Cosima

A Cosmic Simulator for MEGAlib based on Geant4

Designed and implemented by Andreas Zoglauer (zog@ssl.berkeley.edu)

Version of 2017-05-24

1. Content

2. Prelude

2.1. What is Cosima?

Cosima is intended as a universal simulator for low-to-medium-energy gamma-ray telescopes, detecting gamma-rays via photo-effect, Compton scattering, and pair creation. This goal requires accurate simulation from a few keV up to hundreds of GeV, including particles ranging from gamma-ray to cascades triggered by cosmic high-energy particles. It has also been used for simulations for medical imaging and nuclear surveillance applications.
Cosima is full integrated into MEGAlib. For the simulations, one can use the same geometry file used with MEGAlib. The output of Cosima, the sim-file, can be used by MEGAlib's Revan and Sivan programs.

2.2. Installation

Since Cosima is part of MEGAlib, please see the MEGAlib installation instructions for a complete step by step guide.

2.3. Bug reports

If you find a bug or other problem, please email it to me: Andreas Zoglauer, zog@ssl.berkeley.edu

3. Invocation

3.1. Direct invocation

```
cosima <options> <parameter file>
```

Cosima can be started with a variety of command line options. The last option has to be the parameter file which contains the geometry, source, etc. descriptions. If this option is not given, a default parameter file will be loaded.

The other options are:

`-s <integer>`	The seed for the random number generator, which must be an integer larger than zero. By default – i.e. if the -s option is not given – the random number is initialized via the current time. In that case make sure to start multiple instances of cosima at least 1 second apart.
`-v <integer>`	Verbosity: The higher the number the more debugging output is printed to the command line: 0: Just Geant4 output is printed (Cosima itself should be silent) 1: 0 plus Cosima warnings and errors (default)

	2: 1 plus Cosima debug info 3: 2 plus some Geant4 output 4: 3 plus more Geant4 output 5: 4 plus even more Geant4 output
`-m <file name>`	Load and execute a Geant4 macro
`-i`	Enter Geant4 interactive mode --- beware that Cosima doesn't support the full interactive mode you might know from some of the Geant4 examples
`-z`	Gzip' output files after simulation
`-c <file name>`	Diagnostics: During reading the Geomega geometry it is converted to the Geant4 format. This option converts it back to the Geomega format and stores it in the given file name.

An alternate option to launch cosima is mcosima, which will start multiple parallel instances of cosima with the same parameter file (but different random seeds). By default as many parallel instances are started as the computer has CPU cores.

3.2. Multiple parallel runs on the local machine - mcosima

```
mcosima <options> <parameter file>
```

The most important options are (for a full list do `mcosima --help`):

`-t <integer>`	The number of instances to start. The default is as many instances as the CPU has cores.
`-n <integer>`	The nice level. Default is 0.
`-z`	zlib compress the output sim file.
`-w`	Wait until all runs have finished.

The individual simulation files get an additional tag, "p", to differentiate between different runs with `cosima`. For example, you will get Run1.p1.inc1.id1.sim, Run1.p1.inc2.id1.sim, and Run1.p1.sim. The first two contain the actual simulation data of the two instances, and the later one is a combination file, which can be used with other MEGAlib programs and will automatically load and concatenate the former two files.

3.3. Multiple parallel runs on distributed machines - dcosima

Cosima has a few scripts to set up multiple parallel runs on several distributed machines via one simple run invocation via dcosima (d stands for distributed). However, this requires a specific setup of the remote machines, especially all machines must be accessible via password-less ssh login (i.e. using a public key), and run the same MEGAlib version

3.3.1. Invocation

```
dcosima <options> <parameter file>
```

The key options are (attention: the format is a bit different from most of the other MEGAlib formats):

`--instance=<integer>`	The number of instances to start.

`--source=<file name>`	The source file.
`--name=<run name>`	A unique name of the run. This is actually a prefix, as the full name will be postfixed with a random number, i.e. –name=MyRun will result in a name such as MyRun_ID657258263936.
`-z`	Compress the output file name.
`--delay=<seconds>`	There is a small waiting time between searches for free simulation slots (10 seconds by default). If multiple users are running dcosima, giving it a number smaller than the default will automatically give this run a higher priority.

The options can be reduced to the minimum which is unique, e.g. instead of --instance=10 you can write –i=10. For a full list of all options, see `dcosima --help`.

3.3.2. The configuration file .dcosima.cfg

In order for dcosima to run you need a local and remote configuration files, which tell dcosima which machines to use, how many threads to use, where to store the temporary data, and when not to start a run.
A typical dcosima configuration file looks like this:

```
# List the allowed instances if run locally
instances 4

# List the local simulation directory
# Attention, this must be a full path you have write access to.
directory /home/user/dcosima

# List the nice level with which new cosima instances are launched
nice 15

# List the user names which are not allowed to be logged in for the
simulation to start
# Use all when nobody but the simulator is allowed to be logged in
inhibitors cozy omny distry anny

# List the remote machines
# Format:
# machine user@IP:port  priority
machine simy@128.32.13.12:21022   2
machine simy@128.32.13.12:21122   2
machine simy@128.32.13.12:21222   3
machine simy@128.32.13.12:21322   3
machine simy@128.32.13.12:21422   3
machine simy@128.32.13.12:21522   3
```

It contains 5 important sections: instances; directory, nice, inhibitors, and machines:

`instances`	This keyword is important on remote machines, since it contains the maximum number of instances of cosima which are allowed to run in parallel on this machine.
`directory`	This keyword gives the name of the remote directory where to store and afterwards delete the data. It must be a full path.

`nice`	The nice level with which new cosima instances are started (default: 0)
`inhibitors`	A list of user names on the remote machine. When those users are logged in no new runs are started.
`machine`	A list of the remote machines to be used for simulations. Format: machine username@IP:port priority Machines with lower priority number get filled first

3.3.3. Other dcosima tools

There is a set of other dcosima tools which help you to manage your run. For each of them do `--help` for a full list of command line options.

dcosima-updatemegalib:

Installs or updates MEGAlib on the machines. Do not try to install it be yourself. Do not run this command while dcosima simulations are running!

dcosima-checkmegalib:

Gives you an overview which megalib versions are installed where.

dcosima-listallinstances:

Gives you an overview how many instances can be run on a machine, how many are available, and how many are running where.

dcosima-listrunning:

This command lists the running instances by run name.

dcosima-showdiskspace:

This shows how much disk space is available on the remote machines to temporarily store simulation data, and which run directory consumes how much disk space.

dcosima-clean:

It sometime can happen that a run ends prematurely and its temporary data gets not deleted from the remote machines. You can remove individual run directories from all machines with this command – just don't do it for an actually running dcosima run. To figure out the run directories check with dcosima-showdiskspace.

dcosima-kill:

Sometimes it is necessary to kill a dcosima run. For this you first kill the dcosima instance on your computer, the use this command to your kill the running simulations with this command, and finally clean up the temporary disk space on the remote machines with dcosima-clean.

dcosima-rsync:

This command is used internally to rsync the data from the remote machines to your local machines. If something went wrong you can use this command to sync it back by hand.

dcosima-assemblefiles:

This command is used internally to collect all files necessary to run the simulation (source file, data, files, geometry files), and modifies them so they run locally.

dcosima-allowedinstances:

This command is used internally to check how many instances can be run on a given machine.

3.3.4. Setup procedure

Setting up the remote nodes:

Please follow this sequence to set up the remote machines:

1. Set up a user dedicated to run dcosima instances – in this example the user name is "simy"
2. Install MEGAlib in the home directory of simy
3. Create a file ~/.bash_local which initialized the MEGAlib environment when sources
4. Make sure the sshd is installed and running in order to accept ssh connections
5. Create a .dcosima.cfg file in the home directory

The next step is to enable password-less ssh login by using a private-public keypair. On the computer and with the user with whom you want to start dcosima do the following:

1. Check if you already have a key-pair ready which you can use: Check if a file .ssh/id_rsa.pub exists. Only if it does not exist, create one:
   - ssh-keygen -t rsa (Hit return for all questions!)
2. Copy the file .ssh/id_rsa.pub to ALL the remote machines (ATTENTION: Large -P), e.g. for user simy on the machine with IP 128.32.13.12 using ssh-port 21322 do:
   - scp -P 21322 .ssh/id_rsa.pub simy@128.32.13.12:
3. Login to ALL the remote machine and do
   - Make sure .ssh exists, if not create it
   - cat id_rsa.pub >> .ssh/authorized_keys
   - chmod 700 .ssh
   - chmod 640 .ssh/authorized_keys
4. Back on your GSE machine, test if you now can login password free:
   - ssh -p 21322 simy@128.32.13.12

Set up the .dcosima.cfg

This has to be done on your local machine (there the machine keyword is the most important) as well as on the remote machine (there instances, directory, and nice level are important).

Install MEGAlib on the remote machines:

This should be done exclusively with the script dcosima-updatemegalib, since it will setup the correct environment script and installs MEGAlib in the expected path.

Run dcosima

The just run dcosima:

- It is preferable to have smaller runs (1-2 hours) but many of same, instead of just having a few long runs.
- Make sure you always use the option to gzip your output files (option -z in dcosima)
- If you use the dcosima-clean or dcosima-kill scripts make sure you only use them on your own files. Currently there are no protections that you do not kill anybody else runs.

4. The sim file format

The simulation format of Cosima is basically identical with the sim format of the two other simulation interfaces of MEGAlib, GMega and ConvertMGGPOD. It consists of a header section and an event section.

