instruments

Article

DOME: Discrete Oriented Muon Emission in GEANT4 Simulations

Ahmet Ilker Topuz [1,2,*], Madis Kiisk [1,3] and Andrea Giammanco [2]

1 Institute of Physics, University of Tartu, W. Ostwaldi 1, 50411 Tartu, Estonia
2 Centre for Cosmology, Particle Physics and Phenomenology, Université Catholique de Louvain, Chemin du Cyclotron 2, B-1348 Louvain-la-Neuve, Belgium
3 GScan OU, Maealuse 2/1, 12618 Tallinn, Estonia
* Correspondence: ahmet.ilker.topuz@ut.ee or ahmet.topuz@uclouvain.be

Abstract: The simulation of muon tomography requires a multi-directional particle source that traverses a number of horizontal detectors of limited angular acceptance that are used to track cosmic-ray muons. In this study, we describe a simple strategy that can use GEANT4 simulations to produce a hemispherical particle source. We initially generate random points on a spherical surface of practical radius by using a Gaussian distributions for the three components of the Cartesian coordinates, thereby obtaining a generating surface for the initial position of the particles to be tracked. Since we do not require the bottom half of the sphere, we take the absolute value of the vertical coordinate, resulting in a hemisphere. Next, we direct the generated particles into the target body by selectively favoring the momentum direction along the vector constructed between a random point on the hemispherical surface and the origin of the target, thereby minimizing particle loss through source biasing. We also discuss a second scheme where the coordinate transformation is performed between the spherical and Cartesian coordinates, and the above-source biasing procedure is applied to orient the generated muons towards the target. Finally, a recipe based on restrictive planes from our previous study is discussed. We implement our strategies by using G4ParticleGun in the GEANT4 code. While we apply these techniques to simulations for muon tomography via scattering, these source schemes can be applied to similar studies for atmospheric sciences, space engineering, and astrophysics where a 3D particle source is a necessity.

Keywords: muon tomography; GEANT4; Monte Carlo simulations; discreet energy spectra; source biasing; restrictive planes

check for updates

Citation: Topuz, A.I.; Kiisk, M.; Giammanco, A. DOME: Discrete Oriented Muon Emission in GEANT4 Simulations. *Instruments* **2022**, *6*, 42. https://doi.org/10.3390/instruments6030042

Academic Editors: Alan D. Bross, Jacques E. Marteau and Antonio Ereditato

Received: 23 May 2022
Accepted: 14 September 2022
Published: 15 September 2022

1. Introduction

In the past, a variety of source geometries have been utilized for specific applications in muon imaging simulation studies, including planar surfaces and parabolic beams, as well as hemispherical surfaces [1]. In this study, we describe the implementation of two schemes aimed at building a hemispherical muon source where the generated particles are oriented towards a specific point or plane, using what we call the "selective momentum" direction. While there are different schemes to generate 2D/3D sources, we prefer to use the existing algorithms in GEANT4 [2], i.e., G4RandGauss :: shoot() and G4UniformRand() as the distribution function. Whereas the geometrical shape of the 2D/3D sources plays an important role in a particular application, the momentum direction is another variable the user must specify. In this study, we first generate a spherical surface by using three Gaussian distributions for the three components of the Cartesian coordinates and we direct the generated particles from their initial positions on this spherical surface to the preferred location(s) by using a vector constructed, as described in our previous study [3]. This methodology is called discrete oriented muon emission (DOME), where the kinetic energy of the generated particles is intentionally discrete for the computational purposes, as already implemented in another study [4]. In the latter scheme, we generate the initial positions by randomizing the spherical variables, i.e., azimuth and longitude, and we perform the coordinate transformation from the spherical coordinates to the Cartesian coordinates [5–7]. We repeat the same operations as performed in the first scheme. This paper is organized as follows. Section 2.1 describes the first scheme that is based on the Gaussian distribution functions, while Section 2.2 consists of the second methodology founded on the coordinate transformation from spherical to Cartesian coordinates. An alternative focusing scheme is explained in Section 3, and we summarize our conclusions in Section 4.

