win].log.

- If you have used the install_egs script, look in the files install.log (in the directory that you ran install_egs from) and config.log (in directory $HEN_HOUSE/scripts).

17.2   Installing OMEGA/BEAM

Once EGSnrcMP has been successfully installed, you are now ready to install the OMEGA/BEAM system (including BEAMnrc, DOSXYZnrc, BEAMDP, ctcreate, and other utility codes).

The installation can be obtained from the OMEGA/BEAM distribution site at:

http://irs.inms.nrc.ca/software/beamnrc/download.html

Similar to EGSnrcMP, there are 3 methods for installing the OMEGA/BEAM system: using the self-extracting installation wizard (Linux/Unix and Windows), using the installation wizard with files downloaded separately (Linux/Unix and Windows) and using an installation script (Linux/Unix and Mac OSX).

17.2.1   Using the Self-extracting Wizard

The easiest installation method is to use the self-extracting wizard. Download the wizard appropriate for your system and run it by either typing the name of the file (e.g. beam_install_Linux_self.exe) or by double clicking on the file if you are using a Windows-type browser. The self-extracting wizard contains all of the files required to set up a complete OMEGA/BEAM system. When you run the wizard, it will check that you have correctly installed EGSnrcMP by checking for the existence of $HEN_HOUSE (both the directory and the environment variable), $EGS_HOME (directory and environment variable) and $EGS_CONFIG (the config.conf file and the environment variable). If any one of these environment variables is not defined or the directory/file indicated by the variable is not present, then installation stops with a warning message indicating that you must first correctly install EGSnrcMP. However, if EGSnrcMP was correctly installed, then the wizard goes ahead and installs OMEGA/BEAM. OMEGA_HOME is automatically defined as $HEN_HOUSE/omega and the directory structure shown in Figure 1(Section 1.3) is created. In addition, the subdirectory dosxyznrc is added to the $HEN_HOUSE/user_codes area (see Figure 2 in section 1.3, page 15).

   Once all directories/files required for the OMEGA/BEAM installation are in place, the wizard will compile the `beam_build` tool (see section 2.2 page 19), BEAMDP[52, 53], `ctcreate` (see the DOSXYZnrc Users Manual[15], section 15), `readphsp` (see Section 7.4), `statdose` (see section 12 of the DOSXYZnrc Users Manual[54]), `addphsp` (see section 13.4), and `dosxyz_show`[55]. The wizard also determines whether your machine includes a C compiler and, if it does, it compiles the C routines used for parallel runs with DOSXYZnrc, `$HEN_HOUSE/user_codes/dosxyznrc/read_write_pardose.c`, and puts the object file, `read_write_pardose.o`, in the `$HEN_HOUSE/lib/config` directory (where `config` is the name of your machine configuration defined by the variable `EGS_CONFIG`). The wizard then creates the file `$HEN_HOUSE/specs/dosxyznrc_config.spec`, which is used when compiling DOSXYZnrc to determine whether `read_write_pardose.o` will be linked in at compile time or not. See the DOSXYZnrc Users Manual[15] for more information about `read_write_pardose` and `dosxyznrc_config.spec`. Note that if any of the above compilations fail, the installation does not stop, but a warning message is output at the end.

   At this point, all of the OMEGA/BEAM codes have been extracted and the utility programs have been compiled on the `$OMEGA_HOME` area. This completes the actual installation of OMEGA/BEAM. The self-extracting wizard then automatically sets up your user's area (see Figure 3) for OMEGA/BEAM.

   To set up your system, the wizard creates the subdirectory chain `beamnrc/spec_modules` in your `$EGS_HOME` area and copies the example accelerator spec modules (`EX10MeVe.module`, `EX16MVp.module`, `EXphantom.module`, and `EXslabs.module`) into the `spec_modules` subdirectory. The wizard also creates subdirectory `dosxyznrc` off your `$EGS_HOME` area, copies the required source code files from `$HEN_HOUSE/user_codes/dosxyznrc` into it and compiles DOSXYZnrc. If any of these operations fail, the set up does not stop, but a warning message will be output at the end.

   The final step of set up is to define your `$OMEGA_HOME` environment variable. If setting up on Windows, then `$OMEGA_HOME` will automatically be set (to `/full_directory_path_to_$HEN_HOUSE/omega`). If set up is on Unix/Linux/Mac OSX, `$OMEGA_HOME` will not automatically be set, but you will be instructed (in the installation wizard display window) to add the statement:

```
source /full path to your HEN_HOUSE/scripts/beamnrc_cshrc_additions
(or . /full path to your HEN_HOUSE/scripts/beamnrc_bashrc_additions)
```

to the `.cshrc` (or `.bashrc`) file in your `$HOME` area. Then source `.cshrc` (or `.bashrc`). `beamnrc_cshrc_additions` (or `beamnrc_bashrc_additions`) will define `OMEGA_HOME` and aliases useful for running the OMEGA/BEAM codes in Unix/Linux. At this point, configuration of your area is complete.

   The installation wizard also gives you the option to put GUI icons (for the BEAMnrc, DOSXYZnrc and BEAMDP GUIs) on the desktop (Windows and Linux KDE only) and/or in the "Start" menu (Windows only).

   Instructions, warnings and information appearing in the wizard display window, including during configuration, are echoed in the file `$HEN_HOUSE/install_status/config_beam_[unix|win].log`

17.2.2   Using the Installation Wizard with Files Downloaded Separately

You can also install BEAM by downloading the installation wizard and distribution archives separately. Once you have downloaded the wizard appropriate for your system, you have the option of either downloading the `.egszip` distribution archives,

`V4_BEAMnrc.egszip` All OMEGA/BEAM codes, GUIs, documentation, and examples (essential)

`V4_BEAMnrc_phsp.egszip` The phase space file `dosxyznrc_test_electron_medium.egsphsp1` for use with the BEAMDP multiple-source model lab (optional)

`V4_BEAMnrc_CT.egszip` Example CT data in Pinnacle, AAPM, DICOM and CADPLAN formats. Pinnacle data is used in the ctcreate lab (optional),

yourself or letting the wizard download the archives directly from the NRC distribution site. Unlike the self-extracting wizard, this wizard gives you the option to download only the essential BEAMnrc system files (`V4_BEAMnrc.egszip`) and one or none of the optional archives, `V4_BEAMnrc_phsp.egszip` and `V4_BEAMnrc_CT.egszip`. This option is useful if space is a consideration on your system or if you are not interested in the multiple-source model and ctcreate labs.

  By default, when the GUI is first started, it checks for the existence of all `.egszip` archives in the startup directory (*i.e.* the directory that you are running the installation GUI from). If you have already downloaded all of the files to this directory, then you can proceed with installation immediately. If, however, any files are not found in this directory, then a warning appears. Clicking "OK" in the warning window allows you to proceed with installation with:

  a. those archives that HAVE been downloaded to to the startup directory (these will automatically be checked off in the GUI).

  b. archives you have downloaded to another directory. In this case, you will have to check off the archive files that you have downloaded, ensure that the 'Local archive(s)" option has been selected, and use the browser at the bottom of the GUI to specify the complete path to the directory that you have downloaded the file(s) to. Or

  c. archives to be downloaded by the GUI from the distribution site. Check off the archives you want to download and select the 'Download from NRC distribution site" option.

  Once you have selected your particular installation method, press the "Next" button to proceed with installation.

  This wizard also allows you to choose between doing a "Complete system installation", which will install the OMEGA/BEAM system and set up your user area, or a "User system configuration", which will just set up your user area. The latter option is only used if OMEGA/BEAM has already been installed (by you, `root`, or another user) and does not require you to download any of the `.egszip` archives. If you are just setting up the user's

area, the wizard checks that both EGSnrcMP and OMEGA/BEAM have already been in-
stalled by checking for the existence of `$HEN_HOUSE` (environment variable and directory),
`$EGS_HOME` (environment variable and directory), `$EGS_CONFIG` (environment variable and
file) and `$HEN_HOUSE/omega`. If any one of these does not exist, then set up terminates with
a warning. Otherwise, configuration will proceed and `$HEN_HOUSE/omega` will become your
`$OMEGA_HOME`.