In the header section you find the keywords type, version, geometry, date, and MEGAlib, which are common for most other MEGAlib event files.

Key: Type
Parameters: 1: File type
Description: The unique type of this file. In case of a sim file SIM.

Key: Version
Parameters: 1: integer
Description: Total deposited energy in sensitive material

Key: Geometry
Parameters: 1: File name
Description: Full path of the geometry file used for the simulations

Key: Date
Parameters: 1: Date & time
Description: Time and date when the file was created in the format: 2008-11-17 21:16:44

Key: MEGAlib
Parameters: 1: ID
Description: MEGAlib version as string, such as 2.19-alpha7 or 2.19.

The event section starts with the keyword TB and ends with the keyword TE, representing the start and the end of the observation time, thus TE-TB is the total observation time.

Key: TB
Parameters: 1: Time in seconds
Description: Start of the observation time

Key: TE
Parameters: 1: Time in seconds
Description: End of the observation time

Each event description consists of two main parts, the IA-section and the HT-section plus some additional information.

The IA-section contains all information about the initial parameters of the particle (INIT), and all main interactions: Compton scattering (COMP), pair creation (PAIR), photo effect (PHOT), Rayleigh scattering (RAYL), Bremsstrahlung emission (BREM), ionization (IONI), inelastic scattering (INEL), hadron capture (CAPT), decay (DECA), and finally escapes form the world volume (ESCP) as well as immediate absorption in a black absorber (BLAK). However, the IONI entry is only generated when the flag **StoreSimulationInfoIonization** is set to avoid "gigantic" file sizes. For some interactions such as PAIR creation two IA entries are generated, representing the electron and the positron. The information appearing in the fields of individual IA entries depends on the interaction type. In particular, there are many interactions which do not fill all possible fields. For example Rayleigh scattering doesn't generate a secondary particle so the fields for the new particle are empty and only the new parameters of the original particle, the photon, are filled out. In addition, for, e.g., pair creation and photo effect the mother particle direction, polarization, and energy fields are empty since the photon no longer exists, etc.
Moreover, there is a special mode **StoreSimulationInfoWatchedVolumes**, which allows keeping track when particles enter and exit selected volumes via the keywords ENTR and EXIT.
In general, IA entries are generated, when particles undergo a major change and/or new particles are generated. By default, IA entries are not generated when charged particles loose energy via ionization, irrespective of the generation of secondaries – IA entries for particles generated via ionization are only stored if the flag **StoreSimulationInfoIonization** is set to true.
Furthermore, IA entries are generated irrespective if they occurred in active or passive material, but only if the events has raised a trigger or veto signal.
Finally, the IA-section contains the "Monte-Carlo-truth". Positions, directions, and energies are the exact positions at which the interaction happened, i.e. the information is not voxelized into the voxels of the detectors, not noised, etc. All positions can be exact, since only discrete processes, which happen at a discrete position, are stored in the IA section. As consequence, from this information the path of the initial photon can be reconstructed as required for the generation of response matrices.

The HT-section contains all energy deposits as the detector would detect them, but without applying detector noise to energy and depth measurements. The positions used for the energy deposits correspond either to the location of a discrete energy deposit, or, if the energy deposit happened continuously in form of ionization along the path of the particle, half-way between the start and the stop of the simulation step. If the option **DiscretizeHits** is turned on (default), then those deposits are centered in the individual voxels of the detector. This process is also called voxelization of the energy deposits. An exception are the depth in depth-resolving strip detectors (Strip3D) and calorimeters, where the value of the z-axis in the HT-section corresponds to the energy weighted average z-position.
Even if the discretization is turned off there is not necessarily an energy deposit at the exact same position as given in the corresponding IA entry, since charged particles deposit their energy along their path, and the location in the HT-section is half-way between start and stop position of the corresponding simulation step.
In order to connect the HT's with IA's, the HT's contain a section of all interactions which contributed to the energy deposit. Since deposits are usually generated by charged particles, the interaction during which they are generated is given. For example, if an electron is generated during Compton scattering, the IA-ID of this Compton scattering is given.

The various elements in the sim file are:

Key: SE

Parameters: -

Description: "Start Event": Marks the beginning of a new event

Key: ID

Parameters: 1: ID triggered event

2: ID of simulated event causing the triggered event

Description: Represents a unique ID of the event

Key: TI
Parameters: 1: Time in seconds
Description: Observation time of the event

Key: ED
Parameters: 1: Energy in keV
Description: Total deposited energy in sensitive material including guard rings

Key: EC
Parameters: 1: Energy in keV
Description: Total energy of particles escaped from the world volume

Key: NS
Parameters: 1: Energy in keV
Description: Energy deposited in not sensitive material

Key: PM
Parameters: 1: Material name
1: Energy in keV
Description: Energy deposited in this specific passive material

Key: GR
Parameters: 1: x position of detector center in cm
2: y position of detector center in cm
3: z position of detector center in cm
4: Energy in keV
Description: Energy deposited in the guard ring of strip detectors

Key: XE
Parameters: 1: x position of detector center in cm
2: y position of detector center in cm
3: z position of detector center in cm
4: Energy in keV
Description Total energy deposit per drift chamber

Key: DR
Parameters: 1: x position of interaction in cm
2: y position of interaction in cm

3: z position of interaction in cm

4: x direction

5: y direction

6: z direction

7: Energy in keV

Description: Electron direction information for Strip3DDirectional detectors

Key: IA

Parameters: 1: Type
One if the following interaction/process types:
INIT: The initial parameters of the particle
PAIR: Pair creation
COMP: Compton scattering
PHOT: Photo effect
BREM: Bremsstrahlung
RAYL: Rayleigh scattering
IONI: Ionization (activated via StoreSimulationInfoIonization)
INEL: Inelastic scattering
CAPT: Some capture process (e.g. neutron capture)
DECA: Decay
ESCP: Particle escapes the world volume
ENTR: A particle enters a watched volume (see StoreSimulationInfoWatchedVolume)
EXIT: A particle exits a watched volume (see StoreSimulationInfoWatchedVolume)
BLAK: A particle is "killed" after entering a black absorber (see BlackAbsorber)

2: ID of this interaction

3: ID of interaction this particle originated from

4: Detector ID (see table of detector IDs)

5: Time since start of event in seconds

6: x position of interaction in cm

7: y position of interaction in cm

8: z position of interaction in cm

9: ID of original particle (see table of particle IDs)

10: New x direction of original particle

11: New y direction of original particle

12: New z direction of original particle

13: New x polarization of original particle

14: New y polarization of original particle

15: New z polarization of original particle

16: New kinetic energy of original particle in keV

17: ID of new particle (see table of particle IDs)

18: x direction of new particle

19: y direction of new particle

20: z direction of new particle

21: x polarization of new particle

22: y polarization of new particle

23: z polarization of new particle

24: Kinetic energy of new particle in keV

Description: Interaction information (version 23)

Key: HT

Parameters: 1. Detector ID (see table of detector IDs)

2. x position of interaction in cm

3. y position of interaction in cm

4: z position of interaction in cm

5: Energy deposit in keV

6: Time since start of event in seconds

7: Vector of IDs of the interactions which contributed to this hit

Description: The hit information in the active detector material.
(A Geant4 simulation detail: for "AlongStep" processes (e.g. ionization) the given position is the midpoint between start and end of the step, for "PostStep" processes (e.g. Compton scattering) the given position is the end of the step.)

Key: EN

Parameters: -

Description: "End": Marks the end of the event block. No events are allowed beyond this keyword.

Several special keywords give summaries about observation time beginning (TB) and end (TE) as well as the total simulated events.

Key: TB

Parameters: 1: Time in seconds

Description: This keyword tells the time the observation begins.
This keyword appears always before the event block, i.e. before the first SE.

Example `TB 0.0`

Key: TE

Parameters: 1: Time in seconds

Description: This keyword tells the time the observation ends in seconds.
This keyword appears always after the event block, i.e. after the EN keyword.

Example `NS 100.0`

Key: TS

Parameters: 1: Total started events

Description: This keyword represents the total number of simulated (=started) events.
This keyword appears always after the event block, i.e. after the EN keyword.

Example `TB 0.0`

During later analysis, TE-TB gives to total observation time. In case of a premature interruption of Geant4 (e.g. crash) the TE is recovered from the time of the last event in the file. The same happens with TS.