2. Central Focus Scheme

2.1. Generation through Gaussian Distributions

Our objective is to build a hemispherical muon source that surrounds our detector setup [8] similar to the other configurations existing in the literature [9–11], as illustrated in Figure 1. First, the particle locations in Cartesian coordinates are generated by using the Gaussian distributions formally defined as G4RandGauss::shoot() in GEANT4 as:

$$x_0 = G(\bar{x}, \sigma_x, x) = \text{G4RandGauss} :: \text{shoot}(), \tag{1}$$

and

$$y_0 = G(\bar{y}, \sigma_y, y) = \text{G4RandGauss} :: \text{shoot}(), \tag{2}$$

and

$$z_0 = G(\bar{z}, \sigma_z, z) = \text{G4RandGauss} :: \text{shoot}(). \tag{3}$$

where $\bar{x} = \bar{y} = \bar{z} = 0$ and $\sigma_x = \sigma_y = \sigma_z = 1$ by definition. The generated spatial points are renormalized in order to form a unit sphere, as indicated in

$$x_0^* = \frac{x_0}{\sqrt{x_0^2 + y_0^2 + z_0^2}}, \quad y_0^* = \frac{y_0}{\sqrt{x_0^2 + y_0^2 + z_0^2}}, \quad z_0^* = \frac{z_0}{\sqrt{x_0^2 + y_0^2 + z_0^2}}. \tag{4}$$

Given a sphere of radius denoted by R, the initial positions on the spherical surface of radius R in cm in the Cartesian coordinates are obtained as follows

$$x_i = R * x_0^*, \quad y_i = R * |y_0^*| = R * \text{ABS}(y_0^*), \quad z_i = R * z_0^*. \tag{5}$$

where the y-component of the Cartesian coordinates constituting the vertical axis is positively defined in order to yield the hemispherical surface. Then, the generated particles on the spherical surface are directed to the origin

$$x_f = 0, \quad y_f = 0, \quad z_f = 0. \tag{6}$$

By constructing a vector from the hemispherical surface to the origin, one obtains

$$px = x_f - x_i, \quad py = y_f - y_i, \quad pz = z_f - z_i. \tag{7}$$

Thus, the selective momentum direction denoted by $\vec{P} = (P_x, P_y, P_z)$ is

$$P_x = \frac{px}{\sqrt{px^2 + py^2 + pz^2}}, \quad P_y = \frac{py}{\sqrt{px^2 + py^2 + pz^2}}, \quad P_z = \frac{pz}{\sqrt{px^2 + py^2 + pz^2}}. \tag{8}$$

The developed code via the Gaussian distributions is given in Appendix A.

2.2. Generation via Coordinate Transformation

The second scheme is composed of the coordinate transformation, as depicted in Figure 2. To begin, two numbers, q_1 and q_2, are uniformly generated and inserted into the associated expression of the spherical variables as follows

$$q_1 = \text{G4UniformRand}(), \tag{9}$$

and

$$q_2 = \text{G4UniformRand}(). \tag{10}$$

The surface generation is initiated by randomizing θ as well as φ, as shown in

$$\theta = \arccos(2 \times q_1 - 1), \tag{11}$$

and

$$\varphi = 2 \times \pi \times q_2. \tag{12}$$

The coordinate transformation yields the generated points on the hemispherical surface of radius R in Cartesian coordinates, as described in

$$x_i = R \times \cos\theta \times \cos\varphi, \tag{13}$$

and

$$y_i = R \times |\sin\theta| = R \times \text{ABS}(\sin\theta), \tag{14}$$

and

$$z_i = R \times \cos\theta \times \sin\varphi. \tag{15}$$

The y-component of the Cartesian coordinates constituting the vertical axis is positively defined in order to yield the hemispherical surface as usual. Then, the generated particles on the spherical surface are again directed to the origin

