Simulations for HiCARI the high resolution cluster array at the RIBF

Kathrin Wimmer

The University of Tokyo
IEM-CSIC Madrid

November 28, 2019

Contents

0 Versions and lists of changes

Version 1.4, November 27, 2019:
Added angular distributions of γ rays and bug fixes

- in the level scheme file three numbers are added at the end of each transition line for a_0, a_2, and a_4 of the γ-ray angular distribution
- the positions of the Clover detector segments were wrong, they have been fixed (six-fold segmentation was hard coded)
- in case of atomic background simulations, the number of γ-rays per event can be 0, in this case the code crashed. This has been fixed

Version 1.3, September 27, 2019:
Added the possibility to simulate and analyze atomic background

- UCHiCARI: option to read in a histogram containing the energy and angular distribution of background from atomic processes.
- HrROOT: for events without ZeroDegree information, i.e. background events, the Doppler correction is performed with the average velocity β from the settings file.

Version 1.2, August 30, 2019:
For the simulation code a few changes were made to accommodate the Super Clover detectors

- UCHiCARI: added support for the Super Clover detectors, this affects the sizes of arrays in the simulation part in "EventAction.hh", "EventAction.cc", and "GEB.hh"
- geometry updated in the folder "Geometry" use "proposaldefault" or "proposalMINOS"
- added Bismuth as a material

In the analysis corresponding changes were

- the settings used in the analysis step are written to the output file
- if settings are written to the file, they are read in the histogramming step
- the definition of the MINOS z function has been moved to the "Calibration" class this makes the Settings class easier, the MINOS object contains the beta parameter now (derived from the z position and the function including the resolution)
- previously tracking detectors were labeled by a hole number, which refers to the GRETA hemisphere frame, this has been renamed to cluster number and is consistent between Miniball, Clover, and tracking detectors
- added section to the manual on how to gate on different detector types and angular groups, see section 4.3
- added coincidence examples between different detector types, see section 4.3

Version 1.1, June 12, 2019:

- the simulation also seems to work with GEANT4.10.5, version 10.3 does not work because the level scheme data format has changed.
- makefiles for Mac and ROOT6 have been added

- bug that prevented the simulation of cascade decays on Mac computers has been fixed
- small fixes in the example scripts
- description of add-back and tracking in the manual

Version 1.0.a, May 10, 2019:

- fixed bug with naming the analyzed trees

Version 1.0, May 8, 2019:

- first public version

1 Installation

This GEANT4 [1] simulation code was originally developed for simulations of the GRETINA array. The original version of UCGretina was developed by Lew Riley [2]. The present version of UCHiCARI includes the Miniball geometry written by Heather Crawford and some adaptions for the RIBF geometry. The output of the simulation is a .dat file in the GRETINA data format [3], identical to the output of the GRETINA Event Builder GEB. Therefore any GRETINA analysis code can be used to unpack the data. The data format of the payload for the Miniball and ZeroDegree detectors will be explained. The simulation program also calculates some parameters required for the analysis of the data (Tracking, Add-back, and Doppler-correction).

The simulation code has been developed using Geant4-10.4 [4], due to backwards compatibility issues with Geant in general, this version (or newer) is highly recommended. If you have an older version of the Geant4 package, you will not be able to install the present simulation code. For the installation of Geant4-10.4 refer to the guide written by Liliana [5]. Geant4-10.5 has also been successfully tested without any modifications to the UCHiCARI code.

1.1 Installation of the simulation program

Go to a directory of your choice and unpack the tar file:

```
cd /home/wimmer/geant
tar hzvf UCHiCARI.tar.gz
```

This creates a new sub-directory containing the source code and a few examples. To install go to that directory and make the code:

```
cd UCHiCARI
make -j8
```

Make sure that the correct Geant version is installed and the environment variables are set correctly. The script “geant4.10.4”

```
source geant4.10.4
```

can be used to set up the environment if you have several versions of Geant installed. For visualization, I am using DAWN with the Fukui Renderer, but there are many other options. Most of them are better for quick checking. Adjust the “vis.mac” file accordingly. A simple test if everything is working

is

```
UCHiCARI
Idle> /control/execute vis.mac
Idle> exit
```

which gives you the picture on the right (Fig. 1), with the detectors and the target.

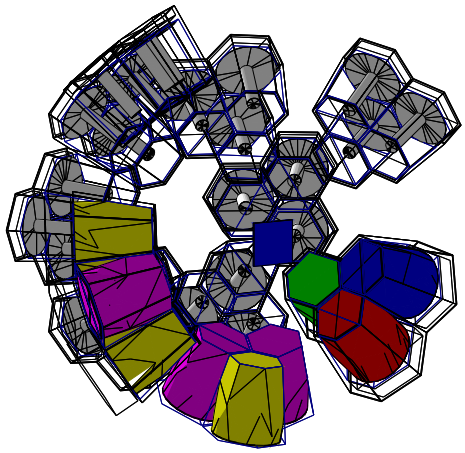

Figure 1: Testing the simulation setup using the default configuration file.

1.2 Installing the event-builder and analysis program

The package is based on the GrROOT code which was originally developed for the analysis of GRETINA data taken at the S800 at NSCL. Adjustments for the Miniball structure and the Doppler corrections based on the simulated ZeroDegree and MINOS data have been made. The output of the event-builder is a root file (structure described in section 4.2) which can be used to plot the data from the command-line. Alternatively, especially for conditional analyses, like coincidences, histograms can be produced.

The analysis software is self contained, ROOT5 needs to be installed and setup. The Makefile for Mac is called "Darwin.Makefile". For ROOT6 use the ROOT6.Makefile and the notes in the appendix 6.1. Unpack the code to a directory of you choice and compile it:

```
cd /home/wimmer/progs
tar hzvf HrROOT.tar.gz
cd HrROOT
```

In the "Makefile" lines 5 and 6 define the location of the executables and libraries.

```
BIN_DIR = $(HOME)/bin
LIB_DIR = $(HOME)/lib
```

Create these directories, if you do not have them, or change the path to your liking. Then compile the code

```
make -j8
```

If there are no errors, you have successful installed the libraries "libCommandLineInterface.so" and "libHRArray.so" into your "LIB_DIR" and the executables "SimCalculate" and "Sim_histos" to "BIN_DIR". Add "LIB_DIR" and "BIN_DIR" to your path and library-path, i.e. add the following lines to you " /.bashrc" or similar and reload it.

```
export PATH=$HOME/bin:$PATH
export LD_LIBRARY_PATH=$HOME/lib:$LD_LIBRARY_PATH
```

Then you are able to run the codes. Without any input parameters, the code will give you a short help of what it expects as input parameters. More about this will be described in section 4.

```
SimCalculate
use SimCalculate with following flags:
        [-lb <int          >: last buffer to be read]
        [-i  <char*        >: input file]
        [-o  <char*        >: output file]
        [-s  <vector<char*>>: settingsfile]
        [-rt                : write raw tree]
        [-st                : write sim tree]
No input file given
```

If you want to interact with the objects in the tree on the command line, then you need to load the library:

```
gSystem->Load("libHRArray");
```

either by executing this command before you load the file, or by adding the line to the "rootlogon.C". provided that the path (called "LIB_DIR" in the Makefile) can be searched by root. This can be achieved by adding a line like below (check your root library path).

```
Unix.*.Root.DynamicPath:    .:/usr/local/root5.34/lib/root:~/lib
```

to the ".rootrc" file.

2 Test examples

With the simulation code a few basic examples are included. You can try them following the instructions, and then modify the input files as described in section 3.1.

Note that all these examples are only illustrative, realistic count rates, thresholds, and resolutions might not be taken into account properly. Check your results!

Unpack the examples to your Geant working directory

```
cd ~/simulation/HRArray
tar hzvf examples.tar.gz
cd examples
```

you find several sub-folders with examples. To try them you need to setup the default geometry, for example:

```
cd simplesource
../change_geometry.sh ../Geometry/default/
```

Note that you have to provide the proper path to where the simulation is installed (a copy of the "Geometry" directory is provided with the examples).

2.1 Simulating a stationary source

There are two options to simulate a source measurement. One is to provide an energy in the input file. This is useful for a simple source simulation with one or a few lines. This example is in the directory "examples/simplesource" To run the example do the following

```
UCHiCARI sim1MeV.mac
SimCalculate -i source1MeV.dat -o source1MeV.root -s sourceana.set
```

In the root file you will find two trees, the simulated events in "simtr", and the detected and analyzed events in "caltr". Additionally the settings are written to the file. One simple option to draw the result is

```
root -l
root [0] TFile f("source1MeV.root")
root [1] caltr->Draw("miniballcalc.fhits.fen>>h(2000,0,2000)")
root [2] caltr->Draw("gretinacalc.fhits.fen:gretinacalc.fhits.fposition.Theta()*
         180/TMath::Pi()>>h(180,0,180,2000,0,2000)","","colz")
```

or similar to draw various parameters from the tree.

