GHENT
24-26 NOV.
Muographers
2021

A muon generator for cosmic-ray muon applications

EcoMug: An Efficient COsmic MUon Generator for cosmic-ray muon applications

D. Pagano [a,b,*], G. Bonomi [a,b], A. Donzella [a,b], A. Zenoni [a,b], G. Zumerle [c,d], N. Zurlo [e,b]

Nuclear Inst. and Methods in Physics Research, A 1014 (2021) 165732

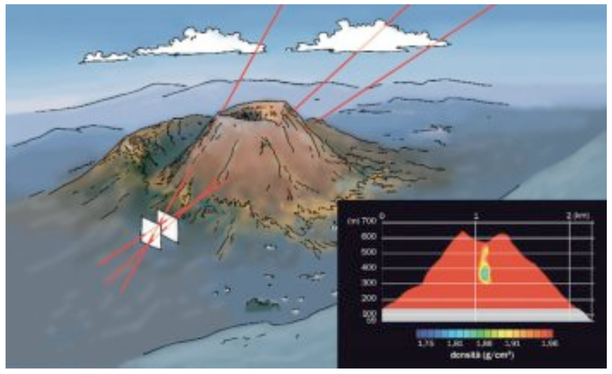
UNIVERSITY OF BRESCIA

Germano Bonomi

G. Bonomi [a,b,*], P. Checchia [c], M. D'Errico [d,e], D. Pagano [a,b], G. Saracino [d,e]

Applications of cosmic-ray muons

Transmission muography, scattering muography, metrology muography …



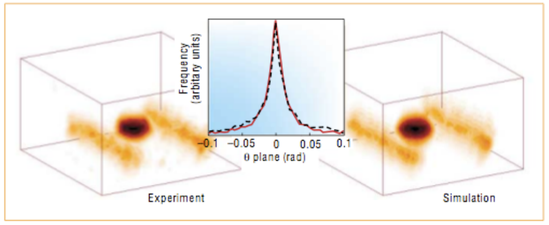


Figure 1 Radiographic imaging with muons of a test object (left) and the reconstructed image of its Monte Carlo simulation (right). The test object is a tungsten cylinder (radius, 5.5 cm; height, 5.7 cm) on a plastic ($35 \times 60 \times 1$ cm^3) plate with two steel support rails. The tungsten cylinder and the iron in the rails are clearly visible in both the experiment and simulation reconstructions. Inset, the widths of the scattering distributions for tracks passing through the tungsten target are very similar for the experimental and simulated data.

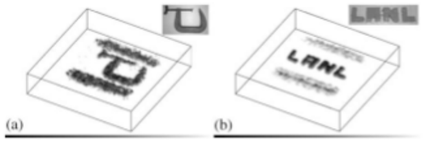
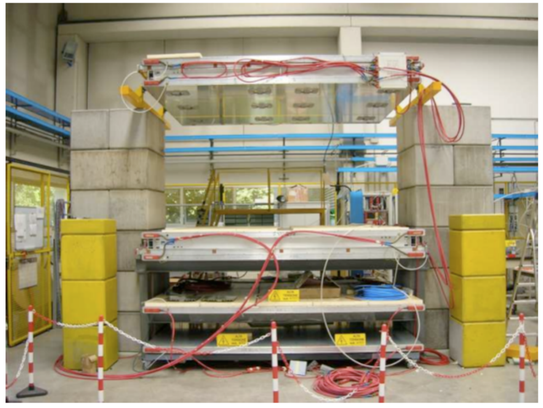
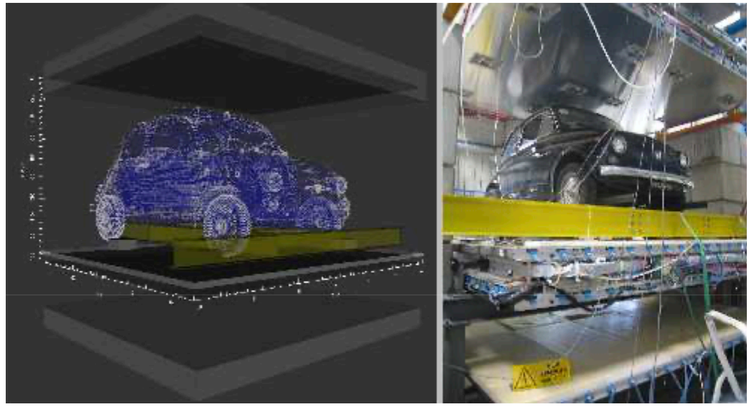
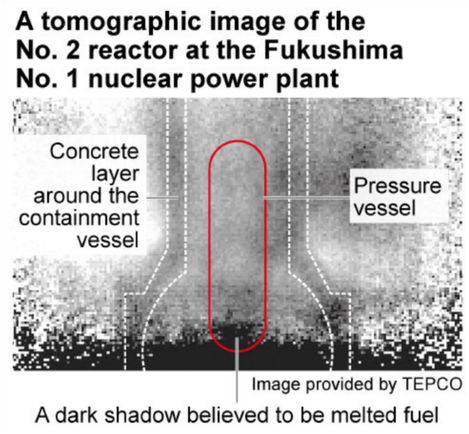
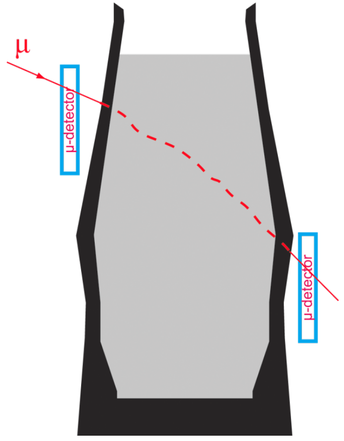


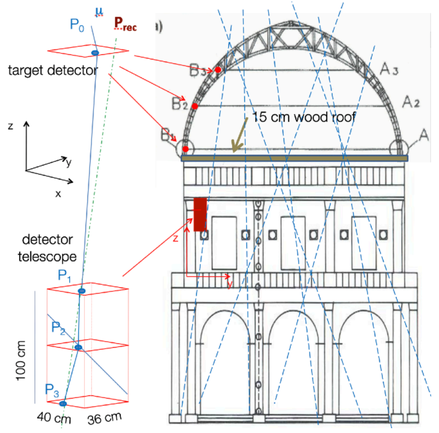
Fig. 5. Experimentally produced cosmic ray muon radiographs of (a) a steel c-clamp, and (b) "LANL" constructed from 1″ lead stock. The bar-like features result from steel beams used to











Nowadays, all applications of cosmic-ray muons rely on simulation tools

CERN Accelerating science Sign in Directory

GEANT4 A SIMULATION TOOLKIT

Download | User Forum
Contact Us | Bug Reports

Geant4

GEometry **AN**d **T**racking

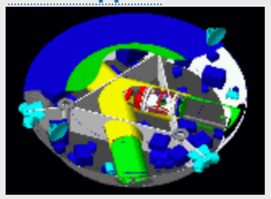
Overview

Geant4 is a toolkit for the simulation of the passage of particles through matter. Its areas of application include high energy, nuclear and accelerator physics, as well as studies in medical and space science. The three main reference papers for Geant4 are published in Nuclear Instruments and Methods in Physics Research A 506 (2003) 250-303, IEEE Transactions on Nuclear Science 53 No. 1 (2006) 270-278 and Nuclear Instruments and Methods in Physics Research A 835 (2016) 186-225.

Applications

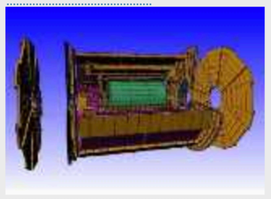
A sampling of applications, technology transfer and other uses of Geant4

User Support



Getting started, guides and information for users and developers

Publications



Validation of Geant4, results from experiments and publications

Collaboration

Who we are: collaborating institutions, members, organization and legal information

It is the most complete, reliable and basically the de facto statutory toolkit for simulations of cosmic-ray applications

All the physics about the interaction of muons with matter is included in a simulation package developed at CERN and called GEANT4. Indeed GEANT4 is a toolkit for **simulating the passage of particles through matter**. It includes a complete range of functionality including **tracking, geometry, physics models and hits**.

The toolkit is the result of a worldwide collaboration of physicists and software engineers. It has been created exploiting software engineering and object-oriented technology and implemented in the C++ programming language. It is being used in applications in particle physics, nuclear physics, **applied physics**, accelerator design, space engineering and medical physics.

General - Simulations

CERN Accelerating science Sign in Directory

GEANT4 A SIMULATION TOOLKIT

Download | User Forum
Contact Us | Bug Reports

Geant4

GEometry **AN**d **T**racking (and detector response)

"Beam"

The primary particle generator: proton, electron, antiproton, …
or **muons** [*position, energy and direction distributions must be defined*]

Geometry

All the "world" of the simulation must be modelled: shape, dimensions, material composition of all the elements

Detectors

Shape, dimensions, material composition, hit response …

Physics and tracking parameters, output format

Physical (interaction) models, physical processes, tracking steps …

MC simulation output

Detector hits, Primary particle information, tracking information, etc.

Generators available packages - Simulations

Nowadays, all applications of cosmic-ray muons rely on a cosmic-ray generator:

Cosmic-ray air shower (CRAS) generators

simulation of the full cascade of secondary particles initiated by primary cosmic rays

CORSIKA

CRY

MCEq

Parametric generators

simulation using a parametrization of the flux of muons, based either on experimental data or on the results from simulations with CRAS generators

GEMC

CMSCGEN

Special generators

specifically designed simulations for underground, high altitude or underwater experiments

MuTev

MUPAGE

Custom generators

...

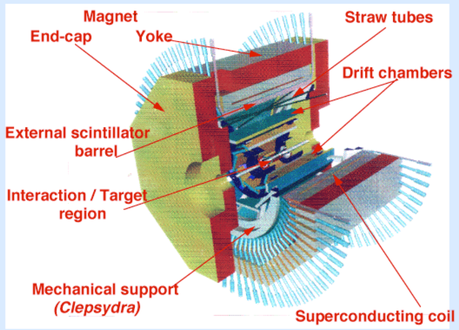
...