$$x_f = 0, \quad y_f = 0, \quad z_f = 0. \tag{16}$$

By constructing a vector from the hemispherical surface to the origin, one obtains

$$px = x_f - x_i, \quad py = y_f - y_i, \quad pz = z_f - z_i. \tag{17}$$

Thus, the selective momentum direction denoted by $\vec{P} = (P_x, P_y, P_z)$ is

$$P_x = \frac{px}{\sqrt{px^2 + py^2 + pz^2}}, \quad P_y = \frac{py}{\sqrt{px^2 + py^2 + pz^2}}, \quad P_z = \frac{pz}{\sqrt{px^2 + py^2 + pz^2}}. \tag{18}$$

The obtained code by means the coordinate transformation is shown in Appendix B. A simulation preview through either scheme is displayed in Figure 3.

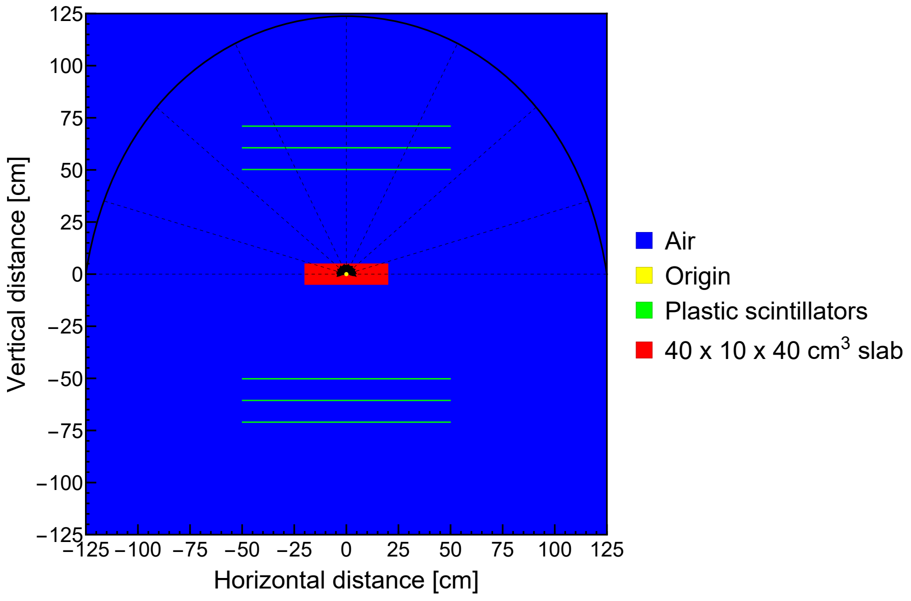

Figure 1. Delineation of the generated particles from the hemispherical source with a momentum direction towards the origin.

