MaGe Evaluation Report

Contact: Xiang Liu, Luciano Pandola

Version 0.5

December 16, 2004

Abstract

In this note we report about our evaluation of the Majorana Monte Carlo package called MJ. In particular, we focused on its suitability for the future joint use and development by the Majorana and Gerda MC groups. We received a released version of the MJ package by the courtesy of the Majorana MC group. We then made some modifications and implemented a simplified Gerda geometry. The updated simulation package for both Collaborations is named "MaGe". During the work we find that the existing package is well-designed, easy to use and flexible for modifications. Therefore, we propose that the Gerda Task Group T10 (*Simulation and background studies*) should collaborate with the Majorana group for the development of a common Geant4-based MC framework.

Contents

1 Introduction 3

2 Gerda MC ongoing activity 3

3 Gerda MC needs a common framework 4

3.1 A common framework 4
3.2 Gerda subgroups in common framework 4
3.3 Gerda MC future needs 5

4 Why Majorana MC package 7

5 What is in Majorana MC package? 7

5.1 Framework provided 7
5.2 Geometry (`geometry`) 8
5.3 Material (`material`) 8
5.4 Event generator (`generators`) 8
5.5 Physics process (`processes`) 9
5.6 Output (`io`) 9
5.7 Macro control (`management`) 9
5.8 Preliminary waveform simulation (`waveform`) 9
5.9 How to implement a new geometry 10

6 First version of MaGe 11

6.1 List of work already done in MaGe 11
6.2 Gerda geometry in MaGe 12
6.3 One steering file 12

7 CVS and database servers 13

8 Conclusion 14

A Implementing Gerda geometry 15

1 Introduction

After the Majorana-Gerda MC workshop at Gran Sasso, we were assigned the task to evaluate the Majorana package. With the help of the Majorana group, we received the source code of the existing MJ package and developed the first version of MaGe, a common MC package for both Collaborations. We are convinced that MaGe is the most viable choice for the Gerda MC group.

The note is arranged in the following parts. The recent Gerda MC activities are summarized in Chapter 2. Chapter 3 then describes why a common and well-designed framework is needed by the Gerda MC developers for a more efficient work and collaboration. Chapter 4 summarizes the main reasons of our proposal to adopt, extend and develop the existing Majorana framework. Details about the framework are given in Chapter 5. Chapter 6 summarizes the work and the contributions made so far in the MaGe package. Chapter 7 lists our suggestions concerning CVS repository and database issues. Detailed instructions of how to implement a new geometry can be found in the appendix.

2 Gerda MC ongoing activity

A short summary of the ongoing MC activities in the different Institutions of the Gerda Collaboration is presented below. They are extracted so far from the presentations given at the LNGS MC workshop, so are by no means complete.

- Luciano Pandola (LNGS) General framework designs.
- Xiang Liu (MPI-Munich) Radiopurity requirements for the Gerda support structures and front end electronics. Suppression of the internal and surface background of the Ge detectors. Future simulation of the test facilities that will be built in Munich.
- Michael Bauer (Tübingen) Cosmic-ray-induced neutron and isotope background from the surrounding environment. A comparison between Geant4 and FLUKA was already accomplished.
- Stephan Scholl (Tübingen) Low-energy neutron recoil with Geant4; feasibility of dark matter search with Gerda.
- Hardy Simgen (Heidelberg) ^{39}Ar background in liquid argon; feasibility study for dark matter search.

- George Rugel (Heidelberg) Simulation of the detector GeMPI-II; the software is presently independent on Geant4. GeMPI-II will be used for the Gerda material screening. If a new Geant4-base simulation is found to be necessary, MaGe can accommodate it.

3 Gerda MC needs a common framework

3.1 A common framework

Geant4[1] is a toolkit for the Monte Carlo simulation of the interaction of particles with matter, based on the Object-Oriented technology. It allows to divide the simulation chain into de-coupled classes: material, geometry, event generator, physics processes, output and visualization.
Users are required to write their own geometry input, select the physics processes to be activated and define their own output format. Once these are done, the Geant4 interfaces can manage the simulation process and the communications among the different components.

Gerda MC Task Group needs a common framework able to

1. provide the complete simulation chain;
2. facilitate the cooperation among the TG members and allow the distributed development of the software.
3. make comparisons between different studies possible, as the same basic tools are used.

A flexible framework providing a suitable set of interfaces and base classes is hence required. In this case the development can proceed in a modular way: each MC developer implements only the specific and concrete classes required for his/her study. The framework ensures the correct "communication" among several de-coupled modules of the software, so that the interfaces do no need to be re-written from scratch. In this way the developers can effectively avoid the duplication of work and cooperate efficiently.