In the case the sim file reaches 95% of the maximum allowed file size on the given operating system, all following events are written to a new sim file. In the name of the new sim file the id-tag in the filename is increased by one. In addition, the last keyword in the old sim file is NF followed by the new file name. Therefore all successive files can be read if only the first file is given for analysis.
Attention: Do not use this keyword to concatenate sim files by your own! If you want to concatenate sim files use the keyword IN instead – see next section

Key: NF

Parameters: 1: File name (in this case sim file)

Description: When an IN keyword is found the given sim file is opened and read. This allows for easy concatenation of sim files. Attention: A sim file is allowed to have either an event section or an include section

Example `NF MyRun.inc1.id2.sim`

A special version of simulation file allows easily concatenating sim files. There, the event section is replaced by an include section. It contains only the keyword IN followed by the filename, and directs the analysis program to read the given sim files.
Attention:

- **A sim file is allowed to have either an event section or an include section, not both!**
- **The keywords BE, TE, TS are not allowed to appear in a sim file with contains an include section!**

Key: IN

Parameters: 1: File name (in this case sim file)

Description: When an IN keyword is found the given sim file is opened and read. This allows for easy concatenation of sim files. Attention: A sim file is allowed to have either an event section or an include section

Example

```
Type        SIM
Version     25
Geometry    Sphere.geo.setup

Date        2008-11-17 21:16:44
MEGAlib     2.18

IN FirstSimFile.sim
IN SecondSimFile.sim
```

ID	Description
1	2D Strip detector (no depth resolution)
2	MEGA style calorimeter – many scintillator bars in one enclosing volume
3	3D Strip detector with depth resolution
4	Universal detector without any position resolution (e.g. calorimeter)
5	**Not fully implemented**: Drift chamber
6	**Not fully implemented**: 3D Strip detector with depth resolution and limited directional resolution
7	Anger camera
8	3D Voxel detector

Table: List of detector Ids – the definition is identical to the definition in MEGAlib

ID	Name
1	γ
2	e^+
3	e^-
4	p
5	anti p
6	n
7	anti n
8	μ^+
9	μ^-
10	τ^+
11	τ^-
12	ν_e
13	anti ν_e
14	ν_μ
15	anti ν_μ
16	ν_τ
17	anti ν_τ

ID	Name
18	deuteron
19	triton
20	^{3}He
21	α
22	generic ion
23	π^+
24	π^0
25	π^-
26	η
27	η’
28	K^+
29	K^0
30	anti K^0
31	K^0_S
32	K^0_L
33	K^-
34	Λ

ID	Name
35	anti Λ
36	Σ^+
37	anti Σ^+
38	Σ^0
39	anti Σ^0
40	Σ^-
41	anti Σ^-
42	Ξ^0
43	anti Ξ^0
44	Ξ^-
45	anti Ξ^-
46	Ω^-
47	anti Ω^-
48	ς^-
49	ς^0
50	ς^+

Table: List of all particle IDs

5. The Parameter file

The simulation is steered by a parameter file *.source (also known as source file). It contains a description of the sources (geometry, energy, intensity, etc.) as well as general information about the simulation, how to store the data, which physics lists to use, geometry, etc.
One extremely important element for the simulation, which is not part of the parameter file, but part of geometry, is the surrounding sphere from which particles in far-field simulations are started (see Figure 1).

All options of the parameter file a described in the following sections. You can find several examples in the directory: $MEGALIB/resource/examples/cosima/sources

5.1. The seed of the random number generator

The seed for the random number generators is not given in the parameter file but as a command line option for cosima, because one should be to launch multiple instances of cosima with the some parameter file. If no seed is given at the command line, then the current time in seconds is used as seed. In this case, make sure that you launch multiple instances of cosima with at least 1 second time difference.
Internally cosima uses two random number generators, the default one of Geant4 (CLHEP) and indirectly the default random number generator of ROOT in classes which are not directly part of cosima but part of MEGAlib.

5.2. Include other files

Keyword	Include
Parameters	1: File name
Description	This option allows to include another parameter files
Example	`Include SubFile.source`

5.3. Geometry

Cosima uses MEGAlib's Geomega library to generate a Geant4 geometry. The geometry file contains the complete geometry, detector and trigger information. Concerning the detector effects engine, in Cosima only the voxelization of the hits into the voxels of the detector, an energy-loss map as a function of energy deposit, as well as a pre-trigger condition is applied to the hits. Here, pre-triggering means that vetoes and trigger thresholds are ignored. The full detector effects engine is only applied when the simulation file is read in by revan.

Keyword	Geometry
Parameters	1: File name
Description	This mandatory keyword contains the name of the Geomega geometry file
Example	`Geometry $(MEGALIB)/MyGeometry.geo.setup`

Keyword	CheckForOverlaps
Parameters	1: Number of random test points on each volumes surface (e.g. 1000)

2: Tolerance of overlaps in cm (e.g. 0.0001)

Description This option searches for overlaps in the volume tree and dumps them to the screen. No special action is taken if overlaps are found.

Example `CheckForOverlaps 1000 0.0001`

Keyword DetectorTimeConstant

Parameters 1: Time in seconds

Description For activation simulations only!
This keyword represents the time within which two decays or de-excitations are considered coincident.
This keyword does NOT do normal coincidences. The reason is that a usual space background simulation contains many background types which are simulated after each other. Therefore normal coincidence cannot be simulated within the simulations but has to be performed after wards.
Activation simulations are an exception since they separate prompt from delayed components. This procedure doesn't allow performing latter coincidences in some special cases. For example inelastic proton scattering has generated an isotope with a short meta-stable state (e.g. 1 ns). Without coincidence simulation in the prompt file the decay until the decay would appear, and in the delayed file, with a random, i.e. unrelated time stamp. Those two hits can not be joined through coincidences and an otherwise non-existent nuclear line is visible in the simulations.

Example `DetectorTimeConstant 0.000005`

5.4. Cuts by range

Two sets of global range cuts can be set via the parameter file, a default cut for all particles and the whole geometry as well as a cut for all particles by region. As usual in Geant4, the cut is a production threshold in particle range. The advantage of this approach is to be able set a higher threshold for the total geometry, but a low threshold for region surrounding the sensitive detector.

The default cut is set via the keyword **DefaultRangeCut** and corresponds to the "defaultCutValue"' in the user physics list.

Keyword DefaultRangeCut

Parameters 1: Particle range in cm

Description Global range cut for all particles

Example `DefaultRangeCut 0.0005`

The second way is to set range cuts in specific regions in the detector. This corresponds to the "Cut by Region" mechanism of Geant4: For a certain logical volume and all its daughter volumes you define a specialized range cut (for details see Geant4 manual).

Keyword Region

Parameters 1: Name of the region

Description Define a region object, used to define a detector volume in which special

	range cuts exist
Subkeyword to Region	**Region.Volume**
Parameters	1: Name of logical volume
Description	Names the logical volume, which defines the region. Please make sure that the volume is not a virtual volume. In addition if the geometry contains virtual volumes, the names of the volume might have changed in geomega. In this case use geomega to determine its new name.
Subkeyword to Region	**Region.RangeCut**
Parameters	1: Particle range in cm
Description	Range cut for all particles in this region

Example	`Region Tracker`
	`Tracker.Volume TrackerVolume`
	`Tracker.RangeCut 0.0005`

What are good cuts?
For example for silicon a cut of 0.000002 cm corresponds to a Compton electron production threshold of 0.8 keV, 0.000001 cm to 0.4 keV, and 0.0000005 cm to 0.15 keV.

If you do not give any of these options, a default cut of 0.0005 cm is used.

Attention:
It is very important to set those cuts wisely: Too low values increase your simulation time significantly, and too high values will give you wrong results, since your production thresholds are wrong!
Thus make sure that the range is lower or equal to 10% of the smallest dimension of your (sensitive) volumes (including all sub-divisioning of the actual volume, if you use e.g. strip or voxel detectors!), and that the range is small enough to produce all secondaries!!
In any case, make sure to read the Geant4 manuals to understand how range cuts work!

5.5. Physics lists

The possible physics lists for electro-magnetic processes are:

- **None**: Do not use an EM-physics list. Definitely not recommended for any normal simulations
- **Livermore**: The Livermore low-energy EM-processes (includes Doppler-broadening)
- **Livermore-Pol**: The Livermore low-energy EM-processes (includes Doppler-broadening and polarization)
- **Penelope**: The Penelope low-energy EM-processes
- **Standard**: The standard EM-processes, which are only suited for high gamma-ray energies

The **Livermore-G4LECS** package is not longer supported and superseded by the default Livermore package.

Keyword	PhysicsListEM
Parameters	ID of the physics lists (not case sensitive)
Description	Set the physics used for simulating electromagnetic processes
Default	Livermore

Example `PhysicsListEM Livermore`

The possible physics lists hadronic processes are:

- **None**
- **QGSP-BIC-HP**: This is one of the standard Geant4 physics lists (for details see Geant4 documentation), covering the energy range of particles interacting in a low- to medium-energy gamma-ray telescopes under space conditions
- **QGSP-BERT-HP**: This is one of the standard Geant4 physics lists suitable for space simulations.
- **FTFP-BERT-HP**: This is one of the standard Geant4 physics lists suitable for space simulations.

As long as you only simulate gamma-ray interaction, you do not need to give and hadron physics list!

Description	PhysicsListHD
Parameters	ID of the physics lists (not case sensitive). The possibilities are “none”, “qgsp-bic-hp”, “qgsp-bert-hp”.
Description	Set the physics used for simulating hadronic processes
Default	None
Example	`PhysicsListHD qgsp-bic-hp`

Attention: Using hadron interactions is currently under development in Cosima and has not yet been verified through comparison with real satellite data!

By default the radioactive decay physics list is always used. But you can choose how they are handled. Currently four modes exist:

- **Normal**: This is the standard Geant4 way to do, which is not very useful: The decays are added to the current event, even is they appear eons later…
- **Ignore**: Radioactive decays are completely ignored, i.e. not happening.
- **Buildup**: The decay is delayed into a new event, which is happening at the correct moment in time, if the simulation has not yet ended of course, i.e. you generate an instable isotope at time t_0, which will decay at t_1. At time t_0 the isotope is stored and the simulation continues. If time t_1 is reached, then we return to the radioactive isotope and let it decay. In this mode a list of future events is kept, which can grow indefinitely (and significantly slow down the simulation) unless a maximum time is known. Thus please use “Time” as stop criterion, so that only those events are kept which are within the given time window. In addition, if the amount of stored events exceeds 10.000.000, the events furthest in the future are deleted.
- **ActivationBuildup**: This mode is used for space activation simulations. The decays are NOT happening during the run, but the generated isotopes are stored in a list (see the chapter Activation simulations for more details). This keyword requires that you add the keyword **IsotopeProductionFile** to all runs in the source file! Do not use it for anything else but step 1 of the activation calculations!
- **ActivationDelayedDecay**: This mode simulates the delayed decays during step three of the activation simulations. Do not use it for anything else but step 3 of the activation calculations!