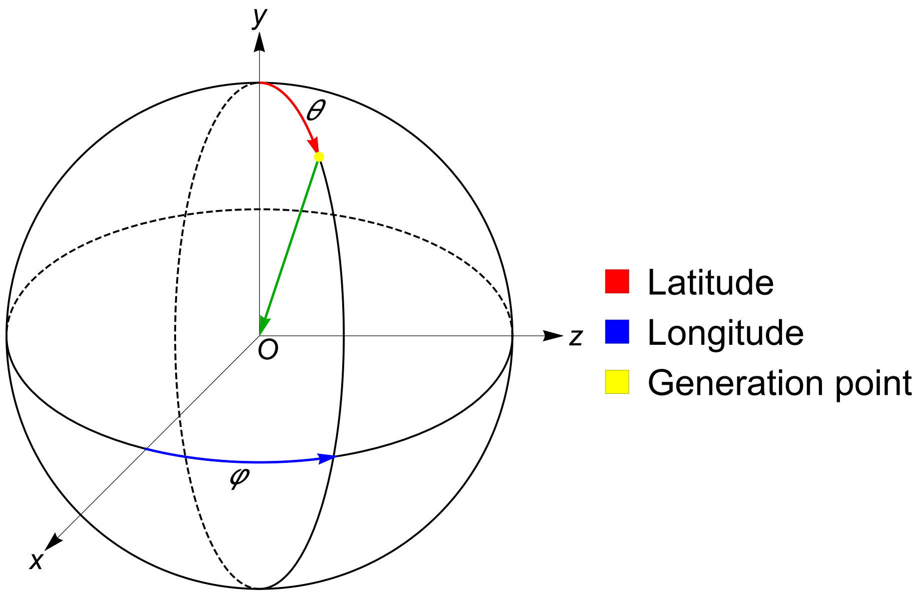

Figure 2. Spherical variables consisting of latitude denoted by θ and longitude indicated by φ with respect to the Cartesian coordinates (x,y,z).

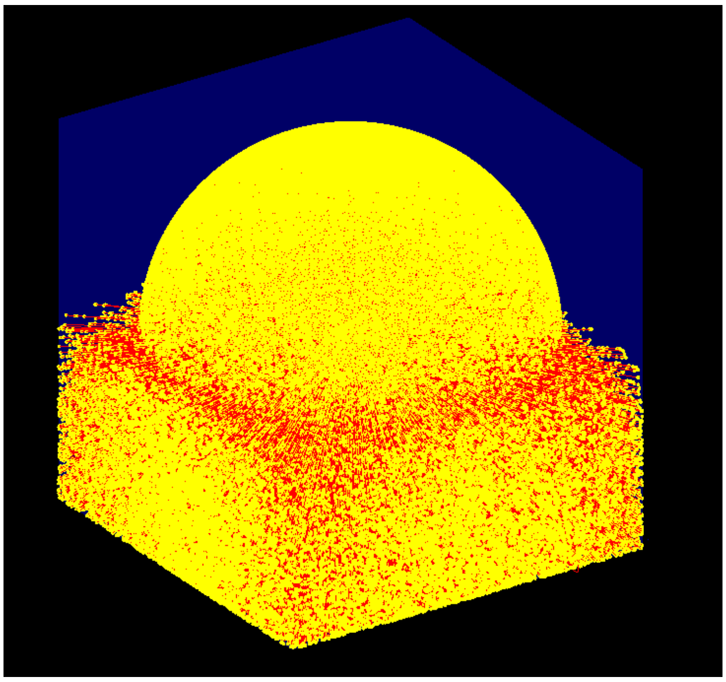

Figure 3. Hemispherical muon source in GEANT4.

3. Restrictive Planar Focus Scheme

As described in another study [3], the generated particles from any initial point on the hemispherical surface can be directed to a location randomly selected on a pseudo plane that restricts the momentum direction and which also leads to the minimization of the particle loss. Thus, the particle locations in cm on a restrictive plane of $2L \times 2D$ cm^2 situated at y = 0 will have the spatial coordinates, such that

$$x_f = -L + 2 \times L \times \text{G4UniformRand}(), \quad y_f = 0, \quad z_f = -D + 2 \times D \times \text{G4UniformRand}(). \tag{19}$$

Then, by constructing a vector from the generated hemispherical surface to the restrictive plane, one obtains

$$px = x_f - x_i, \quad py = y_f - y_i, \quad pz = z_f - z_i. \tag{20}$$

Thus, the selective momentum direction, i.e., $\vec{P} = (P_x, P_y, P_z)$, is

$$P_x = \frac{px}{\sqrt{px^2 + py^2 + pz^2}}, \quad P_y = \frac{py}{\sqrt{px^2 + py^2 + pz^2}}, \quad P_z = \frac{pz}{\sqrt{px^2 + py^2 + pz^2}}. \tag{21}$$