3.2 Gerda subgroups in common framework

When the common framework for Gerda is ready and the default base classes are defined, the Institutions can develop different specific modules in parallel and contribute to the complete simulation chain without overlaps. This is summarized in Table 1.

It will be shown later that the MaGe represents a suitable framework meeting all the requirements of flexibility and modularization described above.

3.3 Gerda MC future needs

The complete simulation of pulse shapes will also be needed in the future. The requirement can be met in two ways:

- with a native (or re-engineered) software integrated in the MaGe framework and directly interfaced with the MC tracking;
- with a stand-alone software (like the FORTRAN-based one developed by the Padua group) which takes as input the output file (energy deposit, etc.) of the MC stage.

For the long term, once Gerda starts operating, many more things are needed.

- Online trigger and DAQ simulation.
- Database for calibrations of individual Ge crystals, including energy scale, geometry, HV, alignment, properties for mirror charge and pulse shape.
- The format of Output data of each triggered event to be saved.
- Off-line analysis tools and access to database.

All these items should be included in the Gerda software package so that online and off-line analysis are consistent, and data and MC events can also be compared consistently. The issue has to be taken jointly by the T9 (*DAQ electronics and software*) and T10 Gerda working groups.

	Material	Geometry	Event generator	Physics process	Output
Default	definition of normal materials and their components	whole Gerda setup, including crystal , cryogenics, supporting and shielding.	Geant4 particle generations with most radioactive isotopes and their decay chain	Geant4 simulation of particles interacting in detector and shielding materials	Root ntuples with energy deposit information
LNGS	default	default	interface for 0ν- and 2ν- 2β decays from Decay0 package $\star$	default	generalize to other analysis tools than ROOT $\star$
Background Munich	materials for supporting structure	provide default Gerda setup $\star$	default	default	provide trajectories and points $\star$
Test-facility Munich	default	own geometry	default	default	own output
Neutron Tübingen	default	default	provide neutron flux	study neutron interaction	default
^{39}Ar Heidelberg	special liquid argon if necessary	liquid Ar cryogenic structure	default	optical photon tracking	default
GeMPI-II Heidelberg	new material definition if necessary	own geometry	default	default	own output
To be defined	default	default	default	mirror charge and pulse shape simulation	energy deposit as function of time

Table 1: Foreseen contributions from the different Institutions to the common framework. $\star$ for already accomplished.

4 Why Majorana MC package

We suggest to collaborate with Majorana MC group and share one single MC package, MaGe. Our suggestion is based on the following considerations.

- Gerda needs a common framework and Majorana provides one with good organization and enough flexibility. The collaboration is of reciprocal advantage, because Gerda and Majorana are studying the same physics process. The development of common tools (e.g. generators, pulse shape simulation) is shared and not-duplicated, while each side takes care independently of the specific concrete parts (e.g. geometry).
- The present MJ interfaces are flexible enough that they does not put *any* constraint to the Gerda side concerning geometry, physics, i/o and generators. All the detector-specific or not-suitable existing components can be extended or replaced with others written from scratch. Gerda is hence not forced to use any of the existing components: each of them can be independently re-written, sharing only the generic interfaces.
- More data from real experiments will be available for the MC validation. Since both Gerda and Majorana have their own R&D research and their own measurements from different test facilities, simulations can be compared and studied more extensively.
- Geant4 is flexible enough for possible framework modifications and is suitable for distributed development.

5 What is in Majorana MC package?

5.1 Framework provided

The most important thing about the Majorana MJ package is that the framework is already designed, implemented and working. After getting the source code, we could define our own geometry and output format and start the events simulation. A key point is that only one executable file is created and the user can choose at run-time (via macros) the specific implementation he/she wants for geometry, physics, generators and i/o. As an example, the procedure for the definition of the default Gerda geometry and a steering file for loading this geometry are described in the next chapter.

The following directories are present in the MJ package:

- `geometry`
- `materials`
- `database`
- `generators`
- `processes`
- `io`
- `management`
- `waveform`

They are described in the following sections. In each section, possible future developments are also briefly discussed.

5.2 Geometry (`geometry`)

It contains a virtual base geometry class for the interface with the rest of the framework. Concrete classes for new geometries just need to inherit the base class and define the specific volumes.

Majorana implemented two experimental setups, describing a clover detector and a clover detector inside a NaI barrel calorimeter.