Note that the self-extracting wizard always does the equivalent of a "Complete system
installation" with all of the archives.

Installation and set up of your user area proceeds in a manner similar to the self-
extracting wizard (see section above) with the exception that only those distribution files
that you have downloaded are uncompressed. All of the essential codes are contained in
`V4_BEAMnrc.egszip`, so, even if this is the only archive you download, the compilations
described in the previous section will be done.

Similar to the self-extracting wizard, instructions, warnings and information appearing
in the wizard display window, including during configuration, are reiterated in the file
`$HEN_HOUSE/install_status/config_beam_[unix|win].log`

17.2.3 Using the `install_beam` Script (Unix/Linux only)

If you are using a Linux/Unix system, then you can also install the OMEGA/BEAM system
using the script `install_beam`. To do this, download the script along with the distribution
`.tar` archives (which can be in compressed format):

`V4_BEAMnrc.tar[ |.gz|.bz2]`

`V4_BEAMnrc_phsp.tar[ |.gz|.bz2]`

`V4_BEAMnrc_CT.tar[ |.gz|.bz2]`

Descriptions of the archives are the same as for the `.egszip` archives described above. Again,
if space is a consideration, then the only essential archive is `V4_BEAMnrc.tar[ |.gz|.bz2]`.

Run the script by typing `install_beam` in the directory that you downloaded it into. As
with the installation wizards, the script checks that EGSnrcMP has been installed correctly
before proceeding. In particular, `$HEN_HOUSE` and `$EGS_HOME` need to be set. You will then
be prompted to input the full path to the directory where you have stored the distribution
archives. The script then copies the distribution archives into your `$HEN_HOUSE` directory,
uncompresses them (if required) and untars them. This sets up the `$HEN_HOUSE/omega` direc-
tory tree, which becomes `$OMEGA_HOME` (see Figure 1 page 14), and also sets up the `dosxyznrc`
subdirectory as part of `$HEN_HOUSE/user_codes` (see Figure 2). Once the distribution files
have been untarred, the `.tar` archives (which are left behind) are removed from `$HEN_HOUSE`.

Beyond this point, installation proceeds in a manner similar to the wizards. The script
compiles the OMEGA/BEAM utility codes `beam_build`, BEAMDP, `ctcreate`, `readphsp`,
`statdose`, `addphsp` and `dosxyz_show`. The script also checks for the existence of a C compiler
on your machine. If there is one, then it compiles the C routines used by DOSXYZnrc[15]

for parallel calculations, `$HEN_HOUSE/user_codes/dosyznrc/read_write_pardose.c` and puts the object file, `read_write_pardose.o` in `$HEN_HOUSE/lib/config` (where `config` is the name of your machine as defined during EGSnrcMP installation). Finally, `install_beam` creates the file `$HEN_HOUSE/specs/dosyznrc_config.spec`, which is used when DOSXYZnrc is compiled to determine whether `read_write_pardose.o` will be linked in or not. Note that if any compilations fail, the installation continues, but a warning will be output at the end.

Basic output from the installation appears on the screen and is also echoed in the file `install_beam.status`, which is written to the directory that `install_beam` was run from (Note that this means you MUST have write permission in this directory). Comprehensive output appears in the file `install_beam.log`, also written to the directory that `install_beam` was run from. The latter file contains the full output from compilations of the individual OMEGA/BEAM codes, etc, and is essential for debugging your installation if something goes wrong.

At this point, OMEGA/BEAM is now installed, but in order to run the codes, you must set up your user area (see Figure 3, page 16). If you are the one installing OMEGA/BEAM, then the `install_beam` script provides you with an option to set up your system automatically at the end of the installation. If you choose to do this, then `install_beam` calls the script `$HEN_HOUSE/scripts/finalize_beam_foruser`, which performs the set up. If, on the other hand, the OMEGA/BEAM system has already been installed (by you, `root`, or another user), then you will have to run `$HEN_HOUSE/scripts/finalize_beam_foruser` separately to set up your system.

The `finalize_beam_foruser` script goes through the same steps as an installation wizard (see above) to set up your system. First of all, it checks for the existence of `$HEN_HOUSE` (environment variable and directory), `$EGS_HOME` (environment variable and directory), `$EGS_CONFIG` (environment variable and file) and `$HEN_HOUSE/omega` (which will become your `$OMEGA_HOME`). If any one of these is not defined, set up terminates with a warning. If everything is defined, the script creates the subdirectory chain `beamnrc/spec_modules` in your `$EGS_HOME` area and copies the example accelerator modules (`EX10MeVe.module`, `EX16MVp.module`, `EXphantom.module`, and `EXslabs.module`) into the `spec_modules` subdirectory. Finally, the script creates subdirectory `dosyznrc` off of your `$EGS_HOME` area, copies the required source code files from `$HEN_HOUSE/user_codes/dosyznrc` into it and compiles DOSXYZnrc. If any operations fail, the set up does not stop, but a warning message will be output at the end.

At the end of the set up, you will be instructed to add the statement:

```
source /full_path_to_your_HEN_HOUSE/scripts/beamnrc_cshrc_additions
(or  . /full_path_to_your_HEN_HOUSE/scripts/beamnrc_bashrc_additions)
```

to the `.cshrc` (or `.bashrc`) file in your `$HOME` area. Then source `.cshrc` (or `.bashrc`). `beamnrc_cshrc_additions` (or `beamnrc_bashrc_additions`) will then define `OMEGA_HOME` and aliases useful for running the OMEGA/BEAM codes in Unix/Linux.

Detailed output from the `finalize_beam_foruser` script (including a reiteration of the instructions for sourcing the `beamnrc_cshrc_additions` or `beamnrc_bashrc_additions` script) is found in the file, `finalize_beam_[username].log` which will be in the directory `$HEN_HOUSE/scripts` if you have run `finalize_beam_foruser` directly from the `install_beam` script or in the directory that `finalize_beam_foruser` was run from on its own.

Note that script-based installation using the `install beam` script does not give you the option to add BEAMnrc, DOSXYZnrc and BEAMDP GUI icons to your desktop.

17.2.4   A Note on Tcl/Tk

Before running the BEAMnrc, DOSXYZnrc and BEAMDP GUIs, you must first have installed Tcl/Tk (this is already installed in most Linux systems) and have included the directory `/full path to Tcl installation/bin` in your `PATH` environment variable. For more information on installing Tcl/Tk, see the GUI Manual[4].

18   Known Problems/Restrictions

If RMAX is interior to any other boundaries in the geometry, the results will be in error since volumes are not correct and also energy past RMAX is always deposited in region(1), even if still inside another region.

The value for a particle does not take into account the distance to `RMAX CM`, i.e it is allowed to approach this boundary with large steps. This will lead to inaccuracies in the dose delivered to these regions. For accelerator simulations these errors will be small, but in unusual situations could be significant.

Variables related to the number of particle histories, such as the input variable `NCASE`, the variable `IHSTRY`, etc, have been changed to `INTEGER*8` to allow for $> 2x10^9$ histories. This is particularly useful for restarts and recombining parallel jobs, where a large number of histories may be accrued. However, some Fortran compilers do not support `INTEGER*8`. We have compiled with `INTEGER*8` successfully on Linux PC, SGI, DEC Alpha, and Sun sparc systems. On our rs6000 system, the compiler gave "length specified is not valid for the specified type" warnings, reverted to `INTEGER*4` and then compiled successfully. On our HP9000 system, the compiler gave "incompatible type-length combination" errors and compilation failed. If compilation fails due to `INTEGER*8`, go into `beamnrc.mortran` and change the macro:

REPLACE {$LONG_INT} WITH {INTEGER*8}

to:

REPLACE {$LONG_INT} WITH {INTEGER*4}

The code will now compile, but note that you are now limited to $2x10^9$ total histories.