4. Conclusions

In this study, we explored the use of random number generators that are defined in the GEANT4 code. This can provide a number of source schemes where the first strategy is based on the Gaussian distributions, whereas the latter procedure requires a coordinate transformation to spherical variables. Finally, we obtain a hemispherical muon source where the kinetic energies of the generated muons are binned, and the momentum directions of these generated muons are selected by means of vector constructions. We call this source discrete oriented muon emission (DOME). DOME has been developed for simulations of muon tomography scenarios where the volume of interest is contained in a gap between detection layers, and the hemispheric source surrounds the entire setup. However, it can find applications in a broader array of use cases. For example, as demonstrated in [1], hemispheric sources are computationally efficient and at the same time unbiased for measurements of the cosmic muon flux where the detector has a complex geometry. Moreover, nothing prevents applications of the same method in simulations of muon radiography setups for volcanoes or pyramids or other very large objects of interest that are distant from the detector [12] where solid angle restrictions can optionally be imposed to increase computational efficiency.

Author Contributions: Methodology, A.I.T.; Software, A.I.T.; Supervision, M.K. and A.G.; Validation, A.I.T.; Visualization, A.I.T.; Writing—original draft, A.I.T.; Writing—review & editing, M.K. and A.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the EU Horizon 2020 Research and Innovation Programme under grant agreement No. 101021812 ("SilentBorder").