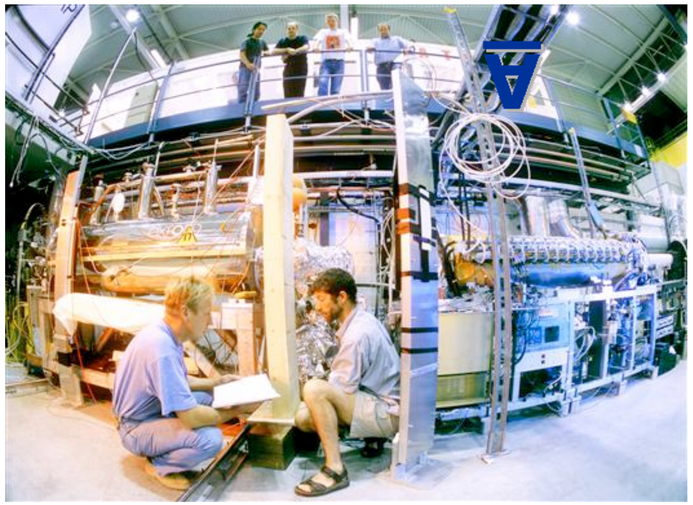
Transmission Muography applications are sensitive to the **angular distribution** of muons, and many applications of **scattering muography** are also sensitive to their **momentum distribution**. For these reasons, an accurate simulation of the dependency of the muon flux on momentum and direction is a key requirement for every generation tool targeting to such applications.

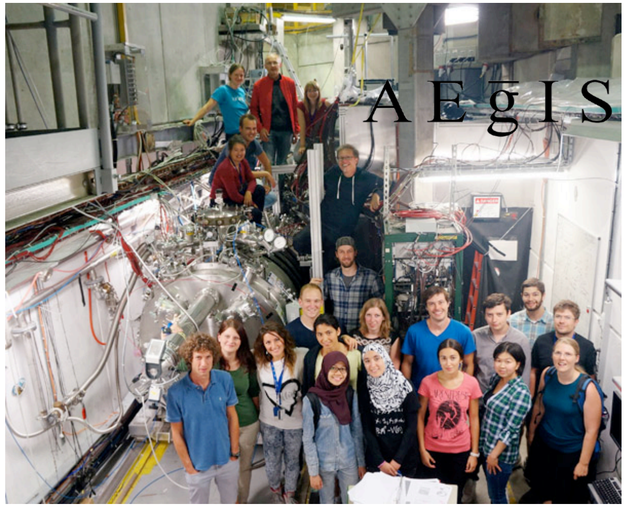
Moreover, as the inspection of large structures requires a very large statistics, **the generator has also to be fast**.

Our custom generator - Simulations

FINUDA experimental set-up







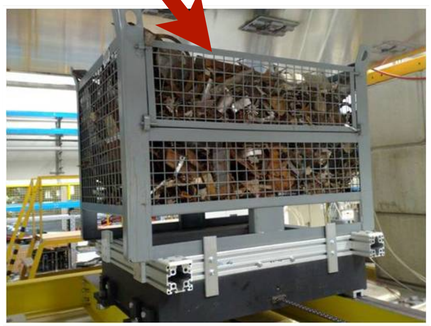
MU-STEEL

RFCS CT-2010-000033

mid 2010

end 2012

Research Fund for Coal & Steel

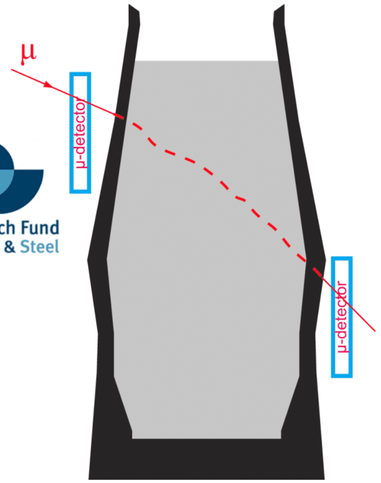


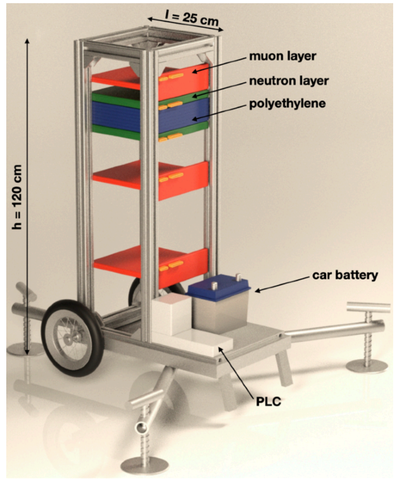
Mu-Blast - RFCS CT-2014-00027

mid 2014- mid 2016

Research Fund for Coal & Steel

INFN swerea MEFOS LKAB





mid '90s

.

.

.

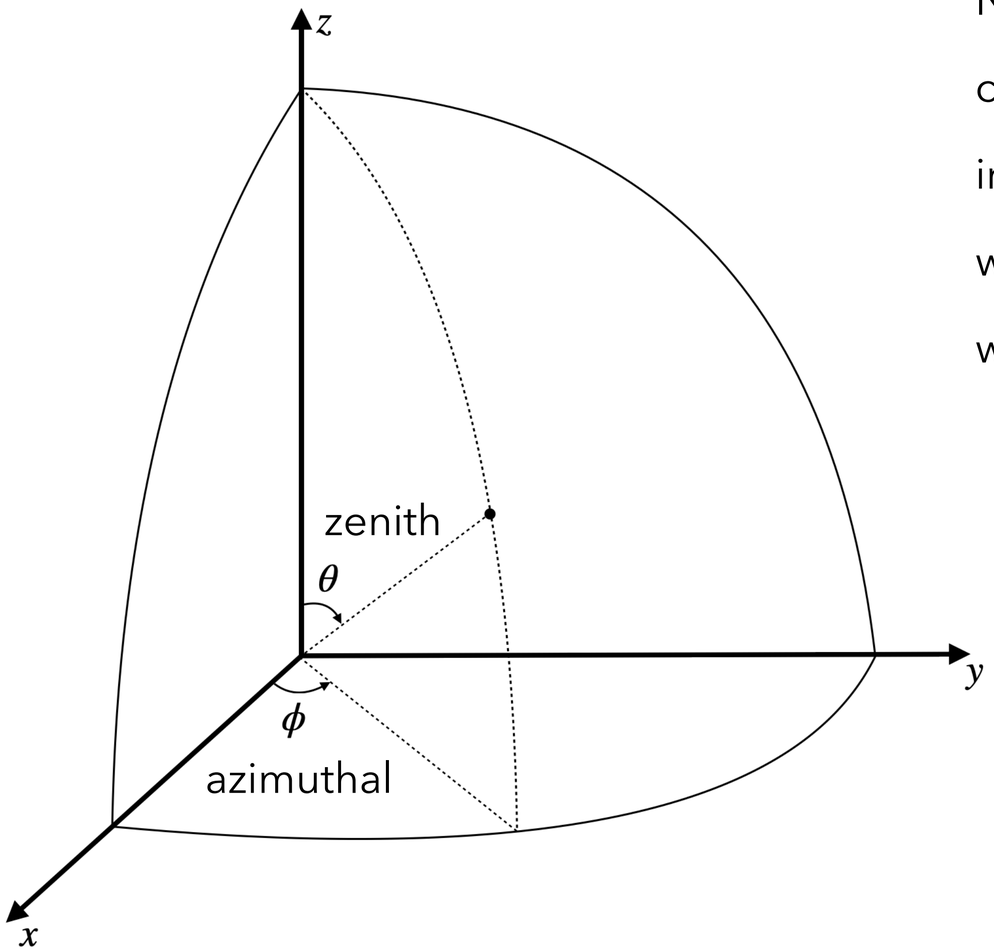
.

.

2021

in 2021 we decided to pack all the developments into a standalone toolkit => EcoMug

Differential flux



Number of cosmic-ray muons

crossing a surface dS_n perpendicular to the muon direction

in a interval of time dt

with momentum between p and p+dp,

within a dΩ solid angle around the direction (θ, Φ)

$$J \equiv J(t, p, \theta, \phi) = \frac{dN}{dt \cdot dp \cdot d\Omega \cdot dS_n}$$

being $d\Omega = \sin\theta \; d\theta \; d\phi$ we can also write

$$J \equiv J(t, p, \theta, \phi) = \frac{dN}{dt \cdot dp \cdot \sin\theta \cdot d\theta \cdot d\phi \cdot dS_n}$$