The script "sim.sh" runs stationary source from 100 keV to 5 MeV to investigate the efficiency.

```
./sim.sh
root -l
root [0] .L plot.C+
root [1] eff()
```

This produces a plot of the efficiency as function of energy as shown in Fig. 2.

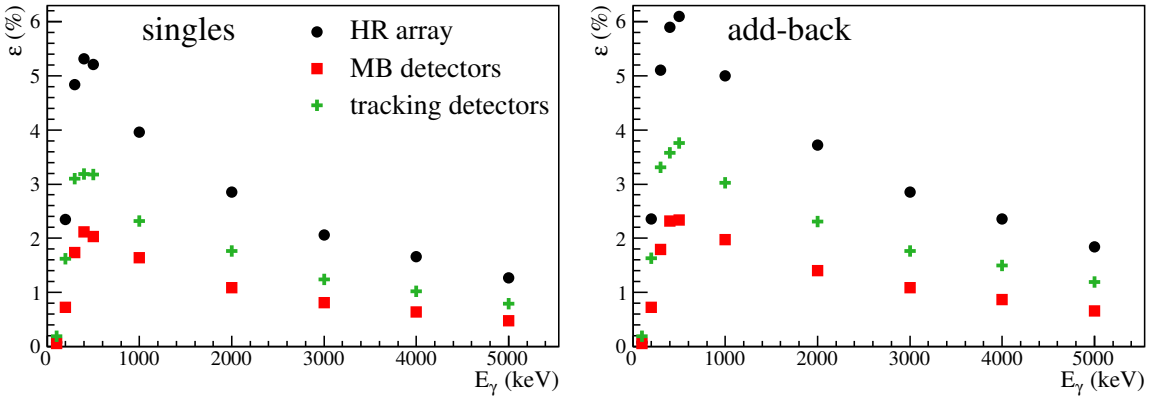

Figure 2: Efficiency for a stationary source, the full efficiency as well as the contribution of the individual detector types are shown. The left panel shows the singles efficiency, the right one the result after simple cluster add-back.

A second possibility is to use the source data that is programmed into the PrimaryGeneratorAction. This example is located in "examples/eu152". In addition to ^{152}Eu, ^{137}Cs, ^{56}Co, ^{60}Co, ^{226}Ra, and ^{241}Am sources are available by modifying the line in the input file to, for example:

```
/Experiment/Source/Set co60
```

To run the simulation, analysis, produce the histogram and compare the spectra do

```
UCHiCARI source152eu.mac
SimCalculate -i source152eu.dat -o source152eu.root -s sourceana.set
Sim_histos -i source152eu.root -o hsource152eu.root
root -l plot.C
```

2.2 A single transition in-beam

An in-beam simulation requires the definition of the beam and the reaction. The example in the directory "inbeam" is for ^{78}Ni produced in one-proton knockout from ^{79}Cu at 200 AMeV.

```
UCHiCARI sim1MeV.mac
SimCalculate -i ni78_1MeV.dat -o ni78_1MeV.root -s inbeamana.set
Sim_histos -i ni78_1MeV.root -o hni78_1MeV.root
root -l plot.C
```

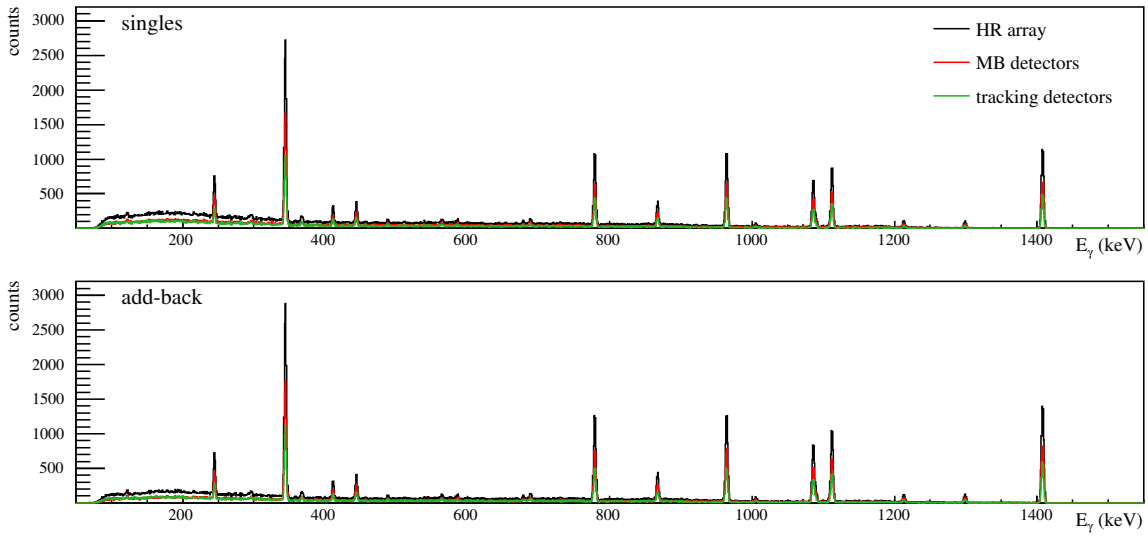

Figure 3: ^{152}Eu source simulation. The full spectrum as well as the contribution of the individual detector types are given. The top panel shows the singles spectrum, the bottom the result of simple cluster add-back.

Using the reconstructed velocity at the reaction and the positions of the γ-ray interactions (in case of the tracking detectors), the weighted detector positions (for Miniball), as well as the event-by-event information for the outgoing particle (position and angle), the Doppler correction is applied. All input for the reconstruction is provided by the simulation step (see section 4 for details). The example above produces the following plot (Fig. 4).

2.3 Complex level scheme

Typically ones wants to simulate more than one transition at the same time. A simple example is one state decaying by two branches. Such a simulation also takes into account the indirect feeding of other excited states and their subsequent decay. The first example in the folder "levelscheme" for the decay of a state at 1 MeV, directly to the ground state with a branch of 60 %, and through a state at 300 keV with 40 %. The code allows to input level scheme files, and select the populated level with the two lines

```
/BeamOut/LevelDataFile test.lvldata
/BeamOut/ProjectileExcitation 1000 keV
```

The simulation is as usually stated by:

```
UCHiCARI level.mac
SimCalculate -i level_branch.dat -o level_branch.root -s cu79ana.set
```

and the resulting spectrum shows the three γ-ray transitions.

More complex examples are also possible. The second example for ^{79}Cu is based on the published data from SEASTAR [6]. States in ^{79}Cu are populated by one-proton knockout from ^{80}Zn. In the present example a secondary Be target is used (for MINOS examples see section 2.5). The level scheme in the file "cu79.lvldata" is shown in Fig. 6. The script "sim.sh" runs the simulation for all the states with relative intensities from ref. [6]. The root script "plot.C" adds the individual states and plots the components as shown in Fig. 5.