Description	DecayMode
Parameters	ID of the decay mode (not case sensitive). For details see text above.
Description	Sets how radioactive decays are handled
Default	Normal
Example	`DecayMode buildup`

Keyword	BlackAbsorber
Parameters	1-N: list of volume names
Description	If any particle enters a black absorber volume its track is immediately killed, and a "BLAK" IA information is generated with the particles last parameters.
Default	not used
Example	`BlackAbsorber Collimator`

5.6. Storing options

Keyword	StoreSimulationInfo
Parameters	1: mode The modes include: "all" or "true" – store all IA information "init-only" – only store the initial interactions (plus ENTR & EXIT if StoreSimulationInfoWatchedVolume is set) "none" or "false" – do not store any IA information
Description	This value controls how much simulation information like deposits in passive material, escaped particles, real interaction positions (IA-section of the sim file), etc., is stored.
Default	"all"
Example	`StoreSimulationInfo all`

Keyword	StoreCalibrated
Parameters	1: true/false
Description	Set this value to true, if the content of the HT-section should be in positions and energies instead of strips/bars and ADC counts
Default	true
Example	StoreCalibrated true

Attention: Setting this option to false is experimental and not fully supported throughout MEGAlib!

Keyword	StoreScientific
Parameters	1: true/false 2: Precision in scientific case (decimal places)
Description	If this value is set to true, the IA- and HT-sections of the output data is stored in scientific format, i.e. "a.cdefgE-xy", and not in fixed format "a.bcdef". The scientific format is more accurate, but the fixed format is more easily readable. The precision only applies to the scientific format, not to the fixed format.
Default	false
Example	`StoreScientific  true 5`

Keyword	StoreSimulationInfoIonization

Parameters 1: true/false

Description If this value is set to true, in the IA-section of the sim file also contains information about ionization – this information is also written if NO secondary has been produced. Since this increases the sim file by up to a factor 10, this option is set to false by default. This flag has only an effect if StoreSimulationInfo is set to "all".
Remark: Due to some strange Geant4 "feature", some ionizations are Geant4-internally called "Transportation". In Cosima they are still called ionization.

Default false

Example StoreSimulationInfoIonization false

Keyword StoreOnlyTriggeredEvents (aka StoreOnlyEventsWithEnergyLoss)

Parameters 1: true/false

Description If this value is set to true (default), then only events which raise a pre-trigger are stored, otherwise all events are stored (although you will only see content in the IA section of the output file).

Default true

Example
```
StoreOnlyTriggeredEvents false
```

Keyword StoreOneHitPerEvent

Parameters 1: true/false

Description In the case there is no coincidence hardware, each hit is stored in its own event. I.e. even if two hits (= hits in different detector voxels) are generated by the same particle they are stored in individual events.

Default false

Example
```
StoreOnlyEventsWithEnergyLoss false
```

Keyword StoreMinimumEnergy

Parameters 1: Energy [keV]

Description Only store events which deposit at least this amount of energy (ideal) in active detectors. It does not matter if the detectors are veto detectors or not.

Attention:
This option just cuts events out. It does not change the number of events which pre-trigger. As consequence, the stored event ID's will have gaps, and when you specify a stop criteria by trigger, it still will stop when the real number of triggered events is reached and thus you will have less than the specified number of events in the file.

Default -numeric_limits<double>::max()

Example
```
StoreMinimumEnergy 450.0
```

Keyword StoreSimulationInfoWatchedVolumes

Parameters 1-N: list of volume names

Description If any particle enters or exits a watched volume, an additional IA key is written to file indicating that the particle entered (ENTR) or left (EXIT) the volume. Obviously, neither if the particle is stopped in the volume, nor when it is created in the volume

	such an entry is created. In general, this information is for example helpful to calculate radiation damage.
Default	not used
Example	`StoreSimulationInfoWatchedVolumes Tracker Calorimeter`

Keyword	DiscretizeHits
Parameters	1: true/false
Description	Do the discretization of the energy deposits during Geant4 “steps” into the voxel/strip size of the detector. Otherwise the energy deposit of each “step” in the Geant4 simulation is stored in the HT-section of the sim file.
Default	true
Example	`DiscretizeHits false`

5.7. Defining a run

The actual simulation within Cosima is split into individual runs. A run defines a file to which the data is stored, a stop criterion, a trigger criterion, and one or more sources.

Keyword	Run
Parameters	1: Unique name of the run (no spaces allowed)
Description	Definition of a run
Example	`Run FirstRun`

Subkeyword	FileName
Parameters	1: Unique file name (no spaces allowed)
Description	Sets the name of the file the data will be stored to. The suffix will be added automatically.
Example	`FirstRun.FileName TestRun`

5.7.1. Stop criteria

The following three keywords **Events**, **Triggers** and **Time** define stop criteria. Please use only one of them!

subkeyword	Events
Parameters	1: Number of events
Description	Defines a stop criterion, fulfilled when the given number of events was simulated. In the case you generate radioactive particles (e.g. through proton irradiation) and use e.g. the build-up mode, not only the primary particles are counted, but also the secondary radioactive decays. *Remark:* In most cases the number of simulated events is (much) larger than the

number of triggered events, since not all events will generate a trigger.

Example `FirstRun.Events 10000`

Subkeyword Triggers

Parameters 1: Number of triggers

Description Defines a stop criterion, fulfilled when the given number of triggers was achieved. *Attention:* Cosima only pre-triggers i.e. ignore thresholds etc. The real triggering is done in MEGAlib's Revan, after applying measurement uncertainties and thresholds.
Remark: The number of triggered events should be the number of events written to the file; in many cases this number is smaller than the number of stimulated events, since usually not all events will result in a trigger.

Example `FirstRun.Triggers 10000`

Keyword Time

Parameters 1: Time in seconds

Description Defines a stop criterion, fulfilled when the given simulation (not CPU) time has passed. In case you perform simulations which include the build-up of radioactive elements please use this stop criterion.

Example `FirstRun.Time 1000`

5.7.2. Orientations

Orientations are used to simulate movements of sources or the detector in Cosima. An orientation in cosima consists of a coordinate system and one (or more) times, translations, and rotations. The coordinate is always either local Cartesian, or, for astrophysical simulations, Galactic. Orientations are defined in a way that everything rotates in the local fixed Cartesian coordinate system.
In general, 3 components in cosima can have orientations: the sky, the detector, or the sources.
The sky should either be fixed to the local coordinate system, or have an orientation in Galactic coordinates, where the orientation represents the pointing of the local x and z-axis in Galactic coordinates.
The detector, can only have an orientation in the local Cartesian coordinate system.
The source, can have an orientation either in the local coordinate system (then the given orientation is the translation/rotation of the beam in the given source coordinates to the local coordinate system), or in Galactic coordinates.

The options for the run are:

Sub-Keyword OrientationDetector

Parameters 1: Local (always)

2: Fixed
Or
2: File
3: Loop or NoLoop
4: file name (no spaces allowed)

Description Orientation of the detector in the local coordinate system

Example `FirstRun.OrientationDetector Local Fixed`

```
SecondRun.OrientationDetector Local File Loop MyFile.txt
```

Sub-Keyword OrientationSky

Parameters
1: Local
2: Fixed (no other options allowed!)

Or

1: Galactic
2: File
3: Loop or NoLoop
4: file name (no spaces allowed)

Or:

1: Galactic
2: Pointing
3: Pointing of the x-axis of the local coordinate system in Galactic coordinates Latitude
4: Pointing of the x-axis of the local coordinate system in Galactic coordinates Longitude
5: Pointing of the z-axis of the local coordinate system in Galactic coordinates Latitude
6: Pointing of the z-axis of the local coordinate system in Galactic coordinates Longitude

Description Orientation of the sky in the local or Galactic coordinate system

Example
```
FirstRun.OrientationSky Local Fixed
SecondRun.OrientationSky Galactic File Loop MyFile.txt
ThirdRun.OrientationSky Galactic Pointing -90 0 0 180
```

5.7.3. Triggers

Obsolete. Cosima uses the trigger criterion from the geometry file. However, it is only used as “pre-trigger”, i.e. vetoes and thresholds are ignored.