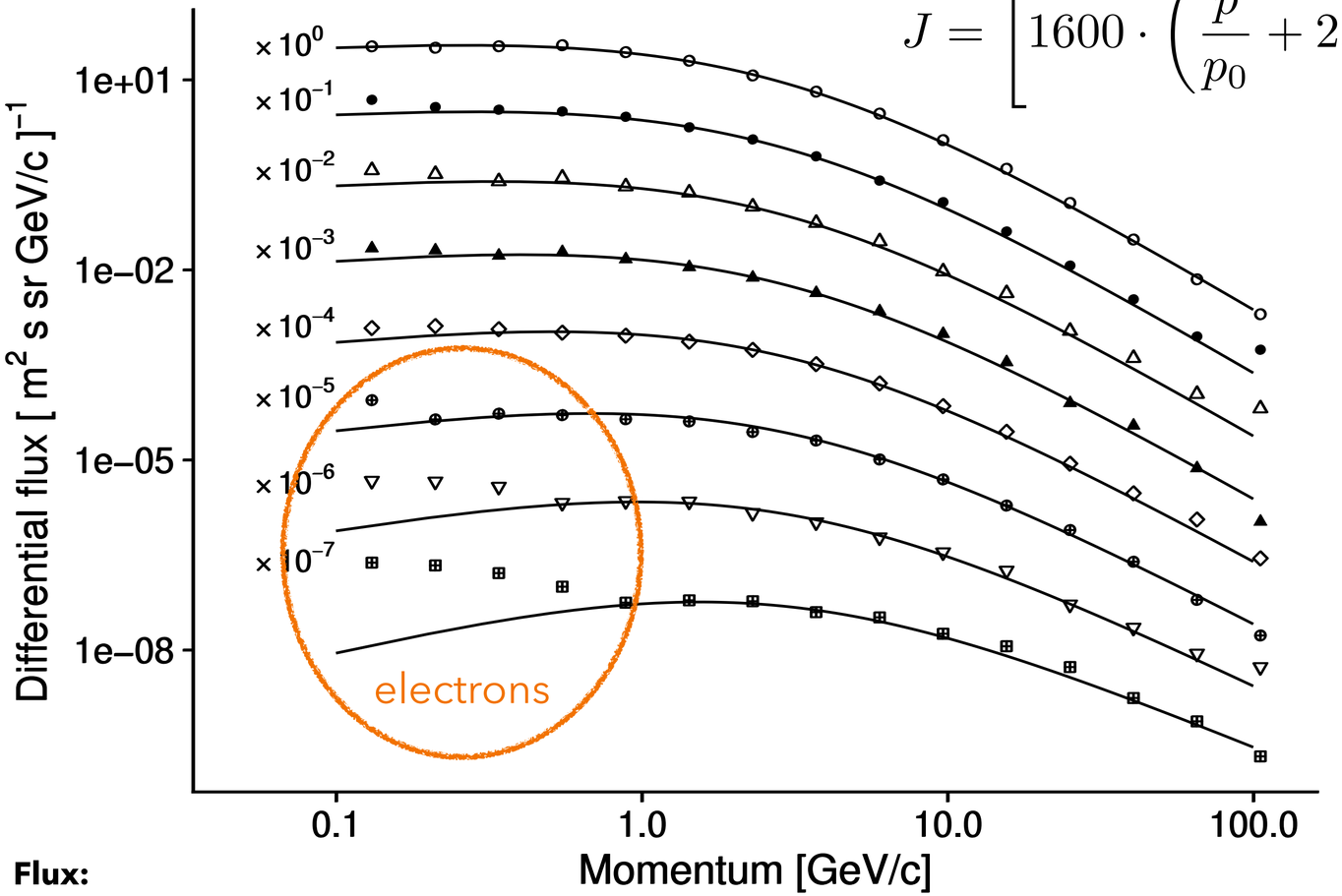
Angular and momentum distributions - EcoMug

The distributions of momentum and arrival direction of the muons in the simulation must be as close as possible to the "real"(*) one.

L. Bonechi, M. Bongi, D. Fedele, M. Grandi, S. Ricciarini, E. Vannuccini,
Development of the ADAMO detector: test with cosmic rays at different zenith angles,
in: 29th International Cosmic Ray Conference Vol. 9, 2005, pp. 283.

$$J = \left[1600 \cdot \left(\frac{p}{p_0} + 2.68\right)^{-3.175} \cdot \left(\frac{p}{p_0}\right)^{0.279}\right] \cdot (\cos\theta)^n \cdot \frac{1}{\mathrm{m}^2 \cdot \mathrm{s} \cdot \mathrm{sr} \cdot \mathrm{GeV/c}}$$

L. Bonechi, Misure di Raggi Cosmici a Terra con l'Esperimento ADAMO (Ph.D. thesis), Università degli Studi di Firenze, Dipartimento di Fisica, 2004.



- ○ 0° < θ < 10°
- ● 10° < θ < 20°
- △ 20° < θ < 30°
- ▲ 30° < θ < 40°
- ◇ 40° < θ < 50°
- ⊕ 50° < θ < 60°
- ▽ 60° < θ < 70°
- ⊞ 70° < θ < 80°

with $p_0 = 1\ \mathrm{GeV/c}$

$p > 0.040\ \ \mathrm{GeV/c}$

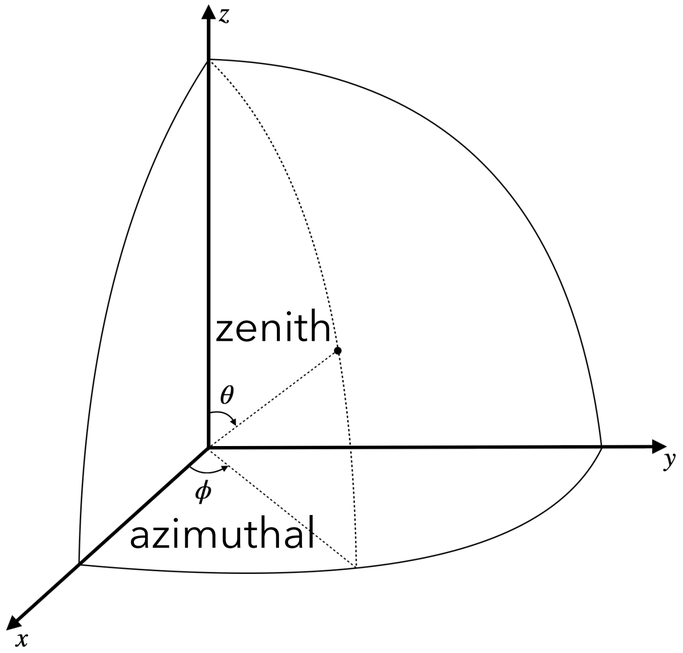
and $n(p) = \max\left[0.1,\ 2.856 - 0.655 \cdot \ln\left(\frac{p}{p_0}\right)\right]$

Flux:
integral over momentum and angle for a horizontal surface: **125 u/(m² s)**
integral over momentum and angle for a vertical surface: **60 u/(m² s) [both sides]**
Important to compute "real world data taking time equivalent" of simulations

*EcoMug generated muons with momentum **up to 1000 GeV/c and zenith angles up to 90°**. The user should keep in mind, though, that the above equation **could not describe accurately the flux outside the range of experimental data**, even though this can be fixed by resorting to reweighting techniques*

(*) Clearly there are no "ones for all real distributions". The momentum, arrival direction and flux may depend on many parameters: solar activity, altitude, air temperature and pressure, etc. etc.

Definitions - EcoMug



Monte Carlo generations

It is more functional to consider the following:

Differential flux (')

Number of cosmic-ray muons

crossing a ***surface dS_n perpendicular to the muon direction***

in a interval of time dt

with momentum between p and p+dp

with zenith angle between θ and θ + dθ

with azimuthal angle between Φ and Φ + dΦ,

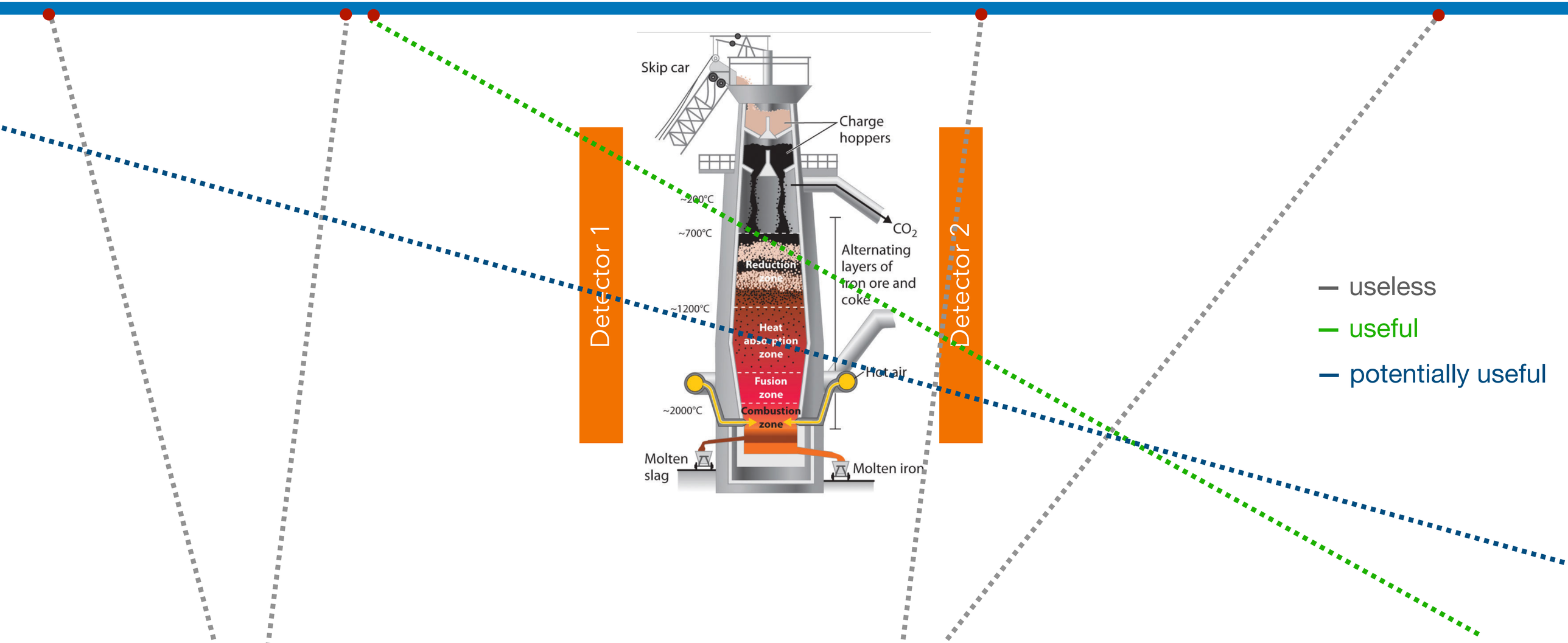
$$J' = \frac{dN}{dt \cdot dp \cdot d\theta \cdot d\phi \cdot dS_n}$$

$$J \equiv J(t,p,\theta,\phi) = \frac{dN}{dt \cdot dp \cdot d\Omega \cdot dS_n} \qquad d\Omega = \sin\theta \; d\theta \; d\phi \qquad J' = \frac{dN}{dt \cdot dp \cdot d\theta \cdot d\phi \cdot dS_n} = J(t,p,\theta,\phi) \cdot sin\theta$$

$$J' = \left[1600 \cdot \left(\frac{p}{p_0} + 2.68 \right)^{-3.175} \cdot \left(\frac{p}{p_0} \right)^{0.279} \right] \cdot (\cos\theta)^n \cdot sin\theta \cdot \frac{1}{\mathrm{m}^2 \cdot \mathrm{s} \cdot \mathrm{sr} \cdot \mathrm{GeV/c}}$$

