Meiga, a Dedicated Framework Used for Muography Applications

A. Taboada,[1] C. Sarmiento-Cano,[1] A. Sedoski,[1] and H. Asorey[1,2] for the MuAr project

[1]*Instituto de Tecnologías en Detección y Astropartículas, (CNEA-CONICET-UNSAM), Buenos Aires, Argentina*

[2]*Instituto Balseiro (CAB-CNEA) and Universidad Nacional de Cuyo, San Carlos de Bariloche, Argentina*

Corresponding author: A. Taboada
Email: alvaro.taboada@iteda.cnea.gov.ar

Abstract

The design and development of detectors for muography are in constant demand of the usage of semiempirical models and simulations. In this contribution, we present Meiga, a framework conceived for simulation and reconstruction of muography applications. This framework takes a simulated muon flux at ground level and propagates it through a given material where the detectors are located. It uses Geant4 as a toolkit for the simulation of traversing particles through the material and computes the signal produced when muons pass through any type of detector at the desired location. The framework provides interfaces to the detector models and a hierarchical structure for data accessing, encompassing the need of simulating different scenarios to optimize detector design in a versatile and easy-to-use framework.

Keywords: muography, detector modelling and simulation, Geant4
DOI: 10.31526/JAIS.2022.266

1. INTRODUCTION

In any scientific project, simulations play a key role in performing feasibility studies of the experiment, obtaining predictions or models, and understanding the results. Therefore, having a simulation tool for a particular project becomes necessary, and muography is not an exception. There are three steps that involve simulations or semiempirical calculations when developing an application for muography: muon flux calculation, muon transport, and detector response. The muon flux can be estimated using methods based on parametrizations, such as CRY [1] and EcoMug [2], or using dedicated air shower simulation software like CORSIKA [3]. Different approaches are taken for the muon transport, such as backward Monte-Carlo techniques implemented in PUMAS [4], and the case of Geant4 [5] which is extensively used for simulating the detector response.

In this proceeding, we present *Meiga*, a simulation framework developed within the MuAr project [6] that integrates the muon flux simulation with the propagation and the detector response in a dedicated and easily adaptable framework. In Section 2, a quick overview of the muon flux simulation used by Meiga is presented. The framework structure and an example application are presented in Section 3. Conclusions are given in Section 4.

2. SIMULATION OF COSMIC RAY MUON FLUX

Obtaining a more realistic muon flux becomes relevant in order to perform feasibility studies of a particular muography detector. For this purpose, we have used the ARTI framework [7] which uses CORSIKA for calculating the muon flux produced in extensive air showers. The flux of galactic cosmic rays is given by

$$\Phi\left(E_{\mathrm{p}}, Z, A, \Omega\right) \simeq j_0(Z, A)\left(\frac{E_{\mathrm{p}}}{E_0}\right)^{\alpha(E_{\mathrm{p}}, Z, A)}, \tag{1}$$

where E_{p} is the energy of the primary cosmic ray of atomic number Z and mass number A, Ω is the solid angle, j_0 is a normalization parameter, and $E_0 = 10^3$ GeV. The spectral index $\alpha(E_{\mathrm{p}}, Z, A)$ is considered independent of the primary energy, i.e., $\alpha \equiv \alpha(Z, A)$, for energies between 10^{11} eV and 10^{15} eV. An example of the cosmic ray flux for different primaries can be seen in Figure 1(left).

Each of these primaries produces a shower of secondary particles in the atmosphere that propagates to the ground. The flux of secondary particles is affected by the geomagnetic field, and therefore, a correction must be applied. ARTI uses the coordinates of the local geomagnetic field, B_x and B_z, as well as the altitude of the observation level to obtain an accurate estimation of the muon flux at the ground.