5.7.4. Defining a source

Sub-Keyword Source

Parameters 1: Name of the source

Description Defines a source

Example
```
FirstRun.Source FirstSource
```

Sub-Sub-Keyword ParticleType

Parameters 1: Particle type (see table particle IDs)

Description Give the type of particle which should be used during the simulation

Example `FirstSource.ParticleType 1`

Sub-Sub-Keyword Spectrum

Parameters 1: Spectral type (see description)
2+: Parameters

Description Give the spectral type of your source. The differential energy spectrum (like the beam profile) has no specific normalization – it only gives the shape of the spectrum. How many photons per cm^2, second, keV, steradian, etc. have to be started in the simulation is calculated internally. The absolute flux is determined via the value given in the keyword.
As consequence, all spectra can be combined with all beam options.
The following spectra are currently implemented:
`Flux`. The possibilities for the spectrum are:
`Mono`: Mono energetic (line source)
`Linear`: Linear distribution between two energies
`PowerLaw`: Power law distribution
`BrokenPowerLaw`: Broken power law distribution
`Gaussian`: Gaussian around a given energy
`BlackBody`: Black body spectrum
`File`: A spectrum given in a file

`Mono` 2: Energy in keV

`Linear` $I(E) \propto const. \quad \forall E \in [p_2, p_3]$
2: Minimum energy in keV
3: Maximum energy in keV

`PowerLaw` $I(E) \propto E^{-p_4} \quad \forall E \in [p_2, p_3]$
2: Minimum energy in keV
3: Maximum energy in keV
4: Photon index

`BrokenPowerLaw` $$I(E) \propto \begin{cases} E^{-p_5} & \forall E \in [p_2, p_4] \\ p_3^{-p_5+p_6} \cdot E^{-p_6} & \forall E \in [p_4, p_3] \end{cases}$$
2: Minimum energy in keV
3: Maximum energy in keV
4: Break energy in keV
5: Photon index min
6: Photon index max
Attention: The break energy must be within the minimum and maximum energy

`Gaussian` $$I(E) \propto e^{-0.5 \cdot \left(\frac{E-p_2}{p_3}\right)^2} \quad \forall E \in [p_2 - p_4 p_3, p_2 + p_4 p_3]$$
2: Mean in keV
3: One sigma in keV
4: Cut-off in number of sigma

BlackBody

$$I(E) \propto \frac{E^2}{e^{E/p_4} - 1} \quad \forall E \in [p_2, p_3]$$

2: Minimum energy in keV
3: Maximum energy in keV
4: Temperature in keV

NormalizedEnergy-BeamFluxFunction (one word without hyphens)

This spectrum has no options, since all input is given in a file. The spectrum requires the beam keyword FarFieldNormalizedEnergyPositionFluxFunction and not giving a flux!.

File

2: File name
The file format is described in the section "Other file formats – 1D Function", an example (Crab.source) can be found in the Cosima example directory. The first value in the DP-section describes the energy in keV, the second value the shape (arbitrary normalization) at this point as a differential energy spectrum, i.e. **the normalization must be per keV**, e.g. something like p/keV, p/s/keV, p/cm^2/s/keV, p/cm^2/s/sr/keV. The absolute normalization does not matter, since it is determined via the **Flux** keyword.

Example

```
FirstSource.Spectrum Mono 1809
```

Sub-Sub-Keyword Beam

Parameters

1: Beam type
2+: Parameters

Description

Give the beam type and all relevant parameters (see below). The beam (like the spectrum) has no specific normalization – it only gives the shape of the spectrum. The absolute flux is determined via the value given in the keyword `Flux`.

There exist two different beam categories: Far field and near field. Far-field sources are so far away that they arrive as a plane wave. To achieve this, for all far-field sources the photons are started on a disk whose center sits on the surrounding sphere, and whose normal vector points towards the center of the surrounding sphere (see Figure 1 for details). The names of all far-field sources start with "FarField". All other sources are near-field sources.

Please pay attention that for far-field sources you have to give a flux in particles/cm^2/s, for all other sources the flux is in particles/s!

ATTENTION: Make your mother/world volume large enough so that all particles are always started from within. Neither Geant4 nor Cosima is capable to detect the cases where the particle is started from outside the world/mother volume, and Geant4 behaves weirdly (sometimes right and sometime wrong) if this happens. Those errors are very hard to detect! The suggestion is to make a large world volume consisting of vacuum – then the performance penalty is minimal. However, if the material is not vacuum, then you might get a large performance penalty if the world volume is too large.

	The possibilities for the beam are:	
FarFieldPointSource	Point source on sphere. The particles are emitted from the disk defined by the surrounding sphere defined in the geometry file (see Figure 1). The direction of emission is given by theta on phi pointing inwards. **This far-field beam requires a flux in particles/cm^2/s!**	2: Theta (polar angle) in degrees 3: Phi (azimuth angle) in degrees *The definition of theta and phi follows the standard mathematical definition of spherical coordinate systems, i.e. theta starts from the positive z-axis, phi from the positive x-axis rotating in direction of the positive y-axis. The same is true for all other angles (see below).*
FarFieldAreaSource	Spherical area source, describing a segment of a sphere. Within this segment you have a homogeneous, "isotropic" emission. The emission scenario is the same as for FarFieldPointSource, i.e. the particles are emitted from the disk defined by the surrounding sphere (see Figure 1). **This far-field beam requires a flux in particles/cm^2/s!**	2: Minimum theta in degrees 3: Maximum theta in degrees 4: Minimum phi in degrees 5: Maximum phi in degrees
FarFieldFile-ZenithDependent	This beam covers all or parts of a sphere in the far-field. The parameters are given in a file, which describes a zenith angle dependent distribution – the values are not integrated over the azimuthal angle. The particles are emitted from the disk defined by the surrounding sphere (see Figure 1). **This far-field beam requires a flux in particles/cm^2/s!**	2: file name The file format is described in the section "Other file formats – 1D Function", an example (Crab.source) can be found in the Cosima example directory. The first value in the DP-section describes the zenith angle in degrees, the second value the shape (arbitrary normalization).
FarFieldNormalized-EnergyBeamFlux-Function (one word without hyphens)	This beam represents a 3D function spanning the energy-theta-phi space. Its content is a function representing flux in ph/cm^2/s/keV/sr As consequence you do not need to give a flux here! This beam requires the spectral option NormalizedEnergyBeamFluxFunction and no flux option!	2: file name
PointSource **Synonym: "Point"**	Point source in Cartesian coordinates. The particle is started with a random direction	2: x in cm 3: y in cm

	(isotropic emission). This beam requires a flux in particles/s!	4: z in cm
RestrictedPointSource **Synonym:** **"RestrictedPoint"**	Point source in Cartesian coordinates. However, only those particles are generated which hit the surrounding sphere. The position has to be outside the surrounding sphere. Consider this beam as an improved implementation of the standard PointSource beam, which only simulates the particles, which can hit the detector. The flux you give is the same as for PointSource, i.e. an isotropic emission is assumed! If you want a real cone beam use ConeBeam. This beam requires a flux in particles/s!	2: Start position x in cm 3: Start position y in cm 4: Start position z in cm
DiffractionPointSource	The emission starts at a point and its direction is defined by a file (x-axis: theta in degree, y-axis: phi in degree) and emitted in 4pi (or any part of it as defined in the file). Attention: The definition of theta and phi is along the standard coordinate system. The rotation is first around the z-axis ("counter-clockwise rotation around z") and then towards the new normal vector: first the inclination angle rotation, and then the azimuth angle rotation.	2: Start position x in cm 3: Start position y in cm 4: Start position z in cm 5: Counter-clockwise rotation around z-direction in deg 6: New normal vector of emission map x 7: New normal vector of emission y 8: New normal vector of emission z 9: file name (the file format is described in the section "Other file formats – 2D Function")
LineSource **Synonym: "Line"**	Line source in Cartesian coordinates. The particle is started with a random direction from a random point on the line. This beam requires a flux in particles/s!	2: x for minimum point in cm 3: y for minimum point in cm 4: z for minimum point in cm 5: x for maximum point in cm 6: y for maximum point in cm 7: z for maximum point in cm
RestrictedLineSource **Synonym:** **"RestrictedLine"**	Line source in Cartesian coordinates. However, only those particles are generated which hit the surrounding sphere. The (infinitely extension of the) line has to be completely outside the surrounding sphere. Consider this beam as an improved implementation of the standard LineSource beam, which only simulates the particles, which can hit the detector. The flux you give is the same as for the LineSource, i.e. an isotropic emission is assumed! This beam requires a flux in particles/s!	2: x for minimum point in cm 3: y for minimum point in cm 4: z for minimum point in cm 5: x for maximum point in cm 6: y for maximum point in cm 7: z for maximum point in cm

BoxSource **Synonym: "Box"**	Box-shaped source in Cartesian coordinates. The particle is started with a random direction from a random position within the box. This beam requires a flux in particles/s!	2: x for minimum point in cm 3: y for minimum point in cm 4: z for minimum point in cm 5: x for maximum point in cm 6: y for maximum point in cm 7: z for maximum point in cm
SphereSource **Synonym: "Sphere"**	Sphere-shaped source in Cartesian coordinates. The particle is started with a random direction from a random position within the sphere. This beam requires a flux in particles/s!	2: Center x in cm 3: Center y in cm 4: Center z in cm 5: Radius x in cm 6: Radius y in cm 7: Radius z in cm
DiskSource **Synonym: "Disk"**	Disk-shaped source in Cartesian coordinates – which can be a ring or a segment of the disk or ring. The disk is defined by an inner and outer radius as well as a height. Assuming a normal vector pointing along the z-axis, the opening angle count starts at the x-axis and goes counter clock wise! The particle is started with a random direction from a random position within the disk/ring (segment). This beam requires a flux in particles/s! Example: Disk.source	2: Center x in cm 3: Center y in cm 4: Center z in cm 5: Normal vector of disk x 6: Normal vector of disk y 7: Normal vector of disk z 5: Inner radius x in cm 6: Outer radius y in cm 7: FULL height z in cm 8: Start opening angle in deg 9: End opening angle in deg *No normalization is needed for the normal vector direction. The same is true for all other directions (see below).*
HomogeneousBeam	Homogeneous beam with circular cross section in Cartesian coordinates. The particle is started with the given direction from a random position within a disk with the given radius and position of center (the normal vector of the disk is pointing towards the emission direction). This beam requires a flux in particles/s!	2: Center of cylinder x in cm 3: Center of cylinder y in cm 4: Center of cylinder z in cm 5: Normal vector of disk x 6: Normal vector of disk y 7: Normal vector of disk z 8: Radius in cm *No normalization is needed for the normal vector direction. The same is true for all other directions (see below).*
RadialProfileBeam	Linear beam, whose beam profile (a 1D radial profile) is given by a file. The particle is started with the given direction (normal vector on the start "disk") from a random position within the extent of the profile. This beam requires a flux in particles/s! Example: Beam.source	2: Center of cylinder x in cm 3: Center of cylinder y in cm 4: Center of cylinder z in cm 5: Normal vector of disk x 6: Normal vector of disk y 7: Normal vector of disk z 8: file name (the file format is described in the section