Generation over a flat sky (horizontal surface)

To generate all the useful cosmic-ray muons, one would need an “infinite” flat sky => cumbersome in terms of computing



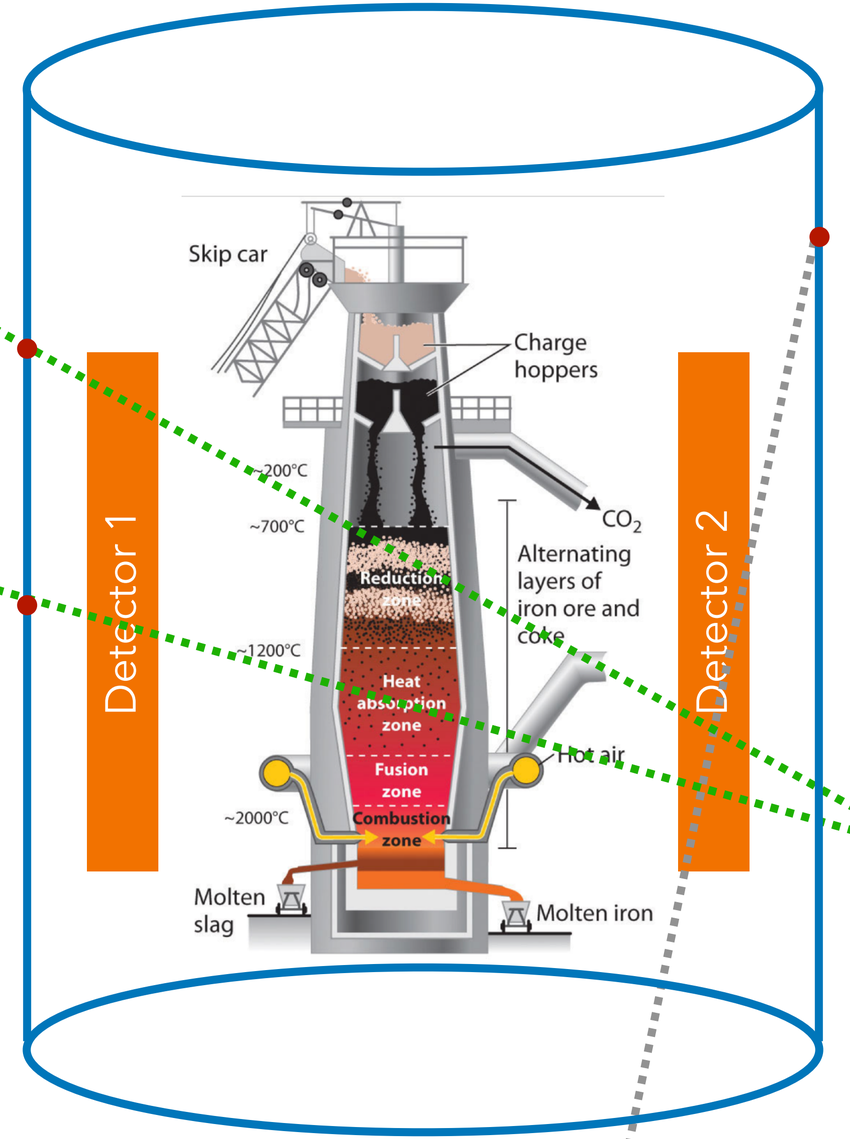
Generation over a cylindrical (lateral) surface

For muon tomography of large and tall structures, generation over a cylinder becomes much more efficient

EcoMug can generate over a:

- **horizontal** surface
  [flat sky generation]
- **cylinder lateral** surface
  [cylindrical generation]
- **hemispherical** surface
  [half-spherical generation]

... the generation of a muon in EcoMug involves the use of the acceptance-rejection method in a 2D space, for a flat sky and cylindrical surface, and in a 4D space, for the half-spherical surface, to sample according to the distributions described in the following ...



Since the statistics are large in these Monte Carlo simulations, a very fast and reliable pseudo-random number generator (PRNG) is useful. EcoMug internally uses a class called EMRandom, which is based on the xoroshiro128+ algorithm, the fastest PRNG algorithm.
The time required to generate 1M muons, using a compiled code with the -O3 flag on a 2.6 GHz Intel Core i7 6 core, was on average:

- **1.3 s for the flat sky generator;**
- **2.6 s for the cylindrical generator**
- **4.9 s for the half-sphere generator**

Differential flux

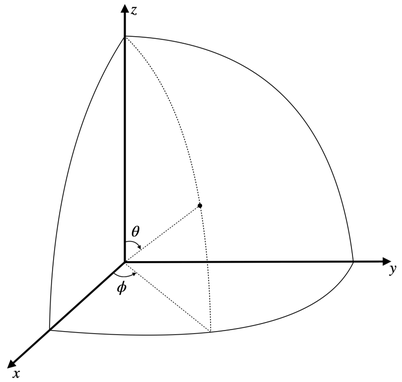
Number of cosmic-ray muons in a interval of time dt,

with momentum between p and p+dp,

with zenith angle between θ and θ + dθ,

with azimuthal angle between Φ and Φ + dΦ,

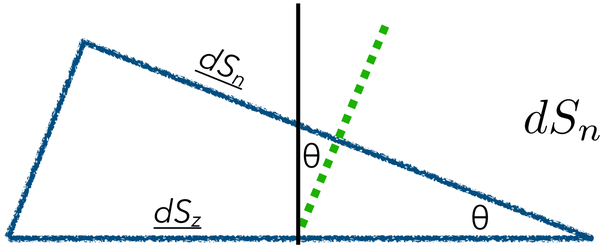
crossing a surface ***dS_n perpendicular to the muon direction***



$$J'(t,p,\theta,\phi) = \frac{dN}{dt \cdot dp \cdot d\theta \cdot d\phi \cdot dS_n}$$

1D simplification

… crossing a ***horizontal surface dS_z***



$$dS_n = dS_z \cdot \cos\theta$$

$$J'_z(t,p,\theta,\phi) = \frac{dN}{dt \cdot dp \cdot d\theta \cdot d\phi \cdot dS_z}$$

$$J'_z(t,p,\theta,\phi) = J'(t,p,\theta,\phi) \cdot \cos\,\theta$$

$$J'_z \equiv \frac{dN}{dt \cdot dp \cdot d\theta \cdot d\phi \cdot dS_z} = \left[1600 \cdot \left(\frac{p}{p_0} + 2.68\right)^{-3.175} \cdot \left(\frac{p}{p_0}\right)^{0.279}\right] \cdot (\cos\theta)^{n+1} \cdot \sin\theta \cdot \frac{1}{\mathrm{m}^2 \cdot \mathrm{s} \cdot \mathrm{sr} \cdot \mathrm{GeV/c}}$$

Cylindrical generation - EcoMug

Differential flux

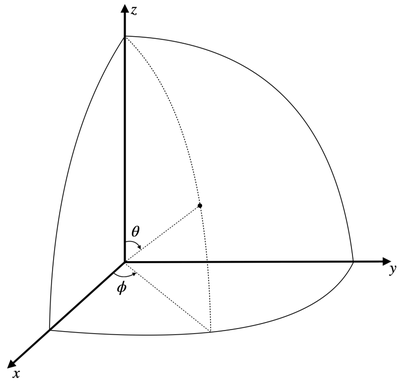
Number of cosmic-ray muons in a interval of time dt,

with momentum between p and p+dp,

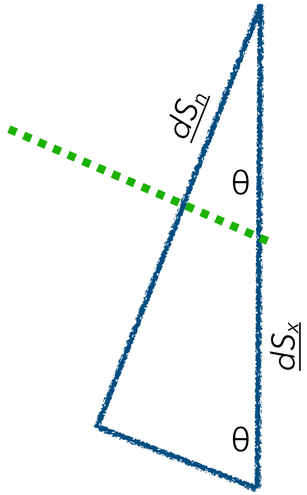
with zenith angle between θ and θ + dθ,

with azimuthal angle between Φ and Φ + dΦ,

crossing a surface ***dS_n perpendicular to the muon direction***



$$J'(t,p,\theta,\phi) = \frac{dN}{dt \cdot dp \cdot d\theta \cdot d\phi \cdot dS_n}$$



1D simplification

… crossing a ***vertical surface dS_x***

$$J'_x(t,p,\theta,\phi) = \frac{dN}{dt \cdot dp \cdot d\theta \cdot d\phi \cdot dS_x}$$

$$dS_n = dS_x \; \hat{x} \cdot \hat{n} = dS_x \sin\theta \cos\phi$$

Cylindrical symmetry => vertical surface dS_x

Cylinder lateral surface => also Φ plays a role

$$J'_x(t,p,\theta,\phi) = J'(t,p,\theta,\phi) \cdot \sin\theta \cdot \cos\phi$$

$$J'_x \equiv \frac{dN}{dt \cdot dp \cdot d\theta \cdot d\phi \cdot dS_x} = \left[1600 \cdot \left(\frac{p}{p_0} + 2.68\right)^{-3.175} \cdot \left(\frac{p}{p_0}\right)^{0.279}\right] \cdot (\cos\theta)^n (\sin\theta)^2 \cos\phi \cdot \frac{1}{\mathrm{m}^2 \cdot \mathrm{s} \cdot \mathrm{sr} \cdot \mathrm{GeV/c}}$$