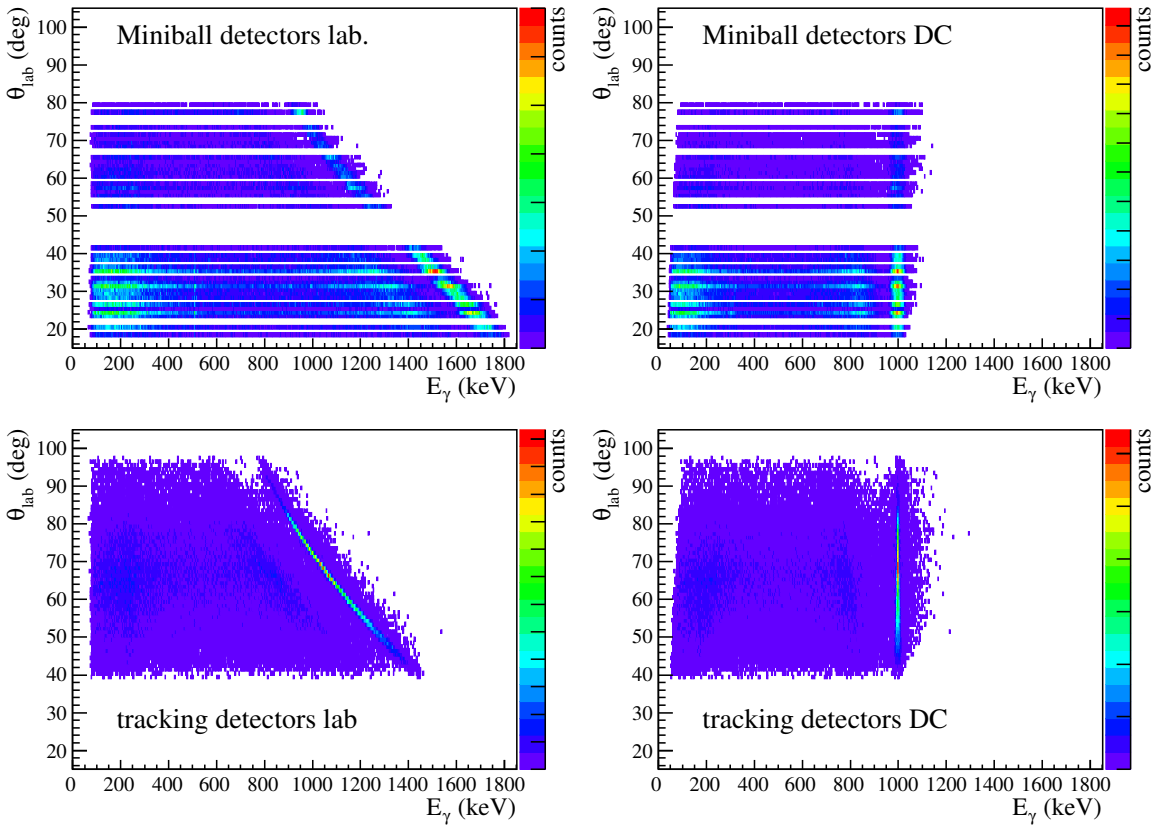

Figure 4: Doppler correction for an in-beam simulation. The top row shows the Miniball data left for the laboratory system, and right after Doppler correction. The bottom row shows the same for the tracking detectors.

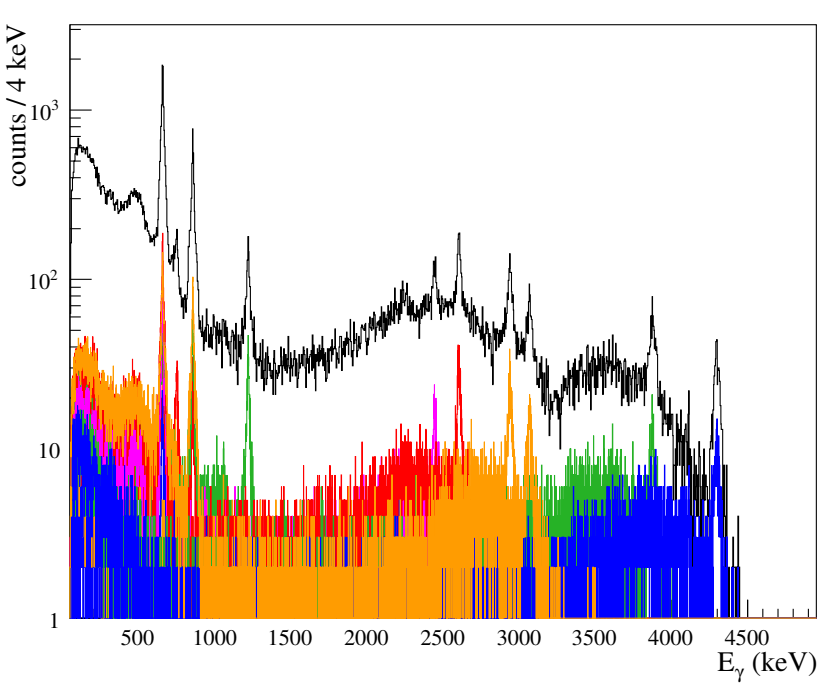

Figure 5: Simulation for ^{79}Cu.

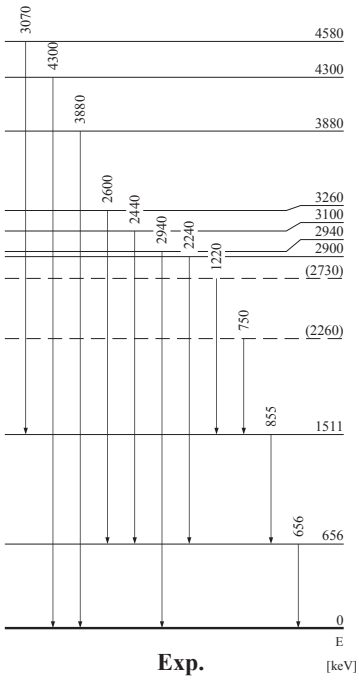

Figure 6: Level scheme of ^{79}Cu from ref. [6].

2.4 Lifetimes

Lifetimes can be implemented into the simulation by changing the level lifetime in the level scheme file. In the directory "lifetimes" a script is used to generate the simulation input. For the analysis it is assumed that the decay happens at the center of the target. States with a finite lifetime will decay further downstream or even outside of the target. This leads to emission at a lower velocity, and at a position different from the target center. For the analysis one first needs to create the generate the average velocities, and the positions of the Miniball detectors. This is done by running a simulation with a representative energy, no lifetime, and sufficient events. The file "MBcoordinates.dat" needs to be saved at a different name.

```
./sim.sh 1000 0 1000000
cp MBcoordinates.dat MBcoordinates_notau.dat
```

At the end of the simulation loop, the code also gives the average velocity β at the reaction point, at the emission point (same in this case without lifetime), and after the target.

```
#average beta at reaction (ctr=1000000):
Target.Beta:    0.536926
#average beta at emission (ctr=999959):
Target.Beta:    0.536911
#average beta after the target (ctr=999998):
Average.Beta.After:     0.524911
```

These need to be copied into the analysis settings file "cr63ana.set". To simulate a 300 keV transition with 100 ps half-life and for comparison without any lifetime use

```
./sim.sh 300 0 50000
./sim.sh 300 100 50000
root -l plot.C
```

In the latter case the emission occurs on average after the target and this leads to a tail in the spectrum as shown in Fig. 7.

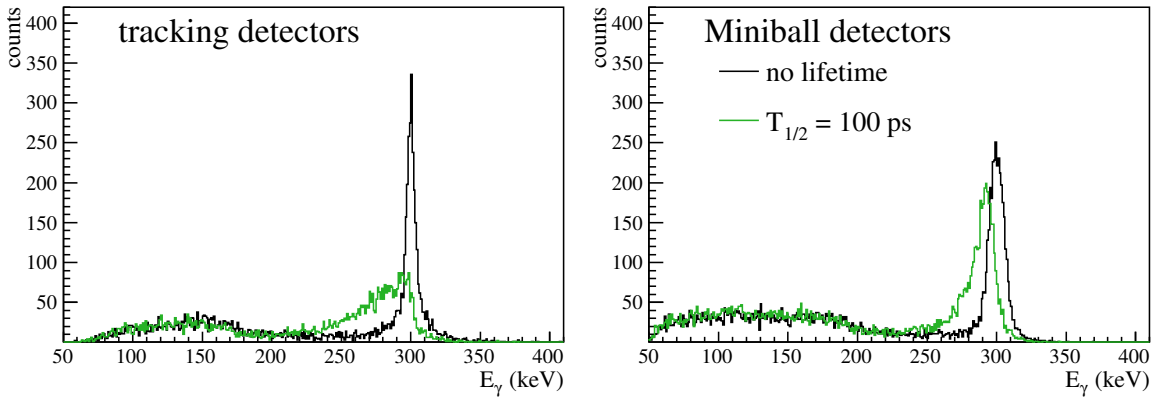

Figure 7: Spectra for the tracking detectors (left) and the Miniball detectors (both angle groups, right) for the decay of a 300 keV state without a lifetime (black) and with a half-life of 100 ps (green).