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Generation via Gaussian Distributions

```
#include "B1PrimaryGeneratorAction.hh"
#include "G4LogicalVolumeStore.hh"
#include "G4LogicalVolume.hh"
#include "G4Box.hh"
#include "G4RunManager.hh"
#include "G4ParticleGun.hh"
#include "G4ParticleTable.hh"
#include "G4ParticleDefinition.hh"
#include "G4SystemOfUnits.hh"
#include "Randomize.hh"
#include <iostream>
using namespace std;

B1PrimaryGeneratorAction::B1PrimaryGeneratorAction()
: G4VUserPrimaryGeneratorAction(),
fParticleGun(0)
//  fEnvelopeBox(0)
{
G4int n_particle = 1;
fParticleGun  = new G4ParticleGun(n_particle);

// default particle kinematic
G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
G4String particleName;
G4ParticleDefinition* particle
= particleTable->FindParticle(particleName="mu-");
fParticleGun->SetParticleDefinition(particle);
}

B1PrimaryGeneratorAction::~B1PrimaryGeneratorAction()
{
delete fParticleGun;
}

//80-bin Discrete CRY Energy Spectrum
void B1PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
//Discrete probabilities
double A[]= {0.0, 0.01253639, 0.02574546, 0.02802035, 0.02706636, 0.03528534, 0.02826496,
0.03157946, 0.03078447, 0.02777574, 0.02546415, 0.03150608, 0.02815489,
0.02580661, 0.02364179, 0.02170935, 0.02152589, 0.02348279, 0.02134243,
0.0196913,  0.02036398, 0.01841931, 0.01718402, 0.01700056, 0.01624226,
0.01539835, 0.01536166, 0.01471344, 0.01422421, 0.01412637, 0.01284215,
0.01260977, 0.01213278, 0.0129033,  0.01248746, 0.01196155, 0.01064064,
0.01057949, 0.0096255,  0.0103838,  0.00928304, 0.00879382, 0.00884274,
0.00793767, 0.00786429, 0.00769306, 0.00709376, 0.00736283, 0.0071916,
0.00721607, 0.00692253, 0.00643331, 0.00678799, 0.00673907, 0.00618869,
0.00634769, 0.00665346, 0.00650669, 0.00561385, 0.00589516, 0.00589516,
0.00578508, 0.00557716, 0.00550378, 0.00434187, 0.0043541,  0.00408503,
0.00364472, 0.00399941, 0.00388934, 0.00396272, 0.00431741, 0.00368142,
0.00363249, 0.00362026, 0.00410949, 0.00336342, 0.00358357, 0.00362026,
0.00348573, 0.0035958};
//Discrete energies
double B[]= {0.0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000,
2100, 2200, 2300, 2400, 2500, 2600, 2700, 2800, 2900, 3000,
3100, 3200, 3300, 3400, 3500, 3600, 3700, 3800, 3900, 4000,
4100, 4200, 4300, 4400, 4500, 4600, 4700, 4800, 4900, 5000,
5100, 5200, 5300, 5400, 5500, 5600, 5700, 5800, 5900, 6000,
6100, 6200, 6300, 6400, 6500, 6600, 6700, 6800, 6900, 7000,
7100, 7200, 7300, 7400, 7500, 7600, 7700, 7800, 7900, 8000};
G4int SizeEnergy=sizeof(B)/sizeof(B[0]);
G4int SizeProbability=sizeof(A)/sizeof(A[0]);

G4double Grid[sizeof(B)/sizeof(B[0])];
double sum=0;
for(int x=0; x < 81; x++){
sum=sum+A[x];
Grid[x]=sum;
std::ofstream GridFile;
GridFile.open("Probability_grid.txt", std::ios::app);
GridFile << Grid[x] << G4endl;
GridFile.close();
}
G4double radius=100*cm; //radius of sphere
for (int n_particle = 1; n_particle < 100000; n_particle++){
G4double x0=G4RandGauss::shoot();
std::ofstream GaussFile;
GaussFile.open("Gauss_x.txt", std::ios::app); //in mm
GaussFile << x0 << G4endl;
GaussFile.close();
//Centerally focused semi-spherical source via Gauss distributions
G4double y0=G4RandGauss::shoot();
G4double z0=G4RandGauss::shoot();
G4double n0=sqrt(pow(x0,2)+pow(y0,2)+pow(z0,2));
//Coordinates on sphere
x0 = radius*(x0/n0);
y0 = radius*abs(y0/n0);
z0 = radius*(z0/n0);
std::ofstream SphereFile;
SphereFile.open("coordinates_on_sphere.txt", std::ios::app); //in mm
SphereFile << x0 << " "<< y0 << " " << z0 << " " << G4endl;
SphereFile.close();
fParticleGun->SetParticlePosition(G4ThreeVector(x0,y0,z0));
//Aimed at origin
G4double x1=0;
G4double y1=0;
G4double z1=0;
G4double mx = x1-x0;
G4double my = y1-y0;
G4double mz = z1-z0;
G4double mn = sqrt(pow(mx,2)+pow(my,2)+pow(mz,2));
mx = mx/mn;
my = my/mn;
mz = mz/mn;
fParticleGun->SetParticleMomentumDirection(G4ThreeVector(mx,my,mz));
G4double Energy=0; //Just for initialization
G4double pseudo=G4UniformRand();
for (int i=0; i < 81; i++){
if(pseudo > Grid[i] && pseudo <= Grid[i+1]){
Energy=B[i+1];
std::ofstream EnergyFile;
EnergyFile.open("Energy.txt", std::ios::app);
EnergyFile << Energy << G4endl;
EnergyFile.close();
}
}
fParticleGun->SetParticleEnergy(Energy);
fParticleGun->GeneratePrimaryVertex(anEvent);
}
}
```