If `ICM SPLIT`=1, *i.e.* you want to split particles going into the first CM, and even though you must input `NSPLIT PHOT` and `NSPLIT ELEC`, no splitting will occur. To get around this bug, if you wish to split particles at CM 1, you can insert a dummy CM 1 and then set `ICM SPLIT`=2.

Not all CMs output geometric information of much value for `EGS Windows` (in particular MESH and ARCCHM).

A restarted run that uses a phase space source with particle recycling will not produce dose/uncertainty results identical to a single run with the same total number of histories. This is because the last particle used before the restart may not have been recycled the full `NRCYCL` times, and restarting automatically skips to the next particle. Results will agree within uncertainty, however.

19   History of Revisions

The files `CHANGES_from_BEAMnrc06.for.BEAMnrc07`, `CHANGES_from_BEAMnrc05.for.BEAMnrc06`, `CHANGES_from_BEAMnrc03.for.BEAMnrc05`, `CHANGES_from_BEAMnrc02.for.BEAMnrc03`, and similar files on `$OMEGA_HOME/doc` contain a description of all the changes made to the BEAM and BEAMnrc system since the release mentioned. `BEAM95` was released at the first course in Oct, 1995.

19.1   Changes from `BEAMnrc08` to `BEAMnrc09`

Major changes between `BEAMnrc08` and `BEAMnrc09` are as follows:

- Modified subroutine `do_rayleigh` in `beamnrc.mortran` so that a new Rayleigh sampling scheme can be used with DBS. Modification includes no longer killing thin photons before Rayleigh interactions to avoid spikes in the photon spectra.

- `LATCH` options now work when DBS is used. They previously did not.

- CM XTUBE was modified to allow the user to have a central region in the outermost target slab comprised of a different material. Should allow more accurate modeling of X-ray tubes.

- Source 24, phase-space source incident from a user-specified angle, added. Concept and much of the coding of the source is courtesy of Patrick Downes at University of Cardiff, Wales.

- Added source 23, BEAM simulation source from a user-specified angle.

- Fixed bug in CM MLC in which a particle right on a leaf boundary with USTEP $\sim 1 \times 10^{-6}$ resulted in an endless loop.

- Fixed bug in source 8 in DOSXYZnrc (phase-space source incident from multiple directions) causing a segmentation fault when many incident angles were specified.

- Fixed bug in BEAMnrc which caused elapsed CPU time stored in the `.egsdat` file to be wrong.

- Changed source 19 from a circular gaussian source to an elliptical source defined by separate gaussian distributions in X and Y.

- Many changes to allow compilation and running in double precision (`REAL*8`).

- Added a new component module, DYNJAWS, to allow simulation of time-dependent JAW settings. In actuality, this is most likely to be used to simulate dynamic wedges. Similar to DYNVMLC, it can be operated in dynamic (beam on during setting changes) and step-and-shoot (beam off during setting changes) modes.

- Fixed a major bug in MLCQ. The distance to the nearest region boundary was calculated incorrectly in the HOWNEAR subroutine. In at least one observed case, this led to an error in electron fluence at the bottom of the CM (incident electron beam) and also caused transport through the CM to be very slow.

19.2   Changes from `BEAMnrc07` to `BEAMnrc08`

Major changes between `BEAMnrc07` and `BEAMnrc08` are as follows:

- Recoded the opening of `.egsinp`, `.egslog`, `.egslst` and `.errors` files in BEAMnrc so that you no longer require `libg2c.a` when using BEAMnrc as a shared library source. Now, BEAMnrc searches for available Fortran units (`i.e.` those not already in use by the driving program) to associate with these files. Thus, the confusion of Fortran units, for which `libg2c.a` was necessary to resolve, is no longer an issue. This should allow BEAMnrc shared libraries to be compiled using `gfortran`, which had problems with `libg2c.a`.

- Introduced dynamic and step-and-shoot modes in the DYNVMLC component module. This allows multiple treatment fields to be simulated in a single run.

- Both BEAMnrc and DOSXYZnrc can now make use of IAEA-format phase space data. In the case of BEAMnrc, IAEA-format data can now be output at scoring planes. This will allow users to use the new IAEA online phase space database. A C++ compiler is required for this functionality.

- Added source 10 to DOSXYZnrc. This is source 9 (BEAMnrc shared library source) but from multiple angles in a single simulation.

- "NRC" bremsstrahlung cross-sections, which include electron-electron bremsstrahlung effects, have been added

- The user now has the option to specify custom Rayleigh form factors for specified media if simulation of Rayleigh scattering is desired.

- The bremsstrahlung cross section enhancement (BCSE) variance reduction technique was introduced. This increases the efficiency of x-ray tube simulations by up to an order of magnitude and can even increase the efficiency of standard 6 MV accelerator simulations by 40%.

- The option to use a rejection plane which can eliminate all fat photons which might interact in the air above the phantom has been added as part of DBS.

- A grid scoring option in each scoring plane has been added so that fluence can be scored in a rectangular grid rather than just in circular or square rings.

19.3 Changes from `BEAMnrc06` to `BEAMnrc07`

Major changes between `BEAMnrc06` and `BEAMnrc07` are as follows:

- Introduced "HOWFARLESS" option for homogeneous phantom calculations in DOSXYZnrc. This option increases efficiency by $\sim 30\%$ in photon beam sources from accelerators simulated using BEAMnrc and by factors of 2.5-3.5 in monenergetic electron beams.

- `EXACT` is now the default boundary crossing algorithm in BEAMnrc (but not in DOSXYZnrc). This was changed from `PRESTA-I` after it was shown[12] that the `PRESTA-I` BCA can result in dose overprediction by up to 2.5% in a CHAMBER phantom.

- The BEAMnrc and DOSXYZnrc GUI's now give the user access to various electron impact ionization theories (only applicable for keV X-Rays) and photon cross-section data other than Storm-Israel (PEGS4).

- The parameter `$BDY_TOL` in `$OMEGA_HOME/beamnrc/beamnrc_user_macros.mortran`, which defines the "fuzzy boundary" used in the `HOWFAR` routines of various CMs to ensure that particles are actually transported beyond region boundaries was changed from 1E-4 cm to 1E-5 cm. The previous value resulted in serious underestimates of contaminant electron dose in high-energy photon beams when the JAWS CM was used.

- Introduced a more efficient range rejection scheme that works together with directional bremsstrahlung splitting (DBS). It can increase the efficiency of DBS by $\sim 20\%$.

- Source 19 (circular beam with Gaussian distribution in X-Y) can now have user-specified angular spread.

- Introduced electron splitting for phase space and BEAMnrc simulation sources in DOSXYZnrc. This is used in conjunction with photon splitting (`n_split`) to ensure that higher-weight contaminant electrons do not compromise statistics.

19.4 Changes from `BEAMnrc05` to `BEAMnrc06`

Apart from many smaller bug fixed and modifications, the major changes between the `BEAMnrc05` and `BEAMnrc06` distributions can be summarised as follows:

- Added the DYNVMLC CM, designed to simulate the Varian Millenium class of multi-leaf collimators. See the BEAM User's Manual for more details. Original coding is from Emily Heath at McGill University.

- Zdenko Sego at Carleton University has fixed and enabled the the use of multiple source model as sources in BEAMnrc (ISOURC=31). This option was disabled for many releases.

- Fixed a major bug in the phase space source for DOSXYZnrc. When particles were recycled, their Z direction cosine (`wsrc`) was not saved from one use to another, resulting in particles that either did not hit the geometry or went in non-physical directions.

Simulations either crashed or produced depth-dose curves wildly different from those expected.

- Added many `USER` macros to the BEAM code which allow a user to easily customize the `AUSGAB` routine, input parameters, data analysis, global variables, etc. These macros are changed in `beamnrc_user_macros.mortran`

- Custom user inputs can appear between the delimiters `:start user inputs:` and `:stop user inputs:` in the BEAMnrc `.egsinp` file and the GUI will handle them (but not give you access to them).