2.5 MINOS

An example with the MINOS geometry is given with the source code.

```
UCHiCARI
Idle> /control/execute minos.vis.mac
Idle> exit
```

The MINOS TPC is not yet included in the simulation framework, the beam pipe serves here as a size model for the TPC. The example with MINOS is located in the folder “minos”. The script “sim.sh”

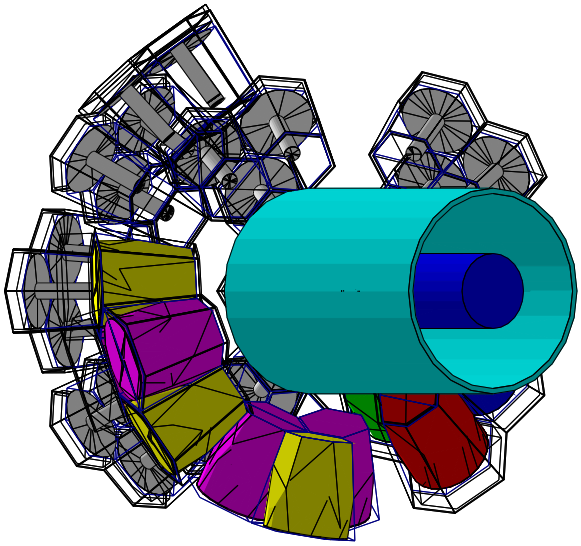

Figure 8: MINOS geometry.

executes the following lines to simulate the proton knockout reaction from ^{54}Ca on a Be and on the MINOS target.

```
../change_geometry.sh ../Geometry/tight/
UCHiCARI k53_minos.mac
SimCalculate -i k53_minos.dat -o k53_minos.root -s minos.set
SimCalculate -i k53_minos.dat -o k53_nominos.root -s nominos.set
../change_geometry.sh ../Geometry/default
UCHiCARI k53_be.mac
SimCalculate -i k53_be.dat -o k53_be.root -s be.set
Sim_histos -i k53_be.root -o hk53_be.root
```

The result after Doppler correction using either the average velocity and position with the Be and 10 cm liquid hydrogen target are shown in Fig. 9. From the vertex position reconstructed by the MINOS device, the reaction position as well as the ejectiles velocity at the reaction point can be reconstructed. Using this information the Doppler correction can be significantly improved as shown by the green histogram in Fig. 9.

2.6 Angular distributions

An example for angular distributions of γ rays is located in the folder “angulardist”. The script “sim.sh” runs a simulation for a 1 MeV transition at $\beta = 0.54$ with an isotropic and a non-isotropic one. The difference can be seen in the “ang.lvldata” file, where the last three numbers in the line describing the transition are the coefficients a_0, a_2, and a_4 (see also 3.1.6). The result is shown in Fig. 10. Note that the distribution in Fig. 10 is for the emitted γ rays at a velocity of $\beta = 0.54$. The shape of the distribution is thus dominated by the Lorentz-boost. It has to be investigated how sensitive the HiCARI array with its geometry is to angular distributions.

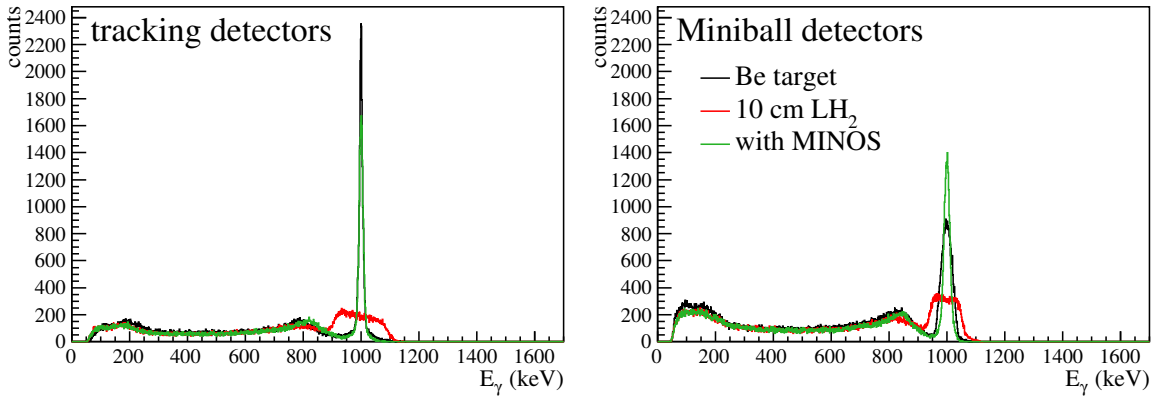

Figure 9: Doppler correction and efficiency with MINOS. The black histogram shows the result of a simulation using a 7 mm thick Be target, the red one a 10 cm thick liquid hydrogen target. The green histogram shows the result using the event by event velocity and reaction position obtained from the MINOS tracking.

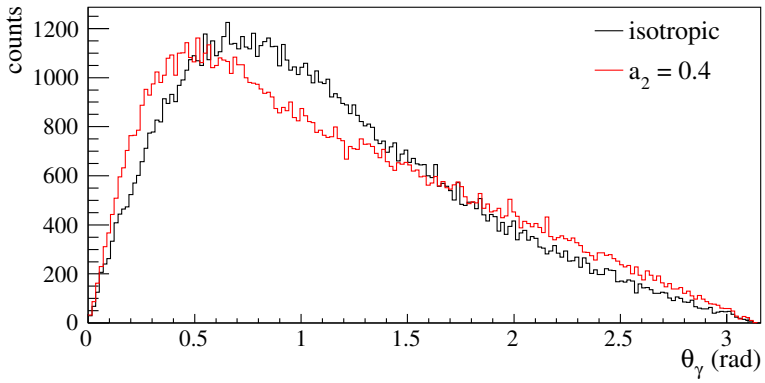

Figure 10: Angular distribution in the laboratory frame for the emitted γ rays.

3 Simulation

This section describes in more detail the simulation, the input file and the output format.

3.1 Input file

The simulation is controlled by an input file specifying the detector geometry, the reaction, the γ-ray properties, and the output file. In the following the most common input parameters are described. Further parameters are discussed in the appendix 6.3. Normally they should be kept at the default values. Note that the order of commands matters. Start from the provided examples.

3.1.1 Detectors

```
/Target/Construct
/BeamTube/Construct
/HRarray/detector/enableCapsules
/HRarray/detector/enableCryostats
/HRarray/Construct
```

The input file should start with these lines. They construct a default target, beam-pipe, and the

high resolution array. Later the parameters can be modified as described below. The geometry and placement of the germanium detectors is defined in the files “aclust”, “aeuler”, “aslice”, “asolid”, and “awalls”. Use the “change_geometry.sh” script to use a different geometry. the default geometry with six Miniball clusters at 30° at a distance of 27 cm, the remaining two Miniball clusters (without the potential additional two non standard ones) are at 65 degrees together with the tracking detectors.

#	type	psi	theta	phi	X	Y	Z
0	0	90	65	-20	-42.5825	15.4988	-21.1309
1	1	0	65	-90	0	45.3154	-21.1309
100	2	60	65	-150	-156.977	-90.6308	84.5237
200	3	0	30	0	135	0	233.8270
201	3	60	30	60	67.5	116.913	233.8270
202	3	120	30	120	-67.5	116.913	233.8270
203	3	180	30	180	-135	0	233.8270
204	3	240	30	240	-67.5	-116.913	233.8270
205	3	300	30	300	67.5	-116.913	233.8270
206	3	0	65	60	90.6308	156.977	84.5237
207	3	150	65	150	-156.977	90.6308	84.5237

Here “type” defines the type of detector, 0 for the RCNP quad, 1 for the LBNL triple, 2 for DAGATA, 3 for Miniball clusters, and 4 for the super clover detectors. “#” is the detector ID, tracking detectors should be placed first, the detector ID of the first Miniball detector needs to be known for the proper data format and analysis (see below). “psi”, “theta”, and “phi” are the angles for the rotation, “X”, “Y”, “Z” the coordinates, they can be calculated with the script “geometry.C” for a given distance.

3.1.2 Target

```
/Target/Material Be
/Target/Thickness 5 mm
/Target/SetPosition_Z 0 mm
```

Sets the target material, thickness, and position. Possible materials are various elements and compounds (others can be easily defined in “Materials.cc”), the default position is in the focus of the array. The default target is square with an area of 50×50 mm^2. When the target material is set to “MINOS” a liquid hydrogen target with a density of 76.140 kg/m^3 is constructed.

```
/Target/Material MINOS
/Target/Thickness 100 mm  #
/Target/SetPosition_Z -10 cm
```

is a typical input for simulating the MINOS device. To print the information to the command-line use

```
/Target/Report
```

3.1.3 Beam-pipe and shielding

The default beam pipe is the standard F8 beam pipe of 5 mm aluminum with a length of 1 m and in inner diameter of 14 cm. However, if needed the beam pipe diameter can be changed.

```
/BeamTube/Material Al
/BeamTube/R_max 7.5 cm
/BeamTube/R_min 7.0 cm
/BeamTube/Length 1 m
```

Usually the beam pipe is covered with shielding layers to absorb low-energy background from atomic processes. Up to two shields are possible currently. A typical choice is

```
/BeamTube/Shield1Thick 1 mm
/BeamTube/Shield1Material Pb
/BeamTube/Shield2Thick 1 mm
/BeamTube/Shield2Material Sn
```

for 1 mm of each lead and tin shields. To print the information to the command-line use

```
/BeamTube/Report
```

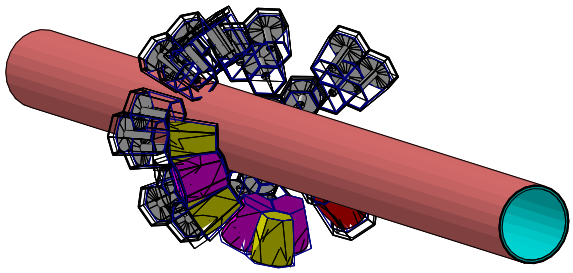

Figure 11: Simulation setup using the default configuration file, with the standard beam-pipe (light blue) covered by Pb and Sn shield (red).