The spectrum of secondary particles at the ground can be seen in Figure 1(right). The secondaries were produced simulating four hours of cosmic rays flux over Buenos Aires (observation level of 25 m a.s.l). While electromagnetic particles (green line) dominate the spectrum at lower energies, muons (blue line) are dominant at higher energies since they are less attenuated in the atmosphere. The hadronic component (brown line) represents the lower contribution of particles at the ground, since most of the hadrons decay into muons or electromagnetic particles during shower development in the atmosphere.

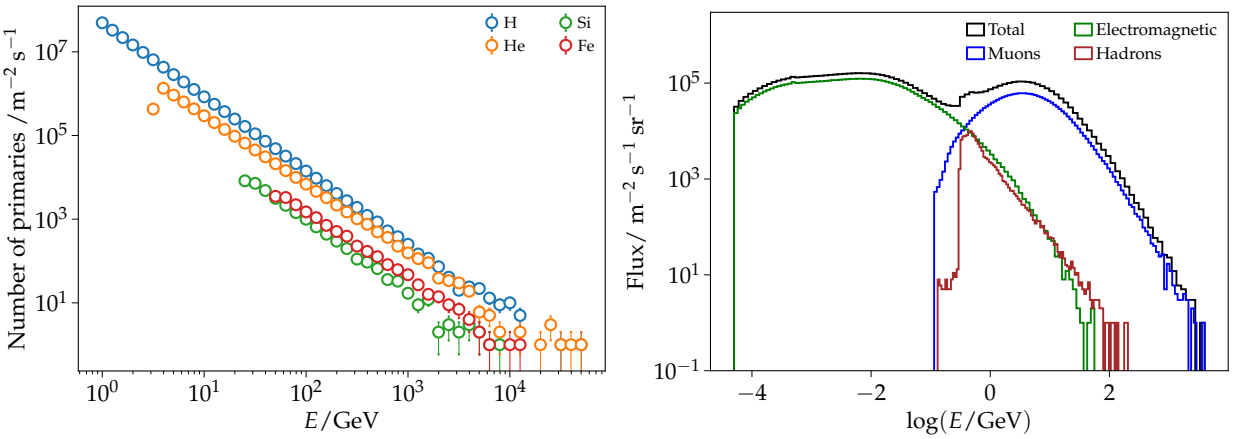

FIGURE 1: Left: energy spectrum of primary cosmic rays with energy from 10^{11} eV to 10^{15} eV calculated using equation (1). In this energy range, the cosmic rays spectrum is dominated by proton primaries. Right: flux of secondary particles at the ground. The flux was obtained simulating four hours of cosmic rays flux over Buenos Aires (25 m a.s.l.). Electromagnetic particles ($\gamma, e^{\pm}$) are shown in green, muons ($\mu^{\pm}$) in blue, and hadrons ($\pi^0, \pi^{\pm}, K^0, K^{\pm}, p, n$) in brown.

The Meiga framework uses the energy spectrum of secondary particles as well as their arrival direction which is given by the momentum components. Different particle types must be taken into account in our detector simulation. For example, electrons and positrons may be a source of noise in a muon detector, especially those placed at the surface where the rate of these particles is high in comparison to muons.

3. THE MEIGA FRAMEWORK

Meiga is a collection of C++ classes that integrate the cosmic ray flux calculation, particle propagation through a given material, and the simulation of the detector response. Both propagation and detector response are performed via Geant4 simulations. In this sense, Meiga uses Geant4 as an external toolkit and provides the necessary interfaces for the detector description and run managers which can be accessed by the user. The framework offers a set of dedicated applications which are easily configurable allowing Geant4 nonexperts to run and access the simulated data. In this section, an overview of the Meiga structure is given, and and an example application is shown.