Half-spherical generation - EcoMug

Differential flux

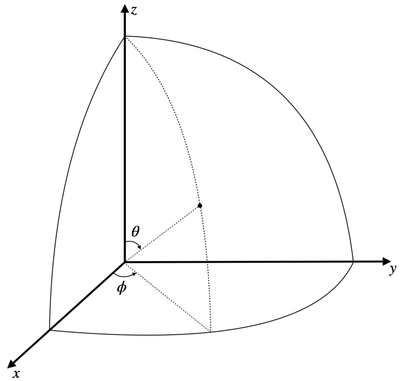
Number of cosmic-ray muons in a interval of time dt,

with momentum between p and p+dp,

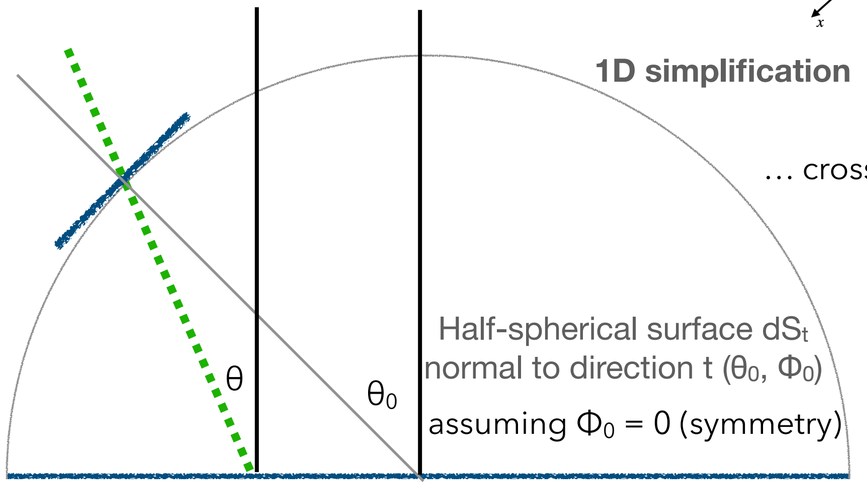
with zenith angle between θ and θ + dθ,

with azimuthal angle between Φ and Φ + dΦ,

crossing a surface ***dS_n perpendicular to the muon direction***



$$J'(t,p,\theta,\phi) = \frac{dN}{dt \cdot dp \cdot d\theta \cdot d\phi \cdot dS_n}$$



… crossing a ***surface dS_t (θ_0, Φ_0)***

$$J'_t(t,p,\theta,\phi) = \frac{dN}{dt \cdot dp \cdot d\theta \cdot d\phi \cdot dS_t}$$

$$J'_t(t,p,\theta,\phi) = J'(t,p,\theta,\phi) \cdot [\sin\theta_0 \sin\theta \cos\phi + \cos\theta_0 \cos\theta]$$

$$dS_n = dS_t \; \hat{t} \cdot \hat{n} = dS_t \, [\sin\theta_0 \sin\theta \cos\phi + \cos\theta_0 \cos\theta]$$

$$J'_t \equiv \frac{dN}{dt \cdot dp \cdot d\theta \cdot d\phi \cdot dS_t} = \left[1600 \cdot \left(\frac{p}{p_0} + 2.68\right)^{-3.175} \cdot \left(\frac{p}{p_0}\right)^{0.279}\right] \cdot (\cos\theta)^n \left[\sin\theta_0 (\sin\theta)^2 \cos\phi + \cos\theta_0 \cos\theta \sin\theta\right] \cdot \frac{1}{\mathrm{m}^2 \cdot \mathrm{s} \cdot \mathrm{sr} \cdot \mathrm{GeV/c}}$$