3.1.4 Incoming beam parameters

```
/BeamIn/A 80
/BeamIn/Z 30
/BeamIn/KEu 200 MeV
/BeamIn/Dpp 0.00
/BeamIn/Focus/DX 10. mm
/BeamIn/Focus/DY 10. mm
```

The parameters for the incoming beam are its mass number "A", the charge "Z", and the kinetic energy per nucleon "KEu". "Dpp" is the momentum spread of the incoming beam. Note that at the RIBF the incoming beam momentum is measured on an event-by-event basis, as a good approximation the momentum spread can therefore be set to 0. "Focus/DX" and "Focus/DY" give the size of the beam spot on target. Tracking by the F8 PPACs allows to measure the target position event-by-event.

3.1.5 Reaction and outgoing beam parameters

```
/BeamOut/TargetA 9
/BeamOut/TargetZ 4
/BeamOut/DA -1
/BeamOut/DZ -1
/BeamOut/AngDistSigmaA 0.002 rad
/BeamOut/AngDistSigmaB 0.002 rad
/BeamOut/Update
```

The parameters for the reaction need the target nucleus "TargetA" and "TargetZ", the change in mass number "DA" and charge "DZ". The parameters "AngDistSigmaA" and "AngDistSigmaB" describe the width of the angular distribution of the ejectile. This is adjusted such that the scattering angle distribution matches the experimental one.

3.1.6 Level scheme and decay

The γ decay is

```
/BeamOut/LevelDataFile level/cu79.lvldata
/BeamOut/ProjectileExcitation 2260 keV
```

where the first line defines a level scheme file, and the second the state populated in the reaction. The documentation of the level scheme file format is not really existing. The "G4LevelReader.cc" source code of your Geant installation can help. The file consists of the levels and their decay properties. The level have a number for identification, a "-" (meaning unknown to me "floating"), an energy, a half-life, a spin, and the number of decay branches. Each line for a level is followed by its decay branches. The decay starts with the level that is populated in the decay, followed by the energy, the branching ratio (internally normalized), the multipolarity, the multipolarity ratio, and the conversion coefficient. The remaining numbers are conversion coefficients. A simple example for two states (ground state and 1 MeV level) is shown below.

```
0  -        0        0.1     0.0   0
1  -     1000      0e-12     2.0   1
   0     1000       100   2      0   3.6e-05    0 0 0 0 0 0 0 0 0 0
```

If you want to include a non-isotropic angular distribution for the γ-ray emission add the coefficients a_0, a_2, and a_4 at the end of the line describing the transition.

```
   0     1000       100   2      0   3.6e-05    0 0 0 0 0 0 0 0 0 0 a0 a2 a4
```

see the angular distribution example.

3.1.7 Atomic background

In order to simulate background events the output of the "abkg" code is used to create a two-dimensional energy versus theta distribution of X-rays. The "abkg" code can be found included with the DALI simulation [7]. Alternatively the python code "atomicBG.py" described in the appendix 6.2 can be used. The required input for the simulation is a line

```
/Experiment/RunBackground BACKGROUND_FILE_NAME
```

for example

```
/Experiment/RunBackground background/110zr_h_250mev.root
```

The simulation program will read the title of the histogram "h1", which contains the total cross section and angular distribution for X-rays from atomic processes. This is used, together with the target thickness, to calculate the number of photons per event. The numbers of events in this case is the number of beam particles. When the simulation runs output is written to stdout which should be carefully checked.

```
Background histogram title:
Total cross section =  6.703E+01 barndSig/dE/dtheta (b/keV/rad)
atomic cross section 67.03 barn
target thickness 100 mm
target areal density 0.7614 g/cm^2
target number density 4.58526e+23 atoms/cm^2
gammas per event 30.735
```

The cross section can also be scaled to different beams, the scaling factor should be Z^2_{beam}. The cross section can be put in by hand (in barn), instead of reading from the histogram title:

```
/Experiment/BackgroundCrossSection CROSSSECTION
```

The maximum number of γ rays simulated per event is given by "MAX_SIM_GAMMAS" in "GEB.hh". Usually this is 10, but for background typically many more γ rays will be created. To change "MAX_SIM_GAMMAS" to 100 a compiler flag has been added (comment or un-comment the line "CPPFLAGS += -DBACKGROUND" in "GNUMakefile"). If you number of γ rays per event exceeds 70 or so, you need to increase this value and recompile. For simulation of "normal" events, it is better to set "MAX_SIM_GAMMAS" back to 10, otherwise many empty lines will be written to the ".dat" output file. The number "MAX_SIM_GAMMAS" must be identical in the analysis code, otherwise the unpacker will crash. The line

```
#define MAX_SIM_GAMMAS          100
```

in the file "Simdefs.h" of the analysis code should read the same number as in "GEB.hh". One can check the number of γ rays per event by plotting the multiplicity:

```
simtr->Draw("fmult")
```

and make sure that the distribution is not cut.

3.1.8 Output description

The last part of the input file specifies the output file and format. The format of the output file is described below in section 3.2. "Mode2" is the format for GRETINA data after signal decomposition. The syntax is the following:

```
/Mode2/PackingRes 5.0 mm
/Mode2/Filename output.dat
/Mode2/FirstInteractions MBcoordinates.dat
/run/beamOn 1000000
```

Interaction points that are within the packing resolution are grouped into one interaction. Geant will place many interaction points which is not realistic when compared to the output of the signal decomposition. The output file is required. Optionally the code writes out the average first interaction points for the Miniball detectors to a file. This file is used as input for the analysis step as described in section 4.1. Lastly, you need to specify how many events to simulate.

If you use MINOS, the vertex information is written to file if the additional option

```
/Mode2/MINOS
```

is used.

3.2 Output format

The output format of the simulation is defined in "GEB.hh" and follows the GRETINA Global Event Builder format. Each detector system hit starts with a header which contains information on the length and type of data following the header.

```
struct GEBHeader{
  int32_t type;
  int32_t length; /*length of payload following the header, in bytes*/
  int64_t timestamp;
}
```

More details can be found here [3]. The "type" identifies the payload, the types and type tags are listed in Table 1. The format for the tracking detector data is defined by the GRETINA group [3].

For Miniball the data contains a type tag, the identification number of the crystal, the number of segments hit, the total energy, followed by "num" times the segment information.

detector	GEB type	type tag
tracking	1	0xabcd5678
Miniball	7	0xf005ba11
Super Clover	7	0xdeadbeef
ZeroDegree	13	0x0de90de9
MINOS	17	0xaffec0c0
simulated γ rays	11	0xabcd1234

Table 1: GEB type identifiers and type tags for the simulation.

```
struct MB_clust{
  int type;
  int crystal_id;
  int num;
  float tot_e;
  SEG seg[MBSEGS];
};
```

For each segment, there is just the number of the corresponding segment and its energy.

```
struct seg{
  int seg_id;
  float e;
} SEG;
```

The data structure for the Super Clover detectors is almost identical to the Miniball detectors, just the number of segments changes from MBSEGS (6) to CLSEGS (8).

```
typedef struct CL_clust{
  int type;
  int crystal_id;
  int num;
  float tot_e;
  SEG seg[CLSEGS];
} CL_DATA;
```

The data for the outgoing beam is the ZeroDegree physics data, containing the outgoing angles "ata" and "bta", as well as "xta" and "yta" the position on the target. Also the velocity after the target is recorded event-by-event.

```
typedef struct ZD_physicsdata {
  int32_t type;   /* defined 0de90de9 for indicating this version */
  float ata;      /* dispersive angle mrad       */
  float bta;      /* non-dispersive angle  mrad  */
  float xta;      /* dispersive position mm      */
  float yta;      /* non-dispersive position mm  */
  float betata;   /* beta velocity after target  */
};
```

For MINOS, if selected in the input file see 3.1.8, the data contains the velocity at the reaction point and the three dimensional coordinates of the reaction.

```
typedef struct MINOS_data {
  int type;       /* defined affeec0c0 for indicating this version */
  float betare;   /* beta velocity at reaction  */
  float x, y, z;  /* reaction vertex  */
} MINOS_DATA;
```