- Fixed some bugs in the new MLCE CM. Among them: it was possible for particles to be trapped in "dead air", not contained within the boundaries of any leaf side-surface, near the bottom of the MLC. Also, leaf overlap restrictions on input were too strict.

- Fixed some bugs in the JAWS CM which resulted in unnecessary warning messages being output. Also, fixed one bug which resulted in occasional particles being discarded when transported backwards ($W<0$) to the top of the jaws.

19.5   Changes from `BEAMnrc03` to `BEAMnrc05`

Apart from many smaller bug fixed and modifications, the major changes between the `BEAMnrc03` and `BEAMnrc05` distributions can be summarised as follows:

- Ported the entire set of codes to work with the EGSnrcMP system.

- Electron impact ionization added to the EGSnrc system.

- Added directional bremsstrahlung splitting (DBS) which increases the efficiency for photon accelerators by more than a factor of 5.

- Included the new CM MLCE which was mostly coded by Nick Reynaert at the University of Ghent.

- Added the BEAM shared library source (source 9) to DOSXYZnrc and other EGSnrc user codes. This required some reorganization of the BEAM code to allow it to be compiled as a shared library for use as a source in DOSXYZnrc and other user codes. Now, `beamnrc.mortran` contains subroutines which are called by either `beam_main.mortran` (regular simulation) or `beam_lib.mortran` (shared library source). Also involved changes to some phase space writing macros in `phsp_macros.mortran` so that, when the BEAM simulation is used a source, particles are dumped into a source "bin" instead of written to a phase space file.

- Introduced a new parallel processing approach which optimizes performance on a system with mixed CPU speeds and utilization factors.

- Updated DICOM CT image reading routine, `readCT_DICOM.c`, used by `ctcreate`. The routine is now independent of libraries from the DICOM CTN (central test node) and is compatible with the latest DICOM format. Thanks to Nick Reynaert at the University of Ghent for sending us the original code.

- Changed uncertainty analysis in BEAMDP so that it uses the history-by-history method used in all user codes.

- Added an option for source 1 in BEAMnrc (isotropically-radiating point source) to be collimated as a rectangle anywhere on the surface of CM 1. Previously, this could only be collimated as a square centred at Z=0.

- Modifications to BEAMnrc, DOSXYZnrc and BEAMDP GUI's to allow them to run codes on Windows.

- Fixed a bug in JAWS in which particles that should have been transported right through the tip of the JAWS (since they were within boundary tolerance of the edge) erroneously had their region numbers assigned to the JAW material. This resulted in many warnings.

- Fixed bugs in CONS3R CM in related to particle region numbers being incorrect when a particle is being transported close to the boundary of the cone.

- Modified the HOWFAR routines in all BEAM CMs so that if a particle is leaving the CM (through the top or bottom) with USTEP=0, then USTEP is given a small positive value $(1.\mathrm{E}^{-16})$ to ensure that AUSGAB gets called and the particle gets scored (if there is a scoring plane at this boundary). This bug was causing significant errors in the phase space file in perverse situations.

19.6   Changes from `BEAMnrc02` to `BEAMnrc03`

- Enabled the RANLUX random number generator in BEAMnrc and DOSXYZnrc. Can now use either RANLUX or RANMAR.

- Fixed 3 bugs in VARMLC and added the `IGNOREGAPS` input which increases the efficiency of range rejection for particles deep in the leaves (i.e., far away from the openings) by ignoring the air gaps between the leaves.

- Fixed a bug in CONS3R which caused particles to be discarded unnecessarily (with a warning) when being transported close to a conical boundary. The region error counter was not being reset properly with each new initial history.

- Put a restriction that all radii in CONESTAK be $\geq$ `$BDY_TOL`. Otherwise, a region check fails and the code could enter an endless loop.

19.7   Changes from `BEAM00` to `BEAMnrc02`

- BEAMnrc is based on the EGSnrc system.

- The statistical analysis package in BEAMnrc is based on the history by history technique which greatly reduces the uncertainty on the uncertainties assigned. See the report on history by history statistics in BEAMnrc and DOSXYZnrc[17] for a complete discussion.

- Range rejection within BEAMnrc is based on the EGSnrc calculation of range at each step but still does range rejection to ECUT, unlike the default vesion of EGSnrc which does it to AE.

- Added sources 7 and 8 to DOSXYZnrc. Source 7 is a parallel rectangular beam incident from multiple, user-selected angles. Source 8 is a phase space source incident from multiple, user-selected angles. This allows modelling of arc therapy.

- Added an option to DOSXYZnrc which allows the the user to output a `.egsphant` file from non-CT data. This allows you to display isodose contours using `dosxyz_show` in non-CT phantoms.

- Added a feature to the GUI for the JAWS CM which allows the X and Y positions of the JAWS to be set to give an arbitrary rectangular field size at a given SSD by diverging from an arbitrary focal point on the axis. Note that this is useful only for the photon jaw field size, not the electron field size (for which the jaws are typically set at a larger size).

- Upgraded VARMLC so that it can now simulate two different classes of multi-leaf collimators: those in which the tongue and groove do not extend beyond the top and bottom of the leaves (the original class which VARMLC was designed to simulate); and those in which the tongue and groove extend to either the top or bottom of the leaves (the new class).

- Corrected some serious bugs in VARMLC. Among them, the distance along the particle trajectory to the nearest vertical boundary parallel to the leaf opening direction was not calculated correctly. This was because some cases were omitted in `$VARMLC_HOWFAR`.

- Changed all variables that store summed quantities in BEAMDP from REAL*4 to REAL*8. This overcomes a potential problem when quantities from a large ($>$ 1 million) number of particles are being summed. Previously, it was possible for summed quantities to become so large that they consumed all decimal places in REAL*4 variables, and any further contributions from individual particles (especially those with low weight) were not included in the sum.

- Fixed a bug in the JAWS CM in which particles were not transported into the jaw material from the air in front of the jaws with a `$BDY_TOL` overshoot. This resulted in endless loops and `WARNING` messages when the transport into the jaws was near the jaw tips.

- Fixed a fairly major bug in DOSXYZnrc in which the `ISMOOTH` and `NRCYCL` options for phase space sources could not be used at the same time or else they caused errors. Previously, incident particle positions and direction cosines were recycled in the DOSXYZnrc phantom coordinate system. When smoothing was also turned on, then these particles were also redistributed in the phantom coordinate system. This was an error, since redistribution is only meaningful in the coordinate system of the phase space source. We solved the problem by taking care of particle recycling and redistribution before rotating the source into its position in the DOSXYZnrc phantom coordinate system.

242 NRCC Report PIRS-0509(A)revL

- Fixed a bug in BEAM in which particles were scored at a scoring plane even though they were scattered back into the CM before actually crossing the scoring plane.

- Fixed a bug in CONS3R CM. Previously, particles with a mismatch between region number and radial position were transported `MIN(USTEP,1.0E-5)` and their region number was automatically changed. This caused problems for particles that were crossing a radial boundary but ended up just short of it (due to roundoff error). In this case the region number was reset to where the particle was coming from even though, clearly, the `MIN(USTEP,1.0E-5)` step took it into the new region. We now check the radial position of the particle after the `MIN(USTEP,1.0E-5)` step before changing the region number.

20 Acknowledgements

clear bug reports, the first version of source 19 and especially the current version of the `BLOCK` CM.

The source code and bug reports explicitly mention the contributions of many others.

21   References

[1] I. Kawrakow and D. W. O. Rogers. The EGSnrc Code System: Monte Carlo simulation of electron and photon transport. Technical Report PIRS–701 (4th printing), National Research Council of Canada, Ottawa, Canada, 2003.

[2] I. Kawrakow, E. Mainegra-Hing, and D. W. O. Rogers. EGSnrcMP: the multi-platform environment for EGSnrc. Technical Report PIRS–877, National Research Council of Canada, Ottawa, Canada, 2003.