		"Other file formats – 1D Function")
MapProfileBeam (former ProfiledBeam)	Linear beam, whose beam pattern (a 2D map) is given by a file. The particle is started with the given direction (normal vector on the disk) from a random position within the extent of the profile. Attention: The orientation of the distribution in the map is in the x-y-plane with a default emission in the positive z-direction. The rotation is first around the z-axis and then towards the new normal vector: first the inclination angle rotation, and then the azimuth angle rotation. This beam requires a flux in particles/s! Example: Beam.source	2: Center of map x in cm 3: Center of map y in cm 4: Center of map z in cm 5: Counter-clockwise rotation around z-direction in deg 6: New normal vector of map x 7: New normal vector of map y 8: New normal vector of map z 9: file name (the file format is described in the section "Other file formats – 2D Function")
ConeBeam	Point source in Cartesian coordinates, emitting a divergent beam in a given direction with a given half opening angle ("cone beam"). Within the beam, directions are homogeneously ("isotropically") distributed, i.e. the beam intensity is the same for all directions. This beam requires a flux in particles/s!	2: Start position x in cm 3: Start position y in cm 4: Start position z in cm 5: Direction x 6: Direction y 7: Direction z 8: Cone angle (half opening angle) in degrees
GaussianConeBeam	Point source in Cartesian coordinates, emitting a divergent beam in a given direction with a given half opening angle ("cone beam"). Within the beam, directions follow a Gaussian distribution of given width (standard deviation) about the beam direction, i.e. the beam intensity is brightest along the beam direction and decreases with increasing angular distance according to a Gaussian distribution. This beam requires a flux in particles/s!	2: Start position x in cm 3: Start position y in cm 4: Start position z in cm 5: Direction x 6: Direction y 7: Direction z 8: Cone angle (half opening angle) in degrees 9: 1-sigma value of Gaussian in degree
FlatMap	A 2D distribution read in by a file. The particle is started with random direction. Attention: The orientation of the distribution in the map is in the x-y-plane with a default emission in the positive z-direction. The rotation is first around the z-axis and then towards the new normal vector: first the inclination angle rotation, and then the azimuth angle rotation. This beam requires a flux in particles/s!	2: Center of map x in cm 3: Center of map y in cm 4: Center of map z in cm 5: Counter-clockwise rotation around z-direction in deg 6: New normal vector of map x 7: New normal vector of map y 8: New normal vector of map z 9: file name (the file format is described in the section

	Example: FlatMap.source	"Other file formats – 2D Function")
IlluminatedDisk	Illuminated disk in spherical coordinates. The disk has a given center, radius, and orientation. The particles are started from the surrounding sphere (not a disk on the sphere) in the beam direction, so that they pass through the disk. Since the intersection points are randomly chosen on the disk and not on a projection of the disk in particle flight direction, the particle line density is not always equal in all directions. If the orientation of disk and beam are identical, then this mode is identical with the beam mode in Cartesian coordinates. This beam requires a flux in particles/s!	2: Center of disk x in cm 3: Center of disk y in cm 4: Center of disk z in cm 5: Radius of disk in cm 6: Orientation of disk theta in deg 7: Orientation of disk phi in deg 8: Orientation of beam theta in deg 9: Orientation of beam phi in deg
IlluminatedBox	Illuminated box in spherical coordinates. The box has a given center, dimension, and orientation. The particles are started from the surrounding sphere (not a disk on the sphere) in the beam direction, so that they pass through the box. This beam requires a flux in particles/s!	2: Center of box x in cm 3: Center of box y in cm 4: Center of box z in cm 5: Half length of square side in cm 6: Orientation of box theta in deg 7: Orientation of box phi in deg 8: Orientation of beam theta in deg 9: Orientation of beam phi in deg
Volume	All particles are started from a random position within this volume (excluding its daughter volumes) und random direction	2: Volume name (must match geomega name)
Activation	Used internally for detector activation	
Example	Isotropic emission from 4π: `FirstSource.Beam FarFieldAreaSource 0.0 180.0 0.0 360.0`	

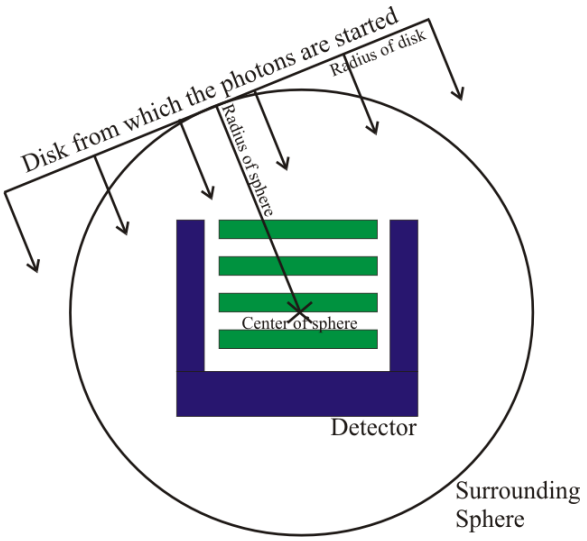

Figure 1: The concept of the surrounding sphere:
In order to simulate plane waves from distant (astrophysical) sources, the surrounding sphere has been introduced. The particles are started from a disk on the surrounding sphere. The disk always points towards the centre of the sphere, i.e. it is tangential on the disk. The direction, from which the particles originate, is defined by the theta and phi values defined in the beam parameters. The direction is given in spherical coordinates (theta: polar angle, phi: azimuth angle), where the origin is the center of the sphere. This therefore defines the start direction of the particles. The start position is a random position on the disk. All parameters (centre and radius of the sphere, radius of the disk) are defined in GEOMEGA. All parameters have to be chosen in a way that from all possible directions the detector is always completely illuminated!
For all sources in Cartesian coordinates, the start point of the particles is given in the beam description as a given point, line box, sphere, disk, etc. Those beams are not started from a disk on the surrounding sphere.

Sub-Sub-Keyword Flux

Parameters 1: Flux in particles/cm^2/s for beams starting with FarField in their name thus arriving as a plane wave **OR** Intensity in particles/s for all other sources

Description In order to enable the combination of every beam with every spectrum (and – not yet implemented – each light curve), neither the beam nor the spectrum have an absolute normalization, they only describe their shapes! The only exception is the beam FarFieldNormalizedEnergyBeamFluxFunction, where the flux is already contained in the 3D function.

The total flux is given by this keyword.

If you have a far-field beam, i.e. each beam starting with "FarField-" such as FarFieldPointSource, or FarFieldAreaSource, then the (average total) flux is given in particles/cm^2/s. All other beams require particles/s!

A special case is the beam RestrictedPointSource: You give the flux as if the source would emit in 4π. But actually simulated are only those events hitting the surrounding sphere.

Examples on how to determine the correct flux:
In order to get the correct flux value if you have a spectrum given in ph/cm^2/s/sr/keV and the FarFieldAreaSource beam type, you have to integrate over keV (in the selected energy band!) and the solid angle in steradian covered by the FarFieldAreaSource. Please take a look at the example "Crab.source".
If you have a FarFieldPointSource and a spectrum given in ph/cm^2/s/keV, then you of course only have to integrate over the energy.
For example, if you have a flat homogeneous beam with a flux given in ph/cm^2/s,

then you still have to integrate over (in this case simply multiply with) the chosen area of the beam to get to the requested flux of ph/s in the near field.

Sanity checks:
If you want to do a sanity check on the number of simulated events (not the number of triggers!) after a certain observation time for a far-field source, then you have to multiply the flux with the observation time t and the start area A, which is πr^2 with r the radius of the surrounding sphere, i.e. N = t * A * F, where t is the observation time and F is the flux (which is always in ph/cm^2/s in the far field).
For all near-field sources (all sources without a "Far Field" in the name) the simulated number of particles N should be simply N = t * F, where t is the observation time and F is the flux (which is always in ph/s in the near field).

Default 1.0

Example `FirstSource.Flux 1.0`

Sub-Sub-Keyword Polarization

Parameters 1: Spectral type (see description)
2+: Parameters

Description Give a polarization to the gamma-ray. Make sure you use a physics-list capable of handling polarization. The following types are available:
None
Random
Absolute
RelativeX, RelativeY, RelativeZ

`None` The polarization vector is set to zero and ignored during the simulation.

`Random` A random polarization vector orthogonal to the direction of the photon is used

`Absolute` Use a polarization vector in global coordinates.
Attention: The vector MUST be orthogonal to the direction of the photon. If this is not the case a random vector is used!
Parameters:
2: Degree of polarization from 0 to 1 where 1 means 100% linearly polarized, 0.5 means 50% are linearly polarized, and 50% have a random polarization vector
3: x-direction of polarization vector
4: y-direction of polarization vector
5: z-direction of polarization vector

`RelativeX`, `RelativeY`, `RelativeZ` Use a polarization vector which is created the following way: Create an initial polarization vector which is orthogonal on the direction vector and the given axis vector (e.g. x-axis for RelativeX). This is a simple cross-product. Then rotate the polarization vector (right-hand-way) around the photon direction by the given rotation angle.
Parameters:
2: Degree of polarization from 0 to 1 where 1 means 100% linearly polarized, 0.5 means 50% are linearly polarized, and 50% have a random polarization vector
3: Rotation around direction vector in degree.