3.1. Framework Structure

The Meiga framework has a hierarchical structure for accessing data. In Figure 2, a diagram with the workflow of Meiga is shown. The *Event* is at the top of the structure and is used for writing and reading data. It serves as a connection between the detector description and the simulated data and is in charge of the information flow during the run. When the program is executed, the input file containing the muon data is read by the *ReadParticleFile*, and this information is stored in the *Event*. The *Detector* class provides an interface to the *Event* from which the detector description can be read. For example, a muon detector made of plastic scintillator bars would consist in a *Detector* class where the bar length would be accessed through a member of that class.

The Geant4 *DetectorConstruction* reads the detector configuration from the *Event*. This can be done manually, i.e., the user has to write the *DetectorConstruction* and the *PhysicsList* on his/her own, or it can be done automatically by means of predefined detector models and physics lists which are contained in the *G4Models* class. The former is user configurable and requires more Geant4 knowledge than the latter. The detector models act as plug-in functions in the *DetectorConstruction* part of the code, enabling the construction of a given detector at a given position.

Simulated data is stored in the *SimData* class which inherits from the *Event*. A hierarchical structure of subclasses of *SimData* is built in order access data at different levels. In this sense, one can access to data at the *event* level, e.g., initial muon flux; and at the *detector* level, e.g., signal produced by passing muons through the detector. The hierarchical structure allows us to access data in an easy way using *setters* and *getters* methods. The *setters* methods are typically called during the Geant4 simulation where information from the Geant4 user action classes is stored in the Meiga *SimData* structure. The *getters* are called by the *DataWriter* functions at the end of the run when the desired data is dumped into output files.

In addition to the *SimData* and the *G4Models*, the Meiga framework contains a set of *Utilities* which include parser of configuration files, physics constants, and geometry and mathematical calculations. XML and JSON formats are used for configuration files that allow users better handling of the simulation settings. For example, a JSON file is used to set the input and output filenames, verbosity, and visualization options. On the other hand, XML files are used to set the *DetectorList*, containing a list of detector types and positions which are then read by the Geant4 detector construction classes. Physics constants are used to set the units and particle definitions of a common system between the external packages CORSIKA and Geant4.

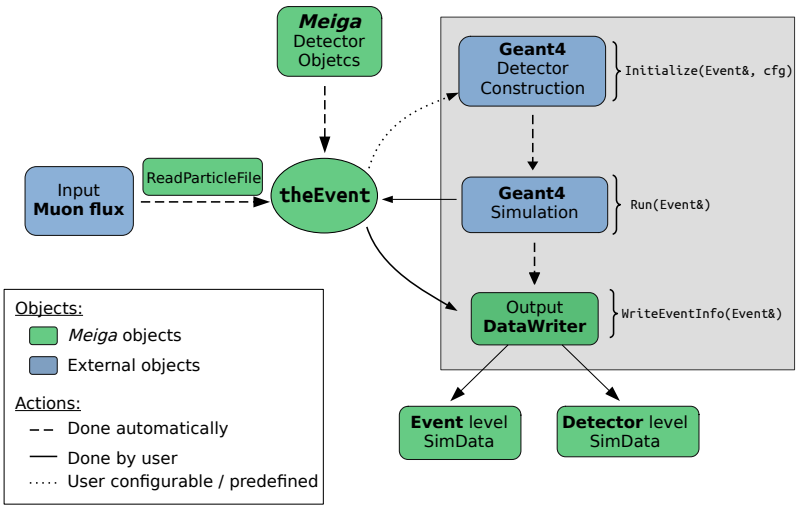

FIGURE 2: Workflow of the Meiga framework. The blue areas represent objects that belong to external packages while the green areas denote the Meiga objects. Arrows indicate the flow of information where dashed lines correspond to actions which are done automatically, solid lines correspond to actions done by the user and dotted lines correspond to actions that can be configured by the user, or performed based on a predefined configuration. The gray box represents the workflow of an *Application*. See text for more details.

3.2. *Meiga Application*

The structure of classes and functionalities described in the previous section represent the base of the Meiga framework. The simulation user task is integrated into what is called *Applications*, where each application is composed of three sequential methods that read the configuration, run the simulation, and write the output files, respectively. The application is run with an executable that is produced during the compilation process. This executable takes a configuration file as input to set the simulation parameters.