5.3 Material (`material`)

Majorana provides a small material database of 22 items, based on postgreSQL. All materials defined in the database are automatically constructed in the simulation; no additional code is required to add new materials. It is hence easy to define and use materia;s.

5.4 Event generator (`generators`)

It contains a virtual base generator class that manages the general interface with the framework. New customized generators inherit this base class. Presently 5 alternative generators are available. Our first concrete contribution to the common package was the implementation of the interface for the DECAY0 generator, which is working and available for Majorana collaborators. Majorana uses the PNNL database for radioactive isotopes and decays. Have to be checked whether it is possible to it. We foresee to implement a new generator for the simulation of cosmic-ray muons inside the Gran Sasso Lab and a new algorithm to randomly sample a point within a volume (a preliminary version is already available).

5.5 Physics process (`processes`)

Majorana defines a unique physics list with two different sets of production cuts: one for dark matter search (low energy threshold), the other for standard double-beta decay search (higher energy threshold and faster speed). Initial studies are done showing that higher energy threshold does not affect physics results for double-beta decay. We would like to have the possibility to switch among alternative physics lists by macro commands, in order to optimize the hadronic processes. This option is still not available but could be implemented in a straightforward way. A further possible development line is the definition of different cuts for the more-interesting (e.g. germanium detectors) and the less-interesting (e.g. air outside the shielding) regions of the set-up, in order to optimize the CPU-time.

5.6 Output (`io`)

The event informations need to be saved on ntuples, or ASCII files or whatever else. The base class for the actions at run level, event level and step level is written. User just needs to define the set of variables and the ways to retrieve and calculate them at each level. The module was extended in order to accomodate other analysis tools alternative to ROOT: the top-level base class for the i/o management is no longer ROOT-specific (as it was in the original MJ package) but it is now completely generic.

5.7 Macro control (`management`)

The MJ package generates only one executable; macro commands can be used to set geometry, tune geometry parameters, select physics process, choose event generator and output scheme. It is fairly easy to add more macro commands if needed. Furthermore, the root of the package class hierarchy (`MJManager`) is a singleton: any class can be accessed via the manager, avoiding the need for global variables.

5.8 Preliminary waveform simulation (`waveform`)

This module (not ready yet) is called pulse shape simulation within Gerda collaboration and is of general interest and the development could then be shared between Majorana and Gerda, with mutual benefit. Standalone waveform simulation codes are available at the semiconductor Lab in Munich and at the INFN-Padua. They will be tested in the future and possible interfaced with the MC side. The Majorana group is independently working on pulse shape simulation as well.

5.9 How to implement a new geometry

The following is a quick guide for the implementaytion of new geometries into the existing Majorana package; it shows that the procedure is easy to understand and to work with. More details about the Gerda geometry already implemented in MaGe can be found in the appendix.

Two steps are required.

First: the new geometry needs to be added.

- add the definitions for the needed elements and materials in the material table, i.e. in the database or in the local file `MaGeGerdaLocalMaterialTable` under `materials`
- Write your own geometry class, which returns the pointer of your main (i.e. outermost) logical volume. The world volume is a $10\times10\times10$ m cube filled of air and it is created by default. The logical volume provided by the user has to contain the whole set-up is placed at the center of the world box.
- Write your own sensitive detector classes and associate them to your geometry.
- Register the geometry in the main framework (by modifying one line and adding two new lines of code) so that it can be loaded at run-time via macro command.

Second: save the event information to ntuples after new geometry is added.

- Write your i/o class which is inherited from class `MJOutputROOT` (if ROOT-based output is wanted, otherwise from `MJVOutputManager`). Within this class override the following member functions:
  - `BeginOfRunAction()` (opens file and defines variables to be saved to ROOT ntuples.)
  - `BeginOfEventAction()` (retrieves true MC informations for each event.)

  - `EndOfEventAction()` (calculates variables after the event is finished.)
  - `EndOfRunAction()` (closes file.)
  - `RootSteppingAction()` (retrieves and saves information after each Geant4 stepping; it can be left empty.)

- Register your i/o class in the main framework (by modifying one line and adding 4 more lines of code) so that the output scheme can be selected by macro commands.

After these, the new geometry can be loaded at run time and results will be saved to ntuples.

6 First version of MaGe

A test version of MaGe with the Gerda geometry is available.

6.1 List of work already done in MaGe

The preliminary MaGe package is based on the original MJ simulation package with the following modifications.