Default None – no polarization, not even random

Example `FirstSource.Polarization RelativeX 1.0 45`

Sub-Sub-Keyword	Orientation
Parameters	1: Local or Galactic (Coordinate system, Galactic can only be used far-field sources!) One of the following: No orientation change (default) 2: Local 3: Fixed 2: Local 3: File 4: Loop or NoLoop 5: Filename 2: Galactic 3: Pointing 4: Galactic Latitude in degree 5: Galactic Longitude in degree
Description	The Orientation given here results in a time dependent translation & rotation relative to the emission position and direction given in the beam parameters. For example, if you have a point source in the near field and with the positions 10/0/10, then a local orientation would contain rotations and translations of this start position in the local coordinate system.
Default	Local Fixed
Example	`FirstSource.Orientation Local File Loop Movement.txt`

An alternative to all the options above is to read in an event list from a file. This would then be the only option set for the source

Sub-Sub-Keyword	EventList
Parameters	1: File name
Description	Format of the event list file: Each event is one line of text with the following elements per line (space seperated): • Event ID • 1 of event is concurrent with previous one • Cosima particle type ID • Particle excitation • Time • Start position X in cm • Start position Y in cm • Start position Z in cm • S tart direction in X • Start direction in Y • Start direction in Z • Polarization direction in X • Polarization direction in Y • Polarization direction in Z

- Energy in keV

Example: resource/examples/advanced/EventList

Default N/A

Example `FirstSource.EventList MyList.txt`

5.8. Activation simulation

ATTENTION: THE ACTIVATION FEATURE IS EXPERIMENTAL AND SHOULD NOT YET BE USED TO PREDICT THE PERFORMANCE OF FUTURE SPACE MISSIONS – THERE ARE CURRENTLY TOO LARGE DIFFERENCES BETWEEN SIMULATIONS AND REALITY!!

ATTENTION: TO PERFORM ACTIVATION SIMULATIONS A PATCH FOR GEANT4 IS CURRENTLY REQUIRED. IT CAN BE FOUND IN THE $MEGALIB/RESOURCE/PATCHES DIRECTORY. CONSULT THE INSTALLATION MANUAL FOR FURTHER INSTRUCTIONS.

Activation simulation is a three step process: First the initial particles are simulated, and all generated nuclei – if their decay/de-excitation is not coincident with the initial particle – are stored in a list. The second step calculates the activation after a certain time of irradiation in orbit. The final step simulates the decays of the radioactive particles.
Each of these steps requires an individual source file with individual keywords. The three ActivationStepX.source files in the cosima example directory serve as a template.

A common keyword is the detector time constant.

Keyword DetectorTimeConstant

Parameters 1: Time in seconds

Description For activation simulations only!
This keyword represents the time within which two decays or de-excitations are considered coincident.

Example `DetectorTimeConstant 0.000005`

This keyword does NOT do normal coincidences just coincidences for decays and de-excitations. The reason is that a usual space background simulation contains many different background types which are simulated after each other. Therefore normal coincidence cannot be handled within the simulations, but has to be performed afterwards.
Activation simulations are an exception since they separate prompt from delayed components. This doesn't allow performing latter coincidences in some special cases. For example inelastic proton scattering has generated an isotope with a short meta-stable state (e.g. 1 ns). Ignoring coincidences, the de-excitation down to the meta-stable state would appear in the prompt data set. The de-excitation down into the ground state would be stored in the delayed data set, however with a random, i.e. unrelated time stamp. As consequence those two hits cannot be joined through coincidences and an otherwise non-existent nuclear line is visible in the simulations.

Step one of the simulation is a standard hadron simulation. Make sure to use the physics list qgsp-bic-hp. In addition, set the keyword **DecayMode** to **ActivationBuildup**. This ensures that delayed decays whose decay/de-excitation is later than the **DetectorTimeConstant**, are not simulated during this first step, but those generated nuclei (isotope, excitation state, volume) are stored in an isotope list.
A special keyword of the run gives the file name of the current **IsotopeProductionFile**:

Subkeyword to Run	IsotopeProductionFile
Parameters	1: File Name
Description	This keyword represents the file name into which all generated isotopes are stored. It also contains the observation time.
Example	`MyRun.IsotopeProductionFile MyIsotopes.dat`

This file is the input for step 2 of the activation simulation, the calculation of the activation after a certain time of irradiation. You again have to give the **DetectorTimeConstant** with the same value as above and you are required to define the **qgsp-bic-hp** physics list. But instead of a run, you define an Activator, with the keywords **IsotopeProductionFile** (input file – same as above), the **ActivationMode**, and the **ActivationFile** (output file with activation data).

Keyword	Activator
Parameters	1: Unique name of the activator (no spaces allowed)
Description	Definition of a activator
Example	`Activator SpaceActivation`

The **IsotopeProductionFile** keyword represents the output of the previous simulation step.

Subkeyword to Activator	IsotopeProductionFile
Parameters	1: File Name
Description	This keyword represents the file name from which the generated isotopes are read, including the observation time.
Example	`SpaceActivation.IsotopeProductionFile MyIsotopes.dat`

The **ActivationMode** keyword describes the length and type of the irradiation.
The mode **ConstantIrradiation** assumes that the irradiation simulated in step 1 was constant during the irradiation time. This of course also assumes that the spectrum of the irradiated particles was constant – or the variations were small enough to be approximated as constant. This is for example the case for cosmic protons irradiation in interplanetary space or in low-earth equatorial orbit.
The mode **ConstantIrradiationWithCoolDown** assumes a constant irradiation for a specific period of time. Then the irradiation stops and is followed by a certain time of cool down. The remaining activation after this cool down is calculated.
The mode **TimeProfile** requires a file with a time profile of the irradiation. This is not yet implemented and the details have yet to be worked out. The goal is to be able to simulate e.g. the effect of multiple SAA passages in detail.

Subkeyword to Activator	ActivationMode
Parameters	1: Mode 2+: Mode parameters
Mode: ConstantIrradiation	2: Length of constant irradiation in seconds
Mode: ConstantIrradiationWithCoolDown	2: Length of constant irradiation in seconds 3: Length of cool down in seconds
Mode: TimeProfile	NOT YET IMPLEMENTED 2: File name 3: Length of irradiation (the time profile can be repeated)

Description	See above text
Example	`SpaceActivation.ActivationMode ConstantIrradiation 31556736` This will calculate the activation after 1 year of constant irradiation.

The **ActivationFile** keyword represents the name of the output file, the activation per isotope, excitation state, and volume, which is of course the input file for the next step.

Subkeyword to Activator	ActivationFile
Parameters	1: File Name
Description	This keyword represents the file name into which all generated isotopes are stored. It also contains the observation time.
Example	`SpaceActivation.ActivationFile Activation.dat`

The final step is the simulation of the delayed decays. You again have to give the **DetectorTimeConstant** with the same value as above and you are required to define the **qgsp-bic-hp** physics list. Instead of defining a Source for run, you define an **ActivationSource**. This is the only data about the source you need to define. It reads the activation data from the file created in the previous step:

Subkeyword to Run	ActivationSource
Parameters	1: File Name
Description	This keyword represents the file which contains all the activation data generated in the last step
Example	`MyRun.ActivationSource Activation.dat`

As an example – or as a template – look at the source files ActivationStep1.source, ActivationStep2.source, ActivationStep3.source in the Cosima example directory, which simulates the irradiation of a Germanium sphere with protons (spectrum and intensity as expected in interplanetary space close to Earth), calculates the activation after one year in orbit, and then simulated the delayed decays.

For a paper on this topic with more details and a real world example see: Zoglauer et al., “Status of Instrumental Background Simulations for Gamma-ray Telescopes with Geant4”, 2008 IEEE NSS Conference Record, 2008.

5.9. Special options

The following contains a list of special options.

Keyword	CreateCrossSectionFiles
Parameters	1: Name of the directory in which the files are stored
Description	This is a special option to create the (macroscopic) cross section files required by revan and mimrec. It is usually only used my geomega to automatically create the cross section files, if the materials have changed. Calling this option ignores all other commands.
Example	`CreateCrossSectionFiles auxiliary`

6. Other file formats

6.1. 1D Functions

For 1D functions a very simple file format is used utilizing only 3 keywords. Here is an example:

```
IP LINLIN
DP 100 1.0
DP 200 1.3
DP 500 1.5
EN
```

`IP` stands for interpolation. You give the type of interpolation on the x- and y axis which you want, either LINLIN, LINLOG, LOGLIN, or LOGLOG. Please use one reasonable for your data. If you use a logarithmic option, make sure all data is positive!
`DP` stands for data point. You give the x and y value of your distribution.
`EN` stands for end of data.

