Gamma-Ray Tracking School 2018: Simulation Hands-On Session

Introduction

Install UCGretina

1. You have been invited to join the bitbucket.org/lriley/ucgretina git repository. If you have not already created a bitbucket account and accepted the invitation, do so now.
2. Download and compile the code.

   ```
   $ git clone https://bitbucket.org/lriley/ucgretina
   ```

   or optionally configure ssh to work with bitbucket (instructions here: https://confluence.atlassian.com/bitbucket/ssh-keys-935365775.html) and use

   ```
   $ git clone git@bitbucket.org:lriley/ucgretina
   ```

   This will save you having to enter your username and password when you interact with the repository.
3. Check out the GRTSchool branch of the code. This is the geant4.10 branch with exercises for the school.

   ```
   $ git checkout GRTSchool
   ```

4. Set up your environment to work with the geant4.10 libraries.

   ```
   $ source <PATH to GEANT4>/bin/geant4.sh
   $ source $G4INSTALL/geant4make.sh
   ```

5. Compile the code.

   ```
   $ cd ucgretina
   $ make
   ```

Resources

- Installation instructions: https://bitbucket.org/lriley/ucgretina/wiki/Home
- Documentation included with the source: ucgretina/README.md
- Built-in help:

  ```
  $ UCGretina
  Idle> help
  ```

1 ASCII and Mode 2 Output

ASCII Output

ASCII output is activated by the macro file command

```
/Output/Filename <Filename>
```

For each event, the emission positions, emission directions, and source β values of the emitted γ rays are described as follows.

```
E <# of emitted gamma rays> <Full Energy> <Event #>
    <Energy> <X> <Y> <Z> <theta> <phi> <beta>
    <Energy> <X> <Y> <Z> <theta> <phi> <beta>
    ...
```

where `<Full Energy>` = 1 if a single γ ray is emitted and its full energy is deposited in a single crystal, and `<Full Energy>` = 1 if a single γ ray is emitted and only part of its energy is deposited in a single crystal. `<Full Energy>` = -1, otherwise.

When an event deposits energy in at least one active volume in the array, the simulated detector response is written in the following format.

```
S <ATA> <BTA> <DTA> <YTA> <Event #>
D <# of decomposed gamma events> <Event #>
C <Crystal ID> <# of interaction points>
    &lt;Segment ID&gt; <Energy> <X> <Y> <Z>
    &lt;Segment ID&gt; <Energy> <X> <Y> <Z>
C <Crystal ID> <# of interaction points>
    &lt;Segment ID&gt; <Energy> <X> <Y> <Z>
    &lt;Segment ID&gt; <Energy> <X> <Y> <Z>
...
```

where the `S` line describes simulated S800 measurements and is only present for in-beam simulations.

Mode 2 Format

In addition to the standard Mode 2 packets found in experimental data, information about emitted γ rays is written in type 11 packets.

```
#define GEB_TYPE_G4SIM 11

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

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;
```

`GRUTinizer` unpacks these into the `TGretSim` class.

In in-beam simulations, simulated S800 data are written by UCGretina to type 9 packets.

```
#define GEB_TYPE_S800PHYSDATA 9

typedef struct S800_physicsdata {
  int32_t type; /* defined abcd1234 for indicating this version */
  float crdc1_x; /* Crdc x/y positions in mm */
  float crdc1_y;
  float crdc2_x;
  float crdc2_y;
  float ic_sum; /* ion chamber energy loss */
  float tof_xfp; /* TOF scintillator after A1900 */
  float tof_obj; /* TOF scintillator in object box */
  float rf; /* Cyclotron RF for TOF */
  int32_t trigger; /* Trigger register bit pattern */
  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
  /* from here corrected values extracted from data above */
  /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
  float ic_de;
  /* TOF values with TOF correction applied (from afp/crdc x) */
  float tof_xfpe1;
  float tof_obje1;
  float tof_rfe1;
  /* Trajectory information at target position calculated from
     a map and afp/bfp/xfp/yfp. New map and you need to re-calc */
  float ata; /* dispersive angle */
  float bta; /* non-dispersive angle */
  float dta; /* dT/T T:kinetic energy */
  float yta; /* non-dispersive position */
} S800_PHYSICSDATA;
```

`GRUTinizer` unpacks these into the `TS800Sim` class. `UCGretina` only populates the ata, bta, dta, and yta values.