Appendix B. Generation by Means of Coordinate Transformation

```
#include "B1PrimaryGeneratorAction.hh"
#include "G4LogicalVolumeStore.hh"
#include "G4LogicalVolume.hh"
#include "G4Box.hh"
#include "G4RunManager.hh"
#include "G4ParticleGun.hh"
#include "G4ParticleTable.hh"
#include "G4ParticleDefinition.hh"
#include "G4SystemOfUnits.hh"
#include "Randomize.hh"
#include <iostream>
using namespace std;

B1PrimaryGeneratorAction::B1PrimaryGeneratorAction()
: G4VUserPrimaryGeneratorAction(),
fParticleGun(0)
//  fEnvelopeBox(0)
{
G4int n_particle = 1;
fParticleGun  = new G4ParticleGun(n_particle);

// default particle kinematic
G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
G4String particleName;
G4ParticleDefinition* particle
= particleTable->FindParticle(particleName="mu-");
fParticleGun->SetParticleDefinition(particle);
}

B1PrimaryGeneratorAction::~B1PrimaryGeneratorAction()
{
delete fParticleGun;
}

//80-bin Discrete CRY Energy Spectrum
void B1PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
//Discrete probabilities
double A[]= {0.0, 0.01253639, 0.02574546, 0.02802035, 0.02706636, 0.03528534, 0.02826496,
0.03157946, 0.03078447, 0.02777574, 0.02546415, 0.03150608, 0.02815489,
0.02580661, 0.02364179, 0.02170935, 0.02152589, 0.02348279, 0.02134243,
0.0196913,  0.02036398, 0.01841931, 0.01718402, 0.01700056, 0.01624226,
0.01539835, 0.01536166, 0.01471344, 0.01422421, 0.01412637, 0.01284215,
0.01260977, 0.01213278, 0.0129033,  0.01248746, 0.01196155, 0.01064064,
0.01057949, 0.0096255,  0.0103838,  0.00928304, 0.00879382, 0.00884274,
0.00793767, 0.00786429, 0.00769306, 0.00709376, 0.00736283, 0.0071916,
0.00721607, 0.00692253, 0.00643331, 0.00678799, 0.00673907, 0.00618869,
0.00634769, 0.00665346, 0.00650669, 0.00561385, 0.00589516, 0.00589516,
0.00578508, 0.00557716, 0.00550378, 0.00434187, 0.0043541,  0.00408503,
0.00364472, 0.00399941, 0.00388934, 0.00396272, 0.00431741, 0.00368142,
0.00363249, 0.00362026, 0.00410949, 0.00336342, 0.00358357, 0.00362026,
0.00348573, 0.0035958};
//Discrete energies
double B[]= {0.0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000,
2100, 2200, 2300, 2400, 2500, 2600, 2700, 2800, 2900, 3000,
3100, 3200, 3300, 3400, 3500, 3600, 3700, 3800, 3900, 4000,
4100, 4200, 4300, 4400, 4500, 4600, 4700, 4800, 4900, 5000,
5100, 5200, 5300, 5400, 5500, 5600, 5700, 5800, 5900, 6000,
6100, 6200, 6300, 6400, 6500, 6600, 6700, 6800, 6900, 7000,
7100, 7200, 7300, 7400, 7500, 7600, 7700, 7800, 7900, 8000};
G4int SizeEnergy=sizeof(B)/sizeof(B[0]);
G4int SizeProbability=sizeof(A)/sizeof(A[0]);

G4double Grid[sizeof(B)/sizeof(B[0])];
double sum=0;
for(int x=0; x < 81; x++){
sum=sum+A[x];
Grid[x]=sum;
std::ofstream GridFile;
GridFile.open("Probability_grid.txt", std::ios::app);
GridFile << Grid[x] << G4endl;
GridFile.close();
}
G4double radius=100*cm; //radius of sphere
for (int n_particle = 1; n_particle < 100000; n_particle++){
//Centerally focused semi-spherical source via coordinate transformation
G4double rand1=G4UniformRand();
G4double rand2=G4UniformRand();
G4double latitude=acos(2*rand1-1);
G4double longitude=2*3.14159265359*rand2;
//Coordinates on sphere
G4double x0=radius*cos(latitude)*cos(longitude);
G4double y0=radius*abs(sin(latitude));
G4double z0=radius*cos(latitude)*sin(longitude);
std::ofstream SphereFile;
SphereFile.open("coordinates_on_sphere.dat", std::ios::app); //in mm
SphereFile << x0 << " "<< y0 << " " << z0 << G4endl;
SphereFile.close();
fParticleGun->SetParticlePosition(G4ThreeVector(x0,y0,z0));
//Aimed at origin
G4double x1=0;
G4double y1=0;
G4double z1=0;
G4double mx = x1-x0;
G4double my = y1-y0;
G4double mz = z1-z0;
G4double mn = sqrt(pow(mx,2)+pow(my,2)+pow(mz,2));
mx = mx/mn;
my = my/mn;
mz = mz/mn;
fParticleGun->SetParticleMomentumDirection(G4ThreeVector(mx,my,mz));
G4double Energy=0; //Just for initialization
G4double pseudo=G4UniformRand();
for (int i=0; i < 81; i++){
if(pseudo > Grid[i] && pseudo <= Grid[i+1]){
Energy=B[i+1];
std::ofstream EnergyFile;
EnergyFile.open("Energy.txt", std::ios::app);
EnergyFile << Energy << G4endl;
EnergyFile.close();
}
}
fParticleGun->SetParticleEnergy(Energy);
fParticleGun->GeneratePrimaryVertex(anEvent);
}
}
```