6.2. 2D Functions

For 2D functions a more sophisticated file format is used, containing 5 keywords

```
IP LIN
XA -1.0 -0.5 0.0 0.5 1.0 (or: XB -1.0 1.0 5)
YA -1.0 -0.5 0.0 0.5 1.0 (or: YB -1.0 1.0 5)
AP 0 0 0.1
AP 1 0 0.5
AP 0 1 0.4
AP 1 1 1.0
AP 1 2 0.3
AP 2 1 0.2
EN
```

`IP` stands for interpolation. You give the type of interpolation you wish, currently only "NONE" – no interpolation – and LIN" – linear interpolation – is implemented.
`XA` stands for x-axis. You give the axis points of the x-axis in cm – only equidistant bins are allowed. As an alternative to XA you can use XB, where the first value corresponds to the first bin center, the second value corresponds to the last bin center, and the last parameter gives the number of bins.
`YA` stands for y-axis. You give the axis points of the y-axis in cm – only equidistant bins are allowed.
`AP` stands for axis point. First value is the x-axis grid ID (counting starts at 0, the ones you have given in XA and YA), the second value is the y-axis grid ID, and the last entry is the function value at this point. The AP's are allowed to be in random order. If they have a content of zero you can skip them. Attention: The values which you are giving are not bin values! They are the values of the given function at this position on the grid! See also Figure 2.
`EN` stands for end of data.

6.3. 3D Functions: Spherical

The format represents a 3D data space spanned by phi, theta, and energy. Its content is flux at the axis position in ph/cm2/s/keV/sr.

The file looks like this:

```
IP LIN

# RA axis in deg:
PA 35.1 35.2 35.3 35.4 35.5
# Dec axis in deg:
TA 5.05 5.10 5.15 5.20 5.25
# Energy axis in keV:
EA 10 15 20 25 30 35 40

AP 0 0 0  0.50
AP 0 0 1  0.25
AP 0 0 2  0.12
AP 0 0 3  0.07

# Skip the rest

EN
```

The IP line gives the interpolation type. Currently only LIN, linear interpolation, is supported.

The next three lines represent the data points on the axes at which the flux is given. PA represents the right phi-axis in degree, TA represents the theta-axis in degree, EA represents the energy-axis in keV.

The following section gives the value at the axis points for the given ID (number starting with zero!) of the data point on the three axis.
For example "AP 3 1 5 1.6" represents a flux of 1.6 ph/cm^2/s/keV/sr for the 4th axis point in the RA-axis (35.4 deg), the 2nd axis point in the DEC-axis (5.10 deg), and the 6th axis point of the energy axis (35 keV).

Make sure the last line in your file is "EN" for "The End".

Lines starting with "\#" are interpreted as comments

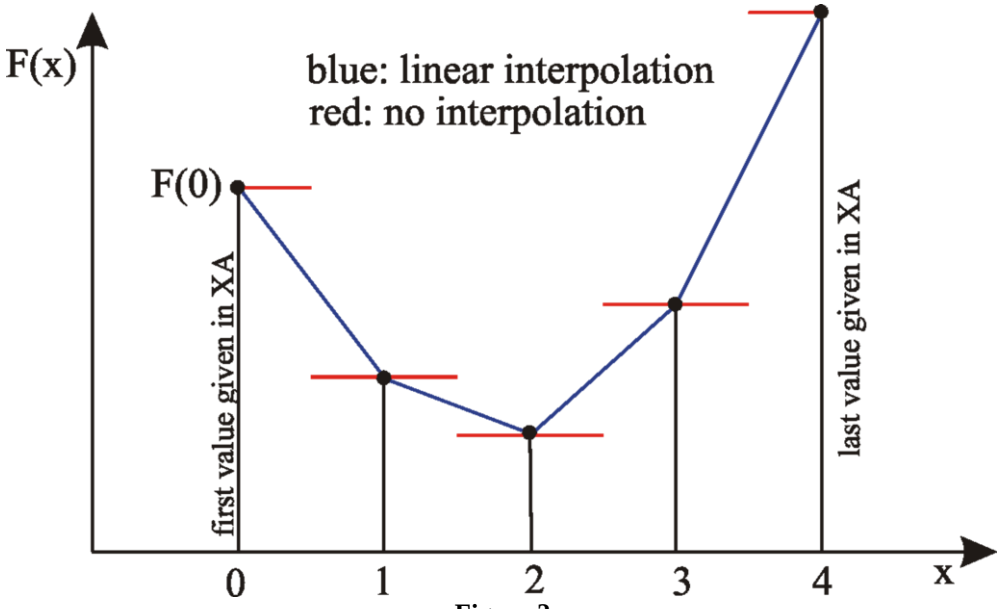

Figure 2
Principle of linear and "no" interpolation of the 2D functions: If you want the no interpolation mode to behave similar to a binned map (i.e. avoid the half binned at the edges), then set the outer bins to zero and just use the inner "bins".

7. Visualization

Visualizing the geometry and individual events is possible via the standard Geant4 mechanisms (see Geant4 manual). Thus you have to make sure to compile the desired visualization engines into Geant4 and set the corresponding `G4VIS_USE_XXX` variables. You can add them to the file "`config/Makefile.user`". However, two of them, the most frequently used ones, `G4VIS_USE_OPENGLX` ("`OGLIX`") and `G4VIS_USE_DAWNFILE` ("`DAWNFILE`"), are automatically detected during the initial configuration of MEGAlib. Visualization can then be achieved by calling a Geant4 macro file. An example can be found in: `$MEGALIB/resource/examples/cosima/macro/Visualize.mac`

8. The MCGeometryConverter class

Cosima contains a special class called MCGeometryConvert. Its purpose is to convert a Geant4 geometry into MEGAlib's Geomega format. However, only in the simplest cases this will work 100%. In the other cases you will have to make modifications by yourself.

In order to use it, you require a program capable of loading your geometry to which you can add the MCGeometryConverter.cc and MCGeometryConverter.hh files. At a point in your program when your geometry is completely initialized, add the following lines of code:

```
MCGeometryConverter* C = new MCGeometryConverter();
```

```
C->Convert("MyNewGeomegaGeometry.geo.setup");
```

After running your program, the file MyNewGeomegaGeometry.geo.setup then contains the geomega geometry.

However, many restrictions apply. The most critical one is that you can only convert volume shapes, which Geomega can understand. So it is mandatory that you investigate the output file and fix all problems, especially you have to define a trigger criteria and your detector descriptions in the geomega file.

9. Tips and Tricks

How to speed up the simulations:

- Make sure you have a hierarchical and not a flat geometry: In a flat geometry all (or most of the) volumes are in the mother volume. In a hierarchical geometry, the volumes are organized in mother volumes with daughter volumes, which have daughter volumes by themselves, and so on. This hierarchical geometry reduces the number of volumes which have to be searched in order to find the volumes which a particle path intersects.
- Define regions (see keywords "DefaultRangeCut" and "Region"): It is a good practice to make high-resolution regions close to your detector, where the particle range is on the order of ~1/5 of the smallest dimension (e.g. if your smallest dimension is a voxel size of 0.5 mm then set your range to around 0.1 mm). Away from the detector (i.e. far away that no low-energy electrons or fluorescence photons can reach it) you can set the range cut to a much larger value (e.g. a few milimeters). Whenever you change your cuts you should make a test simulation to make sure your results don't change (don't just look at the file size, since the number of IA's in the simulation file will be lower).
- Make sure you don't start too many photons which never reach your setup. For far field simulations this involves a tight surrounding sphere. For terrestrial simulations you could add a black absorber behind your source.
- Do not use an unnecessarily large world volume or you particles get tracked too long.
- If you simulate a spectrum ranging over a vast energy range, make sure that the highest energy particles (which take the longest time to simulate) really contribute to you signal. Otherwise you might able to simulate them not at all or separately to assess their impact.
- If this doesn't help use more CPUs: Use mcosima for parallel simulations one machine.

10. Known limitations

The following is an incomplete list of known problems/limitations with Geannt4/Cosima:

- If using **WatchedVolumes**, for some rare cases (1:20,000,000 for my test case) it is possible that no ENTR/EXIT entry is generated even if there was a volume change. The reason is that no step with a volume change is propagated into MCSteppingAction (tested with 9.2.3)
- The strip detectors are currently treated as voxel detectors
- Activation simulations are currently limited by the capabilities of Geant4: some nuclear lines are missing, some have wrong intensities, etc.
- The seed for the random number generator is based on the time in seconds – make sure to start identical simulations a few seconds apart to have different seeds, or supply your own seed (-s command when you start cosima)!

11. Examples

This is a brief description of the examples in the resources/examples/cosima/source directory:

- ActivationStep1.source, ActivationStep2.source, ActivationStep3.source show how to simulate activation. Step 1 simulates the primary protons and collects information about the generated radioactive isotopes. Step 2 calculates the activation after a certain amount in space, and step 3 simulated the radioactive decays.
- AllBeamsAndSpectra.source performs a combined simulation of all beams and all spectra. This shows how to use all beams and spectra, and is a cross-check that all normalizations are correct.
- Beam.source is a more detailed example on how to use the different beam types.
- BlackAbsorber.source shows how to use the black absorber feature in Cosima (all particles entering the black absorber are stopped immediately).
- CrabWithBackground.source, CrabOnly.source,, SuperCrab.source are an example on how to do astropyhsics simulations.
- Disk.source shows how to use a disk source
- EffectiveArea.source is used as an example in the Mimrec documentation showing how to determine effective areas of a space telescope.
- EnergyResolutionTester.source is used in conjunction with the EnergyResolutionTester.geo,setup geometry to test the different energy resolution modes (Gauss, Gauss-Landau, etc.)
- EntrExit.source shows how to use watched volumes resulting in the ENTR and EXIT keywords in the IA section of the sim file
- RadioactiveDecay.source sets up a volume with radioactive elements and lets them decay
- Run.source is the default example started if you launch cosima without a source file
- StartAreaTube.source shows how to use a tube as start area instead of a surrounding sphere
- Tomography.source simulates 3 point sources which have to be resolved by a simple tomography detector
- UseCase1.source is part of the tutorial