A Simple Source Simulation

To generate some output, we'll start with a simulation shooting 100 1 MeV γ rays into the array. The simulation is specified by the macro file `./exercises/simple/simple.mac`:

```
# Detector parameters =======================================================
/Gretina/detector/enableCapsules
/Gretina/detector/enableCryostats
/Gretina/Shell full
/Target/Construct
```

```
/Target/Sled
/BeamTube/Construct
/Gretina/update

# Source parameters ========================================================
/Experiment/RunSource
/Experiment/Source/Set simple
/Experiment/Source/setEnergy 1 MeV

# Output parameters ========================================================
/Mode2/crmatFile crmat.LINUX
/Mode2/GretinaCoords
/Output/Filename simple.out
/run/beamOn 100000
```

Exercises

1. Run the simulation

   ```
   $ UCGretina simple.mac
   ```

   and look at the ASCII output file

   ```
   $ less simple.out
   ```

   Make sure that the output makes sense to you. You should see a lot of events (at least 20) in which the array detected a γ ray, and of these, several should involve two crystals.

2. Generate and sort mode 2 output.

   (a) Replace the `/Output/Filename` command with

   ```
   /Mode2/Filename simple.dat
   ```

   and increase the number of events to 100000.

   (b) Run the simulation and use `GRUTinizer` to sort `simple.dat` into an energy spectrum and a 2D θ vs. ϕ spectrum.

   (c) Use `GRUTinizer` to add simulated energy resolution to your γ-ray energies. To use the `gRandom->Gaus(<mean>, <sigma>)` method, add

   ```
   #include <TRandom.h>
   ```

   to your `GRUTinizer` histogram library code.

2 Source Simulations

2.1 Calibration Sources, Take I: Emit γ-ray Singles with Known Intensities

The simplest and more computationally efficient approach to source simulations is to emit a single γ ray per event, drawing energies from a distribution of known intensities. Several calibration sources

(^{56}Co, ^{60}Co, ^{137}Cs, ^{226}Ra, ^{241}Am) are implemented in this way in `UCGretina`. Cascades of γ rays are not simulated and hence no γ-γ coincidences are produced.

The relevant lines from `./exercises/eu152/eu152_gammas.mac`:

```
# Source parameters ===================================================
/Target/sourceFrame eu152_Z2707
/Experiment/RunSource
/Experiment/Source/Set eu152
```

2.2 Stationary Sources, Take II: Use `G4Radioactive Decay` and `G4PhotonEvaporation`

The decay properties and level schemes of the nuclei in the ENSDF database are included in GEANT4 data files, and GEANT4 can simulate beta decay and photon evaporation (γ decay). In this approach, we simulate a stationary "beam" which decays at rest.

The relevant lines from `./exercises/eu152/eu152.mac`:

```
# Set source/decay parameters ==========================================
/Target/sourceFrame eu152_Z2707

/process/inactivate Reaction # (G4RadioactiveDecay handles the beta decay).
/BeamOut/Source # We're not really shooting a beam at a target

# Stationary 152Eu source
/BeamIn/Focus/Z 0. mm # (Misleading: emission point, not focus)
/BeamIn/A 152
/BeamIn/Z 63
/BeamIn/KEu 0. keV

# Load level schemes of the daughters (152Sm, 152Gd).
/BeamOut/DA 0
/BeamOut/DZ -1
/BeamOut/LevelDataFile z62.a152.lvldata
/BeamOut/DA 0
/BeamOut/DZ +1
/BeamOut/LevelDataFile z64.a152.lvldata
/BeamOut/Update
```

Section 5 gives an introduction to using level data files to specify level schemes.