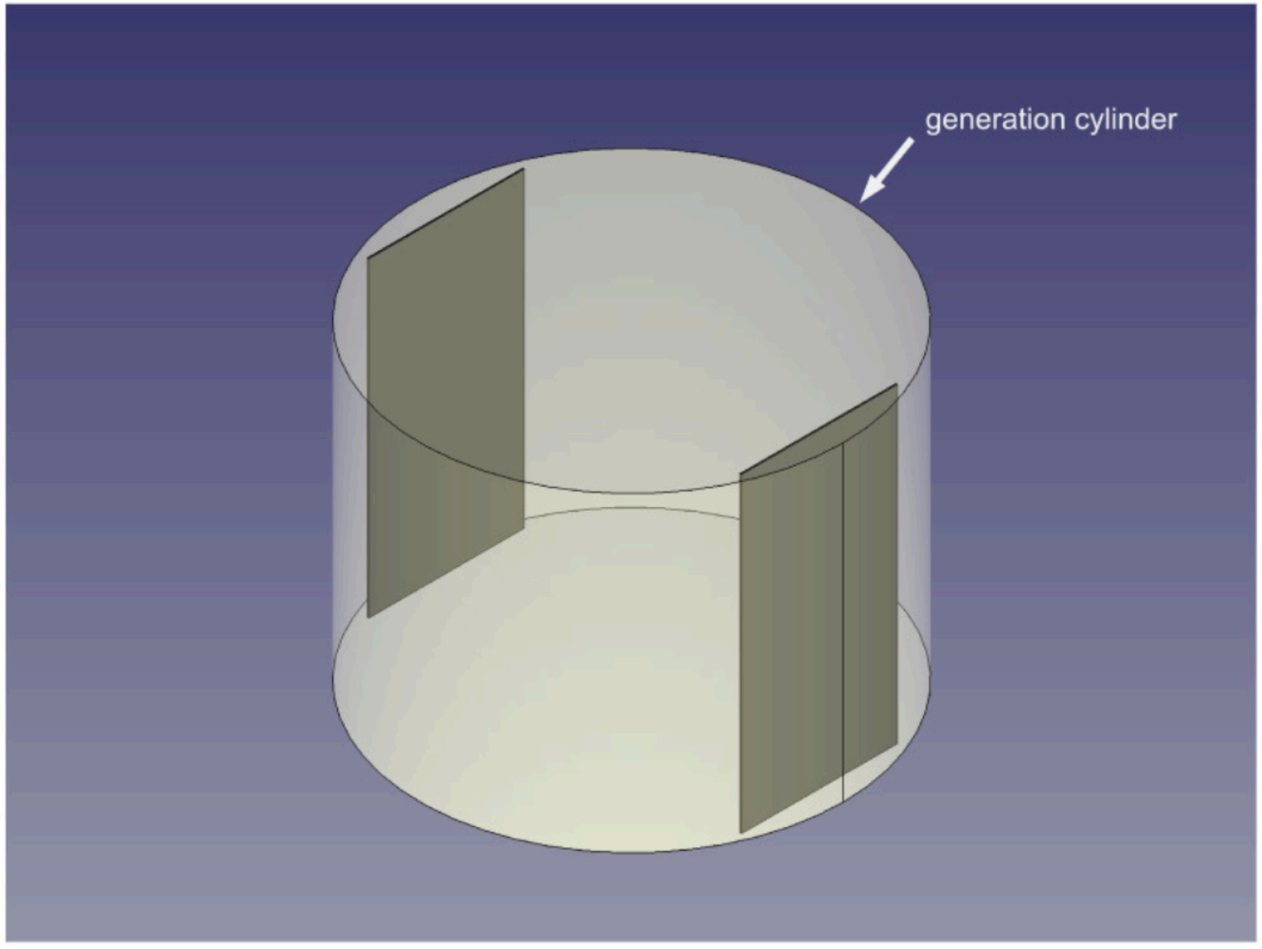
Modes comparison - EcoMug

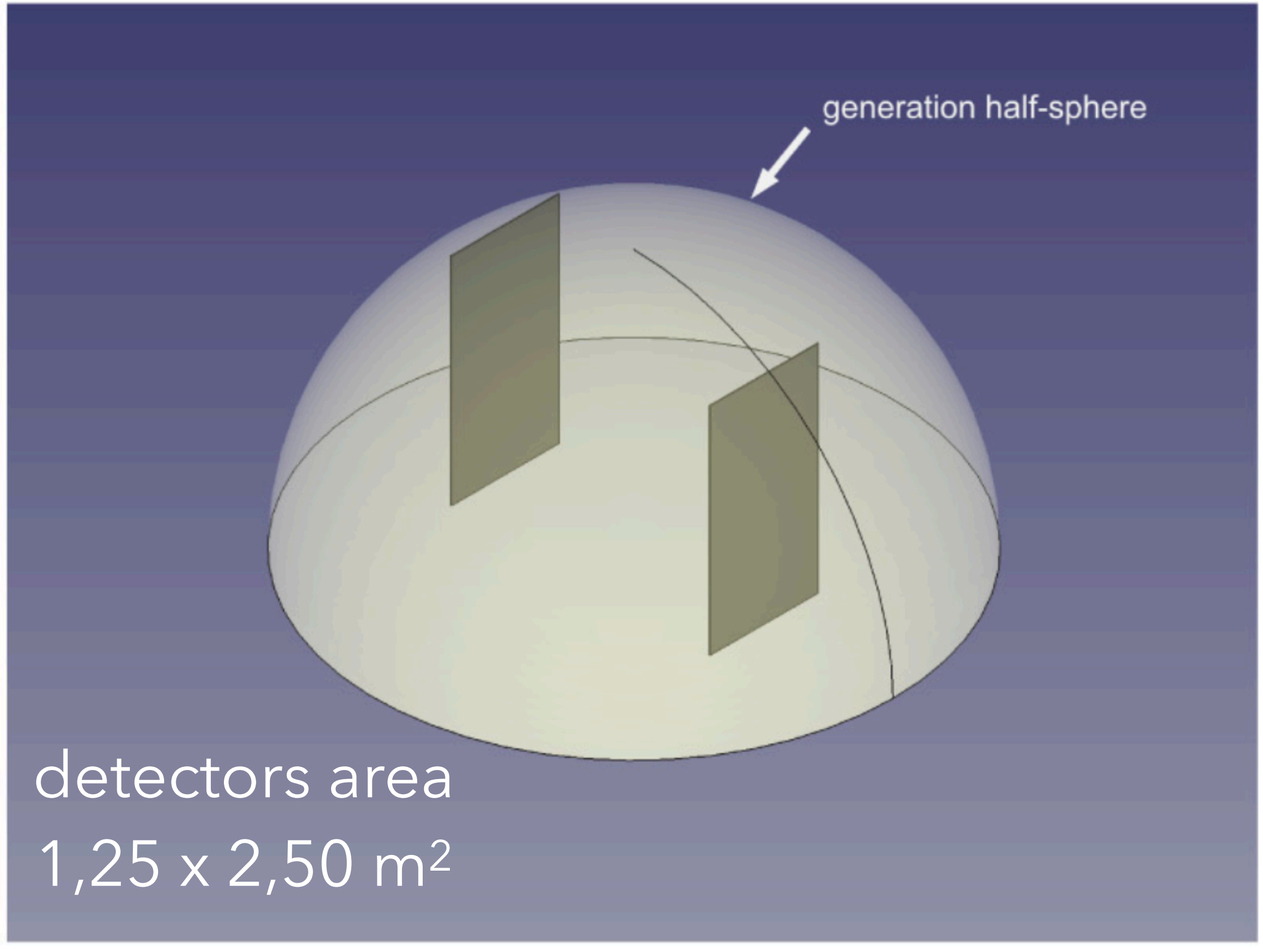
Example 1: “vertical” detectors

- we generate over a cylindrical and a half-spherical surface
- we require the muon to cross both vertical detectors
- we compare the distributions of p, θ and Φ (filtered by the two detectors)

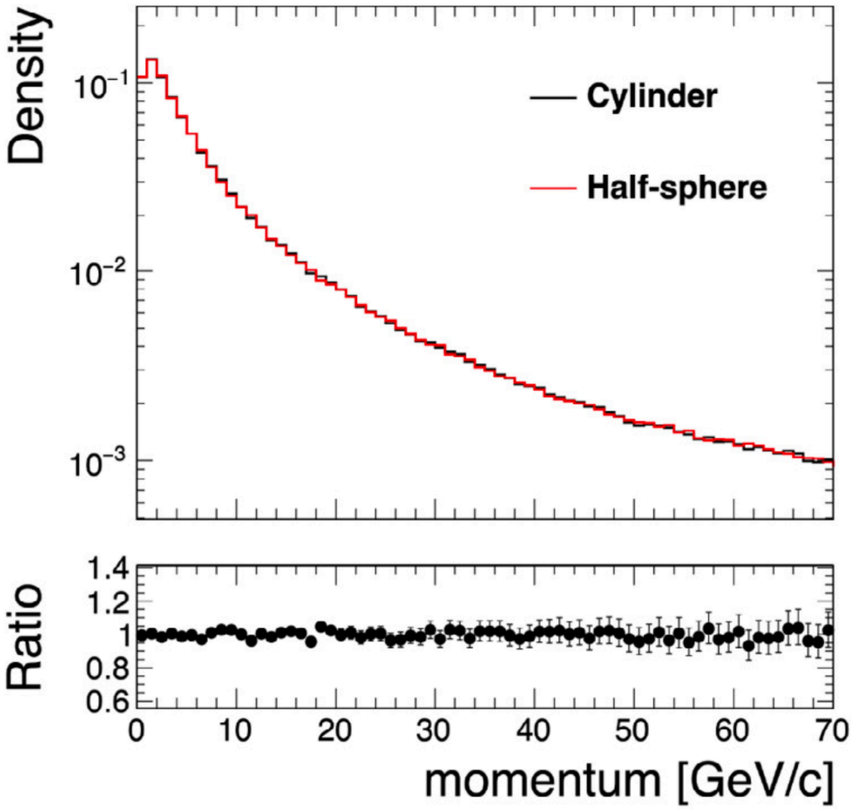
4. Comparison between the different generation methods

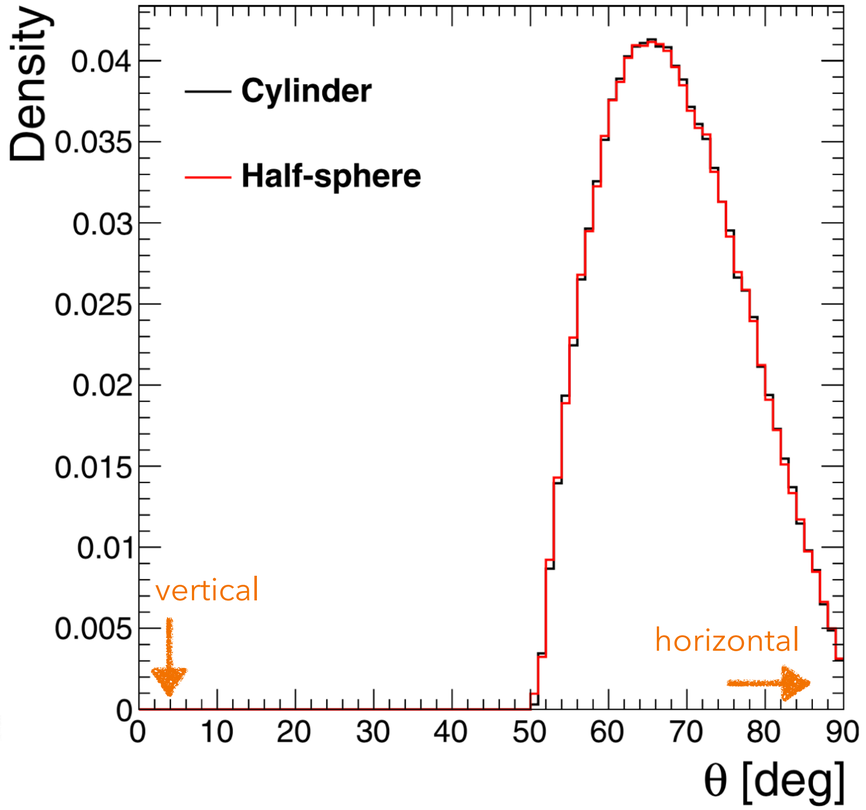
The generation methods discussed above, and implemented in Eco-Mug, are mathematically equivalent, provided that all of them grant the proper coverage of the geometrical acceptance of the detection system. However, depending on the case study, one method could be more effective than the others, in respect to the generation time. In this section, a comparison of the performance of the three generation methods, for three different scenarios, is presented.

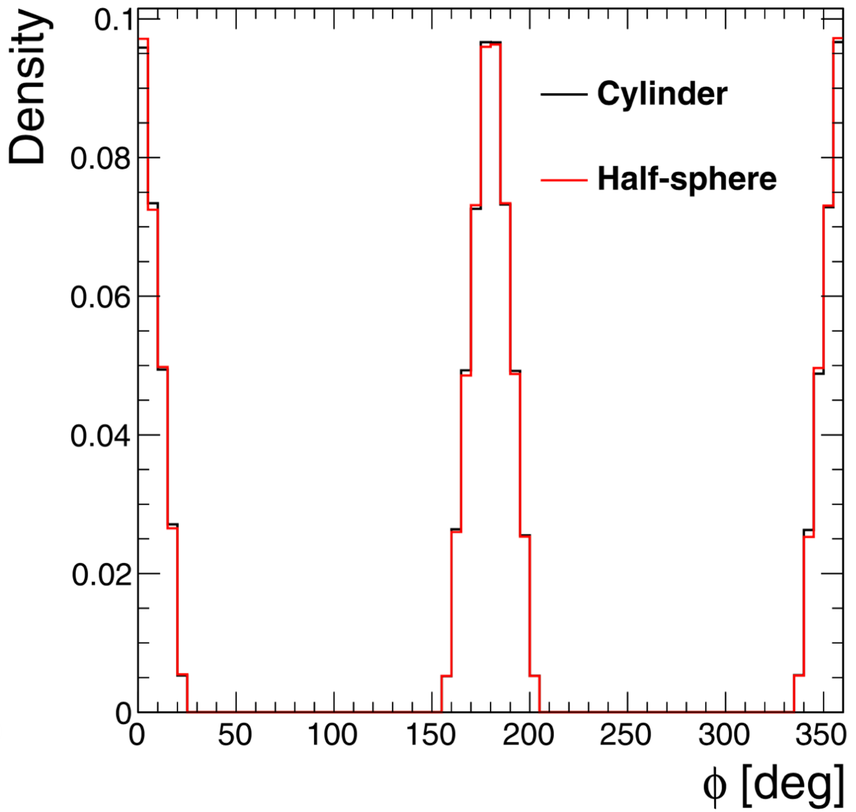




The two generations are equivalent (in terms of “physics”)