References

1. Pagano, D.; Bonomi, G.; Donzella, A.; Zenoni, A.; Zumerle, G.; Zurlo, N. EcoMug: An Efficient COsmic MUon Generator for cosmic-ray muon applications. *Nucl. Instrum. Methods Phys. Res. Sect. A Accel. Spectrom. Detect. Assoc. Equip.* **2021**, *1014*, 165732. [CrossRef]
2. Agostinelli, S.; Allison, J.; Amako, K.; Apostolakis, J.; Araujo, H.; Arce, P.; Asai, M.; Axen, D.; Banerjee, S.; Barrand, G.; et al. GEANT4—A simulation toolkit. *Nucl. Instrum. Methods Phys. Res. Sect. A Accel. Spectrom. Detect. Assoc. Equip.* **2003**, *506*, 250–303. [CrossRef]
3. Topuz, A.I.; Kiisk, M.; Giammanco, A. Particle generation trough restrictive planes in GEANT4 simulations for potential applications of cosmic ray muon tomography. *arXiv* **2022**, arXiv:2201.07068.
4. Topuz, A.I.; Kiisk, M. Towards energy discretization for muon scattering tomography in GEANT4 simulations: A discrete probabilistic approach. *arXiv* **2022**, arXiv:2201.08804.
5. Marsaglia, G. Choosing a point from the surface of a sphere. *Ann. Math. Stat.* **1972**, *43*, 645–646. [CrossRef]
6. Tashiro, Y. On methods for generating uniform random points on the surface of a sphere. *Ann. Inst. Stat. Math.* **1977**, *29*, 295–300. [CrossRef]
7. Weisstein, E.W. "Disk Point Picking". From MathWorld-A Wolfram Web Resource. 2011. Available online: http://mathworld.wolfram.com/ (accessed on 22 May 2022).
8. Georgadze, A.; Kiisk, M.; Mart, M.; Avots, E.; Anbarjafari, G. Method and Apparatus for Detection and/or Identification of Materials and of Articles Using Charged Particles. US Patent 16/977,293, 7 January 2021.
9. Borozdin, K.N.; Hogan, G.E.; Morris, C.; Priedhorsky, W.C.; Saunders, A.; Schultz, L.J.; Teasdale, M.E. Radiographic imaging with cosmic-ray muons. *Nature* **2003**, *422*, 277. [CrossRef] [PubMed]
10. Frazão, L.; Velthuis, J.; Maddrell-Mander, S.; Thomay, C. High-resolution imaging of nuclear waste containers with muon scattering tomography. *J. Instrum.* **2019**, *14*, P08005. [CrossRef]
11. Frazão, L.; Velthuis, J.; Thomay, C.; Steer, C. Discrimination of high-Z materials in concrete-filled containers using muon scattering tomography. *J. Instrum.* **2016**, *11*, P07020. [CrossRef]
12. Bonechi, L.; D'Alessandro, R.; Giammanco, A. Atmospheric muons as an imaging tool. *Rev. Phys.* **2020**, *5*, 100038. [CrossRef]