Exercise

Add a $\gamma - \gamma$ matrix to your `GRUTinizer` histogram library and re-sort both ^{152}Eu simulations. Verify that the expected coincidence relationships are present in the output of eu152.mac. (The 779 - 344 keV cascade in ^{152}Gd should be particularly strongly populated, and the 121.7 keV transition in ^{152}Sm should "see" several strong feeders including 245, 964, 1112, and 1408 keV.)

Figure 1: Mounting shell with hole numbers labeled. (Slot number = hole number - 1.)

3 Detector and Array Geometry

Five text files present in the working directory in which `UCGretina` is run determine the geometry of the crystals, including coaxial and back dead layers (`asolid`), the segmentation of crystals for readout (`aslice`), the assembly of crystals into quads (`acluster`), the aluminum vacuum jacket surrounding the crystals in each quad (`awalls`), and the placement of modules into the array (`aeuler`). The `Gretina_Array` class reads these files and assembles the array. The detector construction code is a modified version of the `AgataDetectorArray` class in the AGATA simulation code. The file `./GretinaGeometry/geometry_description_agata` supplied with `UCGretina` describes the geometry file formats. *Note: the euler angles in `UCGretina` geometry files are expressed relative to the beam axis. This is a departure from the convention used by the AGATA code which expresses euler angles relative to slot 0.*

The `aslice` and `awalls` geometry files are optional. Detector segmentation will not be simulated if the `aslice` file is not present (segment ID = -1 in the output), and the aluminum vacuum jacket surrounding the crystals in each quad will not be constructed if the `awalls` file is not present.

The holes in the GRETINA mounting shell are numbered according to Figure 1. The placement of quads into holes is specified in the euler-angle geometry file. Each line places and orients a quad in a particular hole The first number is a slot number, which is the hole number + 1. (Slots are numbered 0-29, and holes 1-30.) All of the possible quad positions in the full GRETA are listed in `/GretinaGeometry/Greta/G120C4euler.list`. Of these, only 21 are available in the GRETINA mounting shell, due to the S800 quadrupole (holes 1-5 and 10) and the support axles (holes 13 and 18).

Unfortunately, the GRETINA coordinate system shown in Figure 1 is not the standard GEANT4 coordinate system, in which $\hat{z}$ points along the beam axis, $\hat{x}$ points to the left facing downstream, $\hat{y}$ points up. The macro command

```
/Mode2/GretinaCoords
```

produces Mode 2 output in the GRETINA coordinate system.

The macro file command

```
/Mode2/crmatFile crmat.LINUX
```

loads I.Y. Lee's transformations from world to crystal coordinates for mode 2 output.

The `change_geometry.sh` bash script creates soft links to the geometry files with the path and base name supplied as a command line argument. For example

```
$ ./change_geometry.sh <PATH TO GretinaGeometry>/GretinaNSCL/G120C4
```

creates links to the 7-quad configuration used in the GRETINA commissioning at the NSCL.

Exercise

Create a new subdirectory in `ucgretina/GretinaGeometry` called `FMA2018` that describes the FMA configuration of the array with 11 quads placed in holes 14-17, 21, 22, 24, 26-29. Simply copy the files from `ucgretina/GretinaGeometry/GretinaNSCL` and modify the euler-angle file. *Remember that hole number = slot number +1.* Then, move on to Section 4 to visualize your new array configuration.

4 Visualization

The `exercises/vis` directory contains a macro file `vis.mac` illustrating visualization. The visualization manager is only initiated in an interactive session, so we'll need to run `UCGretina` interactively.

```
$ UCGretina

...

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

The product is a VRML2 file `g4_00.wrl` which can be opened in a VRML viewer like `FreeWRL`, `view3dscene`, or `mayavi2`. Subsequent visualization output will be written to `g4_01.wrl`, `g4_02.wrl`, etc.

The macro file `trajectories.mac` emits 100 1 MeV γ rays from the center of the target and draws the trajectories. The color coding is by charge (γ rays in green, scattered electrons in red).

Exercise

1. Run both of the visualization scripts and view the output. You should see a single quad in hole 15 (slot 14) centered at $(\theta, \phi) = (90°, 162°)$. Verify that you can add/remove the mounting shell, beam pipe, etc. by modifying the `vis.mac` file.

2. Use the `./change_geometry.sh` script to change the geometry files to your new ANL2018 configuration, and visualize it.

5 Level Schemes

The level data file follows the format described in the `README` file included with the photon evaporation data files in `GEANT4` versions > 4.10.3. Take a look at it:

```
$ less $G4LEVELGAMMADATA/README-LevelGammaData
```

Exercise

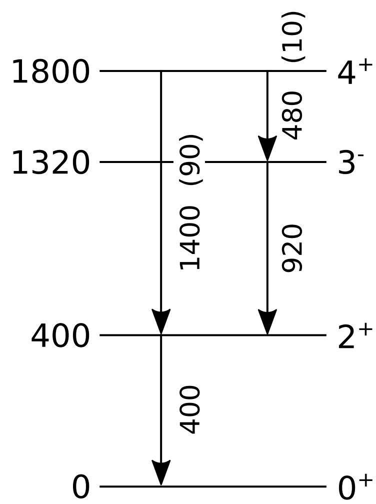

Figure 2: An entirely fabricated level scheme to implement and simulate. Branching ratios are in parentheses.

Create a level scheme file named `myscheme.lvldata` in the `./exercises/inbeam/` directory, corresponding to the level scheme shown in Figure 2. Include spins and parities, and assume transitions have pure multipolarity. Set the total internal conversion coefficient for each transition to 0 (and omit entries 7-16).

6 In-Beam Simulation

`UCGretina` can simulate beam particles undergoing reactions as they pass through a target and emit γ rays in flight. The macro file `./exercises/inbeam/s44.mac` describes inelastic scattering of a 100 MeV/u ^{44}S beam from a thick ^{9}Be target. Your fabricated level data file will be used to generate γ rays. [1]

```
# Detector parameters ========================================================
/Gretina/detector/enableCapsules
/Gretina/detector/enableCryostats
```

[1] Any resemblance to actual spectroscopy of ^{44}S is purely accidental.

```
/Gretina/Shell full
/Target/Construct
/Target/Sled
/BeamTube/Construct
/Gretina/update

# Set reaction parameters ==================================================
/Target/Material Be
/Target/Thickness 2034.6 um # 376 mg/cm^2
/BeamIn/A 44
/BeamIn/Z 16
/BeamIn/KEu 100.0 MeV
/BeamIn/Dpp 0.02 # Momentum acceptance
/BeamIn/Focus/DX 4. mm # Beam spot sigma (non-dispersive)
/BeamIn/Focus/DY 8. mm # Beam spot sigma (dispersive)

/BeamOut/TargetA 9
/BeamOut/TargetZ 4
/BeamOut/DA 0
/BeamOut/DZ 0
/BeamOut/LevelDataFile myscheme.lvldata
/BeamOut/ProjectileExcitation 2280 keV
/BeamOut/AngDistSigmaA 0.006 rad # Angular spread (dispersive)
/BeamOut/AngDistSigmaB 0.006 rad # Angular spread (non-dispersive)
/BeamOut/Update