For each event, detected or not, also the information of the simulated γ rays is written to the output file. Similar to the other data types, an identifier is followed by the number of emitted γ rays, and the full energy.

```
typedef struct g4sim_abcd1234 {
  int type;          /* defined as abcd1234 */
  int num;           /* # of emitted gammas */
  int full;          /* is full energy */
  EG gammas[MAX_SIM_GAMMAS];
} G4SIM_EGS;
```

For each individual γ ray the energy, the position where it was detected and the velocity "beta" at which it was emitted is given.

```
typedef struct g4sim_emitted_gamma{
  float e;
  float x, y, z;
  float phi, theta;
  float beta;
} EG;
```

At the end of the simulation, some output which is needed for the analysis step is written out. Firstly, a file containing the average of the first interactions detected in the Miniball detectors is written. The location of this file is specified with "/Mode2/FirstInteractions" as described in section 3.1.8. For each of the Miniball segments the position of a first interaction hit is recorded and the average is written in the following format.

```
Miniball.Clu0.Cry0.Seg0.X:      196.944
Miniball.Clu0.Cry0.Seg0.Y:      0.100309
Miniball.Clu0.Cry0.Seg0.Z:      223.726
Miniball.Clu0.Cry0.Seg0.Theta:  0.72182
Miniball.Clu0.Cry0.Seg0.Phi:    0.000509326
Miniball.Clu0.Cry0.Seg0.Ctr:    1468
```

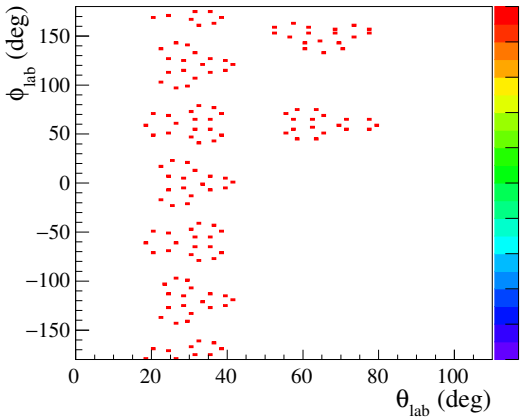

Figure 12: Average first interaction points for Miniball.

These can be plotted and checked using the "plotMBcoords.C" script. In order to get reliable positions for the Miniball detectors, one should simulate a large number of events. The simulation of one million events results in more than 1000 first interactions per segment. These positions are in a global coordinate system. If the target position is not at $z = 0$ one has to correct for this in the analysis (see section 4.1). Additionally, the average velocities at the reaction point, at the emission, and after the target are printed to the standard output. These are needed in the analysis of the data for an event-by-event Doppler correction if MINOS is not used. Depending on your case, the velocity at the reaction is identical to the one at emission (no lifetimes), and depending on which one you use in the analysis, the resulting peak will have a different shape (in case of finite lifetimes).

```
#average beta at reaction (ctr=100000):
Target.Beta:    0.536938
#average beta at emission (ctr=99997):
```

```
Target.Beta:    0.524542
#average beta after the target (ctr=100000):
Average.Beta.After:     0.524915
```

4 Analysis of simulated data

The analysis code is based on the GrROOT package [8] developed for GRETINA and S800 at the NSCL. The analysis consists of two steps. First the data is unpacked from the raw binary format described in the previous section, the structure of the objects is described in section 4.2. Based on the time stamp events are created. These raw events can be written to file, or histograms can be projected. For the analysis of simulated data as described here, this data would typically only used for debugging of the code. In the second step, the analysis is performed. This includes the event-by-event Doppler correction using the information from the ZeroDegree data and MINOS as selected. Various types of add-back and tracking can also be applied. This "calibrated" data is written in a tree which is used to display the data and project histograms.

The analysis is started by

```
SimCalculate -i INPUTFILE -o OUTPUTFILE -s SETTINGSFILE
```

where "INPUTFILE" is the ".dat" file produced in the simulation step, "OUTPUTFILE" is the path for the ".root" file, any existing file with this name will be overwritten. "SETTINGSFILE" controls the analysis, and is described in section 4.1. Additional input flags are "-lb LASTBUFFER" to read only a fraction of the data, "-rt BOOL" to write the tree containing the "raw" data to disk (not very useful), default is off, "-st BOOL" to write the tree containing the simulated γ rays to disk, default is on.

4.1 Settings file

The analysis is controlled by a settings file which is described in this section.

4.1.1 General settings

```
VerboseLevel:     0
```

How much information is printed to the standard output. Only useful for debugging, be prepared to be spammed.

```
EventTimeDiff:    300
```

Time window for the event building. In the simulation a fake time stamp is written to each data fragment. The time stamp is the event number times 1000. Therefore any number larger than 0 and smaller than 1000 is fine for the simulation.

For practical purposes the detector number assigned in the simulation (see "aeuler" file in section 3.1.1) was not consecutive, in the analysis it is possible to reassign this number. The number in the simulation is called "HoleNumber" because it used to refer to the place of the detector within the GRETA sphere. For version 1.2 the naming is changed to "ClusterNumber" to avoid ambiguities. Conversion between the two numbering schemes is provided by the "Settings::Det2Clu(int det)" and "Settings::Clu2Det(int clu)" functions. For the provided default geometries use

```
Detector.0: 1
Detector.1: 2
Detector.2: 101
Detector.3: 201
...
```

or change the settings referring to the resolution, thresholds, and coordinates discussed below.

4.1.2 Resolutions for energy, position, and velocity

The energy depositions in the γ-ray detectors are smeared with a resolution, and a threshold function is applied. The files defining the parameters are defined int he settings file.

```
Sim.Resolution.File:    RES_FILENAME
Sim.Threshold.File:     THRESH_FILENAME
```

The resolution is parameterized as

$$A \cdot \sqrt{1 + E \cdot B} + C \cdot E$$

and the coefficients A, B, and C are defined in "RES_FILENAME" either globally

```
Detector.All.Crystal.All.A: A
Detector.All.Crystal.All.B: B
Detector.All.Crystal.All.C: C
```

or for each detector individually for example by

```
Detector.0.Crystal.0.SimResolution:     1
Detector.0.Crystal.0.A:                 1.15
Detector.0.Crystal.0.B:                 0.0006
Detector.0.Crystal.0.C:                 0.0
```

here the detector number refers to the one starting typically from 0, and not the hole number (see section 4.1.1). The resolutions for the segments are identical to the central contact. Similarly, the thresholds are modeled through a function

$$\frac{1}{2} \cdot \left(1 + \tanh \frac{E - A}{B}\right)$$

whose parameters are defined in "THRESH_FILENAME"

```
Detector.0.Crystal.0.SimThreshold:      1
Detector.0.Crystal.0.E:                 A
Detector.0.Crystal.0.dE:                B
```

and so on.

For the tracking detectors a position resolution can be set by

```
Sim.Gretina.Position.Resolution:   RES
```

which will smear the interaction point positions, potentially over segment boundaries or to even outside of the detector volume. So this is currently not realistic, but the only possibility to include the position resolution.

The resolution of the ZeroDegree spectrometer for the scattering angle, the position on target and the velocity after the target are defined by

```
Target.Angle.Resolution: DA
Target.Pos.Resolution: DP
Target.Beta.Resolution: DB
```

with the σ resolutions "DA", "DP", and "DB".

4.1.3 Add-back

In order to increase the peak-to-background ratio, an add-back procedure can be applied. By selecting

```
AddBackType:    1
```

hits within the same cluster are added together before the Doppler correction is applied. A second hit vector is formed, the position, time, etc. of the hit with the higher energy are kept for the further analysis. If the "AddBackType" is set to 2, all hits are added together, regardless of the crystal or cluster they have been detected in. This calorimeter mode is good to find peaks, especially at high energy where the full energy is divided over several Compton scattering events and also pair production that can be absorbed in different crystals. Or if several a state decays by several branches the summing will lead to a peak at the energy of the state. One has to be careful for realistic data this can cause fake peaks because background events are also summed. Setting "AddBackType" to 3 will create clusters of γ-ray interaction points (only for the tracking detectors) based on the link-cluster algorithm. The maximum angle between two hits (with respect to the target) is defined in the settings file:

```
ClusterAngle:   20
```

The clusters are added to a "vector<vector<HitCalc*>>". This is the input to the tracking (which I will not describe in this manual, but if you are interested, please contact me).