The `Initialize` method is called at the beginning of the program and is used to set up the simulation options. It reads the configuration file using a JSON parser of the C++ *boost* library. This configuration file contains the paths of the input and output files, flags to set the visualization and verbosity options, the detector, and the physics lists. The input file is used to fill the *Event* object with the information about the muon flux. The detector together and the physics lists are needed to set up the Geant4 simulation.

Secondly, the `Run` method is called. First, it reads the information loaded in the *Event* and starts the Geant4 simulation based on the detector configuration which is given by the detector and physics lists. The detector list is an XML file which contains the information of the type and position of the detector. This information is passed through the *Detector* objects to the Geant4 *DetectorConstruction* class. Then, the rest of the Geant4 action methods are initialized.

Once the configuration was successfully set, a loop over the input particle vector is performed in order to start the Geant4 simulation. Particles are injected at ground level with positions randomly distributed and directions given by their momentum components. Each particle constitutes a Geant4 run; i.e., when the injected particle and its products reach zero energy, a new particle of the vector is injected until the loop ends. In this sense, simulation data can be stored *particle-by-particle*. Cuts such as energy or particle production may be implemented by the user in the different action classes which are specific to each application.

An example of the muon transport simulation with a Meiga application is shown in Figure 3. For this particular case, a cylindrical volume of 600 m of height and 150 m of the radius was filled with standard rock (Figure 3(left)). Muons (and their products) are tracked inside the simulation volume. The number of muons that arrived at the bottom of the cylinder is shown in Figure 3(right). These muons were produced in a simulation of one year of cosmic ray flux as explained in Section 2.

An example of the simulation of the detector response is depicted in Figure 4. Methods inside the applications are developed to collect information when particles hit a sensitive volume of the detector. For example, a muon hitting a detector made of plastic scintillator bars with wave-length shifting (WLS) fibers that transport the light to a silicon photomultiplier (SiPM) can be seen in Figure 4(left). The time traces generated by that muon on two SiPMs are shown in Figure 4(right). Time traces are computed by adding the single photo-electron pulses of each photon detected by the SiPM. They are used for signal calculation and for imaging reconstruction algorithms.

After the simulation is done, the `WriteEventInfo` method is called, and the desired information is dumped into output files. The output consists of a collection of ASCII files where muon transport information and the detector traces are stored. The level of verbosity of the output data can be adjusted for the needs o a particular application.