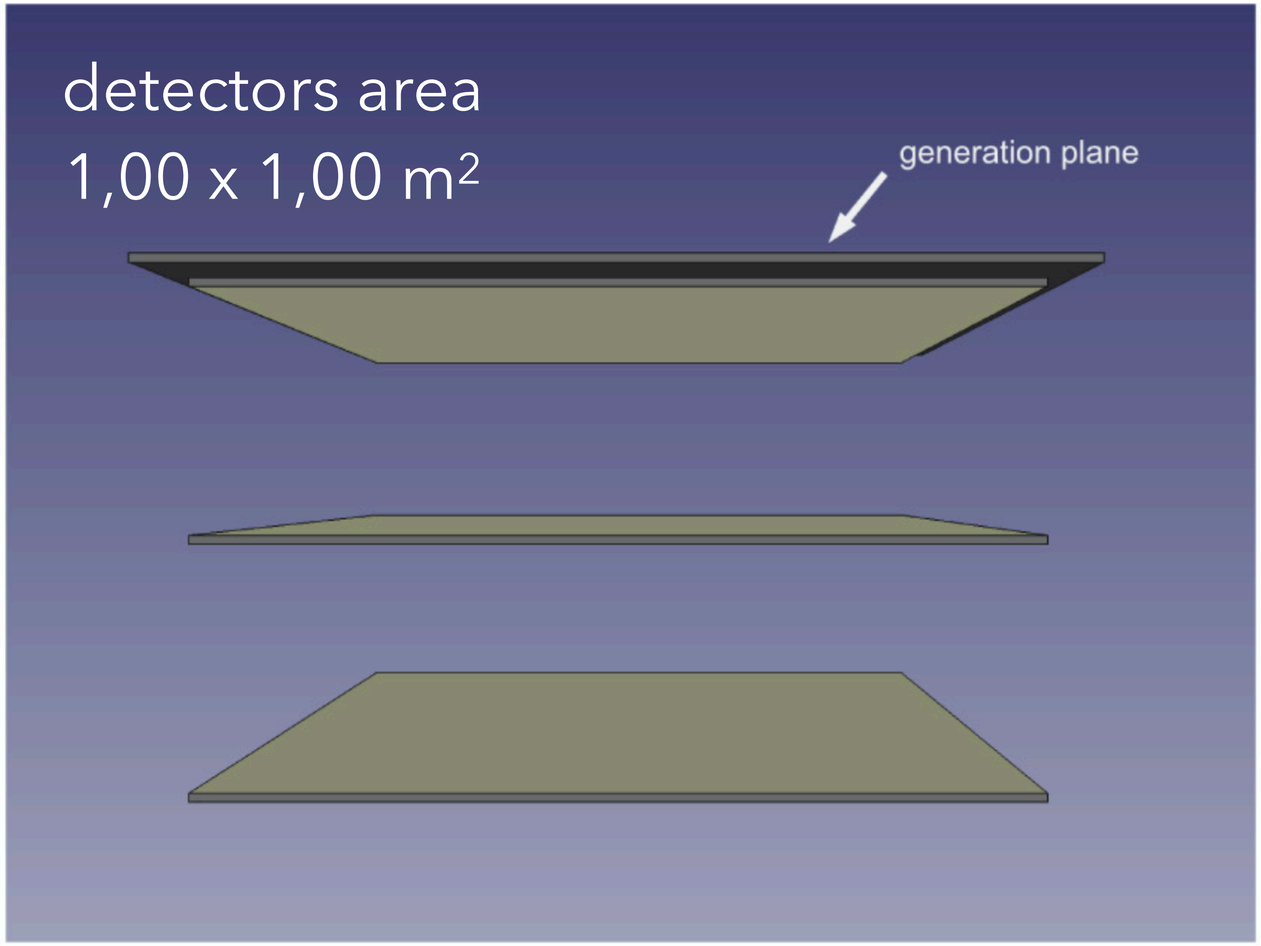


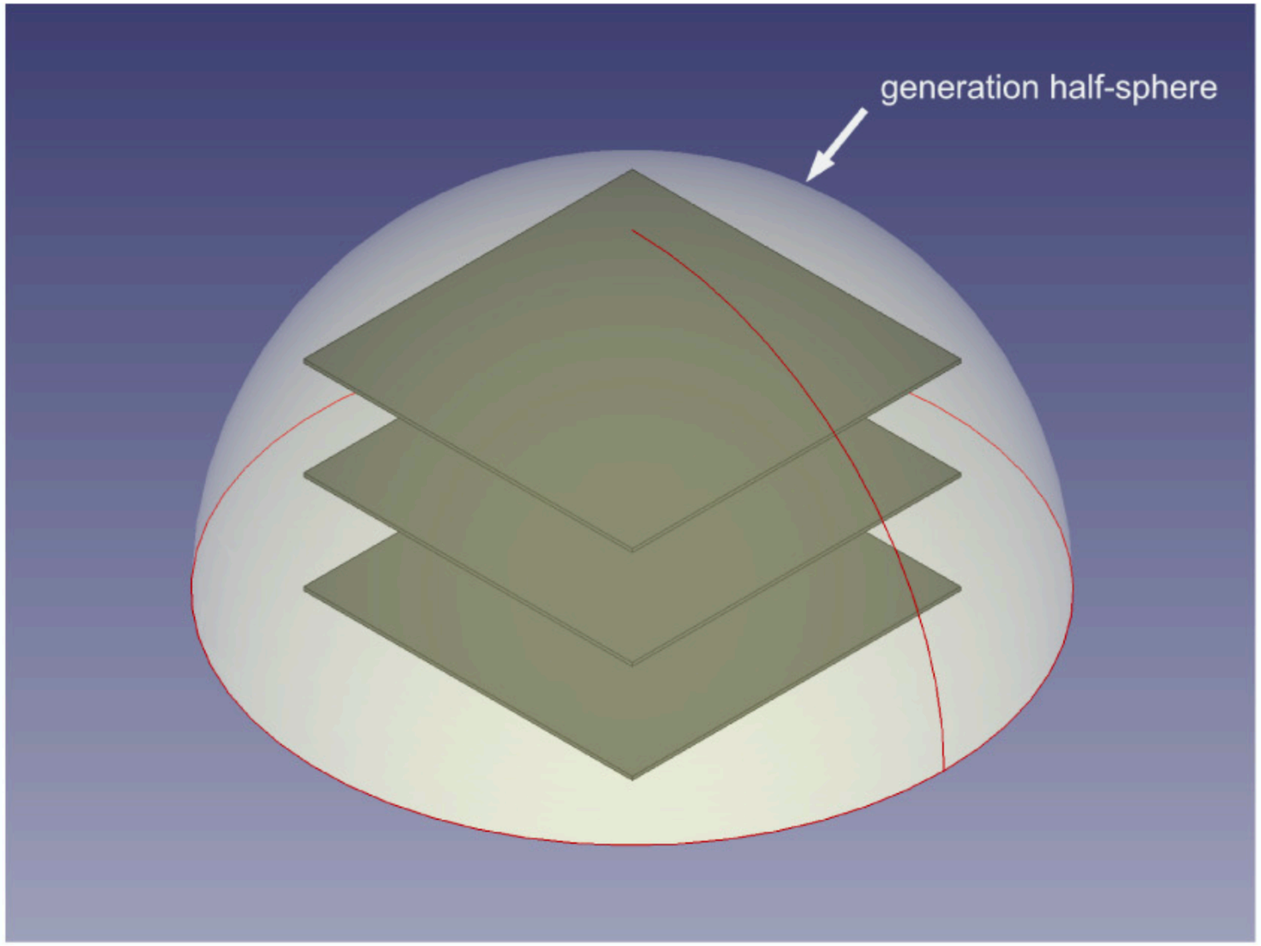


The cylindrical generation is 4/5 times more effective than the half-spherical (in this geometrical configuration)