[3] D. W. O. Rogers, B. A. Faddegon, G. X. Ding, C.-M. Ma, J. Wei, and T. R. Mackie. BEAM: A Monte Carlo code to simulate radiotherapy treatment units. *Med. Phys.*, 22:503 – 524, 1995.

[4] J. A. Treurniet, B. R. B. Walters, and D. W. O. Rogers. BEAMnrc, DOSXYZnrc and BEAMDP GUI User's Manual. *NRC Report PIRS 0623(rev C)*, 2004.

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

[6] D. W. O. Rogers, I. Kawrakow, J. P. Seuntjens, and B. R. B. Walters. NRC User Codes for EGSnrc. Technical Report PIRS–702, National Research Council of Canada, Ottawa, Canada, 2000.

[7] D. W. O. Rogers, C.-M. Ma, G. X. Ding, and B. Walters. BEAM Users Manual. *NRC Report PIRS 509a*, 1995.

[8] D. W. O. Rogers, C.-M. Ma, G. X. Ding, and B. Walters. BEAM Users Manual. *NRC Report PIRS 509(a)revB*, 1997.

[9] D. W. O. Rogers, C.-M. Ma, G. X. Ding, B. Walters, D. Sheikh-Bagheri, and G. G. Zhang. BEAM98 Users Manual. *NRC Report PIRS 509(a)revC*, 1998.

[10] D. W. O. Rogers, C.-M. Ma, G. X. Ding, B. Walters, D. Sheikh-Bagheri, and G. G. Zhang. BEAMnrc Users Manual. *NRC Report PIRS 509(a)revF*, 2001.

[11] D. W. O. Rogers, B. Walters, and I. Kawrakow. BEAMnrc Users Manual. *NRC Report PIRS 509(a)revH*, 2004.

[12] B. R. B. Walters and I. Kawrakow. Technical note: Overprediction of dose with default PRESTA-I boundary crossing in DOSXYZnrc and BEAMnrc. *Med. Phys.*, 34:647 – 650, 2007.

[13] C.-M. Ma, P. Reckwerdt, M. Holmes, D. W. O. Rogers, and B. Geiser. DOSXYZ Users Manual. *NRC Report PIRS 509b*, 1995.

[14] Blake Walters and D. W. O. Rogers. QA for the BEAM System; Component Modules, Variance Reduction Options and Source Routines. *NRC Report PIRS–509k*, 1995.

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

[16] J. A. Treurniet and D. W. O. Rogers. EGS_Windows4.0 User's Manual. *NRC Report PIRS–0669*, 1999.

[17] 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.

[18] Iwan Kawrakow, D. W. O Rogers, and B.R.B. Walters. Large efficiency improvements in BEAMnrc using directional bremsstrahlung splitting. *Med. Phys.*, 31:2883 – 2898, 2004.

[19] D. W. O. Rogers, I. Kawrakow, J. P. Seuntjens, B. R. B. Walters, and E. Mainegra-Hing. NRC User Codes for EGSnrc. Technical Report PIRS–702(RevB), National Research Council of Canada, Ottawa, Canada, 2003.

[20] J. A. Treurniet and D. W. O. Rogers. BEAM, DOSXYZ and BEAMDP GUI User's Manual. *NRC Report PIRS 0623(rev A)*, 1999.

[21] Daryoush Sheikh-Bagheri and D. W. O. Rogers. BEAM Example: A 16 MV photon beam. *NRC Report PIRS–509i*, 1995.

[22] G. G. Zhang and D. W. O. Rogers. BEAM Example: A 10 MeV electron beam. *NRC Report PIRS 509h*, 1995.

[23] Blake Walters and D. W. O. Rogers. BEAM Example: Depth-Dose in a Phantom. *NRC Report PIRS–509j*, 1995.

[24] A. F. Bielajew, R. Mohan, and C. S. Chui. Improved bremsstrahlung photon angular sampling in the EGS4 code system. *National Research Council of Canada Report PIRS-0203*, 1989.

[25] D. W. O. Rogers and A. F. Bielajew. Monte Carlo techniques of electron and photon transport for radiation dosimetry. In K. R. Kase, B. E. Bjärngard, and F. H. Attix, editors, *The Dosimetry of Ionizing Radiation,Vol III*, pages 427 – 539. Academic Press, 1990.

[26] 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.

[27] E. Mainegra-Hing and I. Kawrakow. Efficient x-ray tube simulations. *Med. Phys.*, 33:2683 – 2690, 2006.

[28] R. Capote, R. Jeraj, C.-M. Ma, D.W.O. Rogers, F. Sanchez-Doblado, J. Sempau, J. Seuntjens, and J.V. Siebers. Phase-Space Database for External Beam Radiotherapy. *IAEA Report INDC(NDS)-0484*, 2005.

[29] M. Lüscher. A portable high-quality random number generator for lattice field theory simulations. *Computer Phys. Commun.*, 79:100 – 110, 1994.

[30] F. James. RANLUX: A Fortran implementation of the high-quality pseudorandom number generator of Lüscher. *Computer Phys. Commun.*, 79:111 – 114, 1994.

[31] G. Marsaglia and A. Zaman. A new class of random number generators. *Annals of Applied Probability*, 1:462 – 480, 1991.

[32] G. Marsaglia, A. Zaman, and W. W. Tsang. Toward a universal random number generator. *Statistics and Probability Letters*, 8:35 – 39, 1990.

[33] D. W. O. Rogers. Low energy electron transport with EGS. *Nucl. Inst. Meth.*, 227:535 – 548, 1984.

[34] B. J. Foote and V. G. Smyth. The modelling of electron multiple-scattering in EGS4/PRESTA and its effect on ionization-chamber response. *Nucl. Inst. Meth.*, B100:22 – 30, 1995.

[35] A. F. Bielajew and D. W. O. Rogers. PRESTA: The Parameter Reduced Electron-Step Transport Algorithm for electron Monte Carlo transport. *Nuclear Instruments and Methods*, B18:165 – 181, 1987.

[36] H. W. Koch and J. W. Motz. Bremsstrahlung cross-section formulas and related data. *Rev. Mod. Phys.*, 31:920 – 955, 1959.

[37] S. M. Seltzer and M. J. Berger. Bremsstrahlung spectra from electron interactions with screened atomic nuclei and orbital electrons. *Nucl. Inst. Meth. Phys. Res. B* **12**, 12:95 – 134, 1985.

[38] S. M. Seltzer and M. J. Berger. Bremsstrahlung energy spectra from electrons with kinetic energy from 1 kev to 10 gev incident on screened nuclei and and orbital electrons of neutral atoms with z = 1-100. *Atomic Data and Nuclear Data Tables*, 35:345–418, 1986.

[39] ICRU. Stopping powers for electrons and positrons. ICRU Report 37, ICRU, Washington D.C., 1984.

[40] O. Klein and Y. Nishina. Über die Streuung von Strahlung durch freie Elektronen nach der neuen relativistischen Quantendynamik von Dirac. *Z. für Physik*, 52:853–868, 1929.

[41] R. Ribberfors. . *Phys. Rev. B*, 12:2067, 1975.

[42] J. W. Motz, H. A. Olsen, and H. W. Koch. Pair production by photons. *Rev. Mod. Phys.*, 41:581 – 639, 1969.

[43] F. Sauter. Über den atomaren Photoeffekt in der K-Schale nach der relativistischen Wellenmechanik Diracs. *Ann. Physik*, 11:454 – 488, 1931.

[44] E. Storm and H. I. Israel. Photon cross sections from 1 keV to 100 MeV for elements $Z=1$ to $Z=100$. *Atomic Data and Nuclear Data Tables*, 7:565 – 681, 1970.

[45] J. H. Hubbell and I. Øverbø. Relativistic atomic form factors and photon coherent scattering cross sections. *J. Phys. Chem. Ref. Data*, 9:69, 1979.

[46] I Kawrakow. Electron impact ionization cross sections for egsnrc. *Med. Phys. (abstract)*, 29:1230, 2002.