- Implementation of Gerda-specific geometry and output inside two new directories `gerdageometry` and `gerdaio` (see next sections for more details).
- Accessing trajectories of all particles.
- Interface to the 2ν-2β and 0ν-2β event generator DECAY0.
- Generation of events with random distribution inside some volumes. (preliminary)
- Construction of a local material database and registration in the main frame.
- Generalization of the output interface to beyond ROOT.

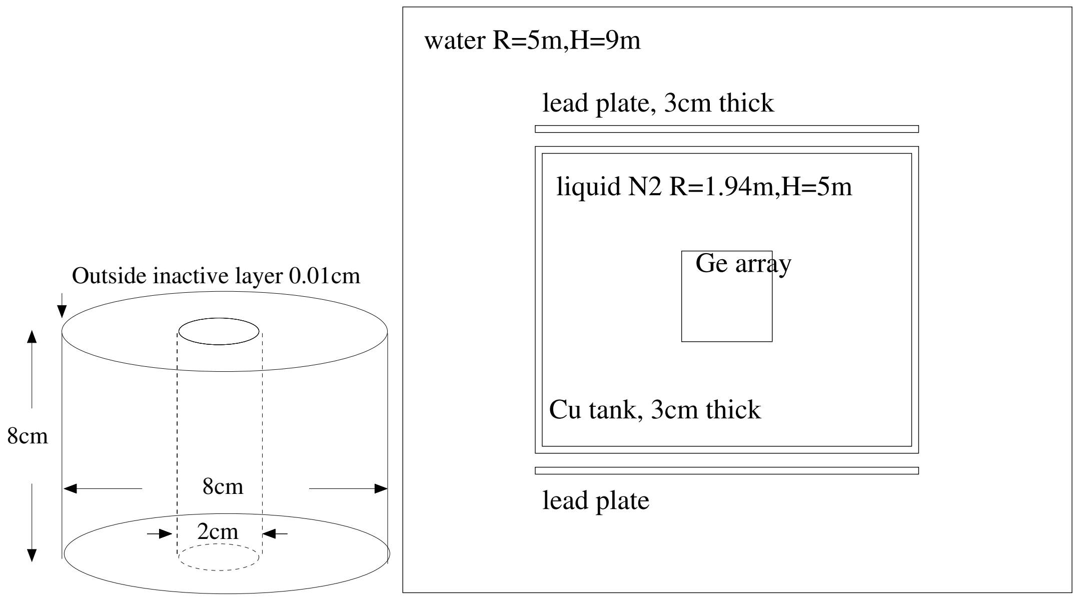

Figure 1: Left plot shows the size of one true coaxial Ge crystal, right plot shows the Gerda setup with liquid N_2 shielding inside and water shielding outside. (not to scale)

6.2 Gerda geometry in MaGe

The implemented Gerda geometry contains 21 Ge crystals surrounded by different shielding materials, as shown in Fig. 1. The single Ge crystal is true coaxial with the size shown in Fig. 1. The Ge array has 3 layers with 7 Ge crystals in each layer. The vertical gap between two layers is 5 cm, and the horizontal gap between two neighboring crystals within the same layer is 1 cm.

6.3 One steering file

The following macro steering file is an example of generating 0ν-2β events with random distributions inside one Ge crystal.

```
/MJ/manager/mjlog trace
/MJ/geometry/detector GerdaArray
/MJ/geometry/database false
/MJ/processes/realm BBdecay
/MJ/eventaction/rootschema GerdaArrayWithTrajectory
/MJ/eventaction/rootfilename testGerda.root
/MJ/eventaction/reportingfrequency 1000
/run/initialize
```

```
9 /MJ/generator/confine volume
10 /MJ/generator/volume Ge_det_0
11 /MJ/generator/select decay0
12 /MJ/generator/decay0/filename ge76.0nubb.1
13 /tracking/verbose 0
14 /run/beamOn 10000
```

Line 1 defines the log-book mode. Line 2 selects the Gerda geometry. Line 3 disables the connection to the LBL database (the local database will be used). Line 4 selects the physics processes for 2β decay study. Line 5 selects the set of variables to be saved to ntuples with the file name specified in line 6. Line 7 asks for reporting the event information for every 1000 events. Line 8 triggers the run initialization (material table and geometry are actually loaded). Line 9 asks the events to be generated randomly inside some volume with the name specified in line 10. Line 11 selects decay0 as event generator with the interface filename specified in Line 12. Line 13 asks for no debugging information during the tracking. Line 14 then starts running for 10,000 events. The simulation reproduced the results from an independent previous simulation.