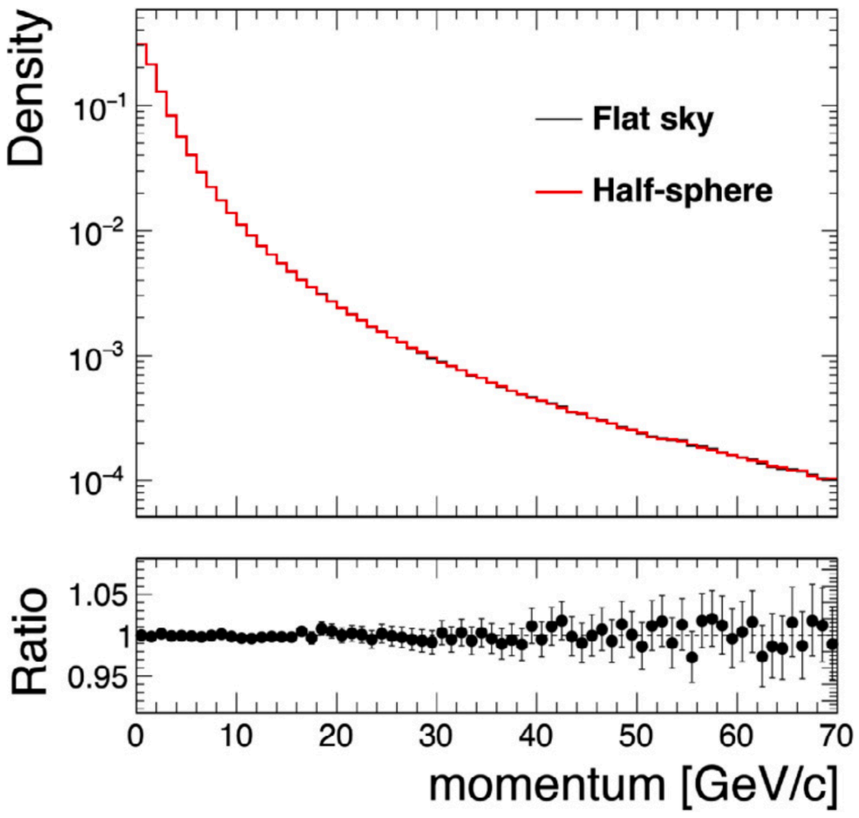
Example 2: a muon telescope

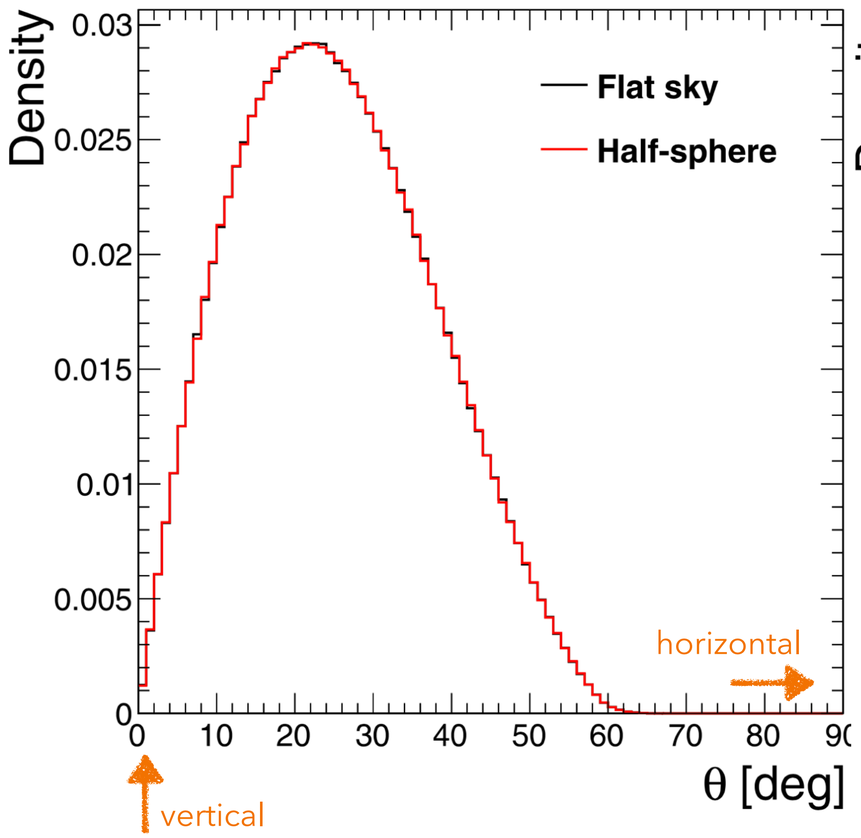
- we generate over a flat sky and a half-spherical surface
- we require the muon to cross all the three horizontal detectors
- we compare the distributions of p, θ and Φ (filtered by three detectors)

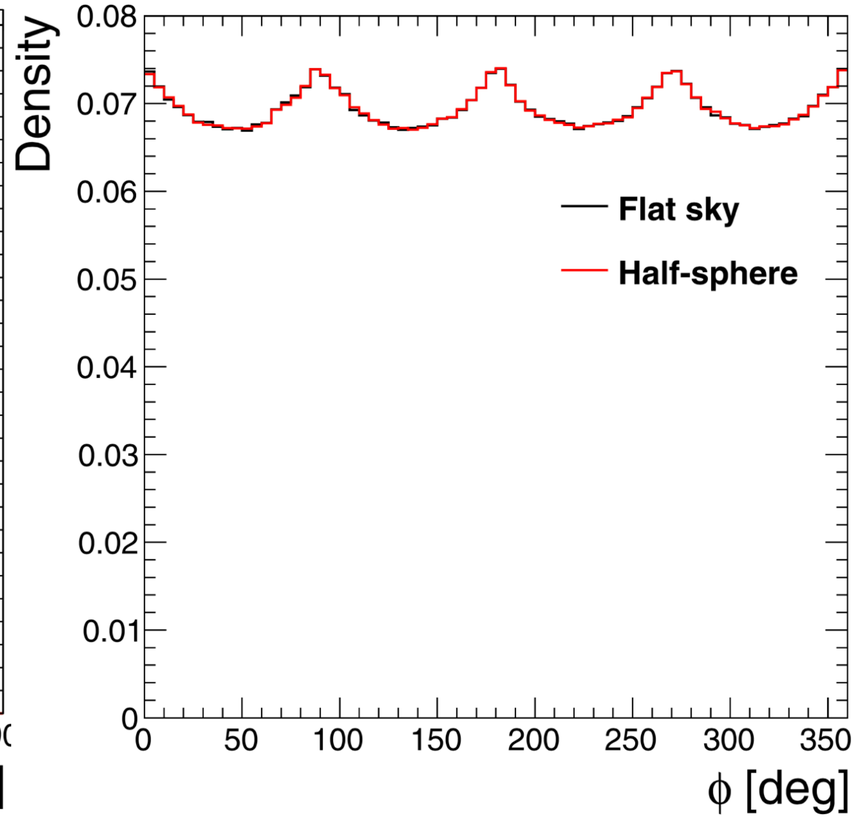




The two generations are equivalent (in terms of “physics”)







The flat sky generation is 4/5 times more effective than the half-spherical (in this geometrical configuration)

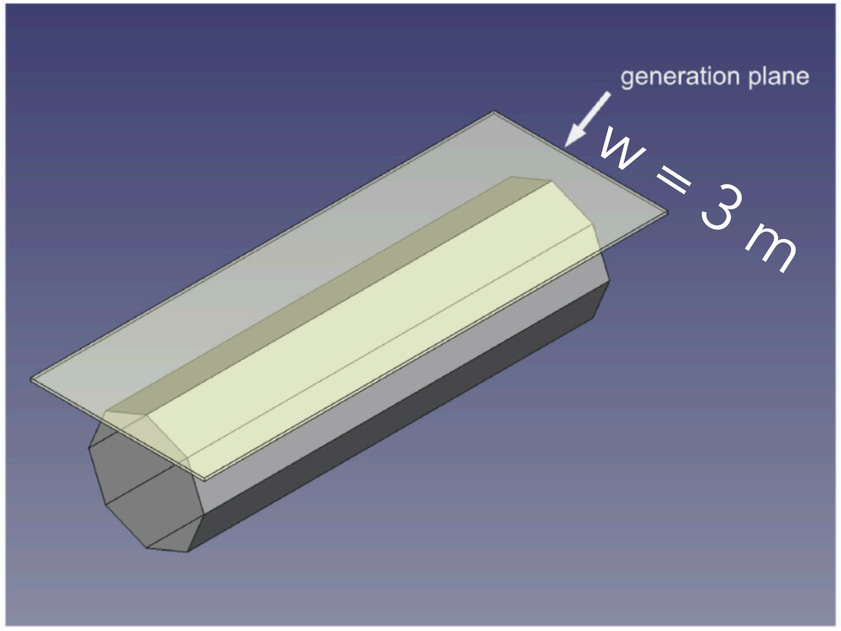
Example 3: a vertex detector

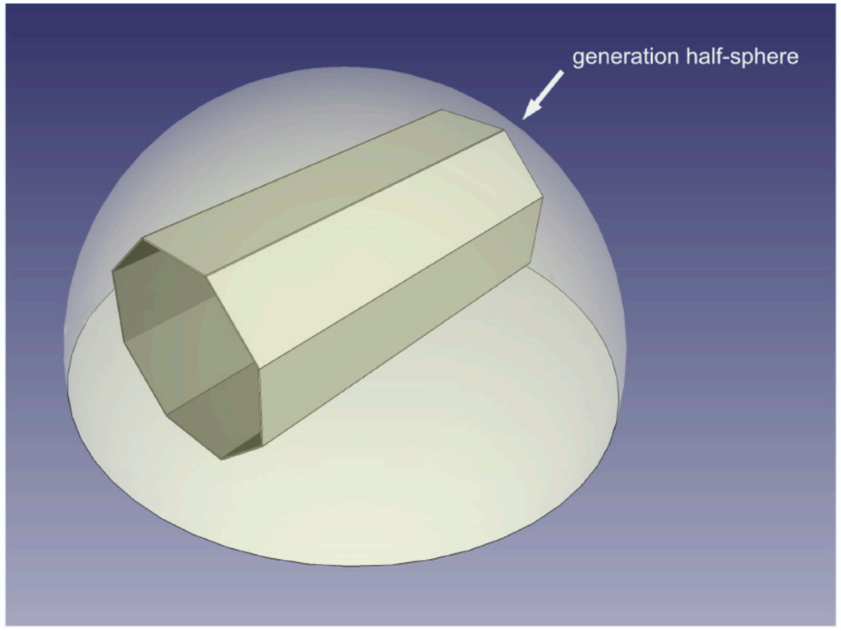
- we generate over a flat sky, a cylindrical and a half-spherical surface
- we require the muon to cross two elements of an octagonal detector
- we compare the distributions of p, θ and Φ (filtered by two elements)

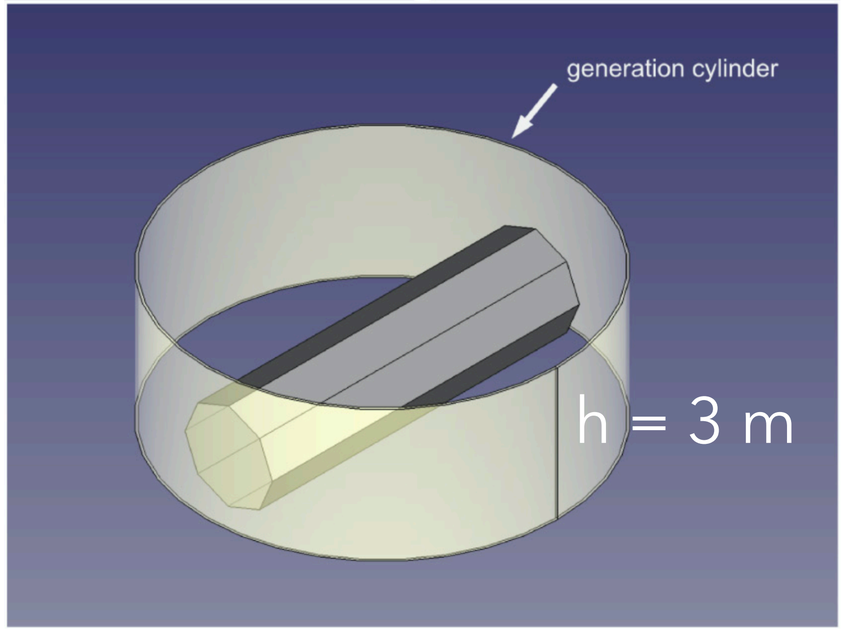
Modes comparison - EcoMug

4. Comparison between the different generation methods

The generation methods discussed above, and implemented in Eco-Mug, are mathematically equivalent, provided that all of them grant the proper coverage of the geometrical acceptance of the detection system. However, depending on the case study, one method could be more effective than the others, in respect to the generation time. In this section, a comparison of the performance of the three generation methods, for three different scenarios, is presented.