[47] D. E. Cullen, S. T. Perkins, and J. A. Rathkopf. The 1989 Livermore Evaluated Photon Data Library (EPDL). *Lawrence Livermore National Laboratory Report UCRL-ID-103424 (Livermore, Calif)*, 1990.

[48] M. J. Berger and J. H. Hubbell. XCOM: Photon Cross Sections on a Personal Computer. Report NBSIR87–3597, NIST, Gaithersburg, MD20899, 1987.

[49] Iwan Kawrakow, Ernesto Mainegra-Hing, and D.W.O. Rogers. EGSnrcMP, the new multi platform version of EGSnrc. *Med. Phys. (abstract)*, 31:1731, 2004.

[50] I. Kawrakow. egspp: the EGSnrc C++ class library. Technical Report PIRS–899, National Research Council of Canada, Ottawa, Canada, 2005.

[51] E. Mainegra-Hing. User Manual for egs_inprz, a GUI for the NRC RZ user-codes. Technical Report PIRS–801(RevA), National Research Council of Canada, Ottawa, Canada, 2003.

[52] C.-M. Ma and D. W. O. Rogers. BEAMDP Users Manual. *NRC Report PIRS 509c(rev A)*, 2004.

[53] C.-M. Ma and D. W. O. Rogers. BEAMDP as a General-Purpose Utility. *NRC Report PIRS 509e(rev A)*, 2004.

[54] B. R. B. Walters and D. W. O. Rogers. DOSXYZnrc Users Manual. *NRC Report PIRS 794 (rev B)*, 2004.

[55] I. Kawrakow. The dose visualization tool dosxyz_show. *NRC Report PIRS 0624*, 1998.

Appendix A:
Specifications for Component Modules for BEAMnrc

D. W. O. Rogers, G.X. Ding, C.M. Ma and B.A. Faddegon

Ionizing Radiation Standards

National Research Council of Canada Ottawa

Printed: October 9, 2012
Last Edited: 2001-02-07 14:30:56-05

Abstract

This report is for BEAMnrc code developers and is not needed for those who just want to use the code. The report specifies what each component module must do, how it must do it, the tools available and the documentation to be followed. There is a separate report describing the QA to be done on a `CM` if it is modified[14]. It is assumed that the reader is an experienced `EGSnrc` user.

A.1   Overview

BEAMnrc is a general purpose code for doing Monte Carlo simulations of radiotherapy beams. One of its design features is that each part of the accelerator or source unit is considered to be a single component module which takes up an horizontal slab portion of the accelerator. These component modules are re-usable and are completely independent. They must communicate with the rest of the system in certain well specified ways. The purpose of this document is to list all the specifications of such a component module including the documentation required for such a component module. The quality assurance required for each component module is described in an associated report by Walters and Rogers (QA for the BEAM System: Component Modules, Variance Reduction Options and Source Routines).

Two `MORTRAN` source files make up each component module. The `MORTRAN` macros specific to component module `CMNAME` are contained in `CMNAME_macros.mortran`. These macros are used by BEAMnrc proper and/or the EGS subroutines and/or the component module subroutines. The set of subroutines specific to component module `CMNAME` are contained in `CMNAME_cm.mortran`.

When you build an accelerator, `beam_build` ensures that within each `CM` the string `$CMNAME` is replaced by a user-supplied identifier everywhere it appears in the `CMs` mortran source code. The subroutine and macro names, `COMMON` name, and variable names specific to a given component module all either begin or end with `$CMNAME`. Every component module must have an unambiguous identifier. This prevents duplication of any of these names when multiple component modules are used in a simulation. Using this convention, the same component module may appear many times in a BEAMnrc code.

`CMNAME_macros.mortran`
 The file `CMNAME_macros.mortran` contains at least two `MORTRAN` replacement macros used by the component module subroutines. The macro `COMIN/CM_$CMNAME/` is the `COMMON` block replacement macro for component module `CMNAME`. The `$CMNAME_CM_HOWNEAR` macro is required by `MORTRAN SUBROUTINE ELECTR` when the `HOWNEAR` replacement macro of BEAMnrc proper is used. This macro calculates the distance from the current location of the particle to the nearest boundary in the component module with identifier `CMNAME`. The macro is also usually used in subroutine `HOWFAR_$CMNAME` as well. Note that in most of the existing CMs, the `$CMNAME_CM_HOWNEAR` macro is just a call to a `HOWNEAR_$CMNAME` subroutine. The subroutine is located in the file `$CMNAME_cm.mortran` and performs the actual calculation of perpendicular distance to the nearest boundary. The use of a subroutine `HOWNEAR_$CMNAME` (with a call to it from the `$CMNAME_CM_HOWNEAR` macro) is recommended in EGSnrc because variables associated only with the calculation of `HOWNEAR` need only be declared inside the subroutine. Otherwise, these variables must be appended to the `$DEFINE-LOCAL-VARIABLES-ELECTR` macro in EGSnrc.

`CMNAME_cm.mortran`
 The `CMNAME_cm.mortran` file contains the following subroutines written for each component module:

- `HOWFAR_$CMNAME` – a standard EGSnrc `SUBROUTINE HOWFAR` for a component module `CMNAME`. It is used during the simulation for defining the geometry via boundary

checking and setting region-dependent parameters.

- HOWNEAR_$CMNAME – a subroutine for component module $CMNAME which calculates the perpendicular distance to the nearest boundary (ie not along the particle trajectory). This is called from the macro $CMNAME_CM_HOWNEAR. See above for a more detailed description of the relationship between the HOWNEAR subroutine and the HOWNEAR macro.

- WHERE_AM_I_$CMNAME – used to determine region of particle upon entry into component module CMNAME.

- INPUT_$CMNAME – prompts for and digests input from the interactive user or the parameter-definition file (*i.e.* file.egsinp) for information related to component module CMNAME.

- ISUMRY_$CMNAME – writes summary of input for component module CMNAME to listing.

A.2 Writing Component Modules

Source for all CMs must be written in MORTRAN3. See the EGS4 manual, SLAC265, for a specification of MORTRAN3.

Component module names are capitalized, from 1 to 8 characters long, and are unambiguous, for example, the component module SLABS is already in use and new component modules must not have this name. When writing component modules it is useful to adhere to the following established convention for code formats:

- MORTRAN3 replacement macros specific to the component module are placed in CMNAME_macros.mortran where CMNAME is the name of the component module.

- Indent 3 spaces for all MORTRAN3 LOOPS, and FORTRAN DO and IF statements.

- Capitalize all MORTRAN3 (this is essential) and FORTRAN code.

- Format for replacement macros should be as follows to allow the CMtoc utility to be used: REPLACE {text} WITH {<CR>

- Incorporate extensive documentation at the start of each component module subroutine file, following the conventions established in SLABS_cm.mortran.

A.2.1 Tips

- To prepare for writing a new component module or modifying an existing one, read the in-line documentation at the beginning of beamnrc.mortran, beamnrc_user_macros.mortran, and beamnrc_cm_macros.hdr. Next read all of SLABS_cm.mortran and SLABS_macros.mortran. Also look at the variables in COMMON/CMs/, which are of general use in writing component modules.

- Write `SUBROUTINE HOWFAR_$CMNAME` in `CMNAME_cm.mortran` first, writing the `COMIN/CM_$CMNAME` macro, which defines all the variables needed, at the same time. The outer boundary of the component module is checked in the `HOWFAR` subroutine in `BEAMnrc` proper, and need not be checked here.

- Next write the `SUBROUTINE HOWNEAR_$CMNAME` for component module `CMNAME` in `CMNAME_cm.mortran`. This macro returns the nearest distance to any boundary from the current particle position (`DNEAR`). Then, write a macro in `$CMNAME_macros.mortran` called `$CMNAME_CM_HOWNEAR` which is just a call to subroutine `HOWNEAR_$CMNAME`. If your geometry is too complex to calculate the nearest perpendicular distance to a boundary, then just write the line:
  `REPLACE {$CMNAME_CM_HOWNEAR(#)} WITH { {P1}=0; }`
  in the `$CMNAME_macros.mortran` file, and omit the subroutine `HOWNEAR_$CMNAME` from `CMNAME_cm.mortran`.

- Write the rest of the component module subroutines for `CMNAME` in the order:
  `WHERE_AM_I_$CMNAME`, `SUBROUTINE INPUT_$CMNAME` followed by `ISUMRY_$CMNAME`.

- Use the macro and subroutine files from an established component module such as `SLABS` or `CONESTAK` as a template for the new component module. Much of the code can be used for the new component module with only minor modifications. This does not generally apply to the `SUBROUTINE HOWFAR_$CMNAME`, the most difficult of the component module subroutines to write.

- When the coding is completed, follow established quality assurance procedures closely. The procedures contained in the QA document listed at the beginning of this report are the minimum.

- Your code will be read and modified by others, so ensure that it is clearly laid out, extensively documented, and conforms closely to the established conventions.

A.3   Specifications–CMNAME_macros.mortran

For every component module, this file must be created. It contains at least the two macros specific to the particular `CM` specified below, but the user may write as many as convenient.

Naming conventions in `CMNAME_macros.mortran`:

- Use `$CMNAME` throughout (*eg.*`$SLABS`) except in those cases (usually output) where you want the original `CM` name to be used, usually to identify the type of `CM` for output purposes.

- The component module name (`$CMNAME`) appears at the start of all `MORTRAN3` macro names (*eg.*`$SLABS_CM_HOWNEAR`).

- The component module name (`$CMNAME`) appends all local variable names (*eg.* `ICM_$SLABS`).

A.3.1 COMIN/CM_$CMNAME macro

This macro defines the `COMIN` which contains the values associated with this specific `CM` and a few links to the rest of the simulation. Nothing is mandatory since only the writer of a given `CM` ever uses these variables. Typically we find it useful to define at least the following variables:

- `ICM_$CMNAME`: an index specifying which `CM` this is, starting from 1 nearest the source. It is usually set in
  `SUBROUTINE INPUT_$CMNAME`.

- `IR_$CMNAME`: the local region number within the CM. This is often used in the `HOWFAR` and `HOWNEAR` subroutines.

- `IRSTART_$CMNAME`: first absolute region number for this `CM`. It is usually set in
  `SUBROUTINE INPUT_$CMNAME` and equals `IR_start_CM(ICM)` from `COMIN/CMs`.

- `IREND_$CMNAME`: last absolute region number for this `CM`. It is usually set in
  `SUBROUTINE INPUT_$CMNAME` and equals `IR_start_CM(ICM+1)-1` from `COMIN/CMs`.

- `TITLE_$CMNAME`: title(60) (character*1). It is usually set in
  `SUBROUTINE INPUT_$CMNAME`.

- `N_GAP_$CMNAME`: flag, = 0 if no air gap this `CM`, = 1 if air gap at top of `CM`. It is usually set in `SUBROUTINE INPUT_$CMNAME`.

As well, all geometric parameters associated with this `CM` are defined in this `COMIN` and filled in `SUBROUTINE INPUT_$CMNAME` usually. These MUST have the string `_$CMNAME` appended at the end of their name to ensure unique names if `COMIN`s are defined more than once.

A.3.2 $CMNAME_CM_HOWNEAR(#) macro

This macro is usually just a call to `SUBROUTINE HOWNEAR_$CMNAME` (See Section A.4.5 below), which returns the perpendicular distance from the particle to the nearest boundary. However, this macro is the only means by which the `HOWNEAR` subroutine is called and is used in EGSnrc and sometimes in the `SUBROUTINE HOWFAR_$CMNAME`.

A.4 Specifications–CMNAME_cm.mortran

Naming conventions in `CMNAME_cm.mortran`:

- Use `$CMNAME` throughout, that is, the name of the component module should not be directly written into the code. This permits a change of identifier simply by changing the `.module` file.

- `$CMNAME` appends all subroutine names.

- $CMNAME appears at the start of all MORTRAN macro names.

- $CMNAME appends all local variable names.

General Requirements

At the top of the source a set of comments on records starting with "I> must define the geometry accurately and the inputs required from unit 5. Use of the "I> comment ensures that we will be able to pick up the description of the input for the next edition of the users manual (*i.e.*this is the primary documentation for input - make it clear).

A.4.1   SUBROUTINE INPUT_$CMNAME

On being called, the following information is available to the routine (all via COMIN/CMs except NMED which is in COMIN/GEOM):

- ICM: the number of this CM, starting from 1 for smallest z.

- IR_start_CM(ICM): the absolute region number this CM starts at. It is set by previous CM's input routine and for the ICM=1 case it is set in main (to 2 since region 1 is the exterior).

- RMAX_CM(ICM): outer boundary of this CM, read in by main for each CM. It is either the radius or 1/2 the side for a square boundary.

- NMED: The number of media *already* asked for in any given simulation.

- MEDIA(24,I),I=1,NMED: names of media already asked for.

The following variables from COMIN/CMs must all be set, not necessarily from user input (*i.e.* the code may define them for the user).

- RMAX_CM_FLAG(ICM), flag for each CM which specifies what quantity RMAX_CM is for this CM (RMAX_CM is set by MAIN). If flag = 0, not used; flag = 1, it is radius of outer boundary of CM; flag = 2, it is distance to edge of square outer boundary of CM (1/2 of side). It is used in main SUBROUTINE HOWFAR to check the outer boundaries, after the individual HOWFAR_$CMNAME routine has done its work. One should avoid redundant calculations.

- IERR_GEOM(ICM), flag which must be initialized to zero and set non-zero to flag input problem. Set < 100, it specifies number of errors detected within CM, >100 specifies that the CM overlaps the one above it. MAIN routine will exit before transport, but input is continued and checked.

- Z_min_thick(ICM,j) and MED_min_thick(ICM,j): minimum thickness in cm and medium number of up to $j = 5$ regions in ICM for an electron going through CM # ICM. It is used for range rejection. It is set in each CM input routine. Region with j=1 is closest to the bottom plane. Often only one is needed or possible and this is given by the total thickness of the CM and medium is air (*i.e.* MEDIUM 1).

- `Z_min_CM(ICM+1)` for next `CM` is set as the back of this `CM`.

- `Z_gap_THICK(ICM)`,thickness of air gap at front of `CM`, if it exists. If `Z_gap_THICK(ICM)`= 0, then there is no air gap at the front of this `CM`.

- `IR_start_CM(ICM+1)`: first absolute region for next `CM`, set once the number of regions in the current `CM` is known.

We find it useful to set the variables in `COMIN/CM_$CMNAME` listed in section A.3.1.

All variables required by the geometry for this `CM` (with `_$CMNAME` at the end of the name) must be input and stored in `COMIN/CM_$CMNAME`.

The following `EGSnrc` variables must be set:

- `ECUT(IRA)` & `PCUT(IRA)` for all regions `IRA` in the `CM` (including the air gap). If not set, the defaults are `AE` and `AP` for the medium in each region.

- `MED(IRA)` the medium index for every region must be set

- `MEDIA(24,I)` must be loaded for new `MEDIA` that are asked for. Note the macro `$MED_INPUT($CMNAME)` handles this automatically. This macro requires `NMED` to contain the current number of different media that have already been read in and stored in `MEDIA`.

- `RHOR(IRA)` the density in each region in g/cm^3. It need only be set by the user if it is different from the value included in the `PEGS4` data set. After the call to `HATCH`, it will include this latter value if left 0.0 prior to the HATCH call.

The following variables in `COMIN/SCORE` must be set.

- `DOSE_ZONE(IRA)` $= 0$ if no dose scored, otherwise =dose zone that dose from this region scored in. Note that one dose zone can include the dose deposited in many geometric regions.

- `NDOSE_ZONE`, largest dose scoring zone number assigned so far.

- `IREGION_to_BIT(IRA)`: mapping from absolute regions to `LATCH` bits (*i.e.* which bit in variable `LATCH` is this region associated with).

- `MAX_BIT`: current largest bit being set.

The following variable in `COMIN/GEOM` must be set: `NREG`: total number of regions in the geometry model up to and including this `CM` (in `COMIN/GEOM`). It should equal `IR_start_CM(ICM+1)-1`.

A.4.2   SUBROUTINE ISUMRY_$CMNAME

This routine, which is called after `HATCH`, must summarize all data related to the use of this `CM` in any given run. Specifically, it must be possible to completely reconstruct the input file from the information in the output listing. Also the Revision No. of the CM must be echoed to the listing. Use already existing CMs as examples.

All output MUST fit in 80 columns and allow `FORTRAN` carriage control to be used for printing purposes (*i.e.* col 0 defines double spacing, new lines *etc.*). However, for screen outputs it is better to just avoid these `FORTRAN` controls.

It must also increment the mass of each scoring zone by the mass of each region in this `CM` which is in that zone. Specifically,
```
 AMASS(IDD) = AMASS(IDD) + RHOR(IRA)*VOLUME(IRA)
```
where `IDD` is the scoring zone associated with region `IRA`, (`IDD = DOSE_ZONE(IRA)`)

A.4.3   SUBROUTINE HOWFAR_$CMNAME

This is a more or less standard `EGSnrc SUBROUTINE HOWFAR` which applies just to this `CM`. See examples from already coded `CMs`.

The routine does NOT consider the `RMAX_CM` boundaries since these are handled afterwards in the main `SUBROUTINE HOWFAR`.

As a particle leaves the `CM` to enter the `CMs` on either side, the `SUBROUTINE WHERE_AM_I(IICM,IDIR)` is useful. This routine is called from `SUBROUTINE HOWFAR_$CMNAME` when a particle reaches the front (`IDIR = 1`) or back (`IDIR = -1`) of a component module. The index of the new component module, `ICMNEW` (passed in `COMIN/CMs/`), is determined and the appropriate `SUBROUTINE WHERE_AM_I_$CMNAME` for the new `CM` is called to determine the new region number, `IRNEW`.

A.4.4   SUBROUTINE WHERE_AM_I_$CMNAME

Subroutine `WHERE_AM_I_$CMNAME` determines the new region number when a particle traverses the front or back boundary of a component module. Whenever a particle is to be transported to a component module boundary in `HOWFAR`, the subroutine `WHERE_AM_I` is called. The current component module, `ICM`, and particle direction (`IDIR`, backwards or forwards) are transferred to `WHERE_AM_I WHERE_AM_I` determines which component module the particle is about to enter and calls the `WHERE_AM_I_$CMNAME` subroutine for that component module, transferring the particle direction. The region number that the particle is about to enter, `IRNEW`, is determined in `WHERE_AM_I_$CMNAME` from the knowledge of which surface the particle is entering through (front if `IDIR=1`, back if `IDIR=-1`) and the (X,Y) coordinates of the particle.

A.4.5   SUBROUTINE HOWNEAR_$CMNAME

This subroutine calculates the perpendicular (minimum) distance from the particle position
to the nearest region boundary (stored globally in the variable `DNEAR`). It is always called
using the `$CMNAME_CM_HOWNEAR` macro (See Section A.3.2 above) See already coded CMs for
examples of how to code this subroutine.

Note: this subroutine does NOT check against the `RMAX_CM` boundary (nor does the main
`CALL HOWNEAR` macro in BEAMnrc). This means that particles do not approach the outer
boundary with lateral correlations turned off (if using the PRESTA-I boundary crossing
algorithm) or in single-scattering mode (if using EXACT boundary crossing algorithm). It
also means that the outer boundary is not used in range rejection.

A.5   Specifications–COMIN/CMs/

This `COMIN` contains the geometrical and range rejection information of interest to all com-
ponent modules:

`E_min_out(ICM)`	minimum energy of electron leaving a CM ICM which can reach the nearest downstream scoring region with energy greater than ECUT in the scoring region. For use in range rejection. Set in MAIN because needs info from all CMs past the current one.
`MAX_CMs`	number of CMs.
`MED_IN`	24-character name of last medium input in INPUT_$CMNAME.
`ICM`	CM index, incremented before call to INPUT_$CMNAME, set in SRCHST and in HOWFAR during particle transport.
`ICMNEW`	Next CM, set in WHERE_AM_I, different than ICM (only?) if particle transported to CM boundary.
`ICM_to_SCORE(ICM)`	scoring plane associated with ICM, 0 => none. Set in main based on input IPLANE_to_CM values.
`IERR_GEOM(ICM)`	geometry-checking flag for each CM, 0=>no errors detected, 1-99 specifies number of errors detected within CM, >100 specifies that CM above overlaps
`IR_start_CM(ICM)`	region number of first region in CM, set by previous CM, read in subroutine INPUT_$CMNAME.
`IR_to_CM(IR)`	pointer used in HOWFAR which denotes the CM region IR is in. It is set in MAIN.
`RMAX_CM(ICM)`	outer boundary of treatment head, particles are discarded if they move outside of this boundary. Read in at step 2 in MAIN.
`RMAX_CM2(ICM)`	square of maximum radius.
`RMAX_CM_FLAG(ICM)`	flag for type of outer boundary of CM: 0--bounds of CM are all set in HOWFAR_$CMNAME, 1--CM is bounded by cylinder of radius RMAX_CM(ICM), 2--CM is bounded by square box, walls at RMAX_CM(ICM) and -RMAX_CM(ICM). Set in INPUT_$CMNAME.

```
Z_min_CM(ICM)       minimum Z for each CM, set by previous CM in INPUT_$CMNAME
                    (back of previous CM) usually following convention that
                    downstream surface of source or accelerator exit window
                    is at Z = 0.0. Last value (ICM = MAX_CMs + 1) is
                    maximum Z of model.
Z_gap_THICK(ICM)    thickness of air gap at front of CM to fill in space
                    between Z_min_CM and front of CM, set in INPUT_$CMNAME
Z_min_thick(ICM,j)  minimum thickness in cm of up to j = 5 regions in
                    ICM for an electron going through ICM. It is used for
                    range rejection. It is set in each CMs input routine.
                    j=1 is closest to the bottom plane. Often only
                    one = total thickness (air).
MED_min_thick(ICM,j)  medium values corresponding to min thicknesses.
ITDOSE_ON           if dose components scored, this flag is 1, otherwise 0
                    -dose components are from LATCH values or incident charge
ICM_CONTAM          If ITDOSE_ON is 1, & ICM_CONTAM is >= 1
                    then dose is broken into 2 components based on the charge
                    entering the front of ICM ICM_CONTAM.
IQ_CONTAM           charge of the particles considered to be the contaminant
                    on entering ICM = ICM_CONTAM (identified via
                    bit 30 in LATCH).
XTUBE_EXISTS        flag is 0 unless first CM in accelerator is XTUBE, in
                    which case it is 1.
ANGLE               = angle between X-ray target surface and z-axis
CMTYPE(ICM)         8 character ordered array with names of CM types
CMLIST(ICM)         8 character ordered array with identifiers for CMs
AIR_INDEX           index for the ''air'' region =1 unless 0 for vacuum
```

A.6   Useful Utilities

If you are running under Unix/Linux, then the following scripts (found in $OMEGA_HOME/beamnrc/CMs) are useful for developing CMs.

checkCM8 checks that once $CMNAME is expanded to 8 characters (as it is when changed to identifier names prior to MORTRAN3 compilation), that all records are less than 80 columns, since that is all that MORTRAN3 handles.

CMtoc CM "table of contents" script. Produces a combined listing of $CMNAME_macros.mortran and $CMNAME_cm.mortran, toc:$CMNAME, with line numbers, page headings and a table of contents at the top which is based on the strings "toc: and %E in the source file and the locations of all REPLACE macros.

Index