4.1.4 Doppler correction

The Doppler correction requires some settings to obtain the positions of interactions and the velocity used for the reconstruction.

```
Target.X:   0
Target.Y:   0
Target.Z:   0
```

Position of the target for the Doppler correction. The coordinates of the γ-ray detectors are relative to $(x, y, z) = (0, 0, 0)$. If the target is placed at a different location in the simulation, this needs to be defined here. Note that the x and y coordinates are obtained from the ZeroDegree data event by event, so normally there should not be a reason the change "Target.X" and "Target.Y". When MINOS is used the reaction coordinates are provided event-by-event, the target position options have no effect in this case.

```
Target.Beta:          0.536924
Average.Beta.After:   0.524914
```

specify the average velocity at the target $\langle\beta_{\text{target}}\rangle$ and in ZeroDegree $\langle\beta_{\text{after}}\rangle$. Both numbers are calculated during the simulation step. For "Target.Beta" one can either use the average at the reaction, or the average velocity at emission resulting in a different Doppler correction if states have finite lifetimes. The event-by-event velocity after the target β_{after} is then used to calculate an event-by-event β for the Doppler correction.

$$\beta = \langle\beta_{\text{target}}\rangle \cdot \left(1 + \frac{\beta_{\text{after}} - \langle\beta_{\text{after}}\rangle}{\langle\beta_{\text{after}}\rangle}\right)$$

```
Ejectile.Mass:      MASS_IN_AMU
```

In order to obtain the energy and momentum of the outgoing particle set the ejectile mass in units of amu. This is not really used, and all the γ-ray analysis works without the proper number here.

The file containing average Miniball coordinates obtained in the first step of the simulation 3.2 needs to be defined in the analysis step as follows

```
Average.MBPositions: COORDINATES
```

otherwise the Doppler correction for Miniball detectors will fail.

4.1.5 Special settings for MINOS

In the case of MINOS the reaction vertex is used to calculate the β event-by-event.

```
Use.MINOS: 1
```

If this option is used for solid targets the results will be unrealistic. The use of the vertex position for the reconstruction also requires that the MINOS data has been written to disc in the simulation step. First the vertex position is smeared with a resolution

```
MINOS.XY.Resolution: Sxy  #mm
MINOS.Z.Resolution: Sz  #mm
```

where "Sxy" and "Sz" are the position resolutions (σ). The event-by-event velocity β is obtained from the smeared vertex position. A polynomial function of second order is used to parameterize the $\beta(z)$ relation. The parameters are given

```
MINOS.Beta.Coefficient.0: QUADRATIC
MINOS.Beta.Coefficient.1: LINEAR
MINOS.Beta.Coefficient.2: CONSTANT
```

for an example of how to determine the parameters see the "example/minos/beta.C" script.

4.2 Output root file

The output file contains per default two trees, "caltr" the tree containing the data for the detected events and "simtr" for the simulated data. For the class structure the user is referred to the header files, or the documentation that can be created by "make doc". Here are just the structure of the final calibrated objects described.

4.2.1 Miniball detectors

For events where a Miniball detector fired, the tree contains a leaf "MiniballCalc". This class essentially contains two vectors of hits, one for the singles and one for the hits after add-back. Their length is also stored as a short.

```
Short_t fmult;
vector<MBHitCalc*> fhits;
Short_t fmult_ab;
vector<MBHitCalc*> fhits_ab;
```

each of the hits "MBHitCalc" contains information on the detector and segment number, the energy (laboratory system and Doppler corrected), position, and the time stamp. For the add-back hits, the number of hits combined into one, as well as the energy of the maximum individual hit, are also stored.

```
Short_t fcluster;
Short_t fcrystal;
Short_t fsegment;
double fen;
double fDCen;
TVector3 fposition;
int fHitsAdded;
double fmaxhit;
long long int ftimestamp;
```

The structure for the tracking detectors "GretinaCalc" and "HitCalc" is very similar.

All the member variables can be access from the root command line. The member functions can be executed. For example to perform event-by-event Doppler correction, but instead of using the position or velocity after the target as described section 4.1.4 but a constant β in one can use:

```
caltr->Draw("gretinacalc.fhits.GetDCEnergy()>>h(1500,0,1500)")
caltr->Draw("gretinacalc.fhits.GetDCEnergy(0.549217)>>hb(1500,0,1500)","","same")
caltr->Draw("gretinacalc.fhits.GetDCEnergy(0.549217,0,0,-5)>>hz(1500,0,1500)","","same")
```

where the last line assumes the target position is shifted by 5 mm upstream.

4.2.2 ZeroDegree and MINOS

The ZeroDegree and MINOS data is very simple. ZeroDegree just contains the angles and position on the target, as well as the velocity after the target.

```
Float_t fata;
Float_t fbta;
Float_t fxta;
Float_t fyta;
Float_t fbetata;
Float_t fazita;
Float_t fscatter;
Float_t fptot;
Float_t fppar;
Float_t fptra;
Float_t fetot;
Float_t fekin;
long long int fts;
```

The momenta and the energy are calculated from the particles velocity. MINOS contains the vertex position, the velocity at the reaction point, and a time stamp. Newly added is the velocity reconstructed from the reaction point (including the resolution).

```
TVector3 fpos;
Float_t fbetare;
long long int fts;
Float_t fbeta;
```

4.2.3 Simulated γ-rays

The class for the simulated data "GammaSim" is just a container for the vector of emitted γ rays including the multiplicity and the time-stamp.

```
long long int ftimestamp;
Short_t fmult;
vector<EmittedGamma*> fgammas;
```

The "EmittedGamma" class has the emission point, direction, velocity, and energy as members.

```
Float_t fenergy;
Float_t femissionX;
Float_t femissionY;
Float_t femissionZ;
Float_t femissionPhi;
Float_t femissionTheta;
Float_t femissionBeta;
```

This data is written to the "simtr" tree of the simulated data.

4.3 Histograms

While simple operations can be done with the root command line, it is recommended to use the histograms made with the program "Sim_histos". The program is executed by simply typing

```
Sim_histos -i INPUTFILE -o OUTPUTFILE
```

Histograms can be allocated in "SimHistograms.cc", the syntax is:

```
Fill(name, nbins, binfrom, binto, value);
Fill(name, nbinsX, binfromX, bintoX, valueX, nbinsY, binfromY, bintoY, valueY);
```

for one- and two-dimensional histograms, respectively. Some basic examples are already included with the source code, and for most purposes this should be enough.

For some applications, like lifetime measurements, it is advisable to separate the spectra by detector type and angle as the sensitivity is different. Tracking and non-tracking detectors are filled into separate histograms starting with the prefix "GR" or "MB". "MB" here includes the Miniball detectors at 30 and 120 degrees, as well as the Clover detectors. The detector ID is coded in "fcluster" and "fcrystal". Cluster numbers for the non-tracking detectors start at 200, in case the provided settings (geometry "euler.list" and "FIRSTMB" in "GEB.hh") are used. You shouldn't change this, unless you really know what you are doing (it has to be consistent everywhere). The settings file provides means to convert this number to smaller ones which are easier to deal with (see 4.1.1). This way gating on individual detector positions and types is facilitated. An example is provided in the "SimHistorgrams.cc" file. The detector ID is accessed via "fSett->Clu2Det(hit->GetCluster())" and corresponds to the number given in the analysis settings, i.e.

```
Detector.2: 201  # 30 deg ring
```

maps cluster ID 201 in the simulation, the first Miniball detector, to detector ID 2. The crystal ID is simply given by "4*fSett->Clu2Det(hit->GetCluster())+hit->GetCrystal()". The histograms called "egamdc_summary" etc. are shown in Fig. 14.

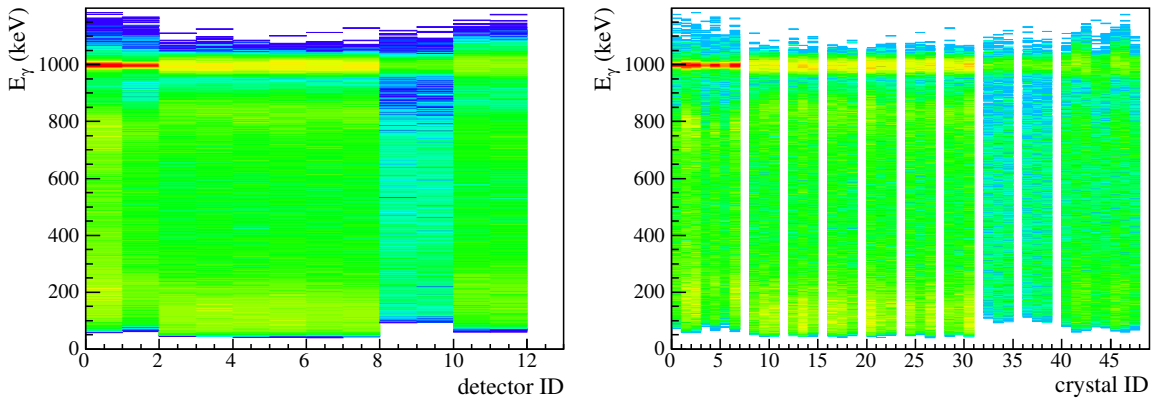

Figure 13:

The histogramming step is also where coincidence spectra and gated histograms are produced. Gating on transitions is straightforward by projecting from the two-dimensional spectra called "egamegamdc". The higher energy hit is sorted on the x-axis. An example is shown below. Because of the different detector types and resolutions, the spectra in the example "SimHistograms.cc" file also includes spectra for tracking and non-tracking detectors on one of the axes. An example would be "GRegamGRegamABdc" were two tracking detector hits with add-back are plotted. For events where one hit is detected in a tracking detector, and another in the segmented detectors, spectra such as "MBegamGRegamdc" are filled. Here the higher of the two hits was measured in the segmented detectors (Miniball and Clover), while a lower energy hit was detected in the tracking detectors.