7 CVS and database servers

A common MaGe CVS repository is needed for Majorana and Gerda people to cooperate. Originally, we suggested two synchronized and independent CVS repositories, one at LBL, one at Munich. However, this proposal turned out to be technically unfeasible. Therefore, we propose to create a single main repository (e.g. at Munich) which is regularly backupped by the other side. Presently a preliminary CVS repository for Gerda collaborators is available at Munich and is being tested extensively by both Gerda and Majorana people. We also suggest to have two separate databases, one at LBL, one at Munich. Users will be able to select which database to connect to through macro commands. The two database could be synchronized weekly.

Right now the MaGe code can be checked out by the command

```
setenv CVSROOT :pserver:gerdaguest@pclg-02.mppmu.mpg.de:/home/pclg-02/cvsroot
cvs login  (enter password gerda)
cvs checkout MaGe
cvs logout
```

Then follow the instructions in
`MaGe/doc/MaGeSetup/setup.ps`
to compile the code.

8 Conclusion

Having investigated in depth the Majorana MC package and successfully instantiated a preliminary Gerda geometry with the corresponding output, we conclude that the Majorana package provides a simulation framework which is well-designed, easy to use, flexible and suitable for future collaborations. Therefore, we suggest to collaborate with the Majorana MC developers and to construct a common software package named MaGe. Each side will have its own geometry setup, output format and specific modules, sharing the development and the know-how of general interfaces and common tools.

A Implementing Gerda geometry

This section describes in detail how the Gerda geometry, including crystals and shielding materials, is implemented in the first version of MaGe. It serves both as an example and as a manual to create new geometries in MaGe.

Step 1. Create Gerda geometry
One new directory `gerdageometry` is created with the following new files:

- `MaGeGeometryGerda` calls the following classes and contains the whole Gerda setup. It inherites the base detector class `MJGeometryDetector` and overrides the member function `ConstructDetector()`. It sets the logical volume pointer of the whole detector to the variable `theDetectorLogical` defined in `MJGeometryDetector`.

- `MaGeGeometryShielding` describes the shielding structure.

- `MaGeGeometryGermaniumArray` contains the array of crystals.

- `MaGeGeometryGermaniumCrystal` defines one single crystal.

The size of the detectors can be tuned as shown in the corresponding `Messenger` classes. New elements and materials can be added in class `MJGerdaLocalMaterialTable` under directory `materials`.

Step 2. Define sensitive volume The class `MaGeGeometrySD` is a general class for sensitive volumes and is used in class `MaGeGeometryGermaniumArray` to define the Ge crystals sensitive.

Step 3. Register the Gerda geometry In class `MJGeometryDetectorMessenger`, include name `GerdaArray` for the new geometry in the line:
`MJGeometryDetectorChoiceCommand->SetCandidates`
so that it can be loaded by macro.

Then in class `MJGeometryDetectorConstruction`, add the following lines in the member function `SetDetector()`

```
else if (detectorType == "GerdaArray")
theDetector = new MaGeGeometryGermaniumArray();
```

This defines the actions for the macro command, which is to instantiate the new geometry.

Step 4. Create Gerda output One new directory `gerdaio` is created with the following file
`MaGeOutputGermaniumArray` with the following member functions

- `BeginOfRunAction()`

- `DefineSchema()` defines the set of variables to be saved.

- `BeginOfEventAction()`
- `EndOfEventAction()`
- `EndOfRunAction()`
- `RootSteppingAction()`

which are already explained in Chapter 5.

Step 5. Register Gerda output In class `MJManagementEventActionMessenger` under directory `management`, include the new output schema `GerdaArray` in the line

```
schemacandidates = ...
```

Then in member function `SetNewValue()`, add the following lines

```
else if (newValues == "GerdaArray") {
fEventAction->SetOutputManager(new MaGeOutputGermaniumArray());
fEventAction->SetOutputName("GerdaArray"); }
```

Notice that `GerdaArrayWithTrajectory` is a different schema requiring the trajectory information to be saved to ntuple as well.

Step 6. Add new commands in macro

Add the following new commands in the macro

```
/MJ/geometry/detector GerdaArray
/MJ/eventaction/rootschema GerdaArray
/MJ/eventaction/rootfilename testGerda.root
```

and start running.

That is it!

References

[1] S. Agostinelli *et al.*, GEANT4 – a simulation toolkit, Nucl. Instr. Meth. A, 506 (2003) 250

[2] Majorana Monte Carlo User's and Developer's Guide, http://neutrino.lbl.gov/majorana/udg/book1.html (internal)