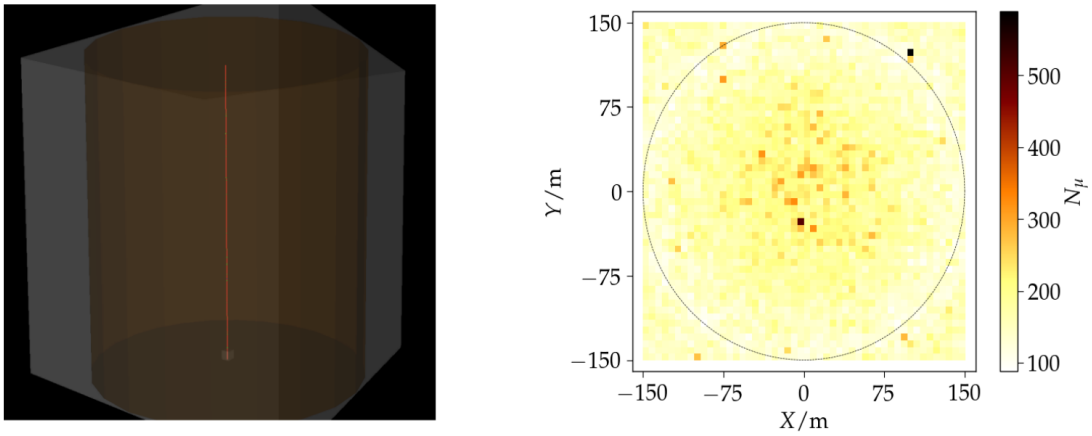

FIGURE 3: Example of muon transport simulation with Meiga. Left: schematics of the simulation volume with Geant4. A muon of 1 TeV of energy is injected in a block of standard rock ($\rho = 2.65\,\mathrm{g/cm^3}$) of 600 m depth. Right: muon number at 600 m of standard rock after injecting one year of secondary muons. The dotted line indicates the volume boundary.

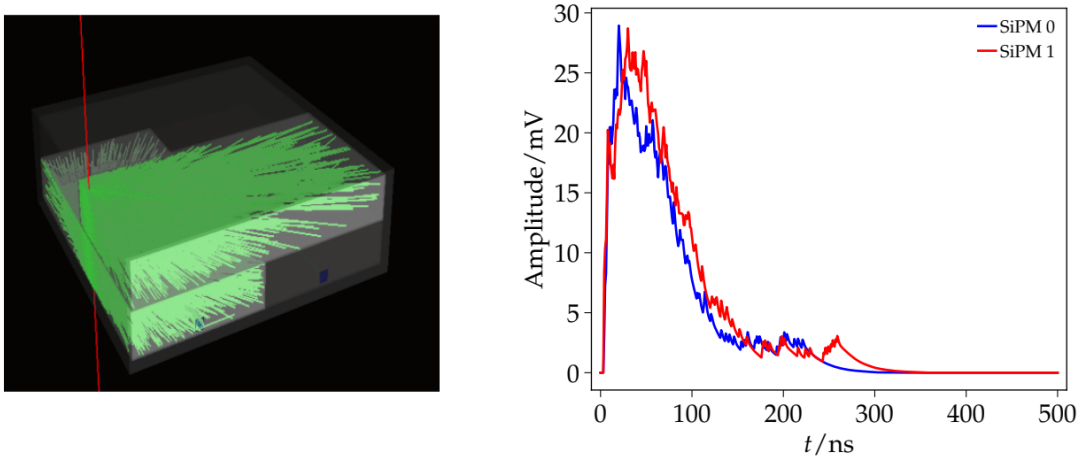

FIGURE 4: Example of a simulation of the detector response with Meiga. Left: the impinging muon (red line) crosses a detector composed of two panels with two scintillator bars, each one of 4 cm wide. Bars are traversed by WLS fibers which transport scintillation light (green lines) to a SiPM (blue squares). Right: SiPM pulses produced by the traversing muon.

4. CONCLUSIONS

The Meiga framework was conceived as a dedicated simulation tool for muography applications. The framework consists of a collection of classes that integrate a more robust calculation of the muon flux with its propagation through the media.

Its hierarchical structure grants data access in an easy way and the usage of configuration files allows Geant4 nonexperts to set up and run the applications. Meiga provides customized detector models commonly used for muography with a structure of classes for material definitions and detector configurations, allowing developers to integrate their own models in the framework.

CONFLICTS OF INTEREST

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] C. Hagmann et al., IEEE Nuclear Science Symposium Conference Record, pp. 1143–1146 (2007).
[2] D. Pagano et al., Nucl. Instrum. Methods. Phys. Res. B **1014**. 165732. (2021).
[3] D. Heck et al., Technical Report **FZKA 6019**, Forschungszentrum Karlsruhe GmbH (1998).
[4] V. Niess et al., Computer Physics Communications **229** (2017).
[5] S. Agostinelli et al., Nucl. Instrum. Methods Phys. Res., A **506** 250-303 (2003).
[6] H. Asorey et al., these proceedings (2021).
[7] C. Sarmiento-Cano et al., EPJ C, submitted, arXiv:2010.14591 [astro-ph.IM] (2021).