# Output parameters ========================================================
/Mode2/crmatFile crmat.LINUX
/Mode2/GretinaCoords
/Mode2/S800KE 3.798 GeV
/Mode2/Filename s44_1329.dat
/run/beamOn 10000
```

The `/Mode2/S800KE` command sets the kinetic energy corresponding to the magnetic rigidity of the S800, which establishes the center of the relative kinetic energy (dta) spectrum.

Exercises

1. Run the simulation. Add the β spectrum of the scattered beam to your histogram library, sort the output, and determine the average β for Doppler reconstruction.

2. Add a Doppler-corrected γ-ray energy spectrum and a 2D Doppler-corrected γ-ray energy vs. θ histogram to your histogram library, and re-sort the simulated data assuming the average β you determined. If the Doppler reconstruction goes well, photopeaks counts should lie along horizontal bands in the γ-ray energy vs. θ histogram.

7 Fitting Simulations to Measured Spectra

7.1 In-Beam Case

The spectrum `s44spectrum` in the root file `./exercises/fit/fakedata/fakedata.root` is a simulated spectrum created by combining simulations populating the levels in the level scheme shown in Figure 2 with various cross sections, combined with a double exponential background. We will treat this as a "measured" spectrum.

Instead of assuming a level scheme *a priori*, we will simulate the response of the array to the individual γ rays observed. The root script `s44Fit.C` reads the "measured" spectrum and simulated Doppler-corrected γ-ray spectra. It then fits a linear combination of the simulated spectra and two exponential functions to the fake "measured" spectrum.

Exercise

1. Create level data files that will produce the individual γ rays in the "measured" spectrum. Name them `s44_400.lvldata`, `s44_480.lvldata`, `s44_920.lvldata`, and `s44_1400.lvldata`.

2. Create macro files `s44_400.mac`, `s44_480.mac`, `s44_920.mac`, and `s44_1400.mac` that will simulate the response of the array to 100000 of each γ ray. Set the output files names to `s44_400.dat`, `s44_480.dat`, etc.

3. To run the simulations and sort the output, use `make`:

   ```
   $ make
   ```

   The resulting root files are named `s44_400_histos.root`, `s44_480_histos.root`, etc.

4. Open the root script `s44Fit.C` in your favorite text editor, and change the spectrum name on line 47 from `"energy/dop_4169_gaus"` to the name you gave the Doppler corrected γ-ray energy spectrum in your `GRUTinizer` histogram library.

5. Run the fit script.

   ```
   grutinizer
   GRizer [0] .x s44Fit.C
   ```

6. Parameters p4-p7 in the fit results are the scaling factors for the four 100000-event simulations giving the best fit to the "measured" spectrum. Use these to determine the number of times each of the three excited states was populated, corrected for feeding via γ decay of higher-lying levels, to produce the "measured" spectrum.

7.2 Source Case

A ^{152}Eu source measurement (`run020_cal_histos.root`) and a room background measurement (`run007_cal_histos.root`) made with the standard 7-quad configuration of the array in the first GRETINA campaign at the NSCL are included in the `./exercises/eu152/data` directory.

The root script `eu152Fit.C` reads the measured spectra and a simulated ^{152}Eu spectrum and fits a linear combination of the simulated spectrum and the measured room background spectrum to the measured ^{152}Eu spectrum.

The source activity was 313100(4400) Bq on 5/1/1978, and the β-decay half life of ^{152}Eu is 13.537(9) years. The measurement was made on 9/17/2012, when the activity was

$$313100(4400)\ \text{Bq} \times \left(\frac{1}{2}\right)^{\frac{34.3819\ \text{years}}{13.537(9)\ \text{years}}} = 53841(760)\ \text{Bq}$$

The duration of the measurement was 595.6 live seconds, yielding $3.21(5) \times 10^7$ decays. We simulated 10^5 events, so we should expect the fit should scale the simulated spectrum by a factor 321(5) to fit the measured spectrum. However, in the case of the γ-ray singles simulation, a correction factor of 1.537 is needed for the average γ-ray multiplicity per decay for the set of γ rays included in the code, giving an expected scaling factor of 493(7).

Exercise

(This is more of a demonstration than an exercise.)

1. Run the simulations and sort γ-ray energy spectra. (You have probably already completed this step.)

2. Open the root script `eu152Fit.C` in your favorite text editor, and modify the spectrum name on line 56 to match that of the γ-ray energy spectrum in your `GRUTinizer` histogram library.

3. Run the fit script with the singles simulation

```
$ grutinizer
$ .x eu152Fit.C ("eu152_gammas_histos.root")
```

   and compare fit parameter p1 with our expectation based on the source activity.

4. Run the fit script with the simulation using `G4RadioactiveDecay`.

```
$ grutinizer
$ .x eu152Fit.C ("eu152_histos.root")
```

   and compare fit parameter p1 with our expectation based on the source activity.