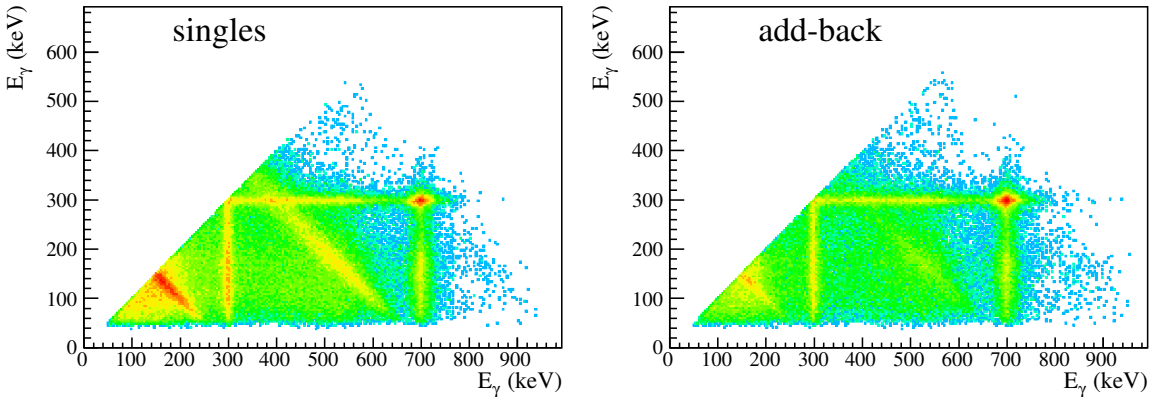

Figure 14:

4.4 Tracking

Tracking of γ-rays is in principle possible by setting

```
DoTracking    true
```

but it is recommended to use a separate program "Track" for this. This program can be provided upon request.

5 Simulations for proposals for the NP-PAC20

In this section the recommended default configuration for the simulations for proposals for the 20th NP-PAC meeting in December 2019 are described. You are requested to use this configuration for the estimation of count rates. Due to the complex mounting structure and the geometry of especially the tracking detectors, changes to the configuration might occur during the mounting at F8. These will have small effects of the efficiency, in-beam resolution, or lifetime sensitivity. The exact configuration will only be known after mounting, laser positioning, and commissioning of the setup.

If you require any modification to the setup, different detector positions (distances can be adjusted), additional particle tracking detectors before or after the target, contact me as soon as possible to investigate the possibility, most likely it is difficult.

For MINOS use the detector configuration "Geometry/proposalMINOS" (previously tight configuration). The target face is aligned with the TPC (modeled in the simulation by the beam pipe). By changing the thickness, 50, 100, and 150 mm are available, the target has to be shifted accordingly. The target shift is 25 cm - half the target length:

```
/Target/Thickness 150 mm
/Target/SetPosition_Z -17.5 cm

/Target/Thickness 100 mm
/Target/SetPosition_Z -20.0 cm

/Target/Thickness 50 mm
/Target/SetPosition_Z -22.5 cm
```

For solid targets the target position can be freely selected, for example for long lifetimes a position more upstream is beneficial. But you should consider that changing the target position also affects the resolution for the scattering angle (needed for Coulomb excitation and projection of momentum transfer) and the transmission of ZeroDegree.

For a solid target the beam-pipe requires a larger diameter of the 30° ring of Miniball detectors has to be increased. The configuration is given in "Geometry/proposaldefault". Shielding around the beam pipe should be included to reduce background.

6 Appendix

6.1 Using ROOT6

For the installation using ROOT6 some additional changes need to be made. Using the "ROOT6.Makefile" additional "Dictionary_rdict.pcm" files are created and placed in "$(LIB_DIR)".

The procedure of loading shared libraries for interactive analysis has changed. Instead of

```
gSystem->Load("libHRArray");
```

you now need to add the two line

```
#include "TVector3.h"
R__LOAD_LIBRARY(libHRArray.so)
```

at the very top of the "rootlogon.C" file, before the "rootlogon()" function. Additionally, root needs to know where the header files are. This is achieved by setting the "ROOT_INCLUDE_PATH", for example, in the ".bashrc" or ".profile" file:

```
export ROOT_INCLUDE_PATH="/home/wimmer/progs/HRArray/inc:$ROOT_INCLUDE_PATH"
```

6.2 Atomic background calculations

The program "abkg" [9], available with the DALI simulation package [7], requires the installation of the "cern-lib" librariers, and does not compile or run on many modern machines. The abkg code, which is based on [10] has been rewritten in python:
https://github.com/wimmer-k/Coulex
The code reproduces the abkg results, it is slower, but should run on any computer. It directly puts out a root file with the same type of histogram as the abkg paw output. Note that the cross section depends on the energy range which is defined in lines 68 of atomicBG.py and the following.

6.3 Further input options for the GEANT simulation

6.3.1 Detectors, Target, Materials

```
/Target/X_length x cm
/Target/Y_length y cm
```

Change the area of the target, default is square of 5×5 cm^2, for circular targets use

```
/Target/Radius r cm
```

to specify the radius.

```
/Target/SetPosition_X  x cm
/Target/SetPosition_Y  y cm
```

To set the target off center with respect to the beam axis.

```
/Target/ScaleDensity number
```

Scales the default target density (GEANT defined) by number. Can be used to adjust energy loss when comparing to experiment.

```
/Target/NStep number
```

Allows to set a G4UserLimits on the step size to Target_thickness/number. Default is 20.

6.3.2 Beam, Reaction, Gamma-decay

`/BeamIn/KE e GeV`

gives the beam kinetic energy instead of the one per nucleon.

```
/BeamIn/Focus/X
/BeamIn/Focus/Y
/BeamIn/Focus/Z
```

sets the beam focus away from $(0, 0, 0)$. The beam is tracked in the experiment as well as in the simulation, and the Doppler correction takes the event by event position into account, so this parameters are not really needed.

```
/BeamIn/Focus/maxAta mAta
/BeamIn/Focus/maxBta mBta
/BeamIn/Focus/Ata0 Ata
/BeamIn/Focus/Bta0 Bta
```

set the "Ata" and "Bta" angles and the divergence "mATa" and "mBta" on target for the incoming beam, can be recovered by the tracking.

`/BeamOut/TargetExcitation`

6.3.3 Output

`/Gretina/detector/PositionResolution r mm`

Set the position resolution to "r" mm. Note that this is done by smearing the interaction point position, and therefore can change the position over segment boundaries. This is not realistic.

References

[1] S. Agostinelli *et al.*, Nucl. Instrum. Methods Phys. Res. A **506**, 250 (2003).

[2] L. Riley, "UCGretina," (2015), `http://gswg.lbl.gov/simulation`.

[3] `http://gswg.lbl.gov/analysis/geb-headers`.

[4] `https://geant4.web.cern.ch/node/1604`.

[5] `http://www.nishina.riken.jp/collaboration/SUNFLOWER/devices/hrarray/installGuide.pdf`.

[6] L. Olivier *et al.*, Phys. Rev. Lett. **119**, 192501 (2017).

[7] `http://ribf.riken.jp/~pieter/shogun/`.

[8] `http://nucl.phys.s.u-tokyo.ac.jp/wimmer/software.php`.

[9] R. Holzman, "Simulation program abkg," GSI (1998).

[10] R. Anholt, C. Stoller, J. D. Molitoris, D. W. Spooner, E. Morenzoni, S. A. Andriamonje, W. E. Meyerhof, H. Bowman, J.-S. Xu, Z.-Z. Xu, J. O. Rasmussen, and D. H. H. Hoffmann, Phys. Rev. A **33**, 2270 (Apr 1986), `https://link.aps.org/doi/10.1103/PhysRevA.33.2270`.