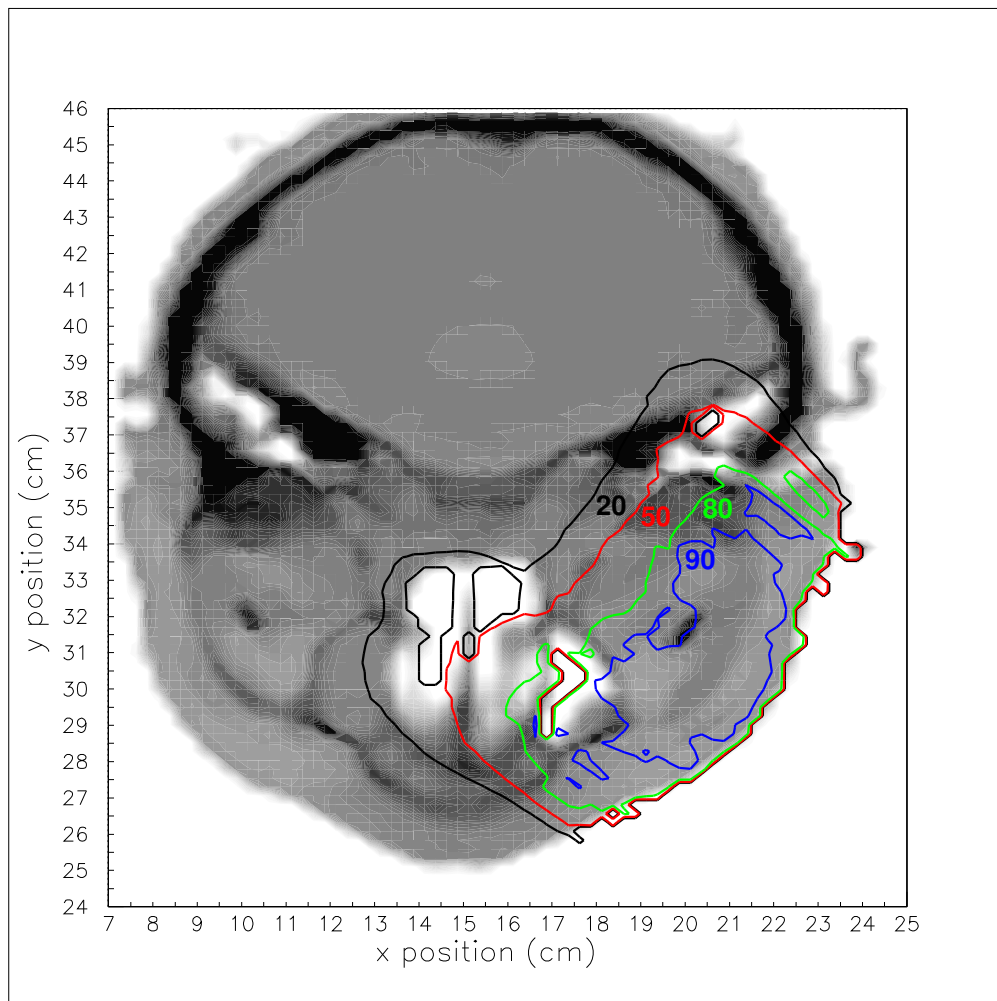# DOSXYZnrc Users Manual

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

Ionizing Radiation Standards
National Research Council of Canada, Ottawa K1A 0R6

Printed: January 13, 2025

Abstract

DOSXYZnrc is an EGSnrc-based Monte Carlo simulation code for calculating dose distributions in a rectilinear voxel phantom and is based directly on the DOSXYZ code developed for the EGS4 code system (see NRC Report PIRS-509B). DOSXYZnrc is part of the OMEGA-BEAM system of codes developed at NRC. Density and material in every voxel may vary. A variety of beams may be incident on the phantom, including full phase-space files from BEAMnrc and beams characterized using Beam Characterization models. The companion program `ctcreate` is capable of reading in a CT data set of Hounsfield numbers and converting it into the information needed by DOSXYZnrc to simulate transport in a phantom (i.e. the appropriate material and density are specified in each voxel). Any of the available beams can be incident on this CT phantom. The code includes a restart facility and can be run on parallel computing platforms. The statistical analysis is based on a history by history method as opposed to the batch method used in DOSXYZ.

This user's manual covers general DOSXYZnrc inputs, geometries and outputs. It contains information on how to compile and run DOSXYZnrc using the EGSnrcMP system. It also describes the use of `ctcreate`.

The figure on the front page shows a PAW visualisation of the isodose curves from a DOSXYZ simulation in which a Clinac 2100c 18MeV electron beam (simulated using a multiple-source model–with 35 million initial histories) was incident on the head and neck of a CT phantom. The visualisation was implemented by Daryoush Sheikh-Bagheri.

Contents

1   Introduction

1.1   Overview

DOSXYZnrc is a general-purpose Monte Carlo EGSnrc[1, 2] user-code for 3-dimensional absorbed dose calculations. EGSnrc/DOSXYZnrc simulates the transport of photons and electrons in a Cartesian volume and scores the energy deposition in the designated voxels. DOSXYZnrc is "stand alone", in the usual EGSnrc sense in that it is controlled by the $\times$ .pegs4dat and .egsinp files and is capable of writing out ASCII formatted dose distribution arrays. The code uses the EGSnrcMP system which is described in detail in its own Users Manual[3]. There is also a graphical user interface (GUI) which allows input files to be created and executed graphically[4]. Much of the information in this manual is conveniently accessible via the GUI's help files.

The geometry is a rectilinear volume with the X-Y plane on the page, X to the right, Y down the page and the Z-axis into the page. Voxel dimensions are completely variable in all three directions. Every voxel (volume element) can have different materials and/or varying densities (for use with CT data). The code allows sources such as a monoenergetic diverging or parallel beam, phase-space data generated by a BEAMnrc simulation, or a model-based beam reconstruction produced by BEAMDP.

DOSXYZnrc has a number of important and unique features such as dose component calculations, a wide variety of source configurations and beam reconstruction techniques, CT to phantom conversion (via ctcreate), restart capabilities, phase-space redistribution, *etc.*

ctcreate is a stand alone program which converts CT data sets into the data needed for DOSXYZnrc to do a simulation. At present it handles ADAC Pinnacle, AAPM, CADPLAN and DICOM formats for the CT files. If you develop extensions, why not share them with your colleagues? Send them to us and we will integrate them into the standard distribution with full acknowledgement of the source.

The DOSXYZnrc code, in common with the BEAMnrc system is written for a preprocessor of Fortran77 called MORTRAN. The user does not need to know MORTRAN to use the code, but its elements are needed for modifications (see refs [5] or [2]).

1.2   History of DOSXYZnrc

DOSXYZ started out as a demonstration code that Dave Rogers wrote in March 1986 to show Ralph Nelson that special purpose coding of rectilinear voxels was faster than using Ralph's more general macros. At about that time it was used to estimate the time required to do a full Monte Carlo treatment planning calculation and the results published 3 years later in a book chapter[6]. It then became the basis for a Monte Carlo timing benchmark[7] which was regularly updated and available on the WWW for many years (until 2000). The OMEGA project took this code over and added a variety of different source routines with coding contributions from Charlie Ma, Bruce Faddegon, George Ding, Dave Rogers, Alex Bielajew, and Paul Reckwerdt. More recent modifications have reduced the array space used by the

code, and added beam characterization inputs (Charlie Ma), btree inputs, (Brian Geiser, but no longer supported), correlated sampling (Mark Holmes, but no longer supported). Blake Walters and Mark Holmes added the CT reading ability in summer 1996. Blake Walters separated out the `ctcreate` code in summer 1997 to make the DOSXYZ code much smaller and thus able to handle much larger array sizes. Blake Walters added the `dsurround` option for reducing simulation time for depth-dose curves and dose profiles and also the coding for parallel processing in 1998. In 1999, an option was added to allow the user to run N parallel jobs using a phase space source that exists in N separate pieces (option now only available using the old `pprocess` script and not with the new built-in parallel processing functionality).

Prior to the 1999 revD of this manual, the authors included Paul Reckwerdt, Mark Holmes and Brian Geiser who had been involved with the original code to read Pinnacle CT data sets, correlated sampling and BTREE beam modelling respectively. These extensions are either no longer used or are not supported and, thus, these authors are no longer included as authors of the users manual. Manuals and/or notes by Geiser and Holmes are available separately describing BTREE[8] and correlated sampling[9]. This latter manual used to be part of the DOSXYZ Users Manual prior to 1999.

In 2001, with the help of Iwan Kawrakow, Blake Walters ported the DOSXYZ code to the EGSnrc system to give DOSXYZnrc. At the same time the statistical analysis routines were converted to a much improved, history by history approach[10] instead of the standard batch approach used in DOSXYZ.

In 2004, Iwan Kawrakow and Blake Walters ported DOSXYZnrc to the new EGSnrcMP system[3]. This eliminated the exclusive use of Linux/Unix scripts and allowed DOSXYZnrc to be compiled and run on Windows-based systems in addition to Linux/Unix platforms. At this time, DOSXYZnrc operates similarly to a standard EGSnrcMP user code, although it is only distributed as part of the OMEGA/BEAM system.

Although Charlie Ma is no longer an author of the DOSXYZnrc version of the code, his major contributions to the original EGS4 version still remain.

1.3   Compatibility of DOSXYZ and DOSXYZnrc

After renaming a DOSXYZ input file from `filename.egs4inp` to `filename.egsinp`, the file can be used directly by DOSXYZnrc. This is because DOSXYZnrc assumes particular default values for all of the additional EGSnrc input parameters needed. However, a better approach is to use the GUI for DOSXYZnrc (`dosxyznrc_gui`) to read in the DOSXYZ file and then output the DOSXYZnrc input file with all of the defaults explicitly stated. For further information, see section 9 on page 91.

The results of calculations with DOSXYZ and DOSXYZnrc will be very similar for most situations. The one systematic difference is that the relativistic spin corrections to the multiple scattering cause the depth-dose curves for electron beams to be about 1.5% more penetrating in water for a given electron energy[11].

2 Compiling/running DOSXYZnrc

2.1 Files Related to DOSXYZnrc and ctcreate

As an EGSnrcMP user code, the DOSXYZnrc files are mostly contained in the directory $HEN_HOUSE/user_codes/dosxyznrc/, while ctcreate files are in $OMEGA_HOME/progs/ctcreate/. For a general description of the file structure see Chapter 1 of the BEAMnrc User's Manual [12].

The following describes some files related to DOSXYZnrc:

dosxyznrc_gui This is the Tcl graphical user interface for creatingi, modifying or executing DOSXYZnrc input files. Files related to this are located in $OMEGA_HOME/progs/gui/dosxyznrc. See the GUI manual[4] for more details.

egsnrc_cshrc_additions (or egsnrc_bashrc_additions) Located in $HEN_HOUSE/scripts. If using a Linux/Unix system, this file must be sourced in the user's .cshrc (or .bashrc) file. It defines useful aliases for compiling/running DOSXYZnrc.

Makefile Located in $HEN_HOUSE/user_codes/dosxyznrc. This file is used by the GNU make utility to handle compilation of DOSXYZnrc. Includes $HEN_HOUSE/specs/config.conf (where config is the configuration you are using) and $HEN_HOUSE/specs/beamnrc.spec files to define environment variables and compiler options. Also, with the RANDOM variable, it defines the random number generator used (current default is ranmar). Finally, it defines the variable SOURCES which determines the macros and MORTRAN sources that are concatenated together to create mortjob.mortran (the code that is actually MORTRAN compiled). There are two versions of this file, called Makefile.MS and Makefile.NOMS, on the distribution. The default Makefile is Makefile.NOMS which does not use multiple source models (source 4, beam characterization models). Makefile.MS is to be used when Multiple Source models are to be used (read the file for instructions).

dosxyznrc.make Located in $HEN_HOUSE/user_codes/dosxyznrc. This is an empty file that must exist for compilation using the make utility.

dosxyznrc.mortran Located in $HEN_HOUSE/user_codes/dosxyznrc. Main MORTRAN source code.

srcxyznrc.mortran Located in $HEN_HOUSE/user_codes/dosxyznrc. Subroutines for source configuration inputs + energy spectrum

srcxyznrc.macros Located in $HEN_HOUSE/user_codes/dosxyznrc. MORTRAN macros required by srcxyznrc.mortran

read_write_pardose.c Located in $HEN_HOUSE/user_codes/dosxyznrc. C subroutines used in DOSXYZnrc to write and read binary .pardose output during parallel jobs. If you have a C or C++ compiler, then this is compiled when the OMEGA/BEAM system

is installed and `read_write_pardose.o` is put in directory `$HEN_HOUSE/lib/config`, where `config` is the name of your configuration. If you do not have a C or C++ compiler, then this file is not compiled, and the built-in parallel functionality of DOSXYZnrc cannot be used.

`dosxyznrc_config.spec` (where `config` is the name of your configuration) Located in `$HEN_HOUSE/specs`. This file is created during OMEGA/BEAM installation and determines whether or not the compiled C routines for reading/writing `.pardose` files, `read_write_pardose.o`, are linked in at compile time or not. If they are to be linked (i.e., you have a C or C++ compiler and `read_write_pardose.c` was compiled successfully), then the variable `PARDOSE_OBJECTS` in this file is set to `$(EGS_LIBDIR)read_write_pardose.o`, where `$(EGS_LIBDIR)=$HEN_HOUSE/lib/config`. If the routines are not to be linked at compile time (i.e., you do not have a C or C++ compiler or `read_write_pardose.c` was not compiled successfully), then `PARDOSE_OBJECTS` is left blank.

`dosyxznrc_user_macros.mortran` Located in `$HEN_HOUSE/user_codes/dosxyznrc`. MORTRAN macros that the user may change - includes defaults for various options such as beam models, *etc.* Note that `dosxyznrc_user_macros` is also used by ctcreate to define the maximum dimensions of the DOSXYZnrc phantom output.

`dosxyznrc.io` Located in `$HEN_HOUSE/user_codes/dosxyznrc`. This file assigns file names to Fortran unit numbers for output files not opened explicitly in `dosxyznrc.mortran`. Currently, the only files that use this are the `.egslst` file (Fortran unit 1) and the `.errors` file (Fortran unit 15).

`phsp_macros.mortran` MORTRAN macros used to read phase space sources. This file is always picked up from the `$HEN_HOUSE/utils` directory.

`iaea_phsp_macros.mortran` MORTRAN macros used to handle IAEA-format phase space sources. Located in the `$HEN_HOUSE/utils` directory, this file is only included if EGSnrc was installed on a machine with a working C++ compiler and the library of IAEA phase space handling routines (`$HEN_HOUSE/iaea_phsp/iaea_phsp.a`) was compiled successfully. Otherwise, these macros are defined as blank ({;}) in `phsp_macros.mortran` and IAEA functionality does not exist.

`beammodel_macros.mortran` MORTRAN macros required by the multiple-source model for beam reconstruction (source 4), stored in `$OMEGA_HOME/progs/beamdp`.

`beammodel_routines.mortran` MORTRAN subroutines required by the multiple-source model for beam reconstruction (source 4), stored in `$OMEGA_HOME/progs/beamdp`.

`DOSXYZnrc_examples/` A subdirectory of `$HEN_HOUSE/user_codes/dosxyznrc`. This directory contains sample input files for DOSXYZnrc.

The following is a description of some of the files related to `ctcreate`:

Makefile Located in $OMEGA_HOME/progs/ctcreate. Used by the GNU make utility, this file directs the compilation of ctcreate. The SOURCES variable defines the macros and MORTRAN sources concatenated to create mortjob.mortran (which is ultimately MORTRAN compiled)

ctcreate.mortran Located in $OMEGA_HOME/progs/ctcreate. This is the main MORTRAN source code for ctcreate.

ReadCT_DICOM.c Located in $OMEGA_HOME/progs/ctcreate. This is a C subroutine for reading CT images in DICOM format. It is linked to ctcreate.mortran at compile time.

tags_ct.h Located in $OMEGA_HOME/progs/ctcreate. This is a C header file used with ReadCT_DICOM.c. It defines the hexadecimal data identifiers used in DICOM image format.

lnblnk1_function.mortran MORTRAN macro to provide the FORTRAN lnblnk function for all configurations. This is picked up from $HEN_HOUSE/src.

2.2   Compiling and Running DOSXYZnrc

The commands for compiling and running DOSXYZnrc are similar to those for other EGSnr-cMP user codes (see the EGSnrcMP Users Manual[3]). For EGS4 users, please note that file extensions have changed from, *eg,* file.egs4inp to file.egsinp.

DOSXYZnrc is normally compiled on your user area as part of the OMEGA/BEAM user set up (see the BEAMnrc Manual [12] for configuration instructions). To compile DOSXYZnrc independently (e.g., necessary if you have changed some parameters in dosxyznrc_user_macros.mortran), ensure that Makefile, dosxyznrc.make, dosxyznrc.mortran, dosxyznrc_user_macros.mortran, srcxyznrc.mortran, srcxyznrc.macros, and dosxyznrc.io exist in your $EGS_HOME/dosxyznrc directory (they should have been copied there automatically from $HEN_HOUSE/user_codes/dosxyznrc during OMEGA/BEAM configuration). Then, from $EGS_HOME/dosxyznrc, compile DOSXYZnrc by typing:

make [options]

The options for make are:

```
make             Compile with default optimization
make opt         turned on.  Default optimization is level 2 (-O2).

make noopt       Compile with no optimization

make debug       Compile executable for debugging.

make fortran     Do mortran compilation only, leaving behind the Fortran
                 file dosxyznrc.F.
```

```
make clean        Remove the Fortran file, mortjob.mortran file,
                  dosxyznrc.mortlst file and the executable.
```

To preserve compatibility with old usage, the `mf` command is also available for compiling DOSXYZnrc on a Linux/Unix system (you must have sourced `$HEN_HOUSE/scripts/egsnrc_cshrc_additions` or `$HEN_HOUSE/scripts/egsnrc_bashrc_additions` from your `.cshrc` or `.bashrc` file). `mf` is aliased to the script `$HEN_HOUSE/scripts/compile_user_code`.

To use `mf`, go into `$EGS_HOME/dosxyznrc` and type:

```
m[f] dosxyznrc [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 (default level 2)
`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 you have successfully compiled DOSXYZnrc, the executable, `dosxyznrc*`, will be left in your `$EGS_HOME/bin/config` directory (where `config` is the name of the configuration that you are using).

To run DOSXYZnrc interactively from the command line, go into `$EGS_HOME/dosxyznrc` and type:

```
dosxyznrc -i inputfile -p pegsdata
```

where the input file is `$EGS_HOME/dosxyznrc/inputfile.egsinp` and the file `pegsdata.pegs4dat` contains the PEGS4 data set (it can be on `$EGS_HOME/pegs4/data` or if not found there, on `$HEN_HOUSE/pegs4/data`).

If you are using a Unix/Linux system, then you can also start an interactive DOSXYZnrc run using the `ex` (aliased to `$HEN_HOUSE/scripts/run_user_code`) command:

```
ex dosxyznrc inputfile pegsdata
```

`ex` is provided to preserve compatibility with old usage.

If you are using a Linux/Unix system then you can also run DOSXYZnrc in batch mode. Batch submission is required for parallel jobs. Batch submission uses the `exb` command, which is aliased to the script `$HEN_HOUSE/scripts/run_user_code_batch`. The syntax of the `exb` command is:

```
exb dosxyznrc inputfile pegsdata [short|medium|long] [batch=batch_sys] [p=N]
```

The `[short|medium|long]` option defines the name of the queue that is used (default is `long` as at NRC). The `batch_sys` input defines the network queuing system to use. Currently, `batch_sys` can be set to `at` (the standard Unix batch command), `pbs` (to use PBS), `keg` (to use Sun's SGE) or `nqs` (for NQS). The default is `at` unless otherwise specified by setting the environment variable `$EGS_BATCH_SYSTEM`. Finally, `N` is used if you are submitting parallel

jobs and is set equal to the number of jobs that you want to split the simulation into.

Once a run is started, a temporary working directory is created as a subdirectory of $EGS_HOME/dosxyznrc. This temporary working directory has the name egsrun_pid_inputfile_hostname, where where pid is the process ID number and hostname is the name of the computer the job is running on. All output files are written to this temporary directory. At the end of the run, the files are moved into $EGS_HOME/dosxyznrc and the temporary working directory is deleted. For more information on temporary working directories, see the EGSnrcMP Users Manual[3].

DOSXYZnrc outputs the following files: inputfile.egslst, inputfile.egslog (for batch runs only, where it contains screen output), inputfile.egsdat (which can be used to restart the calculation) and inputfile.3ddose which contains a summary of the data in all regions and can be used by STATDOSE to create xmgr/xmgrace graphs (see "STATDOSE Users Manual"[13]). If this is a parallel run, then the individual jobs will output binary .pardose files instead of .3ddose files. The .pardose are then combined automatically at the end of the parallel run to create a .3ddose file. See section 10 for more information on parallel runs. Note that the .egslst file can become VERY long and thereby become useless so use it carefully for getting the dose which usually can be more effectively obtained via the .3ddose output file.

The DOSXYZnrc code can also be compiled and run from the dosxyznrc_gui[4]. To compile DOSXYZnrc, select "Compile" from the "Run" menu. This will open up a window which gives you the different make options (ie optimization vs no optimization, debug, etc). To run the code from the GUI, you must first load an existing input file or create a new one (new inputs or changes to an input file must first be saved before running). Then select "Run" from the "Run" menu. This will open up a window in which you can either run DOSXYZnrc interactively or else submit to a queue (or start a parallel run). Batch runs will use the PBS queueing system unless otherwise specified in the $EGS_BATCH_SYSTEM environment variable. Dialog that would normally appear on screen during an interactive run now appears in the GUI run window.

For more information about compiling and running user codes, see the EGSnrcMP Users Manual[3].

2.2.1   Including source 4/beam characterization

To implement beam characterization models in DOSXYZnrc, copy beammodel_macros.mortran and beammodel_routines.mortran, to the user's dosxyznrc area from $OMEGA_HOME/progs/beamdp. Also copy Makefile.MS from $HEN_HOUSE/user_codes/dosxyznrc/ to the user's dosxyznrc area and rename it Makefile. Then recompile.

2.3 Statistical Analysis

The statistical analysis in the original DOSXYZ code was done using a standard batching technique. Starting with DOSXYZnrc the statistics on the doses are determined by grouping scored quantities (i.e., energy deposited) on a history-by-history basis and then determining the uncertainties. For most sources, this simply means grouping quantities by incident particle. However, for phase space sources, where more than one incident particle may be traced back to a single primary history, quantities are grouped by primary history. For more information, see the published paper on history by history statistics in DOSXYZnrc and BEAMnrc[10].

It is worth noting that the method used takes into account the latent variance in any phase space file being used as a source (i.e. the uncertainty introduced by the statistical variations in the phase space file). Hence, one cannot reduce the uncertainty in any dose calculation below that level by recycling the data a large number of times. However, one can get an artificially low statistical result which ignores this latent variance if the phase space source is allowed to restart the phase space file instead of using the recycle option (whereby each particle is used multiple times as it is read in - see section 8.11, page 83). Thus, in order to get accurate uncertainty estimates, restarting the phase space file should be avoided.

3 DOSXYZnrc Input Parameters

3.1 Descriptions in DOSXYZnrc Source Code

This section describes input parameters for DOSXYZnrc. The following descriptions can be found in the beginning of the `dosxyznrc.mortran` source code. The graphical user interface facilitates creation of these input files and contains a great deal of on-line help[4].

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

            *********************************************
            *                                           *
            * dosxyznrc.mortran                         *
            *                                           *
            *********************************************

  A general purpose EGSnrc user code to do cartesian coordinate dose
  deposition studies. Every voxel (volume element) can have different
  materials and/or varying densities (for use with CT data).

  The geometry is a rectilinear volume with a right-handed coordinate system:
  the X-Y plane on the page, X to the right, Y down and the
  Z-axis into the page. Voxel dimensions are completely variable in
  all three directions. For more detail on geometry see subroutine HOWFAR

 Unit Assignments
```

```
    ================

    Unit 1    Output summary and results
    Unit 2    .egsrns file for storing random numbers
    Unit 3    output .3ddose file containing dose arrays
    Unit 4    Raw data output file for restarts
    Unit 5    Input stream - file or terminal
    Unit 6    prompts for and echoes input
    Unit 8    echoes input cross-section data (usually assigned to a null file)
    Unit 17   geometry output file for EGS_Windows
    Unit 13   phase space output file for EGS_Windows
    Unit 12   Input cross section file from PEGS4
    Unit 15   Output of EGSnrc input error messages
    Unit 16   Output for .pardose when IPARALLEL > 1
    Unit 44   Full phase-space data set, such as output from BEAM
    $CTUnitNumber (normally 45)  CT data set input

  DESCRIPTION OF INPUT FILE (on unit 5)
   ========================

  Record 1    TITLE      Up to 80 characters


  Record 2    NMED       Number of media in problem - defaults to 1
              If (NMED=0) then create dosxyznrc phantom from binary CT data


-------------------------------------------------------------------------------
----------if NMED > 0  => non-CT data input-------------------------------------
-------------------------------------------------------------------------------


  Record 3(NMED times) Media names, left justified. Note that
                       entire volume is initially set to medium 1

  Record 4   ECUTIN,PCUTIN,ESTEPM(i,i=1,NMED),SMAX
                   ECUTIN,PCUTIN: Electron (total) and photon global cutoff
                         energies in MeV.  If ECUTIN > ECUT input in EGSnrc
                         parameters (See below), then ECUTIN is used.  The
                         same is true if PCUTIN > PCUT input in EGSnrc parameters.
                         ECUT and PCUT default to AE and AP, respectively.
                   ESTEPM(i,i=1,NMED): Dummy input (used to be estepe, max. energy
                         loss/electron step in each medium)
                   SMAX: Dummy input (used to be max. step length)

                         ESTEPE and SMAX are now handled in EGSnrc inputs (See below),
                         but the dummy inputs are retained for compatibility with
                         EGS4/DOSXYZ input files.

  Record 5    IMAX,JMAX,KMAX,IPHANT (4i10)
                   IMAX,JMAX,KMAX: Number of voxels in the X,Y,Z directions
                         If < 0, it means that (-n) sets of equally spaced
```

```
                      boundaries will be input for that direction.
                 IPHANT: set to 1 to output a .egsphant file for displaying
                      non-CT isodose contours using dosxyz_show


 Record 6 et seq,        repeated for x, y and z directions separately
                      i.e. repeat the following replacing (i and x) by
                      (j and y) and (k and z) respectively
   if IMAX > 0
               input, one per record, the IMAX+1 x boundaries


   if IMAX < 0
               input smallest x boundary, followed by  abs(IMAX) pairs
               one pair/record:
               voxel width, # voxels with this width


   For example, starting at record 5:
               -1,-1,-1
               0.0
               1.0,16
               0.0
               1.0,16
               0.0
               1.0,16
       defines a 16x16x16 cm cube of 1cm**3 voxels with a total of 4097 regions
   Or:
               -1,-1,2
               0.0
               1.0,16
               0.0
               1.0,16
               0.0
               5.0
               10.0
       defines a 16x16x10 cm cube with 1x1x5 cm voxels stacked 2 deep


 Record 7 et seq
               IL,IU, JL,JU, KL,KU, MEDIUM, DENSITY (7i10,f10.0)
               This record is repeated until a blank record is found.
               All regions default to medium 1 with its default density
               unless changed here.
               For all voxels with
                       IL <= i <= IU
                       JL <= j <= JU
                       KL <= k <= KU
               the medium and density are defined.
               If DENSITY=0.0, the default value for that
               MEDIUM is used (faster than entering default density here).
               If IU and IL are non-zero, the rest default to all j,k
```

```
   Record 8 et seq
              IL,IU, JL,JU, KL,KU, ECUTL, PCUTL (6i10,2f10.0)
              This record is repeated until a blank record is found.
              As above but allowing a region by region local definition
              of ECUT and PCUT.
              All regions default to the global values defined in
              record 4 unless changed here.
              Note, this option disabled, but must enter blank record
              for compatibility. Note old code is in place.


   Record 9 et seq
              IL,IU, JL,JU, KL,KU,IZSCAN,MAX20
              This record is repeated until a blank record is found.
              As above except these are the regions for which the
              dose will be printed in .egslst - beware of a paper explosion!
              The default is to print nothing unless asked for here.
              IZSCAN is non-zero to get a z-scan per page, otherwise
              output is an x-scan per page.  If the input parameter
              MAX20 is equal to 1 on any of these input lines, a
              summary of the 20 highest doses is included in the output.



--------------------------------------------------------------------------------
---------------if NMED = 0   input for a CT phantom --------------------------
--------------------------------------------------------------------------------


   Record 3   PhantFileName (A256)   : The full name of the file containing
                                        the CT phantom as output by ctcreate
                                        (should be a .egsphant file).


   Record 4   ECUTIN,PCUTIN,SMAX (3F15.0)
              ECUTIN,PCUTIN: Electron (total) and photon global cutoff
                      energies in MeV.  If ECUTIN > ECUT input in EGSnrc
                      parameters (See below), then ECUTIN is used as the
                      global ECUT.  The same is true if PCUTIN > PCUT input
                      in EGSnrc parameters.  ECUT and PCUT default to AE and
                      AP, respectively.
              SMAX: Dummy input (used to be max. step length)

              SMAX is now handled in EGSnrc inputs (See below),
              but the dummy input is retained for compatibility with
              EGS4/DOSXYZ input files.


   Record 5   zeroairdose,doseprint,MAX20 (3I5)

              zeroairdose: = 1 to zero the dose in air (any
                               material with density < 0.044 g/cm^3)
                               in the .3ddose file
                           = 0(default) to not zero the dose in air
```

```
                    doseprint: = 1 for output of all doses to
                                 .egslst file
                               = 0(default) to suppress this output
                        MAX20: = 1 to print out summary of 20 highest
                                 doses
                               = 0(default) no summary of these doses


-------------------------------------------------------------------------------
--------------From here on the two formats are identical.------------------
-------------------------------------------------------------------------------


   Records 10--12 (6--8 if NMED=0) Refer to srcxyznrc.mortran
*******************************************************************************


             Input source parameters read by srcxyznrc.mortran
             =================================================
             (see section 4 of DOSXYZnrc User's Manual for details)


*******************************************************************************


Record SC1-0  Parallel beam incident on the front with rectangular collimation
                                     (front is the zbound(1) plane)


        iqin,isource,xinl,xinu,yinl,yinu,thetax,thetay,thetaz


            iqin       Charge of the incident beam (defaults to 0)
            isource    = 0
            xinl,xinu  Lower and upper x-bounds on source (in cm)
            yinl,yinu  Lower and upper y-bounds on source (in cm)
            thetax     Angle of the beam relative to the X-axis (degrees)
            thetay     Angle of the beam relative to the Y-axis (degrees)
            thetaz     Angle of the beam relative to the Z-axis (degrees)
                       (incident angles default to 90,90,0)


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


Record SC1-1  Parallel beam incident from any direction with rectangular
           collimation


        iqin,isource,xiso,yiso,ziso,theta,phi,xcol,ycol,phicol


            iqin       Charge of the incident beam (defaults to 0)
            isource    = 1
            x|y|z|iso  x|y|z|-coordinates of the isocenter
            theta      angle between the +z direction and a line joining
                       the center of the beam to the isocenter
            phi        angle  between the +x direction and the
                       projection of the line joining the center of the
                       beam to the isocenter on the xy plane
```

```
                  x|y|col      total x- and y- widths of the beam on the plane
                               perpendicular to the beam direction, defined
                               by the center of the beam and the isocenter
                  phicol       angle by which the collimator is rotated in the
                               collimator plane perpendicular to the beam direction
                               Phicol is determined for theta=0 or 180 and phi=0.
                               The positive sense of rotation is counterclockwise
                               as one sights down the beam direction.
```

Record SC1-2   Full phase-space source file, particles incident on front face

```
       iqin,isource,xiso,yiso,ziso,theta,phi,dsource,phicol,i_dbs,r_dbs,
                                 ssd_dbs,z_dbs,e_split
```

```
                  iqin(iqphsp)= 0 only photons from ph-sp file will be used
                               = 1 only positrons will be used
                               =-1 only electrons will be used
                               = 2 all the particles will be used
                               internally renamed as iqphsp
                  isource      =  2
                  x|y|z|iso    x|y|z|-coordinates of the isocenter
                  theta        angle between the +z direction and a line joining
                               the origin in the phase space plane to the
                               isocenter.  theta=180 degrees for a beam down
                               from the top.
                  phi          angle  between the +x direction and the
                               projection of the line joining the origin in the
                               phase space plane to the isocenter on the xy plane
                  dsource      For a BEAMnrc format phase space source: absolute
                               distance from the isocenter to the
                               source center, which is, by definition, the
                               origin in the phase space plane.  For an IAEA
                               format phase space source: the primary source to
                               isocentre distance (SAD).
                  phicol       angle by which the collimator is rotated in the
                               collimator plane perpendicular to beam direction.
                               Phicol is determined for theta=0 or 180 and phi=0.
                               The positive sense of rotation is counterclockwise
                               as one sights down from the origin in the
                               phase space plane.
                               phicol = 180 to retain same x,y positions passing
                               from BEAM to DOSXYZ  (because of coordinate
                               transformation used).
                  i_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 fat photons not aimed into the splitting
```

```
                             field.  These fat photons may compromise dose
                             statistics.  Set to 0 otherwise.
            r_dbs            DBS splitting radius used in BEAMnrc simulation(cm).
                             Set to 0 to disable this option.  Only needed if
                             i_dbs=1.
          ssd_dbs            SSD at which r_dbs is defined in the BEAM sim. (cm).
                             Only needed if i_dbs=1.
            z_dbs            Z in the BEAMnrc run where the phase space source
                             was scored (in cm).  Only needed if i_dbs=1.
          e_split            No. of times to split charged particles as soon
                             as they enter the DOSXYZnrc geometry.  Used in
                             conjunction with photon splitting (n_split) to
                             prevent higher-weight charged particles from
                             compromising dose statistics.  Recommended setting
                             is e_split=n_split for optimum efficiency.  Not
                             used if n_split<=1.


     For i_dbs=1, photons are projected from z_dbs to ssd_dbs and if they will
     fall outside of r_dbs (based on their Z direction cosine) then they will
     be rejected.  This prevents fat photons from compromising dose
     statistics.


   Note: as currently set, multiple passers and all particles going
         backwards in the phase space file are NOT used.


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


Record SC1-3  Point source incident from the front (along the +ve z-axis)
           employing rectangular collimation

     iqin,isource,xinl,xinu,yinl,yinu,ssd


            iqin       Charge of the incident beam (defaults to 0)
            isource    = 3
            xinl,xinu  Lower and upper x-bounds on source
            yinl,yinu  Lower and upper y-bounds on source
            ssd        Source-surface distance (ssd >0)


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


Record SC1-4  Beam characterization model, particles incident on front face

     iqin,isource,xiso,yiso,ziso,thetax,thetay,thetaz,phicol


            iqin       = 0 only photons from ph-sp file will be simulated
                       = 1 only positrons will be simulated
                       =-1 only electrons will be simulated
                       = 2 all the particles will be simulated
            isource    =  4
```

```
            x|y|z|iso    x|y|z|-coordinates of the isocenter
            theta        angle between the +z direction and a line joining
                         the origin in the phase space plane to the isocenter
            phi          angle  between the +x direction and the
                         projection of the line joining the origin in the
                         phase space plane to the isocenter on the xy plane
            dsource      absolute distance from the isocenter to the
                         source center, which is, by definition, the
                         origin in the phase space plane
            phicol       angle by which the collimator is rotated in the
                         collimator plane perpendicular to the beam direction
                         Phicol is determined for theta=0 or 180 and phi=0.
                         The positive sense of rotation is counterclockwise
                         as one sights down from the origin in the
                         phase space plane.
```

Record SC1-6 Uniform isotropically radiating parallelepiped within DOSXYZ
              phantom

```
    iqin,isource,xinl,xinu,yinl,yinu,zinl,zinu
            iqin         charge of particles from source (defaults to 0)
            isource      = 6
            xinl         min x of active volume (cm)
            xinu         max x of active volume (cm)
            yinl         min y of active volume (cm)
            yinu         max y of active volume (cm)
            zinl         min z of active volume (cm)
            zinu         max z of active volume (cm)
```

NOTE: The active volume must be completely contained within the phantom.

Record SC1-7 Parallel beam incident from multiple, user-selected angles.

```
      iqin,isource,xiso,yiso,ziso,nang,xcol,ycol,phicol

            iqin         Charge of the incident beam (defaults to 0)
            isource      = 7
            x|y|z|iso    x|y|z|-coordinates of the isocenter
            nang         number of incident theta-phi pairs or, if -ve,
                         absolute value is the number of groups of theta-
                         phi pairs.  Within a group:
                             1. only theta or phi, but not both, can vary
                             2. varying theta's or phi's are evenly
                                distributed
                             3. theta-phi pairs have equal probability
            x|y|col      total x- and y- widths of the beam on the plane
```

```
                              perpendicular to the beam direction, defined
                              by the center of the beam and the isocenter
                   phicol     angle by which the collimator is rotated in the
                              collimator plane perpendicular to the beam direction
                              Phicol is determined for theta=0 or 180 and phi=0.
                              The positive sense of rotation is counterclockwise
                              as one sights down the beam direction.


     Note that, similar to source 1, theta is defined as the angle between the +z
     direction and a line joining the center of the beam to the isocenter, and
     phi is defined as the angle between the +x direction and the projection of
     the line joining the center of the beam to the isocenter on the xy plane.


   ----------------------------------------------------------------------------
   Record SC1-8  Full phase-space source, particles incident from multiple,
                 user-selected angles.

       iqin,isource,xiso,yiso,ziso,nang,dsource,phicol,i_dbs,r_dbs,ssd_dbs,
                z_dbs,e_split

                      iqin(iqphsp)= 0 only photons from ph-sp file will be used
                                  = 1 only positrons will be used
                                  =-1 only electrons will be used
                                  = 2 all the particles will be used
                                  internally renamed as iqphsp
                      isource    =  8
                      x|y|z|iso   x|y|z|-coordinates of the isocenter
                      nang        number of incident theta-phi pairs or, if -ve,
                                  absolute value is the number of groups of theta-
                                  phi pairs.  Within a group:
                                     1. only theta or phi, but not both, can vary
                                     2. varying theta's or phi's are evenly
                                        distributed
                                     3. theta-phi pairs have equal probability
                      dsource     For an BEAMnrc format phase space source: the
                                  absolute distance from the isocenter to the
                                  source center, which is, by definition, the
                                  origin in the phase space plane.  For an IAEA
                                  format source: the primary source-to-isocentre
                                  distance (SAD).
                      phicol      angle by which the collimator is rotated in the
                                  collimator plane perpendicular to beam direction.
                                  Phicol is determined for theta=0 or 180 and phi=0.
                                  The positive sense of rotation is counterclockwise
                                  as one sights down from the origin in the
                                  phase space plane.
                                  phicol = 180 to retain same x,y positions passing
                                  from BEAM to DOSXYZ  (because of coordinate
                                  transformation used).
```

                    i_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 fat photons not aimed into the splitting
                                   field. These fat photons may compromise dose
                                   statistics. Set to 0 otherwise.
                    r_dbs   DBS splitting radius used in BEAMnrc simulation(cm).
                                   Set to 0 to disable this option. Only needed if
                                   i_dbs=1.
                  ssd_dbs   SSD at which r_dbs is defined in the BEAM sim. (cm).
                                   Only needed if i_dbs=1.
                    z_dbs   Z in the BEAMnrc run where the phase space source
                                   was scored (in cm). Only needed if i_dbs=1.
                  e_split   No. of times to split charged particles as soon
                                   as they enter the DOSXYZnrc geometry. Used in
                                   conjunction with photon splitting (n_split) to
                                   prevent higher-weight charged particles from
                                   compromising dose statistics. Recommended setting
                                   is e_split=n_split for optimum efficiency. Not
                                   used if n_split<=1.

   For i_dbs=1, photons are projected from z_dbs to ssd_dbs and if they will
   fall outside of r_dbs (based on their Z direction cosine) then they will
   be rejected. This prevents fat photons from compromising dose
   statistics.

   Note: as currently set, multiple passers and all particles going
         backwards in the phase space file are NOT used.
   Also note that, similar to source 2, theta is the angle between the +z
   direction and a line joining the origin of the phase space plane to the
   isocenter, and phi is the angle between the +x direction and the
   projection of the line joining the origin of the phase space plane to
   the isocenter in the xy plane.

Record SC1-9 BEAM treatment head simulation used as source, incident from
          any angle

  This source requires unix/Linux systems to have a working C/C++ compiler
  and requires the BEAM code, BEAM_accelname, to have been compiled as a
  shared library (libBEAM_accelname.so or BEAM_accelname.dll) that exists in
  your EGS_HOME/bin/config directory. See the DOSXYZnrc Manual for more
  details.

        iqin,isource,xiso,yiso,ziso,theta,phi,dsource,phicol,i_dbs,e_split

             iqin(iqinc)= 0 only photons from BEAM simulation will be used
                        = 1 only positrons will be used

```
                            =-1 only electrons will be used
                            = 2 all the particles will be used
                            internally renamed as iqinc
                isource     = 9
                x|y|z|iso   x|y|z|-coordinates of the isocenter.
                theta       angle between the +z direction and a vector
                            joining the isocentre to the origin of the source
                            plane (the plane where the particles are sampled
                            from the BEAM simulation--which is the
                            scoring plane in a standard BEAM simulation).
                            theta=180 degrees for a beam down from the top.
                phi         angle between the +x direction and the
                            projection of the vector joining the isocenter to
                            the origin of the source plane.
                dsource     absolute distance from the isocenter to the
                            origin of the source plane.
                phicol      angle by which the source plane is rotated about
                            the BEAM central axis.  Rotation is counter-
                            clockwise for theta=0.  Set phicol=180 degrees
                            to retain same x,y, positions passing
                            from BEAM to DOSXYZ when theta=180 (beam coming
                            down from top).
                i_dbs       set to 1 if you are using directional bremsstrahlung
                            splitting (DBS) in the BEAM simulation
                            AND you wish to reject fat photons not aimed into
                            the splitting field.  These fat photons may
                            compromise dose statistics.  Set to 0 otherwise.
                e_split     No. of times to split charged particles as soon
                            as they enter the DOSXYZnrc geometry.  Used in
                            conjunction with photon splitting (n_split) to
                            prevent higher-weight charged particles from
                            compromising dose statistics.  Recommended setting
                            is e_split=n_split for optimum efficiency.  Not
                            used if n_split<=1.
--------------------------------------------------------------------------------


Record SC1-10 BEAM treatment head simulation used as source, incident from
              multiple, user-defined angles.


  This source requires unix/Linux systems to have a working C/C++ compiler
  and requires the BEAM code, BEAM_accelname, to have been compiled as a
  shared library (libBEAM_accelname.so or BEAM_accelname.dll) that exists in
  your EGS_HOME/bin/config directory.  See the DOSXYZnrc Manual for more
  details.


        iqin,isource,xiso,yiso,ziso,nang,dsource,phicol,i_dbs,e_split


              iqin(iqinc)= 0 only photons from BEAM simulation will be used
                         = 1 only positrons will be used
```

```
                              =-1 only electrons will be used
                              = 2 all the particles will be used
                              internally renamed as iqinc
                 isource      = 10
                 x|y|z|iso    x|y|z|-coordinates of the isocenter.
                 nang         number of incident theta-phi pairs or, if -ve,
                              absolute value is the number of groups of theta-
                              phi pairs.  Within a group:
                                  1. only theta or phi, but not both, can vary
                                  2. varying theta's or phi's are evenly
                                     distributed
                                  3. theta-phi pairs have equal probability
                 dsource      absolute distance from the isocenter to the
                              origin of the source plane.
                 phicol       angle by which the source plane is rotated about
                              the BEAM central axis.  Rotation is counter-
                              clockwise for theta=0.  Set phicol=180 degrees
                              to retain same x,y, positions passing
                              from BEAM to DOSXYZ when theta=180 (beam coming
                              down from top).
                 i_dbs        set to 1 if you are using directional bremsstrahlung
                              splitting (DBS) in the BEAM simulation
                              AND you wish to reject fat photons not aimed into
                              the splitting field.  These fat photons may
                              compromise dose statistics.  Set to 0 otherwise.
                 e_split      No. of times to split charged particles as soon
                              as they enter the DOSXYZnrc geometry.  Used in
                              conjunction with photon splitting (n_split) to
                              prevent higher-weight charged particles from
                              compromising dose statistics.  Recommended setting
                              is e_split=n_split for optimum efficiency.  Not
                              used if n_split<=1.

        Note that this source requires Record SC1-10a, described below.


 --------------------------------------------------------------------------------
 Record SC1-20 Synchronized phase space source simulating continuous motion of
               the source plane between user-defined control points.  Option
               to run the source through a geometry (usually MLC) defined using
               BEAM or a non-EGSnrc code compiled as a shared library
          - added by Tony Popescu and Julio Lobo


   This source requires unix/Linux systems to have a working C/C++ compiler
   and requires the BEAM code or non-EGSnrc code (e.g particleDmlc) to have
   been compiled as a shared library (libparticleDmlc.so) that exists in
   your EGS_HOME/bin/config directory.



         iqin,isource,nset,i_dbs,r_dbs,ssd_dbs,z_dbs,e_split,i_muidx_out, calflag
```

```
              iqin(iqinc) = 0 only photons from BEAM simulation will be used
                          = 1 only positrons will be used
                          =-1 only electrons will be used
                          = 2 all the particles will be used
                          internally renamed as iqinc
              isource      = 20
              nset         number of settings (control points) (>=2)
              i_dbs        set to 1 if you are using directional bremsstrahlung
                           splitting (DBS) in the BEAM simulation
                           AND you wish to reject fat photons not aimed into
                           the splitting field.  These fat photons may
                           compromise dose statistics.  Set to 0 otherwise.
              r_dbs        DBS splitting radius used in BEAMnrc simulation(cm).
                           Set to 0 to disable this option.  Only needed if
                           i_dbs=1.
            ssd_dbs        SSD at which r_dbs is defined in the BEAM sim. (cm).
                           Only needed if i_dbs=1.
              z_dbs        Z in the BEAMnrc run where the phase space source
                           was scored (in cm).  Only needed if i_dbs=1.
            e_split        No. of times to split charged particles as soon
                           as they enter the DOSXYZnrc geometry.  Used in
                           conjunction with photon splitting (n_split) to
                           prevent higher-weight charged particles from
                           compromising dose statistics.  Recommended setting
                           is e_split=n_split for optimum efficiency.  Not
                           used if n_split<=1.
         i_muidx_out       Set to 1 to include frMU_indx (the fractional
                           monitor unit index) in phase space output if
                           i_phsp_out=1 or 2.
            calflag        set to 1 to skip the calibration run performed to
                           refine the estimate of NRCYCL needed to avoid
                           rewinding the phsp. Only do this if certain that
                           your phsp has enough particles compared to the no.
                           of histories you are running.


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


Record SC1-21 Synchronized BEAM treatment head simulation used as source
            with continuous motion of the source plane between user-defined
            control points.  Option to run the source through a geometry
            (usually MLC) defined using a non-EGSnrc code compiled as a
            shared library.
        - added by Tony Popescu and Julio Lobo


   This source requires unix/Linux systems to have a working C/C++ compiler
   and requires the BEAM code being used as the source, BEAM_accelname, to have
   been compiled as a shared library (libBEAM_accelname.so or
   BEAM_accelname.dll) that exists in your $EGS_HOME/bin/config directory.  See
```

the DOSXYZnrc Manual for more details.

```
iqin,isource,nset,i_dbs,e_split,i_muidx_out

        iqin(iqinc) = 0 only photons from BEAM simulation will be used
                    = 1 only positrons will be used
                    =-1 only electrons will be used
                    = 2 all the particles will be used
                    internally renamed as iqinc
        isource     = 21
        nset        number of settings (control points) (>=2)
        i_dbs       set to 1 if you are using directional bremsstrahlung
                    splitting (DBS) in the BEAM simulation
                    AND you wish to reject fat photons not aimed into
                    the splitting field.  These fat photons may
                    compromise dose statistics.  Set to 0 otherwise.
        e_split     No. of times to split charged particles as soon
                    as they enter the DOSXYZnrc geometry.  Used in
                    conjunction with photon splitting (n_split) to
                    prevent higher-weight charged particles from
                    compromising dose statistics.  Recommended setting
                    is e_split=n_split for optimum efficiency.  Not
                    used if n_split<=1.
        i_muidx_out Set to 1 to include frMU_indx (the fractional
                    monitor unit index) in phase space output if
                    i_phsp_out=1 or 2.

-------------------------------------------------------------------------------
Record SC1-7a, SC1-8a and SC1-10a (required for sources 7, 8 and 10 only)
        if nang>0:
          (theta(i),phi(i),pang(i),i=1,nang)

                theta(i)    incident theta i
                phi(i)      incident phi i.
                pang(i)     probability of a particle being incident at
                            theta(i)-phi(i) (probabilities are automatically
                            normalized to 1).

        if nang<0:
          (ivary(i),angfixed(i),angmin(i),angmax(i),ngang(i),pgang(i),i=1,-nang)

                ivary(i)    =0 to vary phi in group i, 1 to vary theta in
                            group i
                angfixed(i) =fixed theta (ivary=0) or phi (ivary=1) for group i
                angmin(i)   min. value of varying phi (ivary=0) or theta
                            (ivary=1) in group i
                angmax(i)   max. value of varying phi or theta in group i
                ngang(i)    number of equally-spaced phi's or theta's
                            including angmin(i) and angmax(i) (Note: this
```

```
                            means ngang(i) must be at least 2).
                 pgang(i)    probability of a particle being incident in
                            group i (probabilities are automatically normalized
                            to 1).


-------------------------------------------------------------------------------
Record SC1-20a and SC1-21a (required for sources 20 and 21)

  (|x|y|ziso(i),theta(i),phi(i),phicol(i),dsource(i),muIndex(i),i=1,nset)

          |x|y|ziso(i)   x,y,z of isocentre for control point i
              theta(i)   incident theta for control point i
                phi(i)   incident phi for control point i
             phicol(i)   incident phicol for control point i
            dsource(i)   distance from isocentre to origin of BEAM
                         scoring plane for control point i.  For source
                         21 using an IAEA phase space file, this is
                         the primary source to isocentre distance (SAD).
            muIndex(i)   fractional monitor unit index for control point
                         i.  Restrictions: muIndex(i) is on [0,1],
                         muIndex(i-1)<=muIndex(i), muIndex(1)=0.0,
                         muIndex(nset)=1.0

  Note: theta, phi, phicol, and dsource have the same meaning as for
  Source 2 (see above).


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


 Record SC2
      enflag,mode,medsur,dsurround(1),dflag,dsurround(2),dsurround(3),
                                                        dsurround(4)


          enflag    = 0 for monoenergetic beams
                    = 1 for beams with an energy spectrum
                    = 2 for ph-sp beam input or full BEAM sim.
                    = 3 for ph-sp beam input or full BEAM sim. + dose
                         components calculations (use of LATCH filters)
                    = 4 for ph-sp beam modelled with multiple-source
                         (use of LATCH filters)
          mode      = 0 default file format for ph-sp data (enflag=2)
                         (7 variables/record)
                    = 2 special file format for ph-sp data with ZLAST
                         (8 variables/record)
          medsur    :    medium number for the region outside the phantom
                         (default = vacuum if medsur = 0) for enflag > 1
          dsurround(1): if dflag=0, the thickness of the region surrounding
                         the phantom on all sides (default=50cm)
                         OR
                         if dflag=1, the thickness of the region surrounding
```

```
                                the phantom in the x direction (default=0cm)
              dflag     = 0 dsurround(1) applies to all sides of phantom
                        = 1 dsurround(1) applies to x direction; must input
                            dsurround(2) for y direction, dsurround(3) for +z
                            direction (down), dsurround(4) for -z direction (up)
          dsurround(2):    (only required if dflag=1) thickness of region
                            surrounding phantom in y direction (default=0cm)
          dsurround(3):    (only required if dflag=1) thickness of region
                            surrounding phantom in +z direction (default=0cm)
          dsurround(4):    (only required if dflag=1) thickness of region
                            surrounding phantom in -z direction (default=0cm)


      Note:  Currently, medsur, dflag and dsurround(1...4) are only input
             if enflag > 1.


----------------------------------------------------------------------------
 Record SC3a   (if enflag = 0)
            ein
            ein        kinetic energy of the incident particles in MeV


----------------------------------------------------------------------------
 Record SC3b   (if enflag = 1, 2, 3, or 4, each with different inputs)
             FILNAM  or
             the_beam_code, the_input_file, the_pegs_file or
             the_shared_code,the_phsp_file,the_input_file (isource 20) or
   the_beam_code,the_input_file,the_pegs_file,the_vcu_code,the_vcu_input_file
                                                         (isource = 21)
          FILNAM  Name of file containing energy spectrum   (enflag =1)

                  name of file containing phase-space data  (enflag = 2,3
                                                      and isource=2,8)

                  name of file containing source parameters (enflag = 4)

              or for isource = 9,10

         the_beam_code      The name of the BEAM code you are running as a
                            source (ie BEAM_accelname).  This must have been
                            compiled as a shared library (libBEAM_accelname.so
                            or BEAM_accelname.dll) and exist in
                            EGS_HOME/bin/config.
         the_input_file   The input file used to run the BEAM simulation
                            (no .egsinp extension).  This must exist in your
                            EGS_HOME/BEAM_accelname 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 DOSXYZ simulation (no phase space file
                            is scored, however).
```

```
        the_pegs_file    The pegs data set used by the BEAM simulation
                         (no .pegs4dat extension).  This must be in
                         HEN_HOUSE/pegs4/data or EGS_HOME/pegs4/data.


      or for isource = 20


        the_shared_code  BEAM or non-EGSnrc code used to simulate a geometry
                         interposed between phase space source and
                         the phantom.  Must have been compiled as a shared
                         library and exist in $EGS_HOME/bin/config
                         (isource = 20)
        the_phsp_file    Name of file containing phase space data
                         (isource = 20)
        the_input_file   Input file for the BEAM or non-EGSnrc code used
                         to simulate a geometry interposed between the
                         phase space source and the phantom.  In the
                         case where you are using a BEAM accelerator, this
                         must exist in the $EGS_HOME/BEAM_accelname
                         directory and must specify a scoring plane at
                         the bottom of the accelerator. (isource = 20)


       or for isource = 21


        the_beam_code    Same as for isource=9, 10
        the_input_file   Same as for isource=9, 10
        the_pegs_file    Same as for isource=9, 10
        the_vcu_code     (optional) Name of non-EGSnrc code used to simulate
                         a geometry (usually MLC) interposed between the
                         treatment head source and the phantom.  Must have
                         been compiled as a shared library and exist in
                         $EGS_HOME/bin/config.
 the_vcu_input_file (optional) The full path to the input file for
                         the_vcu_code
--------------------------------------------------------------------------------
 Record SC3c   (if enflag = 3)
      I_BIT_FILTER, NBIT1, NBIT2


         I_BIT_FILTER = 0 INclusive/EXclusive for bits: if any of the 1st
                          set of NBIT1 bits are set and none of the 2nd set
                          of NBIT2 bits are set, the particle is used.
                      = 1 EXclusive for bits: if any of the set of NBIT1
                          bits are set, the particle is not used.
                      = 2 INclusive for regions: use secondary particles
                          that originated in any of the specified NBIT1
                          regions.
                      = 3 EXclusive for regions: do not use secondary
                          particles that originated in any of the NBIT1
                          specified regions.
```

```
                                 Note: For I_BIT_FILTER=0,1 the actual bits
                                 specified below by LATBIT(i) are checked.
                                 For I_BIT_FILTER=2,3 the regions of origin
                                 (stored in bits 24 to 28 and specified by
                                 IREGION_TO_BIT) are checked against LATBIT(i)
                                 values.

                       NBIT1: the number of bits or regions of origin to
                              include (I_BIT_FILTER=0,2) or exclude
                              (I_BIT_FILTER=1,3).
                       NBIT2: the number of bits to exclude.  Only has meaning
                              for I_BIT_FILTER=0, otherwise, it is
                              automatically set to 0.

              Restrictions: for I_BIT_FILTER=0:    0<=NBIT1+NBIT2<=29
                            for I_BIT_FILTER=1:    0<=NBIT1<=29
                            for I_BIT_FILTER=2,3:  0<=NBIT1<=24


    (LATBIT(i),i=1,NBIT1)
        (LATBIT(i),i=1,NBIT1): bits/IREGION_TO_BITs to
                               include (I_BIT_FILTER=0,2) or
                               exclude (I_BIT_FILTER=1,3) from the source.

    Next line required only if I_BIT_FILTER=0 and NBIT2>0.

    (LATBIT(i),i=NBIT1+1,NBIT1+NBIT2)
      (LATBIT(i),i=NBIT1+1,NBIT1+NBIT2): bits to exclude (I_BIT_FILTER=0).

          Restrictions: bits should be >=0 and <=28
                        IREGION_TO_BITs should be >=0 and <=23.

    Note: The check on particle charge is done independently before
          these bit filters are applied.

  end of inputs read by srcxyznrc.mortran
--------------------------------------------------------------------------------


  Record 13 (9 if NMED=0)
          NCASE, IWATCH, TIMMAX, INSEED1, INSEED2,
      BEAM_SIZE, ISMOOTH,IRESTART,IDAT,IREJECT,ESAVE_GLOBAL,NRCYCL,IPARALLEL,
      PARNUM,n_split,ihowfarless,i_phsp_out,i_bindos

          NCASE     Number of histories
          IWATCH    0=>no tracking output, 1=>list every interaction
                    2=>list every electron step 4=>EGS_Windows output
                    IWATCH is for debugging/watching what is happening
                    It MUST be ZERO for production runs
          TIMMAX    Not used currently but prints warning: default 1 hour
```

```
        INSEED1&2 Starting random number seeds (0 is ok)
                  Note that, if using the RANLUX random no.
                  generator, INSEED1 is the luxury level and should
                  have a value >0 and <=4.  Otherwise, a default
                  luxury level of 1 will be used.
        BEAM_SIZE Size of maximum square beam field for the dose
                  calculation.  This parameter is only used by phase
                  space sources (sources 2, 8, 20), BEAM simulation
                  sources (sources 9, 10, 21) and the beam model source
                  (source 4).
                  (Defaults to 100 cm for a 100x100 cm square field)
        ISMOOTH   =1 re-distribute particles when re-start a ph-sp file
                   and/or recycle an individual particle
                  =0 re-use the ph-sp data once run out (no redistribution)
        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 just read in the raw data and do the statistical
                      analysis
                  = 4 recombine .pardose files from parallel runs
        IDAT      = 0 output intermediate files for restarts
                  = 1 do not output intermediate data files at all
                  = 2 output the data file for restart at end only
        IREJECT   = 0 (default) do not perform charged particle range
                      rejection.
                  = 1 perform charged particle range rejection.  If
                      particle range to ECUTIN < distance to nearest voxel
                      boundary then terminate history if E <ESAVE_GLOBAL.
     ESAVE_GLOBAL Energy (MeV) below which charged particle will be
                  considered for range rejection.
        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 (provided the
                  particle has the correct charge and latch value) 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, unless the incident charge is +1 (positron) or
                  the user filters incident particles based on their LATCH
                  values (enflag=3), in which case NRCYCL is set to 0.
                  The calculated NRCYCL does not take into account
                  particles that are rejected because they miss the
                  geometry or because they are beyond the beam field
                  being considered (defined by BEAM_SIZE)
                  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 many
                              particles missing the geometry or beyond beam field),
                              then re-run the simulation with NRCYCL set manually to:
                 NCASE/(NPHSP - NSMISS/NRCYCL_prev - NOUTSIDE - NRJCT - NDBSRJCT)-1
                              where NPHSP is the no. of particles of interest in the
                              the ph-sp file (ie total no. of particle, no. of photons,
                              no. of electrons), NSMISS is the no. of particles that
                              missed the geometry in the previous run, NRCYCL_prev is
                              the value of NRCYCL used in the previous run,
                              NOUTSIDE is the no. of particles rejected because they
                              were beyond BEAM_SIZE in the previous run, NRJCT
                              is the no. of particles rejected because they were
                              multiple passers, and NDBSRJCT is the number of photons
                              rejected because they fall outside the directional
                              bremsstrahlung splitting (DBS) field radius at the SSD
                              (see inputs for source 2,8 for more info on rejecting
                              fat photons when DBS is used in BEAM to generate phase
                              space files). All of the aforementioned values are
                              available in the .egslst file. Always round your
                              calculated value of NRCYCL up.

    Note: the following 2 inputs are only relevant if you are manually
          creating/submitting your parallel jobs or if you are using the
          unix pprocess script. In the case of the pprocess script, these
          inputs get set automatically.

          IPARALLEL   set >1 if you are distributing the job among IPARALLEL
                       machines. This will generate
                       a binary .pardose file for each job which will contain
                       the doses in all voxels WITHOUT UNCERTAINTIES. IPARALLEL
                       defaults to 0.
            PARNUM   Only needs to be set if IPARALLEL>1 and you are using
                       a single phase space source. The partition of the phase
                       space source that will be used for this one of the
                       series of parallel input files (for the same total
                       simulation) is given by:
                         (PARNUM-1)*(nshist/IPARALLEL)<nnphsp<=
                                                 (PARNUM)*(nshist/IPARALLEL)
                       where nshist is the total number of particles in the
                       phase space file and nnphsp is the number of the
                       particle chosen for the simulation. Thus, PARNUM
                       should cover the range 1<=PARNUM<=IPARALLEL. If you
                       are using a separate phase space source for each
                       parallel job (ie the phase space source exists in
                       IPARALLEL pieces) and, thus, do not wish to partition
                       the phase space sources, PARNUM can be left at its
                       default value of 0.

          n_split   If set > 1, then all photons will be split

```
                    n_split times.  This option increases the efficiency of
                    dose caluclations more than photon forcing.
                    A rule of thumb for good efficiency is
                        n_split >= No/(1-exp(-Lambda))
                    where Lambda is the approx. number of photon MFP
                    in the geometry of interest and No >= 5.
                    Note that if you use the above, there will
                    be on average approx. No primary interactions per
                    incident photon => reduce the number of
                    histories by this number.

        The n_split algorithm works as follows:
         * dpmfp_i = -log(1 - (eta+i-1)/n_split)
           where dpmfp_i is MFP to the next interaction for the i-th sub-photon
           (i=1,2,...,n_split)
           eta is a random number (the same for all n_split sub-photons)
         * Once at the interaction site, the i'th sub-photon produces
           electrons and/or scattered photons. Scattered photons are
           killed with probability 1/n_split, so that, if they survive,
           they have the weight of the original photon. Electrons have the
           weight of 1/n_split of original weight.
         * In any radiative events (brems, annih, annih at rest), photons
           are killed with probability 1/n_split => they have again the
           weight of the photon that initiated the history, if they survive


        ihowfarless     Set to 1 to use the 'HOWFARLESS' algorithm for
                        transport in a homogeneous phantom.  When 'HOWFARLESS'
                        is used, subroutines HOWFAR and HOWNEAR only consider
                        the extreme outer boundaries of the phantom when
                        calculating the distance along the particle trajectory
                        or nearest perpendicular distance to a region boundary.
                        This removes the restriction that charged particle
                        steps must stop at all voxel boundaries and, thus,
                        speeds up the calculation considerably.  For the purpose
                        of dose deposition in the voxels, the total curved
                        charged particle step is approximated using 2 straight-
                        line steps joined at a hinge point.  See the DOSXYZnrc
                        manual for more details about the approximation.
                        Restrictions on charged particle step length then become
                        the EGSnrc input variables Smax (max. allowable charged
                        particle step length) and estepe (max. fractional energy
                        loss/step).  Leaving Smax set to its default value
                        of 5 cm should maximize efficiency of the 'HOWFARLESS'
                        algorithm at most beam energies.  Efficiency improvement
                        due to 'HOWFARLESS' depends on the incident beam type,
                        DOSXYZnrc voxel size and the boundary crossing algorithm
                        (BCA) used.  For incident photon beams from accelerators
                        simulated with BEAMnrc (phase space or BEAMnrc sim.
                        sources), the efficiency improves by ~30%
```

                          when the PRESTA-I BCA is used, and by factors of
                          2.5-3.5 when the more accurate (but slower) EXACT
                          BCA is used. In monoenergetic electron beams,
                          'HOWFARLESS' improves efficiency by a factor of up to
                          ~4 with the PRESTA-I BCA and by a factor of up to
                          ~15 with the EXACT BCA. Recommended for all homogeneous
                          phantom calculations, but it must NOT be used in
                          nonhomogeneous phantoms.

          i_phsp_out   For outputting IAEA format phase space data on exit
                          from the phantom geometry. If i_phsp_out=1, then data
                          is output in DOSXYZnrc coordinates. If i_phsp_out=2
                          data is output in BEAMnrc coordinates. This option is
                          only available for phase space or BEAM simulation
                          sources (which have a region surrounding the phantom).
                          The default is i_phsp_out=0--no phase space
                          output. If you are using source 20 or source 21
                          (synchronized sources) then you also have the option of
                          storing frMU_indx in the file (see source inputs).

          i_bindos   For outputting sparse binary dose data instead of the
                          dense ASCII .3ddose format. If i_bindos=0, then the
                          dose will be output in .3ddose format. Any other value
                          will output a .bindos file instead.

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

 All input associated with selection of EGSnrc transport parameter
 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 default 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):

      Global ECUT=   Global (in all regions) electron transport cut

```
                         off energy (in MeV). If this input is missing,
                         or is < ECUTIN from the main DOSXYZnrc inputs
                         (See above) then ECUTIN is used for global ECUT.
                         If this input is missing, AE(medium) will be used.
                         [ ECUT ]
     Global PCUT=        Global (in all regions) photon transport cut
                         off energy (in MeV). If this input is missing,
                         or is < PCUTIN from the main DOSXYZnrc inputs
                         (See above) then PCUTIN is used for global PCUT.
                         If this input is missing, AP(medium) will be used.
                         [ 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 is used.  In this case, SMAX
                         will default to 1e10.  However, if either
                          Electron-step algorithm= PRESTA-I
                          or
                          Boundary crossing algorithm= PRESTA-I (the default in
                         DOSXYZnrc), 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 default 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= EXACT, PRESTA-I (default in DOSXYZnrc)
                         There are two selections possible: 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).
                         The second option is PRESTA-I, if selected boundaries
                         will be crossed a la PRESTA, i.e. with lateral
                         correlations turned off and MS forced at boundaries.
                         Default is PRESTA-I for efficiency reasons. This
                         is known not to be exactly correct, and when charged
                         particle equilibrium does not hold or when there is
                         a large difference in size between dose voxels and
                         voxels making up the rest of the phantom, can result
                         in an overestimate of dose by up to 2.5%.  In such
                         cases, swich to EXACT BCA.
                         [ bca_algorithm, exact_bca ]
```

Skin depth for BCA=
                Determines the distance from a boundary (in elastic
                MFP) at which the algorithm will go into single
                scattering mode (if EXACT boundary crossing) or
                switch off lateral correlations (if PRESTA-I boundary
                crossing). Default value is 3 for EXACT or
                exp(BLCMIN)/BLCMIN for PRESTA-I (see the PRESTA paper
                for a definition of BLCMIN). 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 single-scattering mode.
                If you choose PRESTA-I boundary crossing (default
                in DOSXYZnrc) and make Skin depth for BCA large,
                you will get default EGS4 behaviour (no PRESTA).
                [skindepth_for_bca]

        Note that the above defaults have been chosen as a compromise
        between accuracy (EXACT BCA) and efficiency (PRESTA-I BCA)
        since the PRESTA-I BCA algorithm has proven to generally
        produce satisfactory results. Note that the new transport
        mechanics of EGSnrc are maintained away from boundaries and
        that one always has the option of verifying the accuracy
        by doing a long run with the EXACT BCA.

Electron-step algorithm= PRESTA-II (default), PRESTA-I (legacy)
                Determines the algorithm used to take into account
                lateral and longitudinal correlations in a
                condensed history step.
                [transport_algorithm]
Spin effects=   Off, On (default)
                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)
                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 (default), NIST, NRC
                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 significant in the keV energy range. If NRC is
selected, the NRC brems cross-section data base will
be used, which is a version of the NIST data base
with corrected electron-electron brems contributions
(corrections to the NIST data is typically only
significant for low values of the atomic number Z
and for k/T < 0.005).
[ibr_nist]

Triplet production= On or Off (default). Turns on/off simulation
of triplet production. If On, then Borsellino's
first Born approximation is used to sample triplet
events based on the triplet cross-section data.
[itriplet]

Bound Compton scattering= On, Off, Simple or norej (default)
If Off, Compton scattering will be treated with
Klein-Nishina, with On Compton scattering is
treated in the Impulse approximation.
With Simple, the impulse approximation incoherent
scattering function will be used (i.e., no Doppler
broadenning). With norej the actual total bound
Compton cross section is used and there are no
rejections at run time.
Make sure to turn on for low energy applications,
not necessary above, say, 1 MeV.
[IBCMP]

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)get_inputs.mortran).
[radc_flag]

Electron Impact Ionization= Off (default), On, casnati, kolbenstvedt,
gryzinski or penelope. If set to On or ik, then
use Kawrakow's theory to derive EII cross-sections.
If set to casnati, then use the cross-sections of
Casnati (from file $HEN_HOUSE/data/eii_casnati.data).
Similar for kolbenstvedt, gryzinski and penelope.
This is only of interest in kV X-ray calculations.
Note that the user can supply their own EII
cross-section data as well. The requirement is that
the file eii_suffix.data exists in the $HEN_HOUSE/data
directory, where suffix is the name specified.
Entry is case-sensitive except for Off, On or ik.

```
                            [ eii_flag, eii_xfile ]
      Pair angular sampling= Off, Simple (default), KM.
                            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.
                            Default is Simple, make sure you 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 asymmetry in the
                            positron-electron energy distribution.
                            [ pair_nrc ]
      Photon cross sections= Photon cross-section data.  Current options are
                            si (Storm-Israel), epdl (Evaluated Photon Data
                            Library), xcom (default), pegs4, mcdf-xcom and
                            mcdf-epdl:
                            Allows the use of photon cross-sections other than
                            from the PEGS4 file (unless the pegs4 option is
                            specified).  Options mcdf-xcom and mcdf-epdl use
                            Sabbatucci and Salvat's renormalized photoelectric
                            cross sections with either xcom or epdl for all other
                            cross sections.  These are more accurate but can
                            increase CPU time by up to 6 %.
                            Note that the user can supply their own cross-section
                            data as well. The requirement is that the files
                            photon_xsections_photo.data,
                            photon_xsections_pair.data,
                            photon_xsections_triplet.data, and
                            photon_xsections_rayleigh.data exist in the
                            $HEN_HOUSE/data directory, where photon_xsections
                            is the name specified. This entry is case-sensitive.
                            [ 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 ]
      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]
        Rayleigh scattering= Off, On (default), custom
                              If On, turns on coherent (Rayleigh) scattering.
                              Default is On. Should be turned on for low energy
                              applications. If custom, user must provide media names
                              and form factor files for each desired medium. The
                              rest of the media use the default atomic form factors.
                              A PEGS4 data set is not required anymore.
                              [IRAYLR]
        ff media names = A list of media names (must match media found in
                              PEGS4 data file) for which the user is going to
                              provide custom Rayleigh form factor data.
                              [iray_ff_media($MXMED)]
        ff file names = A list of names of files containing the Rayleigh
                              form factor data for the media specified by
                              the ff media names = input above. Full directory
                              paths must be given for all files, and for each medium
                              specified, iray_ff_media(i), there must be a
                              corresponding file name, iray_ff_file(i). For
                              example files, see the directory
                              $HEN_HOUSE/data/molecular_form_factors.
                              [iray_ff_file($MXMED)]
        Photonuclear attenuation= Off (default) or On
                              If On, models the photonuclear effect. Current
                              implementation is crude. Available on a
                              region-by-region basis (see below)
                              [IPHOTONUCR]
        Photonuclear cross sections= Total photonuclear cross sections. User-
                              supplied total photonuclear cross-sections in
                              $HEN_HOUSE/data/photonuc_xsections_photonuc.data,
                              where photonuc_xsections is the name supplied for
                              this input (case sensitive). In the absence of
                              any user-supplied data, or if photonuc_xsections
                              is set to 'default', the default file is
                              iaea_photonuc.data.
                              [photonuc_xsections]
        Photoelectron angular sampling= Off or On (default)
                              If Off, photo-electrons get the direction of the
                              'mother' photon, with On, Sauter's formula is
                              used (which is, strictly speaking, valid only for
                              K-shell photo-absorption).
                              If the user has a better approach, replace the macro
                                  $SELECT-PHOTOELECTRON-DIRECTION;
                              The only application encountered where this option
                              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 ]
           Atomic relaxations= Off, On, eadl (default), simple
                                On defaults to eadl.
                                When simulating atomic relaxations:
                                - 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 sections
                                - Shell vacancies created in photoelectric,
                                  compton and electron impact ionization events
                                  are relaxed via emission of fluorescent X-Rays,
                                  Auger and Koster-Cronig electrons.
                                  The eadl option features a more accurate treatment
                                  of relaxation events and uses binding energies
                                  consistent with those in of the photon cross sections
                                  used in the simulation.  If using mcdf-xcom or
                                  mcdf-epdl photon cross sections, you cannot use
                                  the simple option and this will automatically get
                                  reset to eadl. Make sure to use eadl or simple for
                                  low energy applications.
                                [ IEDGFL ]


        Atomic relaxations, Rayleigh scattering, Photoelectron angular sampling,
        Bound Compton scattering and photonuclear effect
        can also be turned On/Off on a region-by-region basis. An example for
        Atomic relaxations on a region-by-region basis is:

        Atomic relaxations= On in Regions    or
        Atomic relaxations= Off in regions


                                Then define the regions in which you want
                                the feature to be turned on:

        Bound Compton start region=
        Bound Compton stop region=
                 or
        Rayleigh start region=
        Rayleigh stop region=
                 or
        Relaxations start region=
        Relaxations stop region=
                 or
        PE sampling start region=
        PE sampling stop region=
                 or
        Photonuclear start region=
        Photonuclear 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 the input is checked against minimum
                        and maximum region numbers and ignored if
                        start region < 1 or stop_region > $MXREG or
                        start region > stop region.

                        ECUT, PCUT and SMAX can also be set on a
                        region-by-region basis. To do so, iclude
                        in your input file

  Set XXXX=               f_value1, f_value2, ...
  Set XXXX start region= i_value1, i_value2, ...
  Set XXXX stop region=  j_value1, j_value2, ...

                        where XXXX is ECUT, PCUT or SMAX ,
                        f_value1, f_value2,... are the desired values for XXXX
                        and i_value_i and j_value_i are the start and
                        stop regions.
```

4   Source Routines

4.1   Source Types in DOSXYZnrc

The following source types have been developed for DOSXYZnrc:

- Parallel rectangular beam incident from the front (isource = 0)

- Parallel rectangular beam incident from any direction (isource = 1)

- Phase-space source, particles incident from any direction (isource = 2)

- Point source incident from the front (isource = 3)

- Beam characterization model, particles incident from any direction (isource = 4)

- Uniform isotropically radiating parallelepiped within DOSXYZnrc volume (isource = 6)

- Parallel rectangular beam incident from multiple angles (isource = 7)

- Phase-space source incident from multiple angles (isource = 8)

- Full BEAM treatment head simulation as source (isource = 9)

- Full BEAM treatment head simulation as source incident from multiple angles (isource = 10)

- Phase space source incident from multiple angles, with multiple SSD's and isocentres–options to run source through a shared library geometry (defined by BEAM or other MLC code) and synchronize with geometry settings in the case of a BEAM shared library (isource = 20)

- BEAM treatment head simulation incident from multiple angles, with multiple SSD's and isocentres–options to synchronize with BEAM geometry settings and run source through a shared library geometry (other MLC code) (isource = 21)

For all sources, the first input record is:
`iqin, isource, ......`

4.2   isource = 0: Parallel Rectangular Beam Incident from Front

The uniform parallel rectangular beam is always assumed to be incident parallel to the Z-axis from the front of the phantom. The input parameters are:

iqin Charge of the incident beam (-1: electron, 0: photon, 1: positron)

isource $= 0$

xinl,xinu Lower and upper x-bounds on the phantom surface

yinl,yinu Lower and upper y-bounds on the phantom surface

thetax Angle of the beam relative to the x-axis (degrees)

thetay Angle of the beam relative to the y-axis (degrees)

thetaz Angle of the beam relative to the z-axis (degrees)
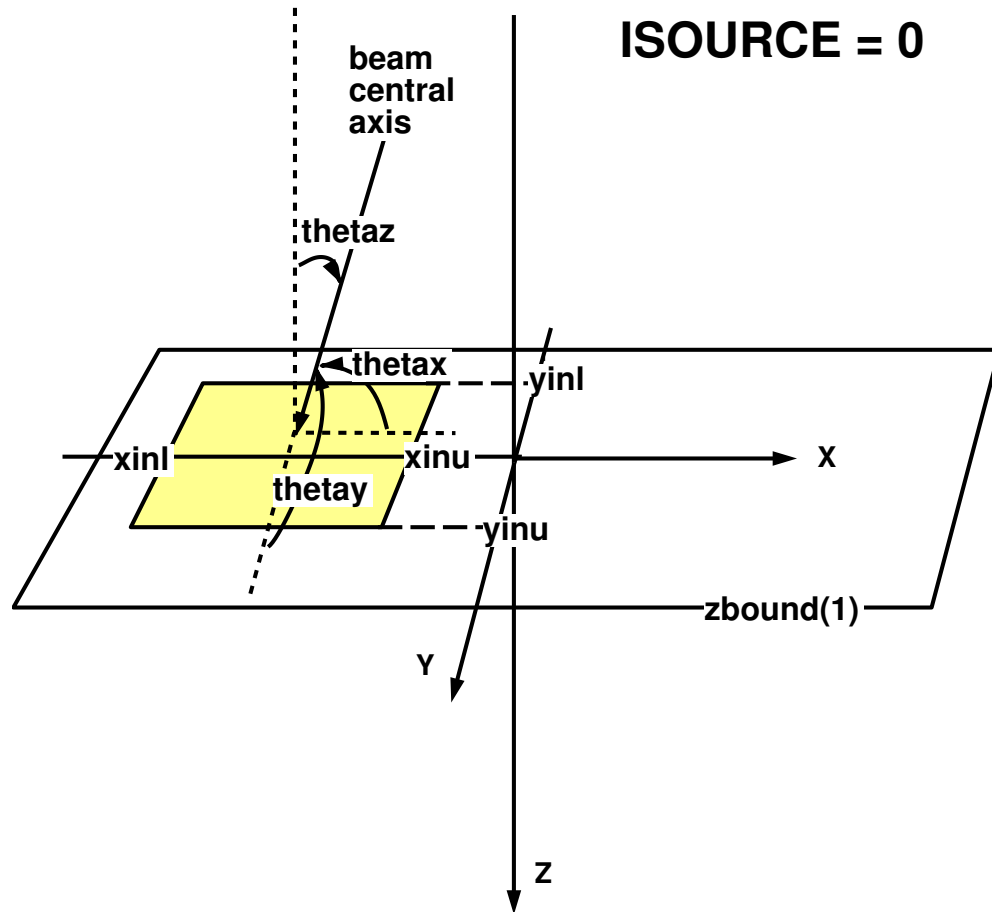
Figure 1: Parallel rectangular beam (isource=0) showing the beam field, defined by `xinu`, `xinl`, `yinu`, and `yinl` on the phantom surface and the beam directions, `thetax`, `thetay`, and `thetaz`. The angles `thetax` and `thetay` are relative to the positive x- and y-axes respectively, while `thetaz` is measured relative to the negative z-axis.

4.3   isource = 1: Parallel Rectangular Beam Incident from Any Direction

This uniform parallel rectangular beam may be incident from any direction. The input parameters are:

iqin Charge of the incident beam (-1: electron, 0: photon, 1: positron)

isource = 1

xiso/yiso/ziso x-, y-, z-coordinates of the isocenter. The isocenter is normally inside the phantom.

theta Angle between the +z direction and a line joining the center of the beam where it strikes the phantom surface to the isocenter. In a polar coordinate system, this angle is known as the polar angle and normally has a range 0-180 degrees. Note that a centred beam incident along the +z-axis (ie from the top) has theta=180 degrees. theta is not to be confused with thetaz for isource=0, for which case thetaz=0 degrees to aim the source=0 beam along the +z-axis.

phi Angle between the +x direction and the projection on the x-y plane of the line joining the center of the beam on the phantom surface to the isocenter on the xy plane. In a polar coordinate system, this angle is known as the azimuthal angle and normally has a range 0-360 degrees.

xcol/ycol Total x- and y-widths of the beam on the plane perpendicular to the beam direction, defined by the center of the beam and the isocenter

phicol Angle by which the collimator is rotated in the collimator plane perpendicular to the beam direction. phicol is determined for theta=phi=0. The positive sense of rotation is counterclockwise as one sights down the beam direction. Note that the effect of setting phicol=90 degrees is the same as switching the values of xcol and ycol when theta and phi are 0 or are both multiples of 90 degrees.
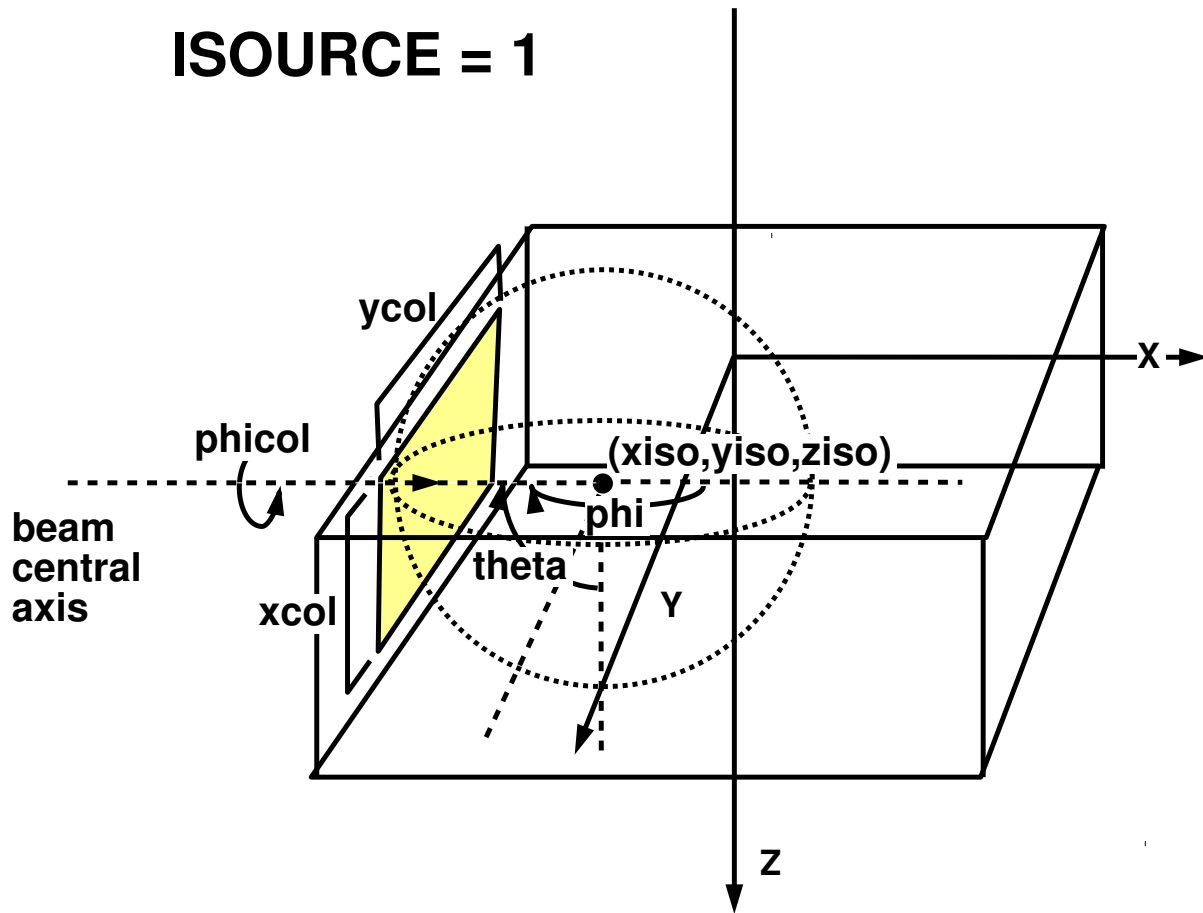
ISOURCE = 1

Figure 2: Parallel rectangular beam incident from any direction (isource=1). A polar coordinate system is set up at the isocenter, (xiso,yiso,ziso). The position of the beam collimator is then defined using the angles, `theta` and `phi`. In the particular example shown in this figure, with the beam incident on the centre of the negative Y face of the volume, `theta` is 90 degrees and `phi` is 180 degrees. The dimensions of the rectangular collimator are given by `xcol` and `ycol`. An additional degree of freedom is available by allowing the collimator to rotate in its own plane by angle `phicol`. Note that source 1 is always incident right on the phantom surface (ie particles do not travel through air to get to the phantom)

4.4   isource = 2: Phase-Space Source Incident from Any Direction

This source uses a phase-space file generated during a BEAMnrc simulation at any flat scoring plane of a linear accelerator geometry. A user can choose any particular type of particles from the phase-space file and score dose components using the `LATCH` filter. The field size of the incident beam can be reduced using the parameter `BEAM_SIZE`. There is also a parameter called `ISMOOTH` which can be set to 1 if one wants to shift the particles to their symmetrical positions with respect to x- and y-axis in the phase-space file for repeated use of these phase-space particles. For `ISMOOTH` = 0, no shift will be made when phase-space particles are re-used. `LATCH` filter, `BEAM_SIZE` and `ISMOOTH` are input later in the file.

Input parameters for a phase-space source are:

iqin Charge of the incident beam (-1: electron, 0: photon, 1: positron, 2: all particle types)

isource = 2

xiso/yiso/ziso x-, y-, z-coordinates of the isocenter (normally located within the phantom)

theta Same as for isource=1.

phi Same as for isource=1.

dsource For a BEAMnrc format phase space source, this is the absolute distance from the isocenter to the source center, which is, by definition, the origin of the phase-space plane (origin may not even be in the beam). For an IAEA format phase space source, this is the distance from the isocentre to the primary source (*i.e.* the source used to generate the phase space file), or SAD.

phicol Angle by which the source is rotated in the source plane perpendicular to the beam direction. `phicol` is determined for `theta=phi=0`. The positive sense of rotation is counterclockwise as one sights down the beam direction.

i_dbs Set to 1 if the phase space source was generated using directional bremsstrahlung splitting (DBS) in BEAM **AND** you wish to reject photons directed outside the splitting field (which will all be fat). Set to 0 otherwise. Note that `i_dbs` is read in as a real and converted to integer.

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

ssd_dbs SSD (in cm) where the DBS splitting field radius was defined in the BEAMnrc simulation used to generate this source. Only needed if `i_dbs` = 1.

z_dbs Z value (in cm) in the BEAMnrc simulation where this phase space source was scored. This will be at the back of a component module (CM). Only needed if `i_dbs` = 1.

e_split Number of times to split charged particles as soon as they enter the phantom geometry. Split particles have their weight reduced by a factor of $1/$e_split. This is only used in conjunction with photon splitting (n_split, see Section 8.16) and prevents higher-weight contaminant electrons from compromising statistics in photon beams. For maximum efficiency, it is suggested that you set e_split=n_split, the photon splitting number.

FILNAM The full name (including extension) of the phase space file (including the directory path). In the case of an IAEA-format phase space source, the full name of the phase space data (.IAEAphsp) file is input here. The header (.IAEAheader) file as assumed to exist in the same directory. For more information about IAEA-format date, see the BEAMnrc Manual.

theta, phi and dsource can be set to place the source anywhere inside the phantom or the surrounding region; the medium and the thickness of the surrounding region is input by the user. Particles from the phase space which are initially outside both the phantom and surrounding region are terminated immediately. If the medium of the surrounding region is air, for example, the phase-space particles will be transported *properly* through air to the surface of the phantom. A particle history is terminated if it is determined that the particle will not make it to the phantom surface (the particle loses all its energy in the surrounding region or it escapes through the outer boundaries of the surrounding region).

Note that, for source 2, the user must set enflag $= 2$ or 3, input the mode of the phase-space file (0 or 2), and input the medium number and the thickness for the surrounding region. The user should also input the phase-space file name after the above inputs. These other inputs are described in sections 5 and 6 below. For more information related to phase-space sources, see the section on phase-space files in the BEAMnrc User's Manual [12].

The value of phicol must be set to 180 degrees in order for the X-Y coordinates in a BEAMnrc-generated phase space source to map onto the equivalent X-Y coordinates in DOSXYZnrc. This is because of the general coordinate transformation done going from BEAMnrc to DOSXYZnrc.

4.4.1 Inputs related to directional bremsstrahlung splitting

If you have used the directional bremsstrahlung splitting (DBS) variance reduction technique (see the BEAMnrc User's Manual[12] for more details) to generate the phase space source, then it is recommended that you use the inputs i_dbs, r_dbs, ssd_dbs and z_dbs to prevent unsplit, high-weight ("fat") photons which are outside the splitting field from compromising your dose statistics. r_dbs and ssd_dbs (the DBS splitting field radius and SSD in cm) are available directly from the inputs for DBS in the BEAM simulation, while z_dbs is simply the Z value at the back of the component module (CM) in the BEAMnrc simulation in which this phase space source was scored. If i_dbs is set to 1, then before a photon is used in the DOSXYZnrc simulation, it is projected along its trajectory from z_dbs to ssd_dbs (we assume ssd_dbs$\geq$z_dbs). If it falls outside r_dbs at ssd_dbs, then the photon is not used in the DOSXYZnrc simulation. In the context of the BEAM simulation this photon will not have been split and will be fat (but this information is not carried in the phase space file

except indirectly because of the larger weight which is not a unique specifier). To be able to ignore these fat photons and thereby increase the efficiency considerably, it is important to ensure that the dose from the photons being ignored is negligible. This can usually be arranged by making the splitting field big enough.

Note that charged particles are not 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 (again, see the BEAMnrc User's Manual for more details).

4.4.2   IAEA format phase space sources

In the case of an IAEA format phase space source, the input, `dsource` (see above), defines the distance from the primary source (*i.e.* the source used to generate the phase space file) to the phantom isocentre, or SAD. This is because, unlike BEAMnrc format phase space files, the incident Z-position of particles is available and read from either the phase space data (in the case of 3-D phase space files) or the header file (in the case of planar phase space files, such as those scored by BEAMnrc). The input value of `dsource` is then subtracted from this Z-position to give the distance along the beam axis from the phantom isocentre to the particle incident position.

The ability of source 2 to automatically handle IAEA phase space files in which the Z-position for each particle is scored (3-D data) is particularly useful in cases where the phase space data available for an accelerator is scored on a non-planar surface (as is the case with the data that Varian has made available for their TrueBeam accelerators) or when using 3-D phase space data scored by an upstream DOSXYZnrc simulation (see Section 8.18).
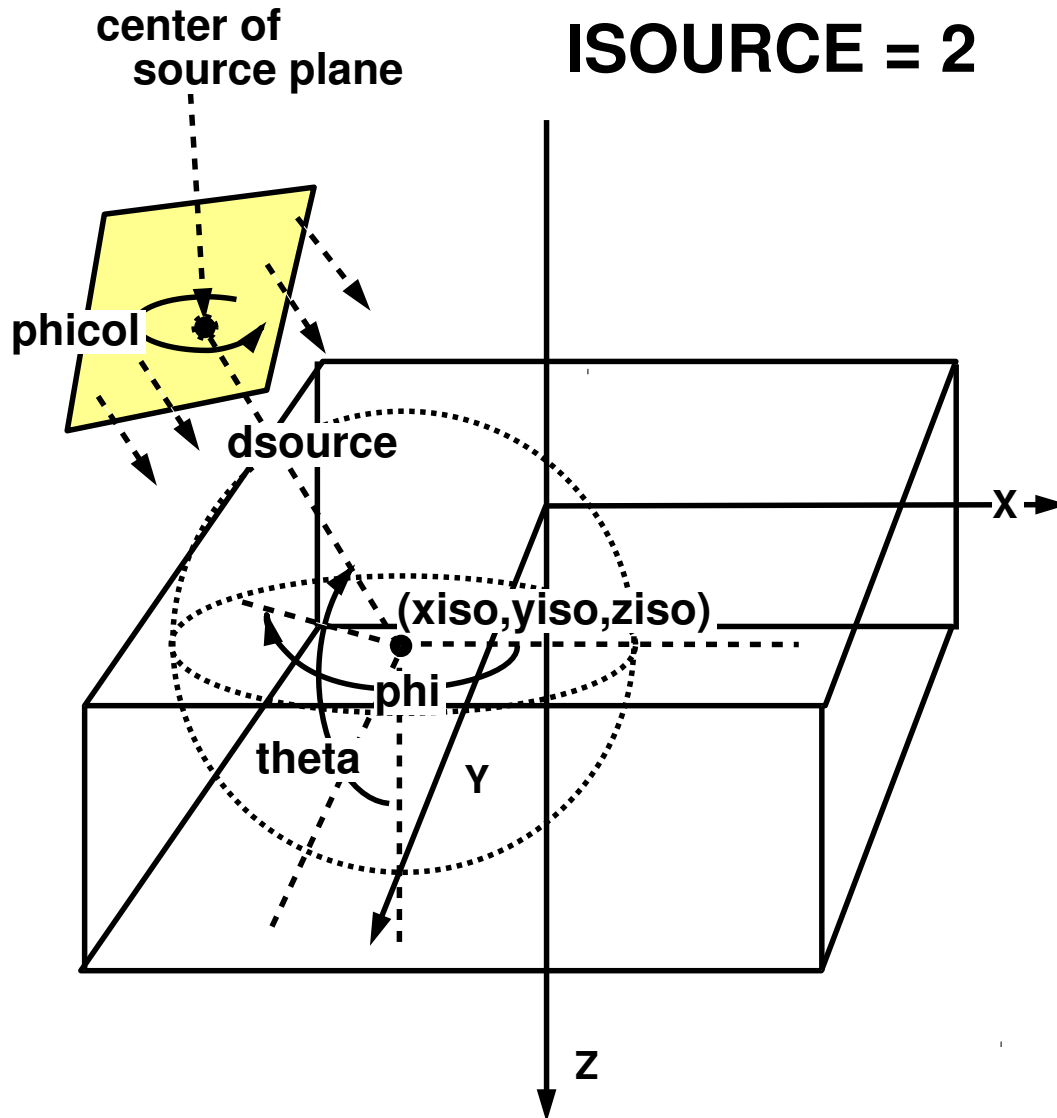
Figure 3: Phase-space source incident from any direction (isource=2). Similar to source 1, a polar coordinate system is set up at the isocenter, (xiso, yiso, ziso). The position of the origin in the phase-space plane is then defined by the angles `theta` and `phi`, and the distance from the isocenter, `dsource`. The source can be rotated in its own plane using the variable `phicol`. The figure shows that the source plane does not necessarily have to be right against the surface of the phantom, however it must be within the surrounding region specified by `dsurround` (see section 5).

4.5   isource = 3: Point Source Rectangular Beam Incident from Front

The isotropically-radiating point source is placed on the Z-axis. It is assumed to be incident on the front surface of the phantom, but can be placed any distance above the phantom. The beam field can be asymmetric.

The input parameters for the point source are:

iqin Charge of the incident beam (-1: electron, 0: photon, 1: positron)

isource $= 3$

xinl,xinu Lower and upper x-bounds of the field on the phantom surface

yinl,yinu Lower and upper y-bounds of the field on the phantom surface

ssd Distance from the point source to the phantom surface (cm)

Note that, between the source and the phantom surface the medium is assumed to be a vacuum for this source.
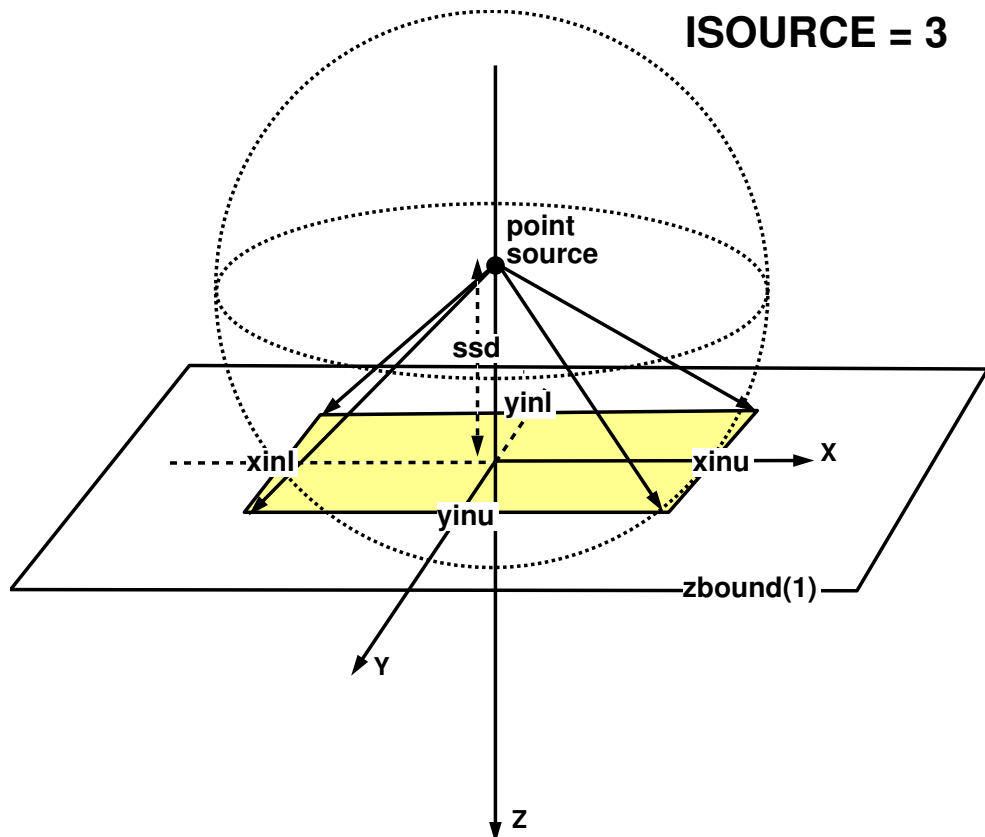
Figure 4: Point source incident from the front (isource=3). The isotropically-radiating point source is located on the Z-axis at distance ssd above the phantom. The source is collimated to a rectangular field defined by xinu, xinl, yinu, yinl on the phantom surface.

4.6   isource = 4: Beam Characterization Model Incident from Any Direction

This source uses the beam characterization models created by BEAMDP from phase space data generated by BEAMnrc. This source consists of a variety of sub-sources for different types of particles coming from different components of a linear accelerator. Each sub-source has its own spectral and (planar) fluence distributions and the correlation between the energy, position and incident angle is retained by sampling the particle positions on the sub-source (the surface of the component) and on the phase-place plane. A beam model input file, generated by BEAMDP, is required by this source. For more details about beam characterization models see the BEAMnrc User's Manual and associated paper[14, 15].

Apart from `isource`, which must be set to 4, input parameters for source 4 are the same as for source 2:

Note this source requires `enflag` = 4 and to input the beam model input file name after the `enflag` input. The field size of the incident beam can be reduced using the parameter `BEAM_SIZE`. These inputs are described in sections 5 and 6 below.

The default version of DOSXYZnrc does not include this source. In order to use it you must recompile DOSXYZnrc using the instructions in section 2.2.1. .

4.7   isource = 6: Uniform Isotropically Radiating Parallelepiped within DOSXYZnrc Volume

This source allows the user to simulate a uniform isotropically radiating rectangular volume (parallelepiped) within the DOSXYZnrc phantom. The active volume is restricted to being completely contained within the DOSXYZnrc phantom. However, it can be shrunk to a point anywhere within the phantom.

The input parameters for the uniform isotropically radiating parallelepiped:

iqin Charge of the incident beam (-1: electron, 0: photon, 1: positron)

isource $= 6$

xinl,xinu Lower and upper x-bounds of the active volume (cm)

yinl,yinu Lower and upper y-bounds of the active volume (cm)

zinl,zinu Lower and upper z-bounds of the active volume (cm)
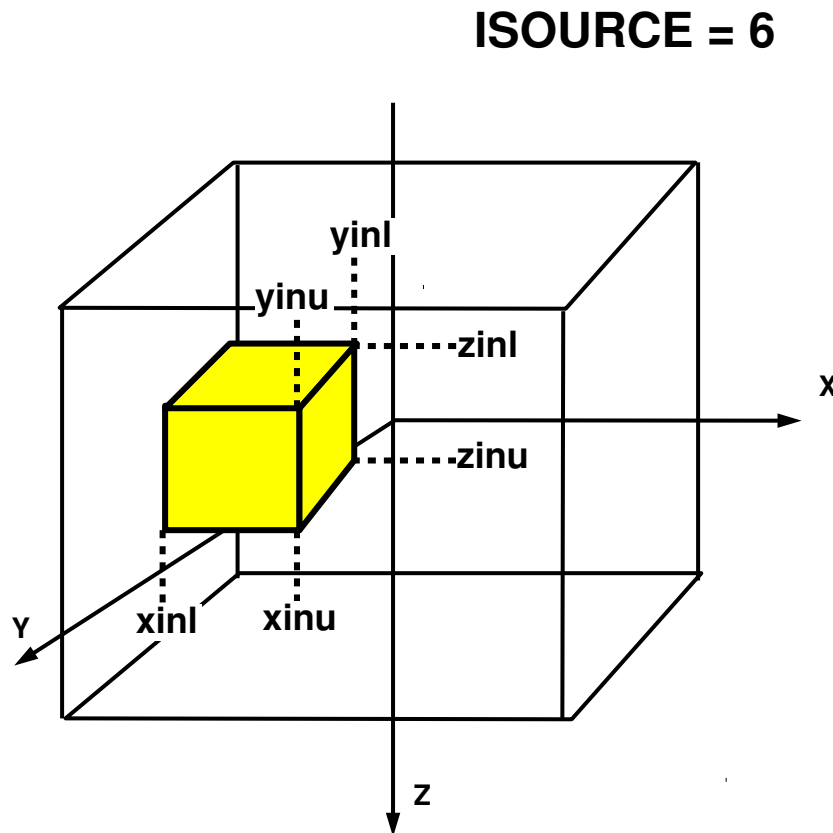
Figure 5: Uniform isotropically radiating parallelepiped within DOSXYZnrc phantom (isource=6). The active volume, specified by xinl, xinu, yinl, yinu, zinl, zinu, must fit within the phantom. The source can be shrunk to a point within the phantom by setting xinu=xinl, yinl=yinu and zinl=zinu.

4.8   isource = 7: Parallel Rectangular Beam Incident from Multiple Directions

This is a uniform parallel rectangular beam similar to source 1, but incident from multiple, user-defined directions (theta-phi pairs). The input parameters are:

  `iqin` Charge of the incident beam (-1: electron, 0: photon, 1: positron)

  `isource = 7`

  `xiso/yiso/ziso` x-, y-, z-coordinates of the isocenter. The isocenter is normally inside the phantom.

  `nang` The number of incident theta-phi pairs or, if negative, then abs(`nang`) is the number of groups of incident theta-phi pairs, where, within a group, all theta-phi pairs have equal probability. The incident angles theta and phi are defined the same as they are in source 1. Theta-phi pairs/groups are specified on separate lines (see below).

  `xcol/ycol` Total x- and y-widths of the beam on the plane perpendicular to the beam direction, defined by the center of the beam and the isocenter

  `phicol` Angle by which the collimator is rotated in the collimator plane perpendicular to the beam direction. `phicol` is defined the same way as in source 1.

And then, on the subsequent `abs(nang)` lines:

If `nang` > 0:

For i=1,`nang`: `theta(i)`, `phi(i)`, `pang(i)`

  `theta(i)` Theta for pair i (degrees).

  `phi(i)` Phi for pair i (degrees).

  `pang(i)` Probability of particle being incident at Theta(i)-Phi(i).

If `nang` < 0:

For i=1,-`nang`: `ivary(i)`, `angfixed(i)`, `angmin(i)`, `angmax(i)`, `ngang(i)`, `pgang(i)`

  `ivary(i)` Set to 0 to vary phi in group i; Set to 1 to vary theta in group i.

  `angfixed(i)` Fixed theta (`ivary(i)`=0) or phi (`ivary(i)`=1) in group i (degrees).

  `angmin(i)` Minimum varying phi (`ivary(i)`=0) or theta (`ivary(i)`=1) in group i (degrees).

angmax(i) Maximum varying phi (`ivary(i)=0`) or theta (`ivary(i)=1`) in group i (degrees).

ngang(i) Number of equally-spaced phi (`ivary(i)=0`) or theta (`ivary(i)=1`) between and including `angmin(i)` and `angmax(i)` in group i (Note that, since it includes `angmin(i)` and `angmax(i)`, `nang(i)` must be ≥ 2).

pgang(i) Probability of particle being in group i. Within a group, all theta-phi pairs have equal probability.

Note that the `pang(i)` are automatically normalized.

4.9   isource = 8: Phase-Space Source Incident from Multiple Directions

This source is similar to source 2, except that it is incident from multiple, user-defined directions (theta-phi pairs).

Input parameters for source 8 are:

iqin Charge of the incident beam (-1: electron, 0: photon, 1: positron, 2: all particle types)

isource $= 8$

xiso/yiso/ziso x-, y-, z-coordinates of the isocenter (normally located within the phantom)

nang The number of incident theta-phi pairs or, if negative, then abs(`nang`) is the number of groups of incident theta-phi pairs, where, within a group, all theta-phi pairs have equal probability. The incident angles theta and phi are defined the same as they are in source 2. Theta-phi pairs/groups are specified the same way as they are in source 7 (see section 4.8 above).

dsource For BEAMnrc phase space files, this is the absolute distance from the isocenter to the source center, which is, by definition, the origin of the phase-space plane (origin may not even be in the beam). For IAEA phase space files, this is the primary source to isocentre distance (SAD).

phicol Angle by which the source is rotated in the source plane perpendicular to the beam direction. `phicol` is determined for `theta=phi=0`. The positive sense of rotation is counterclockwise as one sights down the beam direction.

i_dbs Set to 1 if the phase space source was generated using directional bremsstrahlung splitting (DBS) in BEAMnrc **AND** you wish to reject photons directed outside the splitting field (which will all be fat). Set to 0 otherwise. Note that `i_dbs` is read in as a real and converted to integer.

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

ssd_dbs SSD (in cm) where the DBS splitting field radius was defined in the BEAMnrc simulation used to generate this source. Only needed if i_dbs = 1.

z_dbs Z value (in cm) in the BEAMnrc simulation where this phase space source was scored. This will be at the back of a component module (CM). Only needed if i_dbs = 1.

e_split Number of times to split charged particles as soon as they enter the phantom. The weight of split particles is reduced by a factor of 1/e_split. This is used in conjunction with photon splitting (input variable n_split, see Section 8.16) to prevent higher-weight contaminant electrons from compromising dose statistics. For optimum efficiency, set e_split=n_split.

FILNAM The full name (including extension) of the phase space file (including the directory path). See the description of source 2 (Section 4.4) for more details.

An example may help illustrate this input. The following input:

```
0, 8, 0.0, 0.0, 0.0, -1.0, 80.0, 90.0, 0.0, 0.0, 0, 0.0, 0.0, 0.0
1, 90.0, 0.0, 356.40, 100, 1.0
```

is for source 8. The phase space file rotates about the isocenter at (0,0,0). The centre of the phase-space is 80 cm from the isocenter and the collimator angle is 90 degrees. The -1.0 indicates there is only one group of angles. The 1 at the start of the second line tells us that phi is fixed (at 90 degrees) and that theta is varying, in this case in 100 discrete steps between 0 and 356.4 degrees with equal intensity at each angle. In this example, i_dbs is set to 0, indicating that either directional bremsstrahlung splitting (DBS) was not used in the BEAMnrc simulation that generated this source or else the user does not wish to reject high-weight ("fat") photons created by DBS.

For more information on how to use the inputs i_dbs, r_dbs, ssd_dbs and z_dbs see section 4.4 on source 2 above.

Similar to source 2, source 8 automatically handles IAEA format phase space sources with Z-position scored for each particle (3-D phase space).

4.10   isource = 9: BEAM Treatment Head Simulation Incident from Any Direction

This source uses particles sampled from a BEAM simulation running concurrently with the DOSXYZ simulation. The BEAM accelerator code must be compiled as a shared library (existing in directory $EGS_HOME/bin/config, where config is the name of your configuration) and must be supplied with its own input file and pegs data file. More details about this are given below. Source particles for DOSXYZ are then sampled from what would be

the scoring plane during a normal run of the BEAM accelerator. Thus, this source is similar to `isource`=2 (full phase space file) without the need to store a phase space file.

Note that if you are running on Unix/Linux you must have a working C/C++ compiler to use this source and the file `$HEN_HOUSE/specs/config.conf` (*i.e.* file `$EGS_CONFIG`) must have the variable `BEAMLIB_OBJECTS` set to `$(HEN_HOUSE)lib/$(my_machine)/load_beamlib.o` and
`BEAMLIB_EXTRA_LIBS` set to $-ldl$. If you have installed EGSnrcMP on a Unix/Linux system with a working C/C++ compiler, the installation will automatically compile the C code `$HEN_HOUSE/cutils/load_beamlib.c` to create `load_beamlib.o`, and `BEAMLIB_OBJECTS` and `BEAMLIB_EXTRA_LIBS` will automatically be set to their proper values. Otherwise `BEAMLIB_OBJECTS` and `BEAMLIB_EXTRA_LIBS` will remain undefined (ie definitions left blank).

If you are running on Windows, then there is no requirement to have a working C/C++ compiler to use this source. In this case, the installation automatically sets `BEAMLIB_OBJECTS` in `$HEN_HOUSE/specs/config.conf` to `$(HEN_HOUSE)lib/$(my_machine)/load_beamlib.obj`, where `load_beamlib.obj` is a precompiled version of `load_beamlib.o` included with the installation. `BEAMLIB_EXTRA_LIBS` does not need to be defined and is left blank. See the BEAMnrc Manual[12] for more information about `config.conf` files.

Similar to `isource = 2`, you can select particles from the BEAM simulation to use based on their charge and/or `LATCH` values. You can also select the maximum size of the BEAM field considered using the `BEAM_SIZE` input (discussed in more detail in its own section below).

Input parameters for source 9 are:

> **iqin** Charge of the incident beam (-1: electron, 0: photon, 1: positron, 2: all particle types)
>
> **isource = 9**
>
> **xiso/yiso/ziso** x-, y-, z-coordinates of the isocenter (normally located within the phantom)
>
> **theta** Same as for isource=2. Recall that, for a beam coming down from the top of the phantom, theta=180 degrees.
>
> **phi** Same as for isource=2.
>
> **dsource** Absolute distance from the isocenter to the centre of the source plane, which is, by definition, the origin of the scoring plane in the BEAM simulation.
>
> **phicol** Angle by which the source is rotated about the BEAM central axis. `phicol` is determined for theta=phi=0. The positive sense of rotation is counterclockwise for theta=0. If theta=180 degrees (ie beam coming down from the top), then `phicol` must be set to 180 degrees to have the BEAM x-y coordinates match the DOSXYZ x-y coordinates.
>
> **i_dbs** Set to 1 if directional bremsstrahlung splitting (DBS) is being used in the BEAM simulation **AND** you wish to reject fat photons falling outside the DBS splitting field.

4   SOURCE ROUTINES

This is recommended so that the fat photons do not compromise dose statistics. Note that, with `isource`=9, we have direct access to the BEAM stack variable (`IPHAT`) indicating whether a photon is fat or not, thus we do not need to reconstruct the DBS splitting field using the additional inputs `r_dbs`, `ssd_dbs` and `z_dbs` required in source 2.

`e_split` Number of times to split charged particles as soon as they enter the phantom geometry. Split particles have their weight reduced by a factor of 1/`e_split`. This is only used in conjunction with photon splitting (`n_split`, see Section 8.16) and prevents higher-weight contaminant electrons from compromising statistics in photon beams. For maximum efficiency, it is suggested that you set `e_split`=`n_split`, the photon splitting number.

In addition to the above inputs, the user must input the following information, all on one line, separated by commas. This line of input follows the `enflag` input line, *i.e.* the second line after the above input.

`the_beam_code` The name of the BEAM accelerator simulation (ie `BEAM_accelname`). This code must have been compiled as a shared library that exists in your `$EGS_HOME/bin/config` directory. More information on compiling a BEAM code as a library is given below.

`the_input_file` The input file to use for the BEAM simulation (no `.egsinp` extension). This file must exist in your `$EGS_HOME/BEAM_accelname` directory (ie the accelerator directory). In the BEAM input file, you must define a single scoring plane in the accelerator, where the source particles will be sampled, and the BEAM input parameter `IO_OPT` must be set to output a phase space file at this scoring plane. See the BEAMnrc Manual[12] for more information about scoring planes and `IO_OPT`. `the_input_file` will define a value of `NCASE` (no. of histories) for the BEAM simulation, but this is ignored and the BEAM simulation will always run until `NCASE` for the DOSXYZ simulation is reached.

`the_pegs_file` The pegs data to be used in the BEAM simulation (no `.pegs4dat` extension). This file must exist in either `$HEN_HOUSE/pegs4/data` or your `$EGS_HOME/pegs4/data` directory.

A graphical representation of source 9 is similar to that of source 2 shown in Figure 3. In both sources, the "source plane" is the scoring plane in the BEAM simulation, but in source 9 particles are sampled as soon as they cross the scoring plane rather than stored in a phase space file for later use. Note that the BEAM central axis points from the origin of the source plane to the isocenter.

Similar to source 2, `theta`, `phi` and `dsource` can be set to place the source plane anywhere inside the phantom or the surrounding region; the medium and the thickness of the surrounding region are input by the user. Particles initially falling outside both the phantom and surrounding region are terminated immediately. A particle history is terminated if it is determined that the particle will not make it to the phantom surface (the particle loses all its energy in the surrounding region or it escapes through the outer boundaries of the surrounding region).

The user must also set enflag = 2 or 3 the and input the medium number and the thickness for the surrounding region. These inputs are described in sections 5 and 6 below.

4.10.1   Compiling a BEAM code as a shared library

In order to compile your BEAM accelerator code, BEAM_accelname, as a shared library, you must already have built the code and created your directory $EGS_HOME/BEAM_accelname using the beam_build tool. Building a BEAM code is discussed in detail in the BEAM Manual[12]. Of course, if you have already been running BEAM_accelname as a regular BEAM simulation, then the code will have already been built and $EGS_HOME/BEAM_accelname will exist. To compile the code as a shared library, go into your directory $EGS_HOME/BEAM_accelname and type:

make library

If you are using a Unix/Linux system, this will create the library libBEAM_accelname.so. On a Windows system, the library will be named BEAM_accelname.dll. The library will automatically be copied to your directory $EGS_HOME/bin/config, where config is the name of your configuration (e.g. gcc, win2k, pgf77, etc). See the BEAM Manual for the differences between the codes concatenated to create a BEAM library and those used for a standard BEAM accelerator simulation.

In previous versions of BEAMnrc, the library libg2c.a was required when compiling shared library sources on Unix/Linux to avoid confusion of Fortran units between DOSXYZnrc (the driving code) and BEAMnrc. Recently, however, the opening of files in BEAMnrc has been recoded so that only available Fortran units are used. This solves the problem of confusion between Fortran units and eliminates the need for the libg2c.a library (which caused problems with the gfortran compiler).

Recall that when entering the input parameter the_beam_code, specifying the BEAM simulation to use, you simply use BEAM_accelname, omitting the lib prefix (in the case of a Unix/Linux library) and the .so or .dll extension of the library.

4.10.2   Efficiency of BEAMnrc simulation source vs. a phase space source

A BEAMnrc source has the obvious advantage over a phase space source in that intermediate phase space data need not be stored. For many calculations with reasonable precision, this can save tens of GBytes of disk storage space. In general, the tradeoff will be a reduced simulation efficiency, due to the extra time required to perform a full accelerator simulation to generate source particles.

Recent research[16] has shown, however, that by using variance reduction techniques in the BEAMnrc simulation source and in the DOSXYZnrc calculation, the efficiencies of photon beam dose calculations using BEAMnrc simulation sources can be maximized so that they are only 3–13% (depending on beam energy, field size and phantom voxel size) lower than the peak efficiencies with the equivalent phase space sources. The variance reduction techniques required are directional bremsstrahlung splitting (DBS–see Ref[17] and Section 6.3.4 in the

BEAMnrc Manual) to maximize the efficiency of photon production in the BEAMnrc simulation source in conjunction with photon splitting (n_split–see Section 8.16) to maximize the efficiency of the DOSXYZnrc calculation. Efficiencies quoted for phase space sources include the time required to transport particles through the accelerator jaws (from "fixed" phase space data collected above the jaws) to generate the source, but even if this time is omitted, the peak efficiencies with BEAMnrc simulation sources are only 5–30% lower than those with phase space sources.

4.11   isource = 10: Full BEAMnrc Treatment Head Simulation Incident from Multiple Directions

This source is similar to source 9, except that it is incident from multiple, user-defined directions (theta-phi pairs).

Inputs for source 10 are:

iqin Charge of the incident beam (-1: electron, 0: photon, 1: positron, 2: all particle types)

isource = 10

xiso/yiso/ziso x-, y-, z-coordinates of the isocenter (normally located within the phantom)

nang The number of incident theta-phi pairs or, if negative, then abs(nang) is the number of groups of incident theta-phi pairs, where, within a group, all theta-phi pairs have equal probability. Same as for source 8 (see Section 4.9 above for more details).

dsource Absolute distance from the isocenter to the centre of the source plane. Same as for source 9 (Section 4.10).

phicol Angle by which the source is rotated about the BEAM central axis. Same as for source 9 (Section 4.10).

i_dbs Set to 1 if directional bremsstrahlung splitting (DBS) is being used in the BEAM simulation **AND** you wish to reject fat photons falling outside the DBS splitting field so that they do not compromise dose statistics. Same as for source 9 (Section 4.10).

e_split Number of times to split charged particles as soon as they enter the phantom geometry. This is only used in conjunction with photon splitting (n_split>1). Same as for source 9 (Section 4.10).

the_beam_code The name of the BEAM accelerator simulation (ie BEAM_accelname). Same as for source 9 (Section 4.10).

the_input_file The input file to use for the BEAM simulation (no .egsinp extension). Same as for source 9.

the_pegs_file The pegs data to be used in the BEAM simulation (no .pegs4dat extension). Same as for source 9.

For an example of how the theta-phi pairs are specified, see the description at the bottom of Section 4.9 above.

Note that, similar to source 9, the user must also set enflag = 2 or 3 and input the medium number and the thickness for the surrounding region. These inputs are described in sections 5 and 6 below.

4.12   isource = 20: Synchronized phase space source

This source greatly enhances the capabilities of the phase space source incident from multiple directions (isource=8). Source 20 was developed by Lobo and Popescu[18] to allow the user to simulate continuous motion of the phase space source relative to the DOSXYZnrc phantom over specified ranges of incident directions, SSD's and isocentre coordinates. Moreover, the source allows the user to interpose a geometry, generated using either a BEAM accelerator or a non-EGSnrc code (likely simulating an MLC geometry) compiled as a shared library, between the source plane and the DOSXYZnrc phantom. In the case of a BEAM shared library, the source parameters (range of incident directions, SSD's, isocentres) can be synchronized with the field settings of any synchronized component modules (see the BEAMnrc Manual[19]).

The geometrical parameters controlling the orientation of the source plane relative to the DOSXYZnrc phantom have the same definition as those for sources 2, 8, 9 and 10 and are shown in Figure 3.

Input parameters for source 20 are:

iqin Charge of the incident beam (-1: electron, 0: photon, 1: positron, 2: all particle types)

isource = 20

nset The number of control points. Control points define the beginning and end points of ranges of incident direction angles, SSD's and isocentre coordinates over which continuous motion of the source is simulated (see below). nset must be in the range $2 \leq$ nset $\leq$ $MXANG, where $MXANG is defined in the file $EGS_HOME/dosxyznrc/dosxyznrc_user_macros.mortran.

i_dbs Set to 1 if the phase space source was generated using directional bremsstrahlung splitting (DBS) in BEAMandyou wish to reject photons directed outside the splitting field (which will all be fat). Set to 0 otherwise. Note that i_dbsis read in as a real and converted to integer.

r_dbs Radius (in cm) of the DBS splitting field in the BEAM simulation used to generate this source. Only needed ifi_dbs=1.

ssd_dbs SSD (in cm) where the DBS splitting field radius was defined in the BEAM simulation used to generate this source. Only needed if `i_dbs`= 1.

z_dbs Z value (in cm) in the BEAM simulation where this phase space source was scored. This will be at the back of a component module (CM). Only needed if `i_dbs`= 1.

e_split Number of times to split charged particles as soon as they enter the phantom geometry. Split particles have their weight reduced by a factor of 1/`e_split`. This is only used in conjunction with photon splitting (`n_split`, see Section 8.16) and prevents higher-weight contaminant electrons from compromising statistics in photon beams. For maximum efficiency, it is suggested that you set `e_split=n_split`, the photon splitting number.

i_muidx_out Set to 1 to include the fractional monitor unit index, `MU`, associated with each particle in IAEA format phase space output for particles leaving the phantom geometry. Note that phase space data is not output at all unless `i_phsp_out`, the input parameter in the main code controlling phase space output, is set to 1 or 2. If `MU` is included in the phase space output, then phase space data has a time dimension and is considered 4-D. Scoring of `MU` allows synchronization between the DOSXYZnrc simulation generating the file and any downstream simulations–such as BEAMnrc simulations with synchronized component modules–using the file as a source. See Section 8.18 for a description of DOSXYZnrc phase space output.

calflag Set to 1 to skip the calibration run through a BEAM shared library geometry between the source plane and the phantom. The number of times to recycle each source particle before proceeding to the next one, `NRCYCL`, will then not take into account the ratio of the number of particles emerging from the library geometry to the number of incident particles (survival ratio). The default is to do the calibration run if a BEAM shared library geometry is present.

For i=2,...,`nset` (the no. of control points), the following parameters are entered:

xiso(i)/yiso(i)/ziso(i) x-, y-, z-coordinates of the isocenter.

theta(i) Angle of line connecting phase space plane to isocentre relative to the +Z axis.

phi(i) Angle of line connecting phase space plane to isocentre relative to the +X axis.

phicol(i) 2D Angle of rotation of the phase space plane about its own origin.

dsource(i) The distance from the centre of the phase space plane to the isocentre except when an IAEA format phase space source is used, in which case input the distance from the primary BEAM source to the isocentre (SAD). When an IAEA phase space source is used, the distance from the phase space plane to the isocentre is calculated using the Z position of the phase space plane read in from the IAEA header file.

muIndex(i) A monitor unit index in the range [0,1] defining the fraction of the total number of incident particles delivered up to control point i. Note that `muIndex(i)` $\geq$

muIndex(i-1). Also note that for correct selection of the range of incident source parameters, muIndex(i)=0.0 and muIndex(nset)=1.0. See below for more information on how muIndex is used to sample from a range of incident source parameters.

End of inputs required for each control point.

the_shared_lib The name of the BEAM accelerator code or non-EGSnrc MLC simulation code compiled as a shared library (See below for more information). Leave this blank if there is no geometry interposed between the phase space source and the DOSXYZnrc phantom. This code must have been compiled as a shared library (*e.g.* libBEAM_accelname.so if a BEAM accelerator compiled on Unix/Linux) and exist in directory $EGS_HOME/bin/config, where config is your configuration name.

FILNAM The full name (including directory path and file extension) of the phase space source.

the_input_file Input file for the BEAM or non-EGSnrc code defining the shared library geometry interposed between the phase space source and the DOSXYZnrc phantom. For a BEAM shared library, this input file must exist in $EGS_HOME/BEAM_accelname In the case of a BEAM shared library, this file must specify phase space output at the bottom of the accelerator.

Prior to each incident particle (Note: not necessarily a statistically-independent primary history) a random number, MU_RND∈ [0,1], which can be thought of as a fractional monitor unit, is compared to the muIndex(i) of the control points to determine the incident geometrical parameters. If source 20 is being run through a BEAM shared library geometry which has synchronized component modules, such as SYNCJAWS and/or SYNCVMLC (See the BEAMnrc Users Manual[19]), then the value of MU_RND is passed to BEAM from DOSXYZnrc (more on this below). If muIndex(i-1)≤MU_RND<muIndex(i) then the parameters determining the orientation of the source plane are interpolated between control points i-1 and i using:

$$\text{PARAM} = \text{PARAM}(i-1) + \frac{[\text{PARAM}(i) - \text{PARAM}(i-1)]}{[\text{muIndex}(i) - \text{muIndex}(i-1)]} [\text{MU\_RND} - \text{muIndex}(i-1)] \quad (1)$$

Where PARAM is the value of the source parameter (xiso, yiso, ziso, theta, phi, etc) used in the simulation, and PARAM(i) and PARAM(i-1) are its values defined at control points i and i-1, respectively. Thus, source parameters are evenly distributed between control points i-1 and i.

The generation of a new value of MU_RND for each incident particle means that particles in the phase space source arising from the same primary history will be incident from different source orientations. This was implemented to ensure adequate sampling of source positions even in cases where where relatively few primary histories are represented in the phase space source (*i.e.* if the BEAM simulation used to generate the source used directional

bremsstrahlung splitting with a high splitting number). Note, however, that regardless of how spatially well-sampled a dose distribution is, its uncertainty is limited by the number of primary histories simulated.

An simple example input segment for `nset`=7 is:

```
0, 0, -9, 90, 0, 0, 15, 0
0, 0, -9, 90, 360, 0, 15, 0.1
0, 0, -5, 90, 0, 45, 15, 0.1
0, 0, -5, 90, 360, 45, 15, 0.5
0, 0, 5, 90, 0, 90, 15, 0.5
0, 0, 5, 90, 360, 90, 15, 0.9
0, 0, 9, 90, 360, 135, 15, 1.0
```

This sample input defines four distinct ranges of incident source parameters:

1. The first two control points (`muIndex(1)`=0.0, `muIndex(2)`=0.1) define 10% of incident particles with (`xiso, yiso, ziso`)=(0, 0, -9), `theta`=90° (i.e. incident from the side of the phantom), `phi` evenly distributed over [0,360^o] (i.e. circling the entire phantom in the X-Y plane), `phicol`=0°, and `dsource`=15 cm.

2. Control points 3 and 4, with `muIndex(3)`=0.1 and `muIndex(4)`=0.5, define 40% of incident particles with (`xiso, yiso, ziso`)=(0, 0, -5), `theta`=90°, `phi` evenly distributed over [0,360^o], `phicol`=45°, and `dsource`=15 cm. Note that setting `muIndex(3) = muIndex(2)` is used to define a discontinuity between this range and the previous range, since no parameters will be chosen between control points 2 and 3.

3. Control points 5 and 6, with `muIndex(5)`=0.5 and `muIndex(6)`=0.9, define another range of parameters (discontinuous with the previous one), in which 40% of incident particles have (`xiso, yiso, ziso`)=(0, 0, 5), `theta`=90°, `phi` evenly distributed over [0,360^o], `phicol`=90°, and `dsource`=15 cm.

4. Control point 7 (`muIndex(7)`=1.0) defines 10% of particles (`muIndex(7)`-`muIndex(6)`=0.1) with (`xiso, yiso, ziso`) evenly distributed over [(0, 0, 5),(0, 0, 9)], `theta`=90°, `phi` evenly distributed over [0,360^o], `phicol` evenly distributed over [90°,135°] and `dsource`=15 cm. Note that this range is continuous with the previous one.

A realistic simulation of a treatment plan may involve hundreds, or thousands, of control points. In practice these could either be extracted from a treatment plan or generated using your own code[18]. Note that a large number of control points may require you to increase the value of `$MXANG` in `$EGS_HOME/dosxyznrc/dosxyznrc_user_macros.mortran` (which defines the maximum value of `nset`) and recompile DOSXYZnrc.

It is recommended that the user set the input `i_dbs`=1 if directional bremsstrahlung splitting (DBS) was used to generate the phase space source and they wish to eliminate fat photons, directed outside the splitting field, from the DOSXYZnrc simulation. From the BEAM simulation used to generate the phase space source, the user must then input the DBS

splitting radius, `r_dbs`, the Z-position of the scoring plane, `z_dbs`, and the SSD used for DBS, `ssd_dbs`. See Section 4.4.1 for more details.

Just as with other phase space sources (sources 2, 8), source 20 requires setting the input parameter, `enflag`, to 2 or 3 (with or without `LATCH` bit filtering), that the mode of the phase space file (0–no `zlast` scoring, 2–`zlast` scoring) be input, and that the thickness and medium of the region surrounding the phantom be specified. See Section 5 for more information.

4.12.1 Phase space source incident through a shared library geometry

Using the inputs, `the_shared_lib` and `the_input_file`, the user is able to interpose a geometry defined using a code compiled as a shared library between the phase space source and the DOSXYZnrc phantom geometry. As mentioned above, the code can be a BEAM accelerator or a non-EGSnrc user code, and `the_input_file` is an input file for the code which defines the geometry parameters (e.g. a BEAM input file). The compiled shared library (*e.g.* `libBEAM_accelname.so` if the code is a BEAM accelerator and has been compiled on Unix/Linux) must exist in the `$EGS_HOME/bin/config` directory, where `config` is the name of your configuration.

The system requirements for the use of a shared library geometry are the same as those required for a BEAM treatment head source and are detailed in Section 4.10. Instructions on compiling a BEAM code as a shared library are given in Section 4.10.1.

A 2D schematic showing the differences between running source 20 through a BEAM shared library versus a non-EGSnrc code, `user_code`, compiled as a shared library is shown in Figure 6.

If a BEAM shared library is used the following conditions apply:

- The BEAM input file, `the_input_file` must define a scoring plane at the bottom of the geometry. Particle data at this plane are not actually output but are stored in an internal array for eventual DOSXYZnrc transport through the phantom.

- If the source specified in the BEAM input file is a phase space source, it must not make use of the same phase space file as DOSXYZnrc source 20. This causes a file opening conflict in some versions of Fortran and will cause the run to fail. Note that when BEAM is used to define a geometry for source 20, the BEAM source is not used, so a dummy source of any type can be specified in the BEAM input file.

- With a BEAMnrc-format phase space source, the incident Z-position (in the BEAM frame) of the phase space source is the same as the incident Z-position of the source as defined in the BEAM input file, `the_input_file`. The source 20 input, `dsource` then defines the distance from the phantom isocentre to the centre of the scoring plane.

- If an IAEA-format phase space source is used, the incident Z-position of the phase space source is read from the phase space data itself. Note that this means the user
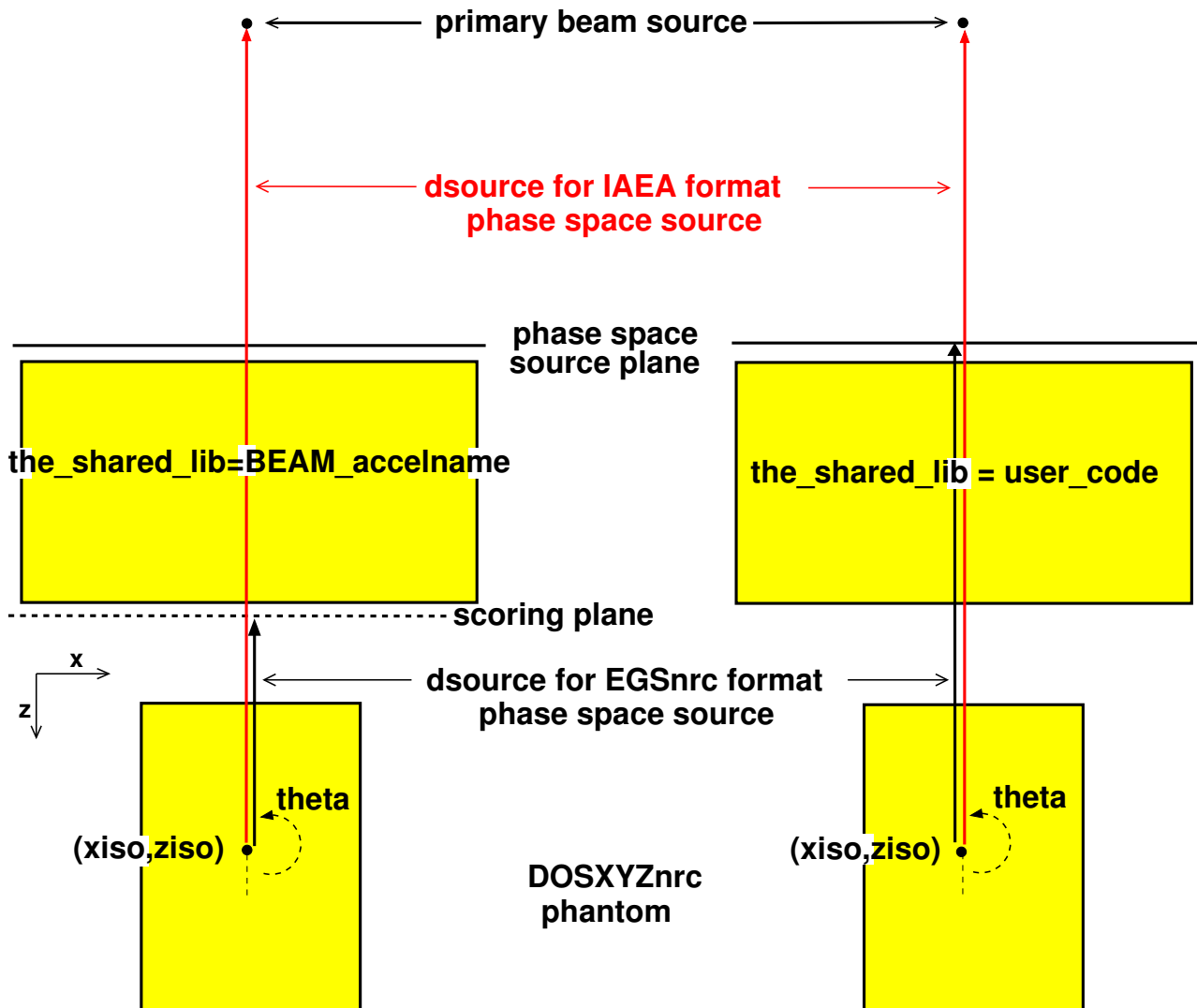
Figure 6: Source 20 incident through a shared library (BEAM or non-EGSnrc user code). When the phase space source is run through a BEAM shared library geometry (left), dsource defines the distance between the centre of the scoring plane at the bottom of the geometry and the isocentre (BEAMnrc phase space source, black arrow), or the distance between the primary source and the isocentre (IAEA phase space source, red arrow). When a non-EGSnrc code is used (right), dsource defines the distance between the phase space source and the isocentre (EGSnrc-format phase space source, black arrow) or between the primary source and the isocentre (IAEA phase space source, red arrow).

must ensure consistency between the geometry defined in BEAM and the incident Z-position of the particles. The input parameter, dsource, defines the distance from the isocentre to the primary beam source (i.e. dsource=SAD, see red arrow in figure)

- The pegs data used in the accelerator simulation is not an input variable and must be the same as that used in the DOSXYZnrc phantom.

If the phase space source is run through a non-EGSnrc shared library (compiled from user_code) the following applies:

- Phase space data below the defined geometry need not be scored.

- If a BEAMnrc-format phase space source is used, the incident Z-position (in the user_code frame) is defined as dsource. Thus, the user must ensure that the source plane is positioned correctly relative to the defined geometry.

- For an IAEA-format phase space source, the incident Z-position is read from the phase space source, and dsource defines the primary source-to-isocentre distance. Again, the user must ensure consistency between the incident Z-position and the dimensions of the geometry defined in user_code.

One issue when running source 20 through a shared library is the determination of how many times to recycle incident particles so that the phase space source does not rewind, causing loss of information about particle correlations and subsequent underestimation of dose uncertainties. As mentioned in Section 8.11 below, DOSXYZnrc automatically estimates the number of times to recycle each particle, NRCYCL, based on the number of particles in the phase space source, the number of histories requested, and the charge of particles requested. However, interactions in the shared library geometry will change the number of particles reaching the DOSXYZnrc phantom per particle incident from the phase space source. Thus, the original estimate of NRCYCL is incorrect. To overcome this when a BEAM shared library is used, an initial calibration comprising 1×10^6 particles run through the BEAM geometry only is used to determine the ratio of the number of particles emerging from the bottom of the BEAM library geometry to the number of incident particles (survival ratio). NRCYCL is then recalculated by multiplying the original estimate by 1/(survival ratio).

Depending on the shared library geometry, the calibration run can consume significant CPU time at the beginning of the simulation. Therefore, if there is a sufficient number of particles in the phase space source that the user is confident rewinding the source is not an issue, the calibration run can be skipped by setting the input, calflag, to 1.

If a non-EGSnrc shared library is used, then the user must provide a routine to estimate the survival ratio of particles passing through the geometry. This will be covered in more detail below.

When a BEAM shared library geometry includes synchronized component modules (CMs), such as SYNCJAWS, SYNCVMLC or SYNCMLCE, then the opening coordinates of these component modules (defining the field) can be synchronized with the orientation of the phase space source plane relative to the phantom. In this case the value of MU_RND, used to set the

incident source orientation (see above), is passed from DOSXYZnrc to BEAM, where it is then used to set the opening coordinates of any synchronized CMs. The user controls the synchronization between CM geometries and the incident source orientation using files (one for each synchronized CM) that associate a field (set of opening coordinates) with a range of fractional monitor units. See the BEAMnrc Users Manual[19] for more information about synchronized CMs.

Non-EGSnrc Geometry Code (`user_code`)

The coding which allows a non-EGSnrc user code compiled as a shared library to communicate with DOSXYZnrc is found in the C code, `$HEN_HOUSE/cutils/load_vculib.c`. The acronym "vcu" stands for "Virginia Commonwealth University" and is a reference to the fact that the Lobo and Popescu[18] used a simplified MLC geometry simulation developed at VCU, called `particleDMLC`[20], as a shared library geometry in simulations with source 20.

System requirements for the use of a non-EGSnrc shared library with source 20 (or 21) are the same as those for a BEAM simulation source (see Section 4.10 above). Unix/Linux systems must have a working C/C++ compiler so that `load_vculib.c` can be compiled to create the object file, `load_vculib.o`. If a working C/C++ compiler is detected on installation of BEAMnrc/DOSXYZnrc then this is done automatically. Windows systems do not require a working C/C++ compiler, since a precompiled version of `load_vculib.o` is provided. In addition, the file, `$HEN_HOUSE/specs/dosxyznrc_config.spec`, where `config` is the configuration name, must have the variable `VCULIB_OBJECTS` set to `$HEN_HOUSE/lib/config/load_vculib.o` (note that a short form for this is `$EGS_LIBDIR/load_vculib.o`) prior to compiling DOSXYZnrc. Again, this is set automatically on installation if `load_vculib.o` was created successfully.

Currently, `load_vculib.c` is optimized for calls to subroutines defined in `particleDMLC`. However, the user can make changes to this file so that it is specific for calls to their own geometry code. For illustrative purposes, the following functions must be defined in the user's geometry code, assuming they have made no changes to `load_vculib.c` or the DOSXYZnrc coding:

1. `initvcu(char *the_input_file, float *survival_ratio)` An initialization routine which reads an input file for the geometry code (`the_input_file`) and generates an estimate of the survival ratio of particles between the phase space source and the bottom of the user code geometry (usually a simulation of an MLC). The input file would likely specify the geometry parameters and would also pass the name of the phase space source to the user code. The survival ratio is used to calculate the number of times to recycle incident particles, `NRCYCL` (see above).

2. `vculib_sample(float *e, float *x, float *y, float *z, float *u, float *v, float *w, float *wt, int *iq, int *latch, int *nhist, int *more_in_container)` A subroutine used to run the geometry code and sample particles reaching the bottom of the geometry. The subroutine returns the phase space data for a particle at the bottom of the geometry (`e`=energy; `x,y,z`=particle x,y,z-position; `u,v,w`=x,y,z-direction cosines; `wt`=particle weight; `iq`=charge; `latch`=latch value), the number of primary histories run to that point, `nhist`, and an integer, `more_in_container`, which is set to

0 if the storage array for particles reaching the bottom of the geometry is empty (i.e. the geometry simulation must be run on the next call to vculib_sample).

3. vculib_finish() A subroutine called at the end of the simulation to close any files used by the geometry code, free up reserved memory, etc.

4.12.2   IAEA-format phase space sources

Similar to other phase space sources (sources 2 and 8) source 20 is implemented such that if an IAEA-format phase space source is used, the incident Z-position of a particle is automatically read directly from the phase space data. This eliminates the restriction imposed by BEAMnrc-format phase space files of all particles being incident in the same plane and, thus, allows more flexibility in terms of the phase space data that can be used.

If the IAEA phase space source includes the fractional monitor unit index, MU, associated with each particle (i.e. 4-D data), then a random value for MU, MU_RND, is not chosen at the beginning of each history, and, instead, the value of MU read from the phase space data is automatically used to set the incident source parameters (isocentre position, incident angle, etc) for the incident particle. MU is automatically scored in IAEA phase space data output by BEAMnrc simulations using synchronized component modules (CMs) with time-varying opening coordinates (see the BEAMnrc Users Manual[19]) and can also be included in IAEA phase space data output by DOSXYZnrc simultions using synchronized sources (20 or 21) by using the i_muidx_out input flag (see description of input above and Section 8.18). Thus, use of MU read from the phase space source allows synchronization of source 20 with upstream BEAMnrc simulations having synchronized CMs or upstream DOSXYZnrc simulations with synchronized sources.

4.12.3   Simulation of Spiral CT Scan

A recent publication by Kim et al[21] details a modification of Source 8 (See Section 4.9 above) in which the phase space plane can be made to rotate in phi about an isocentre which is concurrently translating in Z, thus simulating exposure in a helical pattern. Kim et al have used this to simulate dose due to a spiral CT scan, and their results compare well with measurement.

Spiral scans can be modeled equally well using Source 20 (and 21, see below) using just two control points (nset=2). An example of two control points defining a spiral scan is:

```
0, 0, 0, 90, 0, 0, 15, 0
0, 0, 10, 90, 3600, 0, 15, 1.0
```

These control points define simultaneous translation of the isocentre from z(1)=0 to z(2)=10 cm and rotation of the source plane from phi(1)=0 at z(1) through phi(2)=3600° at z(2). The angular range, phi(2)-phi(1), determines the pitch of the helix. In this example, the source rotates ten times (3600°=10×360°) over a Z translation of 10 cm, so the pitch is equal to 1. The direction of rotation is determined by the sign of phi(2)-phi(1). If this is

positive (as in this example), then rotation is clockwise about the Z-axis; if negative, rotation is counter-clockwise. Note that the X- and Y-positions of the isocentre in this example remain unchanged, and `theta` is fixed at 90°, which means the source-isocentre vector is perpendicular to the Z-axis.

The two control points in this example cover the entire range of fractional monitor units (*i.e.* `muIndex(1)`=0, `muIndex(2)`=1.0), and, thus, include all incident particles. However, using more control points to further subdivide the range, [0,1], of fractional monitor units, it would be possible to generate multiple spiral scans–having different pitch, Z-range, etc–in one simulation.

4.13   isource = 21: Synchronized BEAM Simulation Source

Source 21 defines a BEAM treatment head simulation (compiled as a shared library) source incident over multiple ranges of continuous motion with respect to angle, SSD and isocentre. The source motion can be synchronized with the settings of any synchronized component modules (CMs) in the accelerator. There is also an option to run the source through a geometry (usually MLC) defined by a non-EGSnrc user code, compiled as a shared library, placed between the treatment head and the DOSXYZnrc phantom. Source 21 was developed and contributed by Lobo and Popescu[18]. Just as source 20 represents a significant enhancement over source 8 (phase space incident from multiple angles), source 21 represents a significant enhancement over source 10 (treatment head incident from multiple angles).

The system requirements for running shared library sources are given in Section 4.10 describing source 9 (BEAM treatment head source incident from one direction). Instructions on compiling a BEAM accelerator as a shared library are also given there.

The input parameters for source 21 are:

> **iqin** Charge of the incident beam (-1: electron, 0: photon, 1: positron, 2: all particle types)
>
> **isource** = 21
>
> **nset** The number of control points defining ranges of continuous source motion. Same as for source 20 (see above). `nset` must be in the range $2 \leq$ `nset` $\leq$ `$MXANG`, where `$MXANG` is defined in the file `$EGS_HOME/dosxyznrc/dosxyznrc_user_macros.mortran`.
>
> **i_dbs** Set to 1 if directional bremsstrahlung splitting is used in the BEAMsimulation andyou wish to reject photons directed outside the splitting field (which will all be fat). Set to 0 otherwise. Note that `i_dbs`is read in as a real and converted to integer.
>
> **e_split** Number of times to split charged particles as soon as they enter the phantom geometry. Split particles have their weight reduced by a factor of $1/$`e_split`. This is only used in conjunction with photon splitting (`n_split`, see Section8.16) to prevent higher-weight contaminant electrons from compromising statistics in photon beams. For maximum efficiency, it is suggested that you set `e_split=n_split`, the photon splitting number.

i_muidx_out Set to 1 to include the fractional monitor unit index, MU, associated with each particle in IAEA format phase space output for particles leaving the phantom geometry. Note that phase space data is not output at all unless **i_phsp_out**, the input parameter in the main code controlling phase space output, is set to 1 or 2. If MU is included in the phase space output, then phase space data has a time dimension and is considered 4-D. Scoring of MU allows synchronization between the DOSXYZnrc simulation generating the file and any downstream simulations–such as BEAMnrc simulations with synchronized component modules–using the file as a source. See Section 8.18 for a description of DOSXYZnrc phase space output.

For control points i=1,2,...,**nset**, the following parameters defining the source orientation are entered:

xiso(i)/yiso(i)/ziso(i) x-, y-, z-coordinates of the phantom isocenter.

theta(i) Angle of normal connecting origin of the BEAM scoring plane to the isocentre.

phi(i) Angle of normal connecting the origin of the BEAM scoring plane to isocentre relative to the +X axis.

phicol(i) 2D Angle of rotation of the BEAM scoring plane about its own origin.

dsource(i) The length of the normal from the origin of the BEAM scoring plane to the isocentre.

muIndex(i) A monitor unit index in the range [0,1] defining the fraction of the total number of incident primary histories delivered up to control point i. This is slightly different from source 20, where **muIndex(i)** defines the fraction of incident particles up to control point i. Thus, for source 21, **muIndex(i)** is actually related to the fractional monitor units delivered by the BEAM simulation. Note that **muIndex(i)** $\geq$ **muIndex(i-1)**. Also note that for correct selection of the range of incident source parameters, **muIndex(1)=0.0** and **muIndex(nset)=1.0**.

This is the end of inputs required for each control point.

the_beam_code The name of the BEAM accelerator code (*i.e.* BEAM_accelname) used to run the treatment head simulation. This must have been compiled as shared library (*e.g.* libBEAM_accelname.so on a Linux/Unix system) existing in your $EGS_HOME/bin/config directory.

the_input_file The input file for the BEAM treatment head simulation. This must exist in directory $EGS_HOME/BEAM_accelname and must specify a single scoring plane (usually at the bottom of the accelerator).

the_pegs_file The PEGS data to be used in the BEAM treatment head simulation.

4   SOURCE ROUTINES

the_vcu_code This optional input is the name of a non-EGSnrc user code, compiled as a shared library, that can be used to define a geometry through which particles are run between the bottom of the treatment head and the DOSXYZnrc phantom. The shared library must exist in directory `$EGS_HOME/bin/config`

the_vcu_input_file Input file for the_vcu_code defining the geometry interposed between the treatment head simulation and the phantom.

For each primary history in the treatment head simulation a random fractional monitor unit index, MU_RND$\in$ [0,1], is chosen. MU_RND is either generated by DOSXZYnrc or, if there are synchronized CMs in the treatment head simulation also undergoing motion, it is passed to DOSXYZnrc from the BEAM simulation. More details on the use of synchronized CMs in the BEAM simulation are given below. For all particles associated with a given value of MU_RND, the control points, i-1 and i, within which the source position falls are those for which muIndex(i-1)$\leq$MU_RND<muIndex(i), and the parameters determining the source orientation are calculated using Equation (1) (see Section 4.12 above). Note that, unlike source 20, MU_RND is chosen only for each primary history, not for each incident particle. Thus, the muIndex(i) for the control points are traceable to actual monitor units in the treatment head.

Recall that the source plane is actually the phase space scoring plane in the BEAM treatment head, defined in the_input_file. Phase space data is not output at this plane but is stored in an internal array for use by DOSXYZnrc over the course of the simulation.

Similar to source 9 (See section 4.10), since the BEAM simulation is running concurrently with DOSXYZnrc, if directional bremsstrahlung splitting (DBS) is being used in BEAM, information about whether or not a particle has been split or has survived russian roulette ("fat" particles) is available without having to reconstruct the DBS splitting field after the fact. Thus, fat photons can be rejected from the DOSXYZnrc simulation simply by setting i_dbs=1 and the additional inputs required to reject fat photons from phase space sources, r_dbs, ssd_dbs, z_dbs (See Section 4.12) are not required.

As with other phase space and BEAM simulation sources, the input enflag must be set to 2 or 3 to indicate whether LATCH bit filtering is to be used, the mode of the data (with or without zlast) being read from the BEAM simulation must be indicated, and the medium and thickness of the medium surrounding the phantom (dsurround) must be input. For more information on these inputs, see Section 5.

4.13.1   Use of synchronized CMs in the BEAM simulation

If the BEAM treatment head simulation includes synchronized component modules, such as SYNCJAWS, SYNCVMLC, and SYNCMLCE, and the user is modeling dynamically-changing field coordinates with them, then MU_RND for each primary history is passed from BEAM to DOSXYZnrc, and the user can synchronize the orientation of the source plane with the field coordinates through the muIndex(i) of the individual control points. Synchronized CMs are also synchronized with each other. For more information on the synchronized CMs, see the BEAMnrc Users Manual[19].

Figure 7 below shows visually how the incident source orientation may be synchronized with the CM settings.
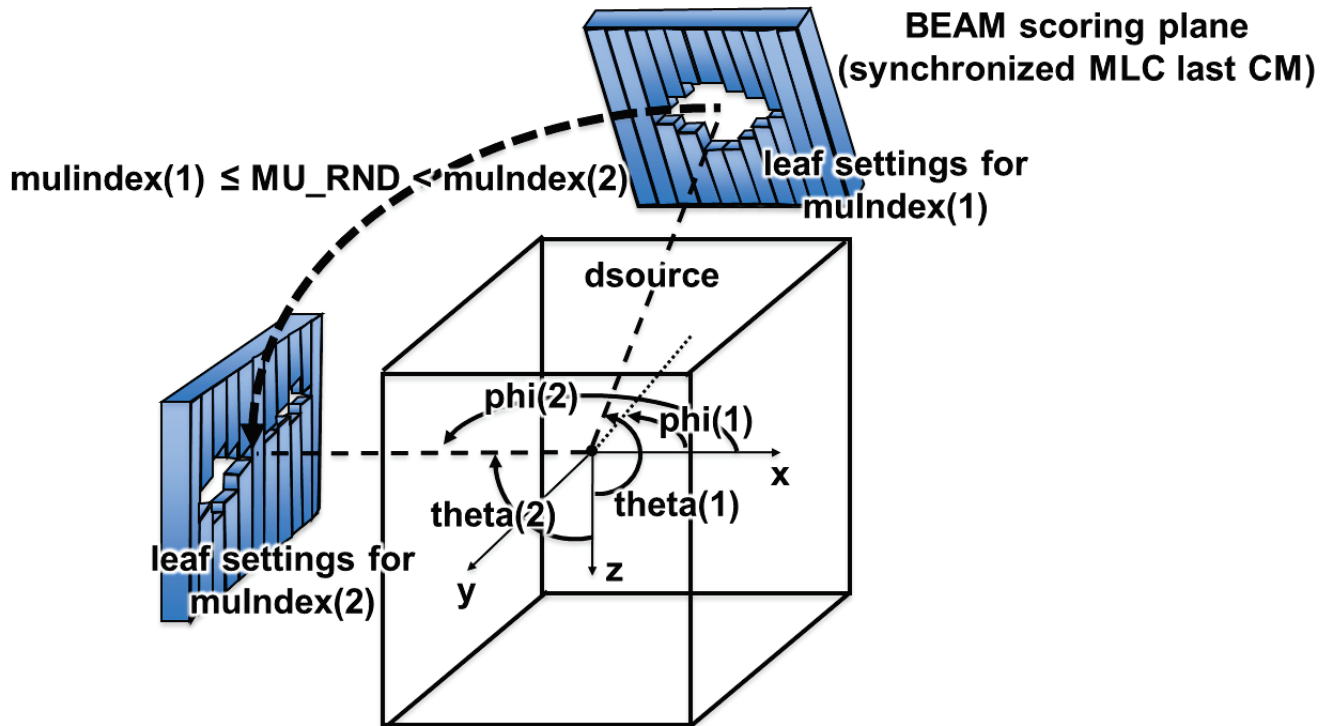
Figure 7: Source 21, including a synchronized CM, incident on a DOSXYZnrc phantom. The figure indicates that the final CM in the treatment head simulation is a synchronized MLC (SYNCVMLC, SYNCMLCE or SYNCHDMLC). Two control points for the source plane, having different `muIndex`, `theta` and `phi`, are indicated. Different MLC leaf settings are also associated with the values of `muIndex` (the user sets this up in a file read by the synchronized CM). Continuous motion of the source between (`theta(1)`, `phi(1)`) and (`theta(2)`, `phi(2)`), along with dynamic motion of the MLC leaves, is simulated for values of `MU_RND` with `muIndex(1)`$\leq$`MU_RND`<`muIndex(2)`. Continous motion in `dource`, isocentre position (`xiso`, `yiso`, `ziso`), and the angle of rotation of the source plane in the plane (`phicol`) can also be simulated.

4.13.2   Running the source through a non-EGSnrc shared library

Using the inputs `the_vcu_code` and `the_vcu_input_file` the user has the option to interpose another geometry (usually an MLC) between the source plane and the DOSXYZnrc phantom. In general, `the_vcu_code` is a non-EGSnrc user code. Lobo and Popescu[18] have used a version of the code particleDMLC[20], developed at Virginia Commonwealth University (hence "vcu" in the variable names), to place a simplified MLC simulation between the source and the phantom.

4   SOURCE ROUTINES

A detailed discussion of potential user codes for creating simplified MLCs is beyond the scope of this manual. However, a summary of the subroutines that must exist in `the_vcu_code` and the parameters required to communicate with DOSXYZnrc is given in the description of source 20 (Section 4.12) above.

Note that with source 21, `dsource` always defines the Z-position, in the BEAM frame, of the source plane (scoring plane in the BEAM simulation), and it is up to the user to ensure that the source is positioned correctly relative to the geometry defined by `the_vcu_code`.

5   Other Source-Related Inputs

After the inputs described above, there is another record input with several other variables related mostly to the source.

5.1   `enflag`

The setting of `enflag` determines whether the source is monoenergetic or has an energy spectrum, or, if the source is a phase space file, whether or not a dose component is to be calculated. The possible settings of `enflag` are:

- $= 0$ (default) for a monoenergetic source (`isource`=0,1,3,6).

- $= 1$ for an energy spectrum input (`isource`=0,1,3,6).

- $= 2$ for a phase space source (`isource`=2) or full BEAM simulation source (`isource`=9), no dose component calculation.

- $= 3$ for a phase space source (`isource`=2) or full BEAM simulation source (`isource`=9), dose component calculation using a bit filter (see section 7 below).

- $= 4$ for a beam characterization model source (`isource`=4).

5.2   `mode`

`mode` only has meaning for a phase space source (`isource`=2). However, an explicit value for `mode` must exist in the input file for a full BEAM simulation source (`isource`=9) or beam characterization model source (`isource`=4), since `isource`=4 and 9 make use of `medsur`, `dflag` and `dsurround`, which occur after `mode` on the same line. The possible settings of `mode` are:

- $= 0$ (default) the phase space source does not contain `ZLAST` (for photons, Z of last site of interaction; for electrons, Z where electron or its ancestor was set in motion by a photon) and, thus, has 7 variables/record.

- $= 2$ the phase space source contains `ZLAST` and, thus, has 8 variables/record.

Note that DOSXYZnrc does not make use of `ZLAST`, but it needs to know whether `ZLAST` is in the file or not so that the appropriate number of variables/record can be read. Thus, if a phase space file is only going to be used as a source for a DOSXYZnrc calculation, you can save space by leaving `ZLAST` out of the file. For more information about `ZLAST`, see the BEAMnrc Users Manual[19].

5.3   `medsur`

`medsur` is the medium number for the region surrounding the phantom as defined by `dsurround` (see below). `medsur` is only input for phase space, full BEAM simulation or beam characterization model sources (`isource`=2, 4, 8 or 9). The default setting of `medsur` is 0, indicating the surrounding region is vacuum. However, the user may set this to correspond to any of the media that they have defined at the top of the DOSXYZnrc input file. In many cases, the user may wish to fill this region with air and have particles transported properly through air before reaching the phantom.

5.4   `dsurround` **and** `dflag`

`dsurround` and `dflag` are required for phase space, full BEAM simulation and beam characterization model sources `isource`=2, 4, 8, 9, 10, 20 or 21. These inputs define the dimensions of the region surrounding the phantom (ie filled with the medium defined by `medsur`). Note that `dsurround` is a 4-dimensional array with `dsurround(1)` occurring before `dflag` on the input line and `dsurround(2...4)`, if required, occurring after `dflag`.

The possible settings for `dflag` and their relation to `dsurround` are as follows:

`dflag=0` (default) `dsurround(1)` defines the thickness if the surrounding region (in cm) on all sides of the phantom, and the user need not input values for `dsurround(2...4)`. `dsurround(1)` defaults to 50 cm if it is set ≤0.

`dflag=1` means that `dsurround(1)` defines the thickness of the surrounding region in the ±x directions, `dsurround(2)` is the thickness of the surrounding region in the ±y directions, `dsurround(3)` is the thickness in the +z direction (bottom of the phantom) and `dsurround(4)` is the thickness in the −z direction (top of the phantom). `dsurround(1...4)` default to 0.

Use of `dflag=1` with `dsurround(1...4)` can save substantial amounts of computing time if the user is only interested in the dose along a certain axis or in a specific slice of a DOSXYZnrc phantom (see section 14 below).

5.5   `ein`

`ein` is the kinetic energy of a monoenergetic incident beam in MeV. It defaults to 1.25 MeV if `ein` is set ≤ 0. Note that `ein` is only required if `isource`=0,1,3,6 and `enflag`=0.

5.6   FILNAM

FILNAM is the file name (with extension) of:

incident beam energy spectrum if `isource`=0,1,3 or 6 and `enflag`=1. In this case, FILNAM has a specific format which corresponds to the ensrc format used in the original EGSnrc system (see `$HEN_HOUSE/spectra` for a large number of example spectra):

ENSRC.V5 File 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 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)

phase space source: if `isource`=2 or 8 and `enflag`=2 or 3. See Section 4.4 for more details.

beam characterization model: if `isource`=4 and `enflag`=4.

the_beam_code, the_input_file, the_pegs_file: if `isource`=9 or 10 and `enflag`=2 or 3. These are the names of the BEAM accelerator code being used as a source, the input file for the BEAM simulation, and the pegs data for the BEAM simulation respectively. See section 4.10 for more details.

5.7   IOUTSP

IOUTSP is only required with an incident energy spectrum (`enflag`=1). The possible settings of IOUTSP are:

= 0 (default) no output summary of incident energy spectrum in `.egslst`.

= 1 output a summary of the incident energy spectrum to the `.egslst` file.

6   Phase Space Sources

The phase-space files output by BEAMnrc and used by DOSXYZnrc for sources 2 and 8 are described in detail in section 7 of the BEAMnrc User's Manual. [12].

The phase-space files are binary files opened with ACCESS = 'direct' and FORM = 'unformatted'. Because they are binary files, and because there are two common byte orders used for binary files by different systems (eg PCs and DEC machines use one form and SUNs and SGIs use the other), files written by one system may not be compatible with another system. The utility `readphsp` found on `$OMEGA_HOME/progs/readphsp` can convert files from one format to the other (see the description in the BEAMnrc User's Manual. [12].)

There are also two types of phase-space files determined by how many variables they contain. The shorter format is called 'MODE0' and ZLAST is not scored and the longer format is 'MODE2' when ZLAST is present. Since DOSXYZnrc makes no use of the ZLAST variable, it saves space to use MODE0 files for use with DOSXYZnrc. 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 energy of any particle in the file, minimum energy of electrons, and minimum energy of photons.

Phase space files generated by BEAMnrc use negative E (energy) to mark the first particle scored by each new primary history. If the file is used as a source, then the negative E marker allows scored quantities (ie energy deposited) to be grouped according to primary history. This ensures that uncertainties estimated account for the correlations between incident particles in a phase space source [10]. If an old phase space file without negative E markers is used as a source, then scored quantities will be grouped according to incident particle instead of primary history. DOSXYZnrc will then output a warning that uncertainty may be underestimated because correlations between incident particles could not be taken into account. This is not a cause for concern, because we have found [10] that in the cases we studied, the underestimate is not significant.

When using a phase space source, if the entire phase space file is read before the requested number of histories is run, then DOSXYZnrc restarts the phase space file from the beginning. For the reasons discussed in section 8.11, page 83, this is not a desirable occurrence and one should strive to avoid this by using the recycling feature (which also saves on reading time).

6.1   IAEA-format Phase Space Sources

In addition to the standard BEAMnrc phase space files described above, DOSXYZnrc can also use IAEA-format phase space data as a source. This allows the user to make use of IAEA's online data base of accelerator phase space data at:
www-nds.iaea.org/phsp/phsp.htmlx

Note that this functionality requires that EGSnrc/DOSXYZnrc be installed on a machine with a working C++ compiler. This is detected automatically during EGSnrc installation and, if a working C++ compiler is found, the library of IAEA phase space handling routines is compiled.

Phase space data in IAEA format comprises both a header (extension `.IAEAheader`) file and a phase space data (extension `.IAEAphsp`) file. When using IAEA phase space data as a source, only the full name and directory path of the `.IAEAphsp` file needs to be specified (See Section 4.4). The `.IAEAheader` file is assumed to be in the same directory.

Note that when IAEA phase space sources are used the incident Z-position of each particle is available and is automatically read, either from the header file in the case of planar phase space data (such as that scored by BEAMnrc), or from the phase space data itself if it is scored in 3-D (such as that scored by DOSXYZnrc). The ability to handle 3-D phase space data greatly enhances the flexibility of DOSXYZnrc and allows it to use non-planar data supplied by manufacturers, such as that supplied by Varian for their TrueBeam accelerators. In addition, IAEA phase space files generated by BEAMnrc simulations with synchronized component modules (CMs) and by DOSXYZnrc simulations using synchronized sources (20 or 21) may include the fractional monitor unit (`MU`) associated with each particle. This is automatically read in and used by DOSXYZnrc's synchronized phase space source (source 20) allowing synchronization between the DOSXYZnrc simulation and the upstream simulation that generated the phase space file. See Sections 4.4.2 and 4.12.1 for more information.

Form more information on the format of IAEA phase space data, see the BEAMnrc Manual and IAEA Report INDC(NDS)-0484[22].

7 Calculating Dose Components with DOSXYZnrc

7.1 Bit Settings

The `LATCH` variable, associated with each particle in a BEAMnrc simulation, is a 32-bit variable used to track the particle's history. It is discussed in detail in the BEAMnrc User's Manual [12].

Each bit in `LATCH` is designated as follows with bit 0 being the lowest value bit:

bit 0 Set to 1 if a photon is created by a bremsstrahlung event or an electron is created by a bremsstrahlung photon; 0 otherwise

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

bits 24-28 Stores `IREGION_TO_BIT` (as a binary number) of the region in which a secondary particle is created; if these bits are all 0, the particle is a primary particle

bits 29-30 Store the charge of a particle at the time `LATCH` is output to a phase-space file.

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.

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

When used with a phase space source, DOSXYZnrc has the capability of selecting from the phase space file only those particles that have user-specified bits (called a "bit filter") set in their `LATCH` variables. Thus, DOSXYZnrc will score only the dose component arising from these user-selected particles. For example, in the BEAMnrc simulation, bit 10 may be associated with a square applicator, so any particle in the phase space file that has been in the applicator will have bit 10 set in `LATCH`. When using this phase space file as a source in DOSXYZnrc, the user may opt to use only those particles that have bit 10 set, thus scoring the dose component from particles that have been in the applicator.

In general, a dose component can be selected 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, and any combination of these.

7.2   Input for Dose Component Calculations

In order to enable dose component calculations with a phase space source or full BEAM simulation source (`isource=2`, 8 or 9), the variable `enflag` must be set to 3. The user then specifies which type of bit filter (`I_BIT_FILTER`) to use. Below is a description of the 4 types of bit filters available and the inputs associated with each.

`I_BIT_FILTER=0` This is an inclusive/exclusive bit filter. On the same line as `I_BIT_FILTER`, the user inputs the integers `NBIT1` and `NBIT2`. `NBIT1` is the number of bits to include and `NBIT2` is the number of bits to exclude. Restriction is that $0 \leq$`NBIT1`+`NBIT2`≤ 29. Both `NBIT1` and `NBIT2` can be set to zero. On the next line, the user inputs `BIT(I)` (I=1,NBIT1), the bits to be included, and on the following line `BIT(I)` (I=NBIT1+1,NBIT1+NBIT2), the bits to be excluded. If any of the first set of `NBIT1` bits are set and none of the second set of `NBIT2` bits are set in the particle's `LATCH` variable, the particle is used in the simulation.

`I_BIT_FILTER=1` This is an exclusive bit filter. On the same line as `I_BIT_FILTER`, the user inputs the integer `NBIT1`, the number of bits to be excluded ($0 \leq$`NBIT1`≤ 29). `NBIT2` is not relevant for this filter and is automatically set to 0. On the next line, the user inputs, `BIT(I)` (I=1,NBIT1), the bits to be excluded. If any of these `NBIT1` bits are set in the particle's `LATCH` variable, the particle is NOT used in the simulation.

`I_BIT_FILTER=2` An inclusive region-of-origin filter. The user inputs `NBIT1` on the same line as `I_BIT_FILTER`, where `NBIT1` is the number of regions of origin to be included. Since regions of origin are distinguished only by their values of `IREGION_TO_BIT`, `NBIT1` can also be seen as the number of distinct values of `IREGION_TO_BIT` to be included. The restriction on `NBIT1` is $0 \leq$`NBIT1`≤ 24. `NBIT2` is not relevant for this filter and is automatically set to 0. On the next line, the user inputs `IREGION_TO_BIT(I)` (I=1,NBIT1),

the IREGION_TO_BIT values of the NBIT1 regions of origin to be included. If the particle originated in any one of these NBIT1 regions, then it will be used in the simulation. Primary particles will be included if IREGION_TO_BIT=0 is one of the NBIT1 values of IREGION_TO_BIT to include.

I_BIT_FILTER=3 An exclusive region-of-origin filter. The user inputs NBIT1 ($0 \leq$ NBIT1 ≤ 24), the number of regions of origin to be excluded, on the same line as I_BIT_FILTER and ignores NBIT2, since it is not relevant for this filter. On the next line the user inputs IREGION_TO_BIT(I) (I=1,NBIT1), the IREGION_TO_BIT values of the regions of origin to be excluded. If a particle originated in any one of these regions, then it is NOT used in the simulation. Note that primary particles will be excluded if IREGION_TO_BIT=0 is one of the NBIT1 values of IREGION_TO_BIT to exclude.

Note that since only one bit filter can be input per simulation, only one dose component may be calculated at a time using DOSXYZnrc. This is because of the large memory requirements for the dose scoring arrays.

8   Other Input Variables

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

8.1   IPHANT

If IPHANT is set to 1, DOSXYZnrc outputs phantom data to a .egsphant file. This file has the same format as the .egsphant file that ctcreate outputs for CT data (see section 16.6) and can be read by dosxyz_show[23], along with the .3ddose dose output file to display isodose contours in the phantom. Note that this option is redundant (and, therefore, unavailable) when using a CT phantom because it is an input file from ctcreate).

8.2   MAX20

When MAX20 is set to 1, a summary of the maximum 20 doses in the phantom is output to the screen (log file) and list file. This summary includes an output of the 20 maximum doses with their fractional uncertainties and coordinates, the average of the 20 maximum doses, the average fractional uncertainty of these doses (not the uncertainty of the average), the average fractional uncertainties of all doses > 50% of the maximum dose, and the average absolute uncertainty of all doses > 50% of the maximum dose as a fraction of the maximum dose.

When not in CT phantom mode, MAX20 is the last variable on the line(s) specifying the regions for which dose will be output (ie record 9). If MAX20 is set to 1 on ANY ONE of these lines (including the line of zeros indicating the end of this record) then the summary of the 20 maximum doses will be output.

In CT phantom mode, MAX20 is input on the same line as zeroairdose and doseprint.

This option is useful for timing/efficiency studies.

8.3 `zeroairdose`

`zeroairdose` is only used in CT phantom mode. Setting `zeroairdose` to 1 will set all dose estimates in voxels with density < 0.044 g/cm^3 to zero in the `.3ddose` file. This has the effect of zeroing dose in voxels filled with air (see the default CT ramp in Figure 10–note that "AIR" has a density ranging from 0.001-0.044 g/cm^3). Air dose will not be zeroed in the `.egslst` file, this file will continue to report the precise estimated dose. `zeroairdose` serves as a dose visualisation parameter for CT phantom simulations, since, in general, the user is only interested in seeing the dose within the patient, not in the surrounding air.

8.4 `doseprint`

This input parameter is also only used in CT phantom mode.

`doseprint` is set to 1 if the user wants a full output of doses in all voxels of the CT phantom in the `.egslst` file. Since, in CT phantom cases, the user is usually not interested in this output and, owing to the large number of voxels in a CT phantom, it can make the `.egslst` exceptionally large, the default is to suppress this output (ie set `doseprint` to 0).

8.5 `NCASE`

`NCASE` is the number of histories to run in a simulation. Minimum value is 100. Default is 100 if `NCASE` is set < 100. The number of histories per output batch is equal to `NCASE/($NBATCH)`, where `$NBATCH` is currently set to 10.

8.6 `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 same information as 0 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 but with complete particle information output at every step; also outputs total dose in a region whenever energy is deposited there

= **3** Similar to 2 but with particle and dose information 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.`

8.7 TIMMAX

`TIMMAX` is the maximum CPU time in hours allowed for a simulation. `TIMMAX` defaults to 0.99 hrs if it is set = 0. However, this time-restriction function is not activated in the current version of DOSXYZnrc except to print a warning as it starts a batch which will exceed this limit.

8.8 INSEED1, INSEED2

These are random number seeds used to initialize the RANMAR[24, 25] or RANLUX[26, 27] random number generator. Within DOSXYZnrc, `INSEED1` is limited to the range 0<`INSEED1`≤31328 (defaults to 1802), and `INSEED2` has the range 0<`INSEED2`≤30081 (defaults to 9373). These ranges/defaults are designed for RANMAR (the default random number generator), however, if you are using RANLUX then `INSEED1` is the "luxury level" of the random number generator and must be in the range 0<`INSEED1`≤4, otherwise, it will automatically be set to the default luxury level of 1 (Note: this means that the DOSXYZnrc default value for `INSEED1` of 1802 will ultimately get reset to 1 by RANLUX).

Note that `INSEED1` and `INSEED2` 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 10000+10000 histories should generate results identical to a single run of the same simulation with 20000 histories. Also note that when running parallel jobs with DOSXYZnrc (see section 10), `INSEED2` must have a different value for each of the individual jobs that make up the simulation. This is taken care of automatically if you use DOSXYZnrc's built-in parallel processing functionality.

In order to switch from the RANMAR random number generator to RANLUX in DOSXYZnrc, go into `$EGS_HOME/dosxyznrc/Makefile` and change the line:

`RANDOM = $(EGS_SOURCEDIR)ranmar`

to:

`RANDOM = $(EGS_SOURCEDIR)ranlux`

Then recompile DOSXYZnrc.

8.9   BEAM_SIZE

BEAM_SIZE allows the user to control the maximum beam size for phase space sources (sources 2, 8, 20), BEAM simulations sources (souces 9, 10, 21) and the beam model source (source 4). BEAM_SIZE is the side of a square field in cm. The default value for BEAM_SIZE is 100 cm. Particles are discarded immediately if they are outside the square defined by BEAM_SIZE centred on the beam axis in the source scoring plane.

One should be careful with the use of this parameter because particles outside the specified field (defined by BEAM_SIZE) may have some effects on the dose distributions calculated and therefore the results may be biased with the use of BEAM_SIZE smaller than the field size of the original phase-space data. It should also be noted that this parameter cannot be used as a beam defining tool, such as a collimator, because in reality particles outside of the inner opening of any beam confining components may interact with and be scattered by the components, resulting in an increase of the particles within the specified field.

8.10   ISMOOTH

When phase-space data are used DOSXYZnrc will re-use the phase-space particles if the number of histories required by the user is greater than the number of particles stored in the phase-space file. Clearly, the dose distributions obtained with more histories have smaller statistical uncertainties. Although some histories may start with the same incident phase-space, the particle trajectories will be different because different random numbers will be used in the simulations in the phantom, resulting in different dose distributions. However, surface doses are mainly affected by the electron fluence distribution and therefore would not be improved by re-using the phase-space particles if the data file contains too few particles (i.e., the calculated doses would have small statistical uncertainties but large systematic uncertainties).

In order to reduce the systematic uncertainties due to a small data set, DOSXYZnrc can re-distribute the phase-space particles **as long as the simulated linear accelerator geometry is symmetric, and the treatment field is centred on the beam axis**. Currently, DOSXYZnrc is allowed to move a particle to 3 symmetrical positions (each with modified direction cosines). This process is accurate as long as the phase space file is symmetric with respect to the x-axis and also with respect to the y-axis. Suppose a particle is at (x,y) with (u,v) the 3 new positions are

$$(-x,y) \text{ with } (-u,v),$$
$$(x,-y) \text{ with } (u,-v),$$
$$(-x,-y) \text{ with } (-u,-v)$$

A user has two options for ISMOOTH:

  = **1** DOSXYZnrc re-distributes the phase-space particles when they are used more than once.

  = **0** (default) DOSXYZnrc re-uses the phase-space particles without changing the particle positions

See also section 8.11 which discusses recycling phase space data.

8.11 NRCYCL

NRCYCL is an essential input when using a phase space source. It determines the number of times that each particle from a phase space source is recycled (*ie*, reused each time it is read). If NRCYCL>0, then each source particle is used a total of NRCYCL $+ 1$ times before moving on to the next particle in the source. When phase space data is sparse, then particles must be re-used to obtain adequate statistics. In addition to recycling, particles may also be re-used whenever a phase space source is restarted (happens automatically when the simulation has reached the end of the source). Restarting is not recommended, however, because it may lead to underestimates of the uncertainty in the final results [10]. We recommend setting NRCYCL to a value that will ensure that the entire phase space source gets sampled (ie the simulation uses almost all the particles in the source) but prevents the source from being restarted.

If you are unsure of the correct value of NRCYCL to use, then run with NRCYCL=0 and DOSXYZnrc will automatically calculate a value of NRCYCL based on the number of histories and the number of particles (with appropriate charge) in the phase space source. There is a possibility that, even with the automatically calculated value of NRCYCL, the phase space source will restart. This may be due to particles that have been rejected because they missed the geometry, were multiple passers and/or were beyond the beam field defined by BEAM_SIZE) or else the algorithm for calculating NRCYCL may have determined that setting NRCYCL> 0 will result in the phase space source not being sampled adequately. If the source has only been restarted once and only a small fraction of it has been covered on the second pass, this is not likely to have a significant impact on the estimated uncertainties. However, if a large portion of the source is covered on the second pass, or if it is restarted more than once, we recommend re-running the simulation with a new value of NRCYCL calculated as:

$$\text{NRCYCL} = \frac{\text{NCASE}}{\left(\text{NPHSP} - (\text{NSMISS}/\text{NRCYCL}_{prev}) - \text{NOUTSIDE} - \text{NRJCT} - \text{NDBSRJCT}\right)} - 1 \qquad (2)$$

where NCASE is the number of incident histories, NPHSP is the total number of particles in the phase space file, NSMISS is the number of particles that missed the geometry in the previous run, NRCYCL$_{prev}$ is the setting of NRCYCL in the previous run, NOUTSIDE is the number of particles rejected because they were outside the field defined by BEAM_SIZE in the previous run, NRJCT is the number of particles rejected because they were multiple passers in the previous run, and NDBSRJCT is the number of photons rejected because they fall outside the directional bremsstrahlung splitting (DBS) field radius at the SSD (only if DBS was used in the BEAM simulation that generated this source AND the user has opted to reject these photons–see section 4.4 for more information about this). Note that NPHSP, NSMISS, NOUTSIDE, NRJCT and NDBSRJCT are all available from the .egslst file of the previous run. Always round your calculated value of NRCYCL up to the nearest integer.

NRCYCL is not automatically calculated if you are only using the positrons in the phase space source or if you are selecting a sub-set of the phase space source based on LATCH settings. In

these cases, the information required for an a-priori calculation of NRCYCL is not available in the header of the phase space source. We recommend setting NRCYCL manually to a "best guess" value. Then, if the source restarts, calculate NRCYCL using Equation 2 where NRJCT includes the number of particles rejected because they were multiple passers, the number of particles rejected because they were the wrong charge and the number of particles rejected because they did not have the right LATCH setting.

If the phase space source is stored on a remote disk, then using NRCYCL to avoid restarting a phase space source also has the beneficial effect of reducing network traffic because it reduces the number of times that a phase space source is accessed during a run (particle data is stored in temporary variables during recycling, not re-read from the source). Repeated accessing of a phase space source on a remote disk can slow a simulation down considerably.

NRCYCL is compatible with ISMOOTH (see section 8.10). So if ISMOOTH=1, then, during the recycling loop, the initial position and direction cosines of the particle are shifted according to the scheme outlined in the ISMOOTH subsection above.

Note that the total number of histories is always limited by NCASE. For example, if the phase space source has 1000 suitable particles and the user sets NRCYCL=9 (so each particle will be used a total of 10 times), but only sets NCASE=5000, then the simulation will only have a chance to use (and recycle) the first 500 particles before the simulation stops. In order to go through the entire phase space source, the user would have to set NCASE=10000. If NCASE>10000 then the phase space source would be restarted at least once during the run. After restarting, particle recycling continues as before.

8.12   IRESTART

The possible settings of IRESTART are:

- **= 0** (default) DOSXYZnrc initiates a new run, deleting all of the output files (.egslog, .egslst, .egsdat, *etc*) if present

- **= 1** Restart of a previous run; DOSXYZnrc opens the .egsdat from the previous run and reads:

    1. $\sum_{i=1}^{nhist} edep_i$ and $\sum_{i=1}^{nhist} edep_i^2$ for all voxels, where $edep_i$ is energy deposited by primary (non-phase space) history i and nhist is the number of primary histories in the previous run.

    2. number of histories and number of primary histories from the previous run

    3. the time taken by the previous run

    4. the state of the random number generator at the end of the previous run

    5. other data relating to particle fluence in previous run, number of electron steps, number of particles rejected from phase space source (if applicable), etc.

    After the current run is complete, dose and uncertainties are calculated using data from the current run and from the previous run [10]. Note that the number of histories

to run and the total CPU time allowed for the simulation do not include the histories and CPU time from the previous run. Also note that, currently, although a restarted simulation with, for example 100000+100000 histories will have identical dose results to a one-off simulation with 200000 histories, the uncertainty estimates on the doses may not be the same. We are currently investigating why this is so.

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

= **3** DOSXYZnrc opens the `.egsdat` file from a previous run, reads the data enumerated above and calculates doses and uncertainties; no simulation is run.

= **4** DOSXYZnrc recombines the binary `.pardose` files from parallel jobs and creates `.egslst` and `.3ddose` output files from the recombined data (see section 10 for more about parallel jobs).

8.13 IDAT

The possible settings of the variable `IDAT` are:

= **0** (default) DOSXYZnrc outputs `.egsdat` file with restart data after every batch.

= **1** DOSXYZnrc does not output a `.egsdat` file at all.

= **2** DOSXYZnrc outputs a `.egsdat` with restart data only at the end of the entire run.

For large phantoms, writing this file will take a lot of time. For production runs use 2 usually.

8.14 IREJECT

`IREJECT` is a switch for turning on charged particle range rejection. Range rejection can save simulation time by terminating particle histories immediately if they cannot reach the boundary of the current voxel with energy > `ECUT` and their current energy is less than `ESAVE_GLOBAL`. This is the same as the `IREJECT_GLOBAL` = 2 option in the BEAMnrc code as discussed in the BEAMnrc User's Manual [12].

The possible settings of `IREJECT` are:

= **0** (default) DOSXYZnrc will not perform charged particle range rejection.

= **1** DOSXYZnrc will immediately terminate the history of a charged particle and deposit its remaining energy in the current voxel if its energy is < `ESAVE_GLOBAL` (see description in next section) and if it cannot reach the nearest voxel boundary with an energy > `ECUT`.

It is found that for 5 mm³ voxels, range rejection can save 10 to 17% on computing time but for smaller voxels it saves less time (3 to 4% for 2.5 mm³ voxels). For non-CT phantoms where one can arrange to have at least some of the voxels quite large, the savings will be correspondingly larger, especially using the dsurround option (see section 14).

Similar to BEAMnrc, range to `ECUT` is determined by subtracting the range from `ECUT` to `AE` (determined using EGSnrc macros and EGSnrc-calculated tables of range to `AE` as a function of particle energy) from the particle's `range` to `AE` (calculated in EGSnrc) at every charged particle step. Rejection of particles based on range to `ECUT` is performed by a DOSXYZnrc macro and not by the EGSnrc's built-in range rejection macro. This is because the EGSnrc range rejection is based on range to `AE` and not `ECUT`.

8.15   ESAVE_GLOBAL

`ESAVE_GLOBAL` is the maximum energy (in MeV) for which range rejection calculations will be performed (ie a particle cannot be rejected if its energy is $\geq$ `ESAVE_GLOBAL`). This option is to prevent termination of high-energy electrons which are likely to generate bremsstrahlung.

8.16   n_split

`n_split` is used to control DOSXYZnrc's photon splitting option. If `n_split` is set > 1 all photons are split into `n_split` photons, each with a weight equal to $\frac{1}{\texttt{n\_split}}$ times the weight of the original photon. For each photon, i, where i=1,2,...,`n_split`:

- The mean free path to its next interaction, $DPMFP_i$, given by

$$DPMFP_i = -\texttt{ln}\left(1 - \frac{\eta + i - 1}{\texttt{n\_split}}\right) \tag{3}$$

   where η is a random number, chosen once for all `n_split` photons.

- At the interaction site, each photon i produces charged particles and/or scattered photons. Russian roulette is played on all scattered photons with a survival probability of $\frac{1}{\texttt{n\_split}}$. Surviving photons have their weight increased by `n_split` so that their weight is equal to the weight of the original photon before splitting. All charged particles survive with weight equal to $\frac{1}{\texttt{n\_split}}$ times the original weight.

- If these charged particles undergo radiative events (bremsstrahlung, annihilation, annihilation at rest), Russian roulette is played on the resultant photons with a survival probability of $\frac{1}{\texttt{n\_split}}$. Again, surviving photons have their weight increased by `n_split` so that their weight is equal to the weight of the original photon before splitting.

- Photons whose weight has been restored to the original weight are subject to splitting again.

Photon splitting has the potential to increase the efficiency of a dose calculation more than photon interaction forcing. A good rule of thumb for the setting of `n_split` is:

$$\texttt{n\_split} >= \frac{N}{1 - e^{-\lambda}} \qquad (4)$$

where λ is approximately equal to the number of photon mean free paths in the geometry of interest and $N \geq 5$. This will increase the number of primary interactions per incident photon by approximately N, so reduce `NCASE` by a factor of N.

It has recently been shown[16] that the use of photon splitting with a phase space source (Section 4.4) or BEAMnrc simulation source (Section 4.10) can increase the efficiency of dose calculations in simulated photon beams by a factor of up to 6.5 (depending on beam energy, field size and phantom voxel size). Moreover, the optimum efficiency (*i.e.* at the optimum value of `n_split`) with a BEAMnrc simulation source is only 3–13% lower than that with the corresponding phase space source, potentially eliminating the need to store phase space data. Thus, it is highly-recommended that you use photon splitting to increase the efficiency of photon beam dose calculations. The optimum setting of `n_split` for phase space and BEAMnrc simulation sources depends on incident beam energy, field size and phantom voxel size. For BEAMnrc simulation sources, splitting numbers of 40 ($0.25 \times 0.25 \times 0.25$ cm^3 voxels) or 32 ($0.5 \times 0.5 \times 0.5$ cm^3 voxels) should give efficiencies close to the optimum, and for phase space sources, splitting numbers of 32 ($0.25 \times 0.25 \times 0.25$ cm^3 voxels) or 24 ($0.5 \times 0.5 \times 0.5$ cm^3 voxels) will be close to the optimum. Note that these settings of `n_split` are higher than those calculated using the rule of thumb given in the paragraph above. This is because much of the efficiency improvement is due to the fact that `n_split` reduces the number of source particles required, thus reducing the CPU time spent generating the source particles (transport through the jaws in the case of a phase space source, performing an entire treatment head simulation in the case of a BEAMnrc simulation source), whereas the rule of thumb is based on the efficiency improvement being solely due to the efficiency inherent in the splitting algorithm itself.

When `n_split` is used with a phase space or BEAMnrc simulation source, then contaminant electrons may compromise dose statistics because they are fewer and will have a higher weight than the split photons (which contribute most of the dose). To avoid this, the phase space sources (`isource`=2,8) and the BEAMnrc simulation source (`isource`=9) have an input, `e_split`, which can be used to split charged particles `e_split` times as soon as they enter the phantom geometry. The weight of the particle is reduced by 1/`e_split`. To maximize efficiency, it is recommended that you set `e_split`=`n_split` if you are using photon splitting with a phase space or BEAMnrc simulation source. For more information about `e_split` see Sections 4.4 and 4.10.

8.17   ihowfarless

If `ihowfarless` is set to 1, then DOSXYZnrc uses the "HOWFARLESS" algorithm for transport in the phantom.

The "HOWFARLESS" algorithm is used to significantly increase the efficiency of dose calculations in a homogeneous phantom. When the option is used, the `HOWFAR` and `HOWNEAR`

subroutines in DOSXYZnrc only consider the extreme outer boundaries of the phantom when calculating the distance along the particle trajectory to the next region boundary and the perpendicular distance to the nearest region boundary respectively. This eliminates the need to stop at voxel boundaries and, hence, speeds up charged particle transport considerably. For the purposes of dose deposition, the total curved charged particle step is approximated by two straight-line steps joined at a hinge point. The straight-line steps can be calculated based on either the known initial position/direction of the particle or its known final position/direction. As coded, the "HOWFARLESS" algorithm uses a 1:1 mixture of step approximations based on the initial position/direction and approximations based on the final position/direction. This has been found to give accurate dose results over all energies and maximum allowed step lengths (input variable SMAX, see paragraph below).

When the "HOWFARLESS" option is used, the limitations on step length within the phantom become the maximum allowable charged particle step length, SMAX (see Section 9.3), and the maximum fractional energy loss per electron step, ESTEPE (see Section 9.4). It is recommended that you leave SMAX at its default value (5 cm if you are using the PRESTA-I boundary crossing algorithm or electron step algorithm, 1e10 cm if you are using the EXACT boundary crossing algorithm and the PRESTA-II electron step algorithm). At most beam energies, this will ensure that step length is only limited by ESTEPE, which it is not recommended that you change.

The efficiency gained using the "HOWFARLESS" algorithm depends on the source type, energy, field size, phantom voxel size, and the boundary crossing algorithm (BCA) used. For a photon beam from a BEAMnrc-simulated linac (using either a phase space source or full BEAMnrc simulation source), use of "HOWFARLESS" increases the efficiency by ∼30% when the PRESTA-I BCA is used, and by a factor of 2.5-3.5 when the more accurate (but much slower) EXACT BCA is used. In the case of a simple photon beam source (*e.g.* a parallel beam simulated using isource=1) with an energy spectrum, the efficiency gain with "HOWFARLESS" is a factor of 1.5-2.5 with the PRESTA-I BCA and a factor of 5-9 with the EXACT BCA. The highest efficiency gains occur in monoenergetic electron beams, where "HOWFARLESS" increases the efficiency by a factor of 3-4 with the PRESTA-I BCA and 8-14 with the EXACT BCA. Efficiency gains are much greater with the EXACT BCA because calculations with standard HOWFAR use the BCA at every voxel boundary, while "HOWFARLESS" calculations use the BCA only at the extreme outer boundaries of the phantom. Thus, the standard calculation becomes much slower when the BCA is switched from PRESTA-I to EXACT, while the speed of the "HOWFARLESS" calculation does not decrease appreciably. For more information about BCA's see Section 9.6.

Use of "HOWFARLESS" is recommended in all homogeneous phantom calculations, such as those used for beam commissioning. More information about this option can be found in the "HOWFARLESS" paper by Walters and Kawrakow[28].

8.18   i_phsp_out

If i_phsp_out is set to 1 or 2, DOSXYZnrc outputs phase space data in IAEA format for particles leaving the phantom geometry. The phase space data is in 3-D, meaning that it

includes the (X,Y,Z) position for each particle. If `i_phsp_out`=1, then the particle positions are in the DOSXYZnrc coordinate system, and if `i_phsp_out`=2, the particle positions are transformed into the BEAMnrc, or source, coordinate system. Note that phase space output requires the existence of a region surrounding the phantom geometry and, thus, is limited to phase space sources (source no.'s 2, 8, 20), BEAMnrc simulation sources (source no.'s 9, 10, 21) and the multiple source model (source no. 4), all of which require the input of `dflag` and `dsurround` to define the dimensions and medium of a surrounding region (see Section 5.4).

If phase space data is output when source 20 (synchronized phase space source) or 21 (synchronized BEAM simulation source) is used then, through the `i_muidx_out` input for those sources (see Sections 4.12 and 4.13), the user has the option to also include the fractional monitor unit index, `MU`, associated with the particle in the output phase space data. Thus, the data includes a time dimension and is considered 4-D. Scoring 4-D data allows simulations downstream of the DOSXYZnrc simulation–*e.g.* BEAMnrc simulations with synchronized component modules or further DOSXYZnrc simulations with synchronized sources–that use the phase space data as a source to be synchronized with the DOSXYZnrc simulation that generated the phase space file.

Since it includes the Z-position of each particle, the size of an IAEA phase space file written by DOSXYZnrc may be slightly larger than that written by BEAMnrc, with the minimum record size (bytes/particle) for DOSXYZnrc being 33 bytes while that for BEAMnrc is 29 bytes. If `MU` is included in the phase space data, then the record size increases to 37 bytes. Note that DOSXYZnrc does not have an option to include `ZLAST` (Z position of last site of photon interaction or creation of secondary charged particle) in the phase space output.

Unlike BEAMnrc, DOSXYZnrc does not have the possibility of defining multiple phase space scoring planes. Thus, the naming scheme for phase space space files output by DOSXYZnrc does not include the scoring plane number, and the files are simply named `inputfile.IAEAheader` (header file) and `inputfile.IAEAphsp` (phase space data).

The default directory for phase space output is `$EGS_HOME/dosxyznrc` (*i.e.* the user code directory). The user can change the phase space output directory by editing the file `$EGS_HOME/dosxyznrc/dosxyznrc_user_macros.mortran`, changing the macro `$DIRECTORY-FOR-PHSP` to the name of the desired output directory (include the full path), and then recompiling DOSXYZnrc.

For more information on the IAEA phase space file format, see the BEAMnrc Users Manual[19].

8.19   `i_bindos`

If `i_bindos` is set to 1, DOSXYZnrc outputs a sparse binary dose file instead of a dense ASCII dose file. The resulting file typically has a much smaller memory footprint than the default `.3ddose` format. See section 12.3 for a description of the format.

8.20   ECUTIN

ECUTIN is used together with the `Global ECUT` input in the EGSnrc input parameters (see section 9) 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 14 of the BEAMnrc manual.

Selection of ECUT is complex in general and is very dependent on what is being calculated[29, 6]. 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.

8.21   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.

8.22   ESTEPM, SMAX

These are dummy inputs that used to define the maximum fractional energy loss per step (ESTEPM) and maximum step length (SMAX). These transport parameters are now handled in the EGSnrc inputs. The dummy inputs have been preserved to ensure compatibility with EGS4/DOSXYZ input files.

9   EGSnrc inputs

The use of EGSnrc to simulate charged particle and photon transport in DOSXYZnrc allows the user a greater degree of control over the transport physics than was previously available in EGS4 versions of DOSXYZ. For most accelerator applications, the DOSXYZnrc default settings of the EGSnrc parameters should be adequate, 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 DOSXYZnrc input file between the delimiters :start mc transport parameter: and :stop mc transport parameter:. The format follows that of the general purpose EGSnrc user-codes[30].

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. Of course, if you are using the DOSXYZnrc 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 DOSXYZ input files to be used directly with DOSXYZnrc. A better approach is to read the old DOSXYZ input file into the dosxyznrc_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 DOSXYZnrc. For more information, see the EGSnrc manual[2]. The actual internal variable name associated with each input appears in brackets.

9.1   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 DOSXYZnrc input section of the input file (the other is Global PCUT described below). Specifically, if ECUTIN in the main DOSXYZnrc 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 8.20 for a more detailed discussion of ECUT.

9.2 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 DOSXYZnrc input section of the input file. If PCUTIN in the main DOSXYZnrc 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 8.21 for a more detailed discussion of PCUT.

9.3 Global SMAX (SMAXIR)

Global SMAX defines the maximum electron step length in cm. If the default EGSnrc electron step electron algorithm (see section 9.8) 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 PRESTA-I 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.

9.4 ESTEPE (ESTEPE)

ESTEPE is the maximum fractional energy loss per electron step. For accurate electron transport with default EGSnrc electron step algorithm (see section 9.8 below) ESTEPE should not exceed 0.25 (the default). ESTEPE should not be changed unless PRESTA-I is being used as the electron transport algorithm.

9.5 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.

9.6 Boundary crossing algorithm (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). In the PRESTA-I case boundary crossing is carried out in a manner similar to EGS4. Specifically, lateral pathlength corrections are turned off if the perpendicular distance from the electron to the boundary is less than Skin depth for BCA (see section 9.7 below) and then, once the electron reaches the boundary, a multiple scattering event is forced. If EXACT boundary crossing is used, 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 9.7 below).

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 [31]. Although the `PRESTA-I` BCA can be up to 3 times more efficient than the `EXACT` BCA it has been shown to result in dose overestimation by up to 2.5% in simulations where charged particle equilibrium does not hold (*e.g.* small beam field on voxels with dimensions $\sim$ field size) or when dose voxels are much smaller than the voxels making up the rest of the phantom[16]. Under such conditions, you must manually switch to the `EXACT` BCA for accurate results.

9.7   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 for this case.

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.

9.8   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` (the name "PRESTA" is preserved only for historical reasons) is the new, more accurate, algorithm developed for use with EGSnrc[2]. `PRESTA-I` is the original PRESTA algorithm with some modifications [32, 33]. 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 (where elastic scattering is weak), it is not recommended for low energy applications.

9.9   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.

9.10 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[34, 35]. 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/DOSXYZ, with some modifications.

9.11 Brems cross sections (IBR_NIST)

This input determines the cross-section used for bremsstrahlung interactions. If `Brems cross sections= BH` (the default), then Bethe-Heitler cross-sections (Coulomb corrected above 50 MeV)[35] are used. These cross-sections are similar to those used by EGS4/DOSXYZ. If `Brems cross sections= NIST`, then cross-sections from the NIST bremsstrahlung cross-section data base[36, 37] are used. The NIST cross-sections are the basis for radiative stopping powers recommended by the ICRU[38]. The difference between `BH` and `NIST` is negligible for energies > 10MeV, but becomes significant in the keV energy range. 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).

9.12 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[39] is used to determine cross-sections for Compton scattering. This is similar to the treatment of Compton scattering in EGS4/DOSXYZ. If `Bound Compton scattering= On`, then the original Klein-Nishina formula is augmented with the impulse approximation[40] 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 9.18). 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 DOSXYZnrc GUI. Note that the `Norej` option cannot be used on a region-by-region basis.

9.13   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, `default_compton.data`, is included in the EGSnrc system.

9.14   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 `$EGS_HOME/dosxyznrc/Makefile` (See Section 2.1 above) must be modified to include `$(EGS_SOURCEDIR)rad_compton1.mortran` just before `$(EGS_SOURCEDIR)get_inputs.mortran`.

9.15   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 the method used to determine positron/electron emission angles in the original version of EGS4. If `Pair angular sampling= KM`, then eqn 3D-2003 in Motz et al[41] is used to determine the positron/electron emission angles. This option is similar to the sampling technique

used by the current version of EGS4/DOSXYZ. Finally if `Pair angular sampling= Simple` (the default), then only the first term in the Motz et al eqn 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`.

9.16   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.

9.17   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`, then photoelectrons inherit the direction of the incident photon. If `Photoelectron angular sampling= On` (the default), then Sauter's formula [42] 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.

9.18   Rayleigh scattering (`IRAYLR`)

This input determines whether Rayleigh (coherent) scattering is simulated or not. If `Rayleigh scattering= On` (the default), then Rayleigh events are simulated using the total coherent cross-sections from Storm and Israel[43] and atomic form factors from Hubbell and Øverbø[44]. This data must be included in the PEGS4 material data set. If `Rayleigh scattering= Off`, then Rayleigh events are not simulated. Rayleigh scattering is only recommended for low-energy (< 1 MeV) simulations. Also, for proper simulation of Rayleigh events, bound Compton scattering (see section 9.12 above) must 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=` .

9.19   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` (the default), 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`, 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/DOSXYZ, 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.

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.

9.20   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[45] is used. For the other selections, the theory associated with the name given is used. See future editions of the EGSnrc Manual[46] 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 BEAMnrc simulations.

9.21   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", "epdl" and "xcom" (the default). 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.

9.22   Photon cross-sections output (xsec_out)

The input `Photon cross-sections output` can be set to `On` to output the file `$EGS_HOME/dosxyznrc/inputfile.xsections` which contains the photon cross section data used in the simulation. Default is `Off`.

10   Parallel Runs using DOSXYZnrc

A DOSXYZnrc simulation may be split into parallel jobs, distributing the simulation among different processors and greatly reducing the elapsed time required for a simulation.

In order to take advantage of the parallel functionality described in this section, the C routines for reading/writing `.pardose` files, `read_write_pardose.c`, must have been successfully compiled during OMEGA/BEAM installation (this requires that your system have a C or C++ compiler), and the `$HEN_HOUSE/specs/dosxyznrc_config.spec` created during installation must have the variable `PARDOSE_OBJECTS` set to `$(EGS_LIBDIR)read_write_pardose.o` (done automatically if `read_write_pardose.c` was compiled successfully). For more information about `read_write_pardose.c` and `dosxyznrc_config.spec`, see section 2.1(page 6).

Note that one must be careful when performing parallel calculations with phase space sources and/or high resolution phantoms, since the network traffic generated can be very high.

Previously, parallel calculations were submitted using the `pprocess` script, which created separate input files for each job, with the same number of histories (total histories/no. of jobs) in each job. These input files were then submitted to the batch queueing system. The DOSXYZnrc input variables `IPARALLEL` and `PARNUM` were essential in this former approach

because they allowed a phase space source to be divided up into equal partitions. `IPARALLEL` was set to the number of parallel jobs and `PARNUM` took on values 1,2,...,`IPARALLEL`, with each input file having a different value of `PARNUM`. For a given input file, then, particles from a phase space source were sampled from a partition given by:

$$(\texttt{PARNUM} - 1) * \frac{\texttt{nshist}}{\texttt{IPARALLEL}} + 1 \leq \texttt{nnphsp} \leq \texttt{PARNUM} * \frac{\texttt{nshist}}{\texttt{IPARALLEL}}$$

where `nshist` is the total number of particles in the phase space source and `nnphsp` is the number of the source particle selected. This ensured that the entire phase space source was sampled evenly over all parallel jobs.

The previous approach to parallel calculations was limited by the fact that each machine ran the same number of histories, making the total calculation only as efficient as the slowest CPU.

The current approach to parallel calculations with DOSXYZnrc is similar to that used by other EGSnrcMP user codes (including BEAMnrc) and uses built-in functions to run the jobs. You must be using a Unix/Linux system and have a batch queueing system, such as PBS or NQS, installed.

To submit a parallel job, use the `exb` script with the following syntax:

`exb dosxyznrc inputfile pegsdata [short|medium|long] batch=batch_sys p=N`

where `N` is the number of parallel jobs and `batch_sys` is the name of the batch queuing system (currently `batch_system=pbs`, for PBS, `batch_system=keg`, for Sun's SGE and `batch_system=nqs`, for NQS, are supported). See section 2.2 for more on `exb`. Note that parallel jobs will not run with the standard Unix batch command `at`. Thus, you must explicitly input the `batch_system` variable as shown above unless you have set the environment variable `$EGS_BATCH_SYSTEM` to something other than `at`, in which case that is the default.

The details of how a parallel run is carried out are similar to those described in the BEAMnrc Users Manual[12]. Basically, jobs are controlled by a job control file, `$EGS_HOME/dosxyznrc/inputfile.lock` (created by the first job submitted). This file is read from and updated by all parallel jobs and contains the total number of histories remaining to be run and the number of jobs running, among other information. Jobs run until there are no histories remaining in the job control file. Rather than each job running a fixed `NCASE`/N histories (as in the old `pprocess` scheme), a job is only allowed to run a fraction, or "chunk" of this number at a time. Thus each run consists of `NCASE`/(N*`$N_CHUNK`) histories, where `$N_CHUNK` is defined in `$HEN_HOUSE/src/egsnrc.macros` and is set by default to 10. Breaking the simulation into smaller chunks allows jobs using faster CPU's to run more histories than those running on slower CPU's, increasing the efficiency of this new parallel processing scheme. If a job finds there are no histories left in `inputfile.lock`, then it analyzes the results for its own runs for output to `inputfile_w[i_parallel].egslst`, where `i_parallel` is the parallel job number (1,2,...N), and then quits. The last job to quit calls a subroutine in DOSXYZnrc (`combine_results`) which automatically combines and analyzes the output from all parallel jobs and outputs the results to `inputfile.egslst` and `inputfile.3ddose`.

Similar to BEAMnrc, each parallel job must begin with a different random number seed.

This is accomplished by incrementing JXXIN, the second random number seed, for each parallel job submitted using:

$$\text{JXXIN} = \text{JXXIN}_{input} - 1 + \text{i\_parallel} \tag{5}$$

where JXXIN_{input} is the value of JXXIN in `inputfile.egsinp` and `i_parallel` is the parallel job number (1,2,...,N).

During a parallel run, output of `.3ddose` files is suppressed, since these files can be large and outputting one for each parallel job could consume large amounts of CPU time/memory. Instead, each parallel job outputs a binary `.pardose` file. These files are output by the C routine `write_pardose.c` which is found in the file `read_write_pardose.c` and is linked with DOSXYZnrc at compile time. A `.pardose` file contains the following data:

1. no. of primary histories in the job

2. incident fluence in the job

3. the number of voxels in X, Y and Z directions

4. the voxel boundaries

5. $\frac{\sum_{i=1}^{nhist} edep_i}{rho}$ for all voxels where $edep_i$ is the energy deposited by primary history i during the job, `nhist` is the number of primary histories in the job, and `rho` is the density of the voxel

6. $\frac{\sum_{i=1}^{nhist} edep_i^2}{rho^2}$ for all voxels

Thus, a `.pardose` file contains enough information to recreate a `.3ddose` file. At the end of a parallel run, when all jobs are finished, the last job uses the C subroutine `read_pardose.c` (also found in `read_write_pardose.c`) to read the `.pardose` files from all jobs and then recombines the data to create `inputfile.3ddose`.

`.pardose` files can be recombined separately by re-running DOSXYZnrc with the input parameter, IRESTART=4 after all jobs have completed. Use of IRESTART=4 is generally not necessary now that the last job automatically recombines parallel results, however, it may be useful if, for some reason, all of the `.pardose` files were not moved out of their temporary working directories or if you wish to add more `.pardose` files from a separate group of parallel runs. See section 8.12(page 84) for more on IRESTART.

Note that `.egsdat` files could have been used for recombining after a parallel run (similar to BEAMnrc). However, `.egsdat` files from DOSXYZnrc runs can get quite large. Thus, we opted to use the much smaller `.pardose` files, allowing the user to set IDAT=1 (option to not output `.egsdat` files – see section 8.13, page 85) to further save CPU time/memory during a parallel run.

Similar to parallel BEAMnrc runs, if you are using a phase space file as a source, then this source is partitioned so that it is sampled evenly over all jobs. Each chunk of the run uses

a different partition of the phase space file, where the number of particles in each partition p_per_phsp_chunk, is given by:

$$\text{p\_per\_phsp\_chunk} = \frac{\text{nshist}}{(\text{N} * \$\text{N\_CHUNKS})} \tag{6}$$

where **nshist** is the total number of particles in the phase space source. See the BEAMnrc Manual[12] for more details.

Parallel runs can be restarted, provided that you have the .egsdat files from all of the previous parallel jobs available. 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.

11   Adjustable Parameters in the Source Code

Within the MORTRAN language, one can set a variety of variables which are used at the compilation stage. These can all be adjusted by the user and then the code recompiled as normal. Many of the parameters the user might want to adjust are in the file $EGS_HOME/dosxyznrc/dosxyznrc_user_macros.mortran. If you want to change any of these parameters from their default values (values are echoed at the top of the .egslst and .egslog files) simply go into this file, change the relevant parameters and recompile DOSXYZnrc. Note that a copy of the original dosxyznrc_user_macros.mortran file remains in the $HEN_HOUSE/user_codes/dosxyznrc directory so that you can revert back to the original defaults at any time.

$MXMED the maximum number of media allowed in the phantom (default 5)

$MXSTACK: The maximum stack allowed by EGSnrc (default 15, code will warn you if this is too small)

$IMAX, $JMAX, $KMAX: the maximum number of dose scoring regions in the x, y, z directions within the phantom.

$DOSEZERO: a flag which is either 1 (default) which implies all doses with uncertainties greater than 50% are zeroed in the .3ddose output file (since these are usually air regions with wildly fluctuating doses which destroy display routines) or the flag is 0 which implies all values are output to the .3ddose file. In all cases, the values are output to the .egslst file if requested. Note that the input variable zeroairdose also tends to get rid of these artifacts (see section 8.3).

12   Format of Dose Outputs

The dose distributions calculated by DOSXYZnrc can be found in the output files, ".egslst", ".3ddose", ".bindos" and ".pardose". The file ".egslst"   contains not only the dose (when asked for) and statistical data but also the information about simulation geometry, number of histories run, CPU time used, etc. The dose output file ".3ddose" contains the information about the simulation geometry and the calculation results in a format that can be read by STATDOSE for generating xvgr/xmgr/xmgrace plots.

12.1   Format of .3ddose

The following explains the format of .3ddose":

   Row/Block 1 — number of voxels in x,y,z directions (e.g., n_x, n_y, n_z)

   Row/Block 2 — voxel boundaries (cm) in x direction(n_x +1 values)

   Row/Block 3 — voxel boundaries (cm) in y direction (n_y +1 values)

   Row/Block 4 — voxel boundaries (cm) in z direction(n_z +1 values)

   Row/Block 5 — dose values array ($n_x n_y n_z$ values)

   Row/Block 6 — error values array (relative errors, $n_x n_y n_z$ values)

General rules for reading the dose data:

   1. Read one by one (across columns) to get dose (error) readings in x direction

   2. Read every (n_x)-th value to get readings in y direction

   3. Read every ($n_x n_y$)-th value to get readings in z direction

12.2   A Sample .3ddose File

Table 1 shows the dose distributions in a 4x4x4 1 cm^3 cube with x between -2 and 2, y between -2 and 2 and z between 0 and 4.

12.3   .bindos Files

A ".bindos" file is output instead of a ".3ddose" file if the flag "i_bindos" is set to 1. The same dose content is contained in each file, but ".bindos" is a sparse format, hence only information about non-zero dose voxels is included. The linearized start and end indices for blocks of contiguous non-zero doses are stored. Both integers and floats are stored as 4 bytes each.

   1. Number of voxels in the x, y, z direction (e.g., n_x, n_y, n_z, 3 ints)

Table 1: Sample `.3ddose` output file

Row (block) Number		Column Number				
		1	2	3	4	5
1	(1)	4	4	4		
2	(2)	-2.0000	-1.0000	0.0000	1.0000	2.0000
3	(3)	-2.0000	-1.0000	0.0000	1.0000	2.0000
4	(4)	0.0000	1.0000	2.0000	3.0000	4.0000
5	(5)	1.0000	2.0000	2.0000	1.0000	2.0000
6		8.0000	8.0000	2.0000	2.0000	8.0000
7		8.0000	2.0000	1.0000	2.0000	2.0000
8		1.0000	2.0000	4.0000	4.0000	2.0000
9		4.0000	16.000	16.000	4.0000	4.0000
10		16.000	16.000	4.0000	2.0000	4.0000
11		4.0000	2.0000	3.0000	6.0000	6.0000
12		3.0000	6.0000	24.000	24.000	6.0000
13		6.0000	24.000	24.000	6.0000	3.0000
14		6.0000	6.0000	3.0000	4.0000	8.0000
15		8.0000	4.0000	8.0000	32.000	32.000
16		8.0000	8.0000	32.000	32.000	8.0000
17		4.0000	8.0000	8.0000	4.0000	
18	(6)	1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01
19		1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01
20		1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01
21		1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01
22		1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01
23		1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01
24		1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01
25		1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01
26		1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01
27		1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01
28		1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01
29		1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01
36		1.0000E-01	1.0000E-01	1.0000E-01	1.0000E-01	

2. Voxel boundaries (cm) in x direction(n_x +1 floats)

3. Voxel boundaries (cm) in y direction(n_y +1 floats)

4. Voxel boundaries (cm) in z direction(n_z +1 floats)

5. Number of non-zero dose voxels (n_{nz}, 1 int)

6. Number of voxel index pairs (n_b, 1 int)

7. Linearized voxel index pairs indicating the [start, end) of contiguous non-zero voxel doses. ($2 \times n_b$ ints)

8. Nonzero voxel doses (n_{nz} floats)

9. Nonzero voxel uncertainties (n_{nz} floats)

The linearization formula is as follows:

$$v_{lin} = (n_x \times n_y) \cdot v_z + (v_x) \cdot v_y + v_x \tag{7}$$

where v_x, v_y, v_z are the voxel indices in the x, y, z axes, respectively.

12.4   `.pardose` Files

The ".`pardose`" file is output instead of the ".`3ddose`" file by parallel jobs (see section 10,page 98). ".`pardose`" is a binary file containing the number of histories in the run, the number of voxels in the X, Y and Z directions, the voxel boundaries, and information about energy deposited and energy^2 in each voxel. The binary format allows for much faster reading and writing and smaller file size. Enough information is contained in ".`pardose`" files from a group of parallel runs to reconstruct a ".`3ddose`" file from them.

13   Dose Normalization

Dose normalization depends on what source you are using. For sources with a well-defined beam area on the surface of the phantom (sources 0,1,3,7) dose is normalized by the incident particle fluence, `ainflu`, given by

$$\texttt{ainflu} = \frac{\texttt{NCASE} + \texttt{ncaseold} - \texttt{nmissm}}{(\texttt{xinu} - \texttt{xinl}) * (\texttt{yinu} - \texttt{yinl})}$$

or

$$\texttt{ainflu} = \frac{\texttt{NCASE} + \texttt{ncaseold} - \texttt{nmissm}}{\texttt{xcol} * \texttt{ycol}}$$

where `NCASE` is the number of histories in this run, `ncaseold` is the number of histories from previous runs (if this is the first run then this will be 0) and `nmissm` is the total number of particles from the source that missed the geometry, including in any previous runs. (`xinu -`

xinl)*(yinu - yinl) is the beam area for sources 0 and 3, while xcol*ycol is the beam area for sources 1 and 7. If the incident beam area happens to be 0, then dose is normalized by (NCASE+ncaseold-nmissm).

For phase space sources (sources 2,8,20), dose is normalized by an estimate of the number of particles incident from the original, non-phase space source, NP, given by:

$$NP = NINCSRC * \left[\frac{NCASE + ncaseold + nsmiss + (NRCYCL + 1) * (nsrjct + nsoutside + ndbsrjct)}{nshist} \right]$$

where (nsrjct+nsoutside+ndbsrjct) is the total number of particles rejected because they had the wrong charge, LATCH bit setting, were going backwards, had crossed the phase space plane more than once, were beyond the user-selected field (BEAM_SIZE), or were fat photons (if directional bremsstrahlung splitting (DBS) was used in the BEAM simulation that generated this phase space source), nsmiss is the number of particles rejected because they missed the geometry, NRCYCL is the number of times that each particle is to be recycled, nshist is the total number of particles in the phase space file and NINCSRC is the number of particles from the original, non-phase space source used to generate this phase space source. The quantity NCASE+ncaseold+nsmiss+(NRCYCL+1)*(nsrjct+nsoutside+ndbsrjct) is an estimate of the total number of particles read from the phase space file in the simulation. By dividing this by nshist we obtain an estimate of the number of times that the phase space source is used which we then multiply by NINCSRC to obtain an estimate of the equivalent number of particles incident from the original source.

For the isotropically radiating source (source 6), dose is simply normalized by the total number of histories, NCASE+ncaseold. For beam characterization models (source 4), dose is normalized by NCASE+ncaseold-nsmiss, since no information about the beam area or the number of particles incident from the original, non-phase space source is available.

For the full BEAM simulation source (sources 9,10), doses are normalized by the number of primary histories incident in the BEAM simulation. This is similar to sources 2 and 8 (phase space sources), where this number is estimated, but with source 9 and 10, because the BEAM simulation is being run concurrently with DOSXYZ, we have access to the exact number of primary histories.

14   Saving Time in Depth-Dose and Dose Profile Calculations using dflag and dsurround

When enflag > 1 (*i.e.* phase space source, full BEAM simulation, or multiple source model) the user can specify the medium and thickness of a region surrounding the DOSXYZnrc phantom using medsur, dflag and dsurround(1...4) (see section 5 above). Setting dflag to 1 allows the user to input separate thicknesses for the region surrounding the phantom in the ± x directions, the ± y directions, the +z direction and the −z direction. This flexibility allows the user to divide only a portion of the DOSXYZnrc phantom into voxels. For example, voxels may be specified in a column (for depth dose calculations) or slice (for dose profile and depth dose calculations) within a larger phantom of the same medium (see figure 8 below).

Since much of the time in a DOSXYZnrc simulation is taken transporting particles to and away from voxel boundaries, having voxels only on the axis/slice of interest and then defining the rest of the phantom as essentially one region using `dsurround(1...4)` can save substantial amounts of CPU time in depth-dose and dose profile calculations. CPU time can be reduced even further by turning on range rejection. Since range rejection of charged particles is based on whether or not a particle can reach the boundary of a region, range rejection in the "`dsurround`" region will be more efficient because, on average, the distance from the particle to a boundary of the region will be larger than if the entire phantom was divided into small voxels.
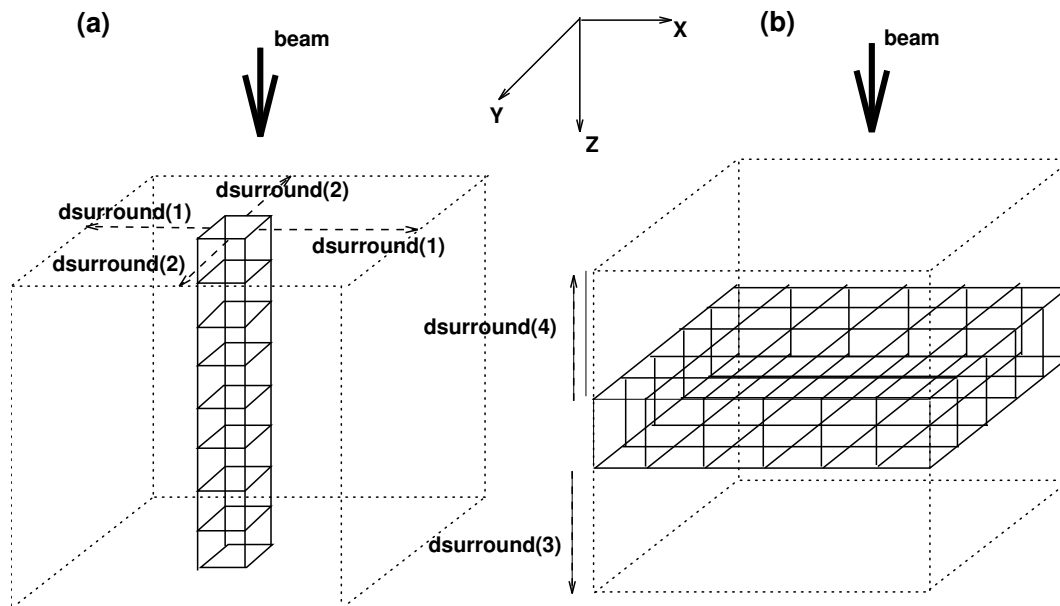
Figure 8: Two examples of how `dflag` and `dsurround(1...4)` can be used to specify a small group of voxels within a larger phantom of the same medium. In (a) voxels are only specified in a single column for scoring depth-dose, while the rest of the phantom is defined using `dsurround(1)` and `dsurround(2)` (`dsurround(3)` and `dsurround(4)` are 0 in this particular example). (b) shows voxels specified in a horizontal slice for scoring the dose profile at a given z value. In this latter example, the dimensions of the phantom below and above the slice are defined by `dsurround(3)` and `dsurround(4)` respectively. `dsurround(1)` and `dsurround(2)` are set to 0 in (b).

Table 2 below shows the results of timing studies performed using a beam of 10 MeV electrons and a beam of 6 MV photons incident on a 59x59x10cm water phantom. For each beam, 4 cases were simulated: the entire volume filled with 1cm^3 voxels; 1cm^3 voxels specified in a column (1x1x10) down the central (z) axis of the volume (similar to Figure 8(a)); 1cm^3 voxels specified in a vertical slice (1x59x10) through the phantom in the y direction at x=0; 1cm^3 voxels specified in a horizontal slice (59x59x1) through the phantom at z=2.5cm ($\sim$d$_{max}$) (similar to Figure 8(b)). In all cases, simulations were run with and without range rejection.

The table shows that if a depth-dose curve is all that is required, then specifying voxels only in a 1x1x10cm column can decrease simulation time by a factor of 5.5 for the electron beam and a factor of 3 for the photon beam. In general, the CPU time increases as the number

Table 2: Timing results for circular beams of 10 MeV electrons and 6 MV photons incident on a water phantom. In all cases, the phantom size is the same (59x59x10 cm^3), but the 1x1x1cm^3 voxels take up differing parts of the volume (see text).

source	number of voxel regions (all 1 cm^3)	CPU time (hrs)	
		range rejection off	range rejection on
10 MeV electrons (r=10 cm, 1x10^6 histories)	59x59x10 uniform	0.576	0.559 (`ESAVE=5MeV`)
	1x1x10 central axis	0.181	0.100
	1x59x10 vertical plane	0.208	0.141
	59x59x1 horizontal plane	0.345	0.300
6 MV photons (r=10 cm, 30x10^6 histories)	59x59x10	1.623	1.525 (`ESAVE=3MeV`)
	1x1x10	0.785	0.479
	1x59x10	0.824	0.551
	59x59x1	0.969	0.711

of voxels specified increases, but orientation of the volume in which voxels are specified plays a role as well. The 59x59x1cm horizontal slice is, relatively, the least efficient use of `dsurround` (saving a factor of ∼2 in CPU time) due to the fact that it has more voxels than the central-axis phantom and the vertical slice phantom but also due to the fact that most of the primary and secondary particles from the beam have to be transported across the horizontal voxel boundaries defining the slice. Finally, note that range rejection is more efficient when using `dsurround` with a smaller number of voxels: turning range rejection on decreases the simulation time for the case where voxels are only specified in a 1x1x10cm column by a factor of almost 2, while its effect on the simulation time when the entire phantom is divided into voxels is negligible.

Note that this option allows electrons to take much bigger steps in the `dsurround` regions and this can cause some small inaccuracies unless the default EGSnrc transport parameters are used.

15 Pegsless mode

As of 2013, DOSXYZnrc can be run independent of the pegs4 cross section data file. Photon cross sections had been calculated on the fly since 2006, so the move towards a completely pegsless version of EGSnrc-based user codes was a logical one.

When running in pegsless mode, all media used in the simulation must be defined in the `.egsinp` file, between the delimiters, `:start media definition:` and `:stop media definition:`. `.egsinp` file. Here, the user has the option of specifying the name of a mate-

rial data file, containing the compositions, bulk densities, etc, of the media and/or defining media parameters directly in the `.egsinp` file. The latter method is useful for defining media not included in the material data file or for overriding parameters in the material data file. The user also specifies the particle energy limits for cross section calculation: `AE` and `UE` for electrons, `AP` and `UP` for photons. For more information about the inputs between the `:start media definition:` and `:stop media definition:` delimiters, including defaults, you are urged to refer to the BEAMnrc Users Manual[19].

Pegsless mode is accessible through the DOSXYZnrc GUI by selecting "Change PEGS4 file" from the "File" menu and then opting to "Go PEGSless" instead of selecting a PEGS4 file. Once you have done this, a "Define Media" button at the bottom of the main input window becomes active. Pushing this button will open up a sub-window through which you have access to the inputs between `:start media definition:` and `:stop media definition:` in the `.egsinp` file.

DOSXYZnrc can be run interactively in pegless mode using the command line input:

`dosxyznrc -i inputfile`

where `inputfile` is the name of the input file (with no `.egsinp` extension).

Pegsless batch runs use the command line syntax:

`exb dosxyznrc inputfile pegsless [short|medium|long] [batch=batch_system] [p=N]`

This is identical to the syntax of a run with pegs data (see Section 2.2 above) but with the word `pegsless` in place of the name of the pegs data file.

On termination, pegsless runs output a file, `inputfile.mederr`, containing information about where the specifications for each medium in the simulation have been read (*i.e.* from the material data file or directly from `inputfile.egsinp`) and various warning messages related to inputs that have been left blank and have been set to their default values. The media compositions and settings for other parameters necessary for calculating cross sections are output to `inputfile.egslst` and `inputfile.egslog` (for batch runs).

16   CT Based Phantoms/`ctcreate`

The CT phantom option of DOSXYZnrc allows calculation of dose distributions in phantoms that are derived from CT data sets. This allows simulations in realistic anthropomorphic phantoms. (Please note that the previous sentence does not use the word patient.) The creation of CT phantoms from CT data is performed using the stand-alone code, `ctcreate`.

At this point in time, `ctcreate` supports CT data in DICOM format (with some restrictions, see below), ADAC Pinnacle format, and CADPLAN format. A tool for converting the AAPM CT format into Pinnacle format is also available.

The process by which CT phantoms are created by `ctcreate` are outlined here and described in detail in the following sections.

1. Read in the format of the CT data

2. Read in the CT header parameters (binary or ASCII).

3. Read in the binary CT data.

4. Choose a subset of the CT data set (if desired).

5. Resample the CT data to correspond to volume elements that dose will be scored in.

6. Convert the CT data to materials and densities for each voxel.

7. Transfer the data via a file to be input to DOSXYZnrc.

The relevant CT phantom information is written into the file, `*.egsphant` (prefix is the same as the original CT data set name). A flowchart for the CT phantom code, showing how it relates to DOSXYZnrc is shown in figure 9.

Nearly all of the CT phantom functionality takes place outside of `ctcreate` proper in subroutines. This modularity allows for easy changes in the code by ambitious users (for example subroutines for other CT file formats). Specifically, this is to provide the user with the means of reading their own CT file formats and writing their own dose distribution files without rewriting large chunks of `ctcreate.mortran` and `dosxyznrc.mortran`. This can be accomplished by creating their own versions of the subroutines `ReadCT` in `ctcreate.mortran` and `write_dose` in `dosxyznrc.mortran`. The specifications as to what is required of these subroutines can be found in the codes themselves and in section 16.5 of this document.

Note that the input CT data set defines the coordinate system for the calculation, which may be different from the coordinate system of the accelerator simulation (using BEAM). For example, for Pinnacle CT data sets, the Z-axis is typically down the centre of the patient whereas in the accelerator simulation the Z-axis is the beam's central axis. This causes no problems, but must be kept in mind.

16.1   Using the CT Phantom Option in DOSXYZnrc

The CT phantom option is used by setting the number of materials, `nmed` in record 2 of the DOSXYZnrc input file, to zero. This will cause the program to execute differently than when `nmed` is > 0, and, instead of geometry, material and density data for the phantom being input explicitly in the DOSXYZnrc input file, DOSXYZnrc reads these data from a CT phantom file which has been created using `ctcreate`. The input for the CT and non-CT modes of DOSXYZnrc are common again after this input (see section 3.1, page 11).

In short, the input file for DOSXYZnrc is very different if the CT option is being used. From section 3.1 it can be seen that in CT mode, input records 3-9 in non-CT mode are replaced by records 3-5, which specify the name of the file containing the CT phantom data, the transport parameters, and some output parameters.
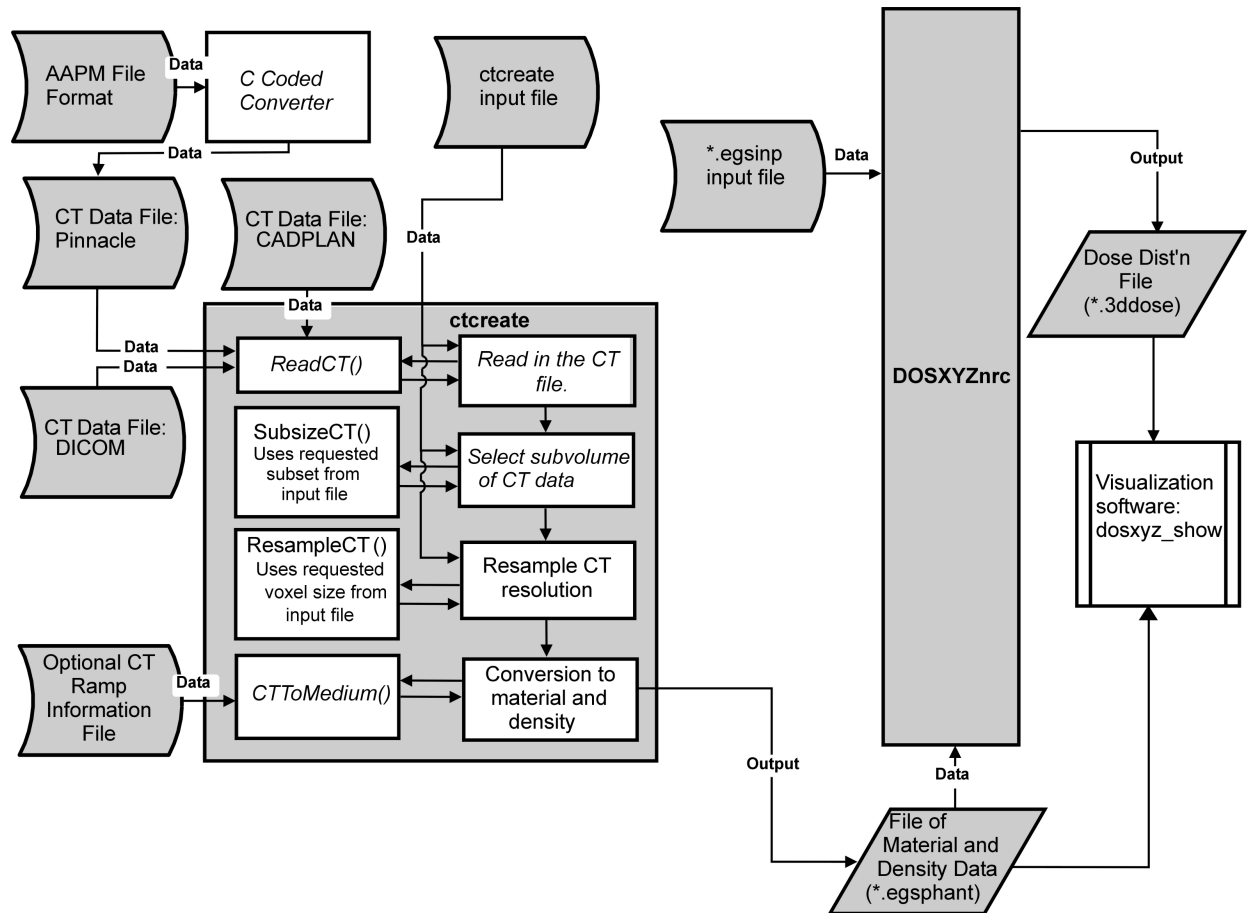
Figure 9: A flowchart for use of CT data with `ctcreate` and DOSXYZnrc.

16.2   Using `ctcreate`

This section covers the input parameters that `ctcreate` requires to create a `*.egsphant` CT phantom file that can then be read in and used by DOSXYZnrc.

It is fairly simple to obtain a CT phantom from the CT data set since all the required material and geometry information is contained in the data set. The additional information that the user is required to provide are the CT data format, the name of the file containing either the header info (Pinnacle) or containing the names of the actual CT data files (CADPLAN and DICOM), voxel dimensions for the phantom. The user can also select a subset of the full CT data and specify their own CT ramp (ie function for converting CT data to the densities and materials required for the DOSXYZnrc phantom). Input of a custom CT ramp is highly recommended.

The following are the input parameters for `ctcreate`. This description is found at the beginning of the `ctcreate.mortran` code.

```
  DESCRIPTION OF INPUT FILE                              toc:
  =========================
  (Revision: 1.19  last edited Date: 2013/03/25 14:40:01 )


 CT Record 1              ctformat (A60)
                              The format of the CT data.  Currently Pinnacle,
                              CADPLAN and DICOM formats are handled for all
                              architectures.
                              AAPM format requires conversion to Pinnacle through
                              the $OMEGA_HOME/progs/ctcreate/CT/AAPM/aapm2pinnacle
                              code.


 CT Record 2              CTFilename (up to 256 characters)
                            For Pinnacle format:
                             CTFilename is the full name of the .header file.
                             Assumes that the binary CT data is stored in a file
                             with the same prefix but with a .img extension.
                            For CADPLAN and DICOM formats:
                             CTFilename is the full name of a file in which is
                             stored the full names of the CADPLAN data files
                             that make up the full CT image (one file/slice).
                             Files must be in order of increasing Z.


 CT Record 3              xctsubmin,xctsubmax,yctsubmin,yctsubmax,zctsubmin,
                          zctsubmax (6F10.4) (ON ONE LINE)

                          xctsubmin,xctsubmax: lower and upper x boundaries (cm)
                                                 of the subset of the CT
                                                 data to be considered for the
                                                 dosxyznrc phantom.
                          yctsubmin,yctsubmax: lower and upper y bounds (cm)
                                                 of the subset of the CT data
```

                                             to be considered for the dosxyznrc
                                             phantom.
                            zctsubmin,zctsubmax: lower and upper z bounds (cm)
                                             of the subset of the CT data
                                             to be considered for the dosxyznrc
                                             phantom.

                         If all are set to 0, then the entire CT volume
                         is selected. If, in any dimension, the lower
                         bound is >= the upper bound, then the entire
                         thickness in that dimension will be used. Finally,
                         note that the subvolume is always expanded to
                         include an integer number of CT voxels.

CT Record 4   xyz_xthickness,xyz_ythickness,xyz_zthickness (3F15.0);
                         The x, y and z voxel dimensions (cm) to be used
                         for the dosxyznrc phantom. Restrictions:
                             (xctsubmax-xctsubmin)/$IMAX<=xyz_xthickness<=
                                                 (xctsubmax-xctsubmin)
                             (yctsubmax-yctsubmin)/$JMAX<=xyz_ythickness<=
                                                 (yctsubmax-yctsubmin)
                             (zctsubmax-zctsubmin)/$KMAX<=xyz_zthickness<=
                                                 (zctsubmax-zctsubmin)

                          Note that voxel dimensions are always increased to
                          fit an integer number of dosxyznrc voxels on the CT
                          sub-volume selected.

CT Record 5   num_material, material_ct_lower_bound (2I10)
                         num_material: This is the number of materials and ramps
                                     that are to be read from the file.
                         material_ct_lower_bound: lower CT no. for the first
                                             medium in the ramp.

                         If this input is 0,0 then the default CT ramp is used.
                         See subroutine CTToMedium for a description of the
                         default ramp.

                         If the default ramp is not used then the format for
                         user input of the materials and ramps
                         is shown in CT Records 6.a and 6.b. These line pairs
                         are repeated num_material times.

 Repeat 6.a and 6.b for i=1 to num_material (only used if num_material > 0).

CT Record 6.a   material_name (24A1)
                         The PEGS4 name of material i.

CT Record 6.b   material_ct_upper_bound(i),

```
                         material_density_lower_bound(i),
                         material_density_upper_bound(i) (I5,2F15.0)
                         (all on one line)

                         material_ct_upper_bound(i): max CT no. for material i
                         material_density_lower_bound(i): min density for
                                                          material i (g/cm^3)
                         material_density_upper_bound(i): max density for
                                                          material i (g/cm^3)
```

Parameters may be input interactively, by typing:

ctcreate

and responding to the prompts; or stored in an input file and `ctcreate` run with the input file by typing:

ctcreate inputfilename

`ctcreate` input parameters are described in more detail in the subsections below.

16.2.1 ctformat

The first input required by `ctcreate` is the format of the CT data. Currently, `ctcreate` supports `Pinnacle`, `CADPLAN` and `DICOM` formats. If the data is in AAPM format, then it can be converted to Pinnacle format using the code **$OMEGA_HOME/progs/ctcreate/CT/AAPM/aapm2pinnacle**

16.2.2 CTFilename

`CTFilename` is the full name of a file, the contents of which depend on the CT data format:

Pinnacle format `CTFilename` is the full name of the header file of the CT data set (including the `.header` extension). The code will read in all the information that it requires from this file before moving on to read in the entire binary data file (which has an assumed extension `.img` and the same prefix as the header file).

CADPLAN and DICOM formats `CTFilename` is the full name (*i.e.* including directory path) of a file containing the full names of the individual CT data files (1 file/image slice). The file names MUST appear in order of increasing Z position of the slice.

On output, the CT phantom will be named:
CTFilename(minus .header extension if using Pinnacle format).egsphant

16.2.3   xctsubmin,xctsubmax,yctsubmin,yctsubmax,zctsubmin,zctsubmax

xctsubmin,xctsubmax, yctsubmin, yctsubmax, zctsubmin, and zctsubmax are used to cre-
ate six planes which describe a cube. The subsection of the original CT data contained in
this cube will be used to create the CT phantom. In this manner only the portion of interest
in the original CT data is used in the simulation. This allows the particle simulation to
be performed at a higher resolution than if the calculation used the entire CT volume, and
allows the user to trim some of the air that surrounds a typical CT image from considera-
tion in the phantom. Note that if the sub-volume selected by the user does not fit on an
integer number of voxels from the original CT data, then the sub-volume will automatically
be expanded until it does so.

16.2.4   xyz_xthickness,xyz_ythickness,xyz_zthickness

The third line of the input to ctcreate is the spatial resolution that the user requires for
the simulation. xyz_xthickness, xyz_ythickness, xyz_zthickness are the 3 dimensions
of the voxels to be used in the CT phantom. The maximum voxel dimension in a given
direction is the distance between the planes delimiting the subset of the CT data to be used
in that direction (see previous section). The minimum voxel dimension in a direction is the
distance between the planes delimiting the subset of CT data in that direction divided by
the maximum number of voxels allowed in that direction (set at compilation). ctcreate
will alert the user when dimensions less than the minimum or greater than the maximum
are input. Note that, if necessary, the voxel dimensions will always be increased to fit an
integer number of DOSXYZnrc voxels on the CT sub-volume selected by the user.

16.2.5   num_material, material_ct_lower_bound, and other CT ramp inputs

num_material is the number of materials in the ramp function used to convert CT number
in each voxel to material and density. If this is set to zero then a default CT ramp is used
in the conversion. The default ramp is shown in Figure 10 and in the sample input file
given in the next section. material_ct_lower_bound defines the lowest CT number of the
first ramp (*i.e.* the lowest CT number for material 1). If the default ramp is used, then
material_ct_lower_bound defaults to -1024, which is typically the minimum CT number
for air in DICOM format data.

If num_material >0 then the CT ramp is read from the input file. For each medium
in the ramp, there are two lines of input. The first of these lines is the name of the
material (material_name) which must be identical to a material in the PEGS4 material
data file being used in the DOSXYZnrc simulation. The second line specifies the max-
imum CT number for the medium, material_ct_upper_bound, the minimum density of
the medium, material_density_lower_bound, and the maximum density of the medium,
material_density_upper_bound. Note that the ramp is assumed to be continuous in CT
number. Thus, for material n, material_density_lower_bound(n) corresponds to a CT
number of material_ct_upper_bound(n-1) (or the overall minimum CT no. of the ramp,
material_ct_lower_bound, if n=1), and material_density_upper_bound(n) corresponds to

`material_ct_upper_bound(n)`. Also note that the ramp must be input in order of increasing `material_ct_upper_bound`.

The CT ramp is then used to determine the medium and density in each voxel of the CT data. If a voxel's CT number is $\leq$ `material_ct_upper_bound(n)` and $>$ `material_ct_upper_bound(n-1)` (or `material_ct_lower_bound` if n=1) then it is assigned medium `n`. The density is then calculated using a linear interpolation between the medium's density limits:

$$
\begin{aligned}
\mathtt{rhor}_{i,j,k} = {} & \mathtt{material\_density\_lower\_bound(n)} \\
& + \left(\frac{\mathtt{material\_density\_upper\_bound(n)} - \mathtt{material\_density\_lower\_bound(n)}}{\mathtt{material\_ct\_upper\_bound(n)} - \mathtt{material\_ct\_upper\_bound(n-1)}} \right) \\
& * \left(\mathtt{CT}_{i,j,k} - \mathtt{material\_ct\_upper\_bound(n-1)} \right)
\end{aligned}
\tag{8}
$$

The default ramp is shown in Figure 10. This is a modified version of a CT ramp given in Kawrakow et al[49]. Note that this ramp has been optimized for typical DICOM format data, where the lowest CT number for air `material_ct_lower_bound` is -1024.

If the CT number in a voxel is less than `material_ct_lower_bound` then the medium in that voxel is set to vacuum with density zero. Those voxels with CT numbers above the last material's upper CT number (`material_ct_upper_bound(num_material)`) will have the density set to the maximum density for the last material.
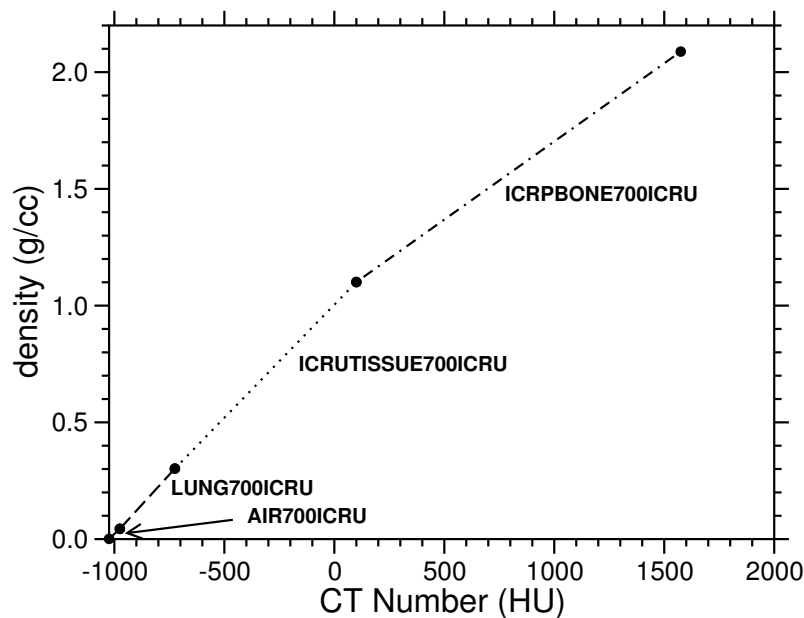
Figure 10: The default ramp for converting CT values to material and density in `ctcreate` modified from Kawrakow et al[49].

The default CT ramp may be suitable for displaying a `.egsphant` file using `dosxyz_show`[23] and for example calculations. However, we strongly recommend that you explicitly enter your own CT ramp function for more detailed simulations, since the CT ramp is dependent upon the imager and the data acquisition method.

16.3   Sample `ctcreate` CT Phantom Input File

Table 3 shows an example `ctcreate` input file, and a sample DOSXYZnrc input file (excluding EGSnrc inputs) which uses the CT phantom data from `ctcreate`.

16.4   Location of `ctcreate` and How to Compile It

`ctcreate.mortran` and related files (see section 2.1 for a complete list) reside in the subdirectory `$OMEGA_HOME/progs/ctcreate`. Normally, it is compiled as part of the OMEGA/BEAM installation (see the BEAMnrc Manual[12] for installation instructions), however it can be compiled separately by going into this directory and typing `make`.

16.5   `ReadCT()` Subroutines

`ReadCT` is the most important subroutine in `ctcreate`. It handles the reading in of header information and binary CT data. Currently, there are three `ReadCT` subroutines within `ctcreate`, one for each of the CT formats supported: `ReadCT_Pinnacle` for Pinnacle, `ReadCT_CADPLAN` for CADPLAN data and `ReadCT_DICOM` for DICOM data.

16.5.1   `ReadCT_Pinnacle`

`ReadCT_Pinnacle` is contained within the `ctcreate.mortran` file and calls other subroutines and functions in this file:

`ReadReal` Reads real values from the image header (`.header`) file.

`ReadInt` Reads integer values from the header file.

`read_ct_data` Reads binary CT data from the image (`.img`) file.

`swap_bytes` Swaps bytes of CT data if there is a byte order mismatch between the data and the machine you are using.

Note that in Pinnacle format CT numbers are > 0. Thus, the default CT ramp in `ctcreate` cannot be used to convert CT numbers to media and densities. The example `ctcreate` inputfile, `$OMEGA_HOME/progs/ctcreate/ctcreate_examples/CT_create.inp` (used in Lab VI in the BEAMnrc Workshop), is used to convert Pinnacle format data and contains a custom CT ramp which is just the default ramp shifted up by 1025 HU.

16.5.2   `ReadCT_CADPLAN`

`ReadCT_CADPLAN` is also contained within the `ctcreate.mortran` file. Unlike `ReadCT_Pinnacle`, `ReadCT_CADPLAN` is a self-contained subroutine. Byte swapping is not required since single bytes are read at a time.

Table 3: Sample input files for `ctcreate` and DOSXYZnrc

Input Parameter	Description of input fields.
	for `ctcreate`
`DICOM`	CT data is in DICOM format
`CTfname`	in DICOM format, this is the name of a file containing the names of the individual image slices (in order of increasing Z).
`-17.64,17.64,-17.64,17.64,-0.7,4.3`	The subvolume of the CT data set that is to be used for the phantom.
`1.0, 1.0, 1.0`	The phantom voxel size.
`4, -1024`	`num_material`, `material_ct_lower_bound`. If `num_material`=0 the default ramp will be used. The default values are listed explicitly in this example.
`AIR700ICRU`	The material_name for the first material.
`-974,0.001,0.044`	CT ramp parameters for the first material: `material_ct_upper_bound`, `material_density_lower_bound`, `material_density_upper_bound`
`LUNG700ICRU`	The material name for the second material.
`-724,0.044,0.302`	The ramp parameters for the second material.
`ICRUTISSUE700ICRU`	The material name for the third material.
`101,0.302,1.101`	The ramp parameters for the third material.
`ICRPBONE700ICRU`	The material name for the fourth material.
`1976,1.101,2.088`	The ramp parameters for the fourth material.
	for DOSXYZnrc
`Sample CT Input File`	Title of the simulation and phantom.
`0`	`NMED` - If this is zero the code switches to CT_Phantom mode.
`CTfname.egsphant`	The name of the CT phantom file.
`0.7,0.01,5.0`	`ECUTIN`, `PCUTIN`, `SMAX` (dummy inputs)
`1,0,1`	`zeroairdose`, `doseprint` and `MAX20`
`-1,0,-1.25,1.25,-2.25,2.25,90.0,` `90.0,0.0`	The source description record. At this point the CT phantom input meets up with the manual phantom inputs.
`0`	monoenergetic particle spectrum.
`20.0`	particle energy.
`100,0,500.,7,3,100.,0,0,0,1,5.0,` `0,0,0,0`	Monte Carlo information (note range rejection below 5MeV).

16.5.3 `ReadCT_DICOM.c`

`ReadCT_DICOM.c` is a separate C routine whose object file is linked to `ctcreate` at compile time. `ReadCT_DICOM.c` is automatically compiled when you `make` ctcreate. `ReadCT_DICOM.c` requires the C header file `tags_ct.h`, which defines hexadecimal tags, or labels, for the DICOM data. These tags are defined in the DICOM standard, and if the user modifies `ReadCT_DICOM.c` to deal with additional tags, then these can be added to the `tags_ct.h` file. Currently, `ReadCT_DICOM` assumes the DICOM data has been collected on a machine with little endian byte order (ie compatible with Linux), however, if the data was collected on a machine with big endian byte order, you must go into `ReadCT_DICOM.c` and change the line:

`#define DICOM_ENDIAN 1`

to:

`#define DICOM_ENDIAN 0`

and recompile `ctcreate`. The endianness of the machine you are running `ctcreate` on is automatically detected by `ReadCT_DICOM` and then bytes are swapped if necessary.

The DICOM format is known to vary from imager to imager and institute to institute. Thus, we cannot guarantee that `ReadCT_DICOM` will work with your images without modification. As a minimum, the current version of `ReadCT_DICOM` requires the following information in the image (slice) header(s):

1. Number of rows (data tag `0x00280010`). This defines the number of voxels in the X-direction.

2. Number of columns (data tag `0x00280011`). This defines the number of voxels in the Y-direction.

3. Pixel spacing (data tag `0x00280030`). Defines the X- and Y-dimensions of the voxels.

4. Image position (data tag `0x00200032`). Defines the (X,Y,Z) of the centre of the first voxel. The position of the first slice (slice with lowest Z) is used to determine the X-, Y- and Z-offset (starting position) of the CT data, and the Z-positions of the first two slices are used to established voxel Z-dimension (*i.e.* slice thickness). The Z-positions of slices as read from the header are also used to sort slices in order of increasing Z, in case the user has not already done so in the file containing the slice names.

5. A pixel data tag (`0x7fe00010`) indicating the size of the data array in bytes. This tag must appear last in the header, immediately preceding the CT data itself.

`ReadCT_DICOM` assumes slices are contiguous in Z. Thus, once the Z-position of the first slice and slice thickness (Z-dimension of voxels) are established, `ReadCT_DICOM` positions subsequent slices so that there are no gaps or overlaps between them. A warning message is output if the Z-position of a slice determined by `ReadCT_DICOM` does not equal the Z-position of the slice as read from the image position in its header.

We note that `ReadCT DICOM.c` has limited flexibility in terms of slice layout, and it has not been tested on all possible variations of the DICOM format. Therefore, if you require more flexibility or are having trouble converting your DICOM files, you are strongly urged to either modify `ReadCT DICOM` or use your own stand-alone routine for converting DICOM images to `.egsphant` files.

The initial version of `ReadCT DICOM.c` was provided by Nick Reynaert at the University of Ghent.

16.5.4   Generic `ReadCT`

Essentially, `ReadCT` is the subroutine or stand-alone code which the user will have to customize or program from scratch to deal with any CT format not currently handled. The structure and number of subroutines within `ReadCT` is largely a matter of the programmer's taste. However, if `ReadCT` is to be included in `ctcreate` (or linked as an object file when compiling `ctcreate`), it must pass certain essential data back to `ctcreate`. An example call to a generic `ReadCT` routine is:

`Subroutine ReadCT(fname,asize,ctdata,offset,vsize,error)`

where `fname` is a character string containing the name of the particular CT data set. The essential CT data returned to `ctcreate` are:

`asize(3)` An array returning the number of CT voxels in the x,y,z directions.

`vsize(3)` An array returning the dimensions of the CT voxels

`offset(3)` An array returning the lower bounds of the CT data in the x,y,z directions in cm.

`ctdata($CTIMAX,$CTJMAX,$CTKMAX)` A 3-D integer array returning the CT data (Hounsfield numbers). In `ctdata`, x→i, y→j, z→ k, so that `ctdata(i,j,k)` is equal to the Hounsfield number in the voxel with lower bounds `x=offset(1)+(i-1)*vsize(1)`, `y=offset(2) + (j-1)*vsize(2)`, `z=offset(3)+ (k-1)*vsize(3)` and upper bounds `x=offset(1)+ i*vsize(1)`, `y=offset(2)+ j*vsize(2)`, `z=offset(3)+ k*vsize(3)`

`error` Returns the error status of the CT read operation. Currently not used.

16.6   Description of the `*.egsphant` File

The `*.egsphant` file output by `ctcreate` or by DOSXYZnrc itself is an ASCII file containing the following information necessary for DOSXYZnrc to simulate the CT phantom and for the display program, `dosxyz show`[23], to display the density information:

1. The number of media in the phantom

2. The names of the media

3. The `ESTEPE` value for each medium (now a dummy input)

4. The number of voxels in the X, Y and Z directions

5. A list of the voxel boundaries in the X direction

6. A list of the voxel boundaries in the Y direction

7. A list of the voxel boundaries in the Z direction

8. For each Z slice, an X-Y array containing the medium number in each voxel

9. For each slice in the Z direction, an X-Y array containing the densities in each voxel

In addition to passing the necessary medium information to DOSXYZnrc, the X-Y arrays of medium numbers (item 8 above) provide a rough slice-by-slice view of the CT phantom. A sample X-Y array of medium numbers from one slice of a CT phantom derived from a lung image is shown below (hold at a distance, and raw data on screen is easier to see):

```
111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111
111111111111111111111222222221311111111111111111111111111111
111111111111112222222222223333333222221111111111111111111111
111111122233333333333333333333333333333333333222111111111111
111222333333333333333334444444444343333333333333333221111111
333333333333333333333334433333333344333333333333333333322221
333333333333333332222233333333333333333322333333333333333333
333333333334332222222233333333333333333322222222233333333333333
333333333333222222222333333333333333333332222222222333333333333
333333333332222222222333333333333333333332222222222333333333333
333333333322222222223222233322233333333332222222222333333333333
333333333322222222223322223333333333333333222222222233333333334
433333333332222222222322223333444433333333332322222222234333333344
443333344322222222223322223344444443222222222322222234433333344
343333334432222222222223444444444322222222222222233333333443
334333333333332222222223344444333444433333333332223333333334433
333443333333333333333344344444444434433222333333344333333344333
333333443333344444333333333333334333334433333333334444333334433333
333333333344333333333333333333333333333333333333333343333333333
333333333333344333333333333333333333333333333333333333344333333333333
```

The user can use these slice-by-slice views to determine whether the CT data is being read in correctly by `ReadCT` or to verify that the sub-volume of the original CT data selected includes the physiology of interest.

16.7   Files and Macros for implementation.

Below is a short description of the files and macros required to compile and run `ctcreate` and DOSXYZnrc with CT phantoms.

dosxyznrc_user_macros.mortran ctcreate uses the version stored in
$HEN_HOUSE/user_codes/dosxyznrc to obtain $IMAX, $JMAX and $KMAX, the maximum X, Y and Z dimensions of the CT phantom allowed in DOSXYZnrc.

$CTIMAX, $CTJMAX, $CTKMAX Macros defining the maximum X, Y and Z dimensions of the CT data to be read in. These macros are in ctcreate.mortran

lnblnk1.mortran MORTRAN macro to provide the FORTRAN lnblnk function for Linux, rs6000 and HP9000 machines. ctcreate makes extensive use of this function and DOSXYZnrc uses it in CT phantom mode. It is read from the $HEN_HOUSE/src directory

17 Known Bugs/Restrictions

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.

It is recommended that when using the built-in parallel processing functionality you do not restart parallel runs that use a phase space source. In this case, parallel jobs have lost track of which chunks of the phase space source they were using in the original run and the uncertainty analysis after recombining the restarted runs will not take into account correlations due to reusing chunks of the phase space source.

18 Acknowledgments

Charlie Ma was a major contributor to the DOSXYZ code and was the senior author of the original DOSXYZ manual. Since he is not directly involved with DOSXYZnrc, he is no longer an author of the manual, but his contribution to the original code while he was at NRC still plays a significant role in the current version.

A variety of people have contributed various pieces of code to the DOSXYZnrc program. We wish to thank: Julie Zachman of UW for providing the aapm2pinnacle tool; Daryoush Sheikh-Bagheri for developing PAW macros for displaying dose distributions and CT files; Geoff Zhang and Daryoush again for extensive use of the code and drawing our attention to bugs; Alex Bielajew for early work on the code; Marc Lauterbach and Joerg Lehmann for initial coding to read CADPLAN CT data sets. We also wish to thank Paul Reckwerdt for his work on the original code to read Pinnacle CT data sets; Mark Holmes for his extensive work on reading CT data and for coding the correlated sampling routines[9]; Brian Geiser for the BTREE beam modeling method[8].

19 References

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

[2] I. Kawrakow and D. W. O. Rogers. The EGSnrc Code System: Monte Carlo simulation of electron and photon transport. Technical Report PIRS–701, National Research Council of Canada, Ottawa, Canada, 2000.

[3] 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.

[4] J. A. Treurniet and D. W. O. Rogers. BEAM, DOSXYZ and BEAMDP GUI User's Manual. *NRC Report PIRS 0623(rev A)*, 1999.

[5] W. R. Nelson, H. Hirayama, and D. W. O. Rogers. The EGS4 Code System. Report SLAC–265, Stanford Linear Accelerator Center, Stanford, California, 1985.

[6] 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.

[7] A. F. Bielajew and D. W. O. Rogers. A standard timing benchmark for EGS4 Monte Carlo calculations. *Med. Phys.*, 19:303 – 304, 1992.

[8] B. P. Geiser, P. J. Reckwerdt, T. R. Mackie, M. Holmes, and G. X. Ding. BTREE vector compression applied to Monte-Carlo phase space. *Med. Phys. (abstract)*, 12:1007, 1995.

[9] M. Holmes. Correlated Sampling in DOSXYZ. *NRC Report PIRS-0509Bi*, 1999.

[10] 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.

[11] B. R. B. Walters, J. Treurniet, D. W. O. Rogers, and I. Kawrakow. QA tests and comparisons of the EGSnrc system with EGS4. Technical Report PIRS–703, National Research Council of Canada, Ottawa, Canada, 2000.

[12] D. W. O. Rogers. Accuracy of the Burns equation for stopping-power ratio as a function of depth and R_{50}. *Med. Phys.*, 31:2961 – 2963, 2004.

[13] H. C. E. McGowan, B. A. Faddegon, and C-M Ma. STATDOSE for 3D dose distributions. *NRC Report PIRS 509f*, 1995.

[14] C.-M. Ma and D. W. O. Rogers. Beam characterization: a multiple-source model. *NRC Report PIRS 509d*, 1995.

[15] C.-M. Ma, B. A. Faddegon, D. W. O. Rogers, and T. R. Mackie. Accurate characterization of Monte Carlo calculated electron beams for radiotherapy. *Med. Phys.*, 24:401 – 416, 1997.

[16] I. Kawrakow and B. R. B. Walters. Efficient photon beam dose calculations using DOSXYZnrc with BEAMnrc. *Med. Phys.*, 33:3046 – 3056, 2006.

[17] 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.

[18] J. Lobo and I. A. Popescu. Two new dosyznrc sources for 4d monte carlo simulations of continuously variable beam configurations, with applications to rapidarc, vmat, tomotherapy and cyberknife. *Phys. Med. Biol.*, 55:4431 – 4443, 2010.

[19] D. W. O. Rogers, B. Walters, and I. Kawrakow. BEAMnrc Users Manual. *NRC Report PIRS 509(a)revL*, 2009.

[20] J. V. Siebers, P. J. Keall, J. O. Kim, , and R. Mohan. A method for photon beam Monte Carlo multileaf collimator particle transport. *Phys. Med. Biol.*, 47:3225 – 3250, 2002.

[21] S. Kim, T. T. Yoshizumi, F.-F. Yin, and I. J. Chetty. Spiral computed tomography phase-space source model in the BEAMnrc/EGSnrc Monte Carlo system: implementation and validation. *Phys. Med. Biol.*, 58:2609 – 2624, 2013.

[22] 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.

[23] I. Kawrakow. The dose visualization tool dosxyz_show. *NRC Report PIRS 0624*, 1998.

[24] G. Marsaglia and A. Zaman. A new class of random number generators. *Annals of Applied Probability*, 1:462 – 480, 1991.

[25] G. Marsaglia, A. Zaman, and W. W. Tsang. Toward a universal random number generator. *Statistics and Probability Letters*, 8:35 – 39, 1990.

[26] M. Lüscher. A portable high-quality random number generator for lattice field theory simulations. *Computer Phys. Commun.*, 79:100 – 110, 1994.

[27] F. James. RANLUX: A Fortran implementation of the high-quality pseudorandom number generator of Lüscher. *Computer Phys. Commun.*, 79:111 – 114, 1994.

[28] 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.

[29] D. W. O. Rogers. Low energy electron transport with EGS. *Nucl. Inst. Meth.*, 227:535 – 548, 1984.

[30] 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.

[31] 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.

[32] 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.

[33] H. W. Lewis. Multiple scattering in an infinite medium. *Phys. Rev.*, 78:526 – 529, 1950.

[34] 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.

[35] H. W. Koch and J. W. Motz. Bremsstrahlung cross-section formulas and related data. *Rev. Mod. Phys.*, 31:920 – 955, 1959.

[36] 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.

[37] 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.

[38] ICRU. Stopping powers for electrons and positrons. ICRU Report 37, ICRU, Washington D.C., 1984.

[39] 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.

[40] R. Ribberfors. . *Phys. Rev. B*, 12:2067, 1975.

[41] J. W. Motz, H. A. Olsen, and H. W. Koch. Pair production by photons. *Rev. Mod. Phys.*, 41:581 – 639, 1969.

[42] F. Sauter. Über den atomaren Photoeffekt in der K-Schale nach der relativistischen Wellenmechanik Diracs. *Ann. Physik*, 11:454 – 488, 1931.

[43] 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.

[44] J. H. Hubbell and I. Øverbø. Relativistic atomic form factors and photon coherent scattering cross sections. *J. Phys. Chem. Ref. Data*, 9:69, 1979.

[45] I Kawrakow. Electron impact ionization cross sections for egsnrc. *Med. Phys. (abstract)*, 29:1230, 2002.

[46] 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.

[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] I. Kawrakow, M. Fippel, and K. Friedrich. 3D Electron Dose Calculation using a Voxel based Monte Carlo Algorithm (VMC). *Med. Phys.*, 23:445 – 457, 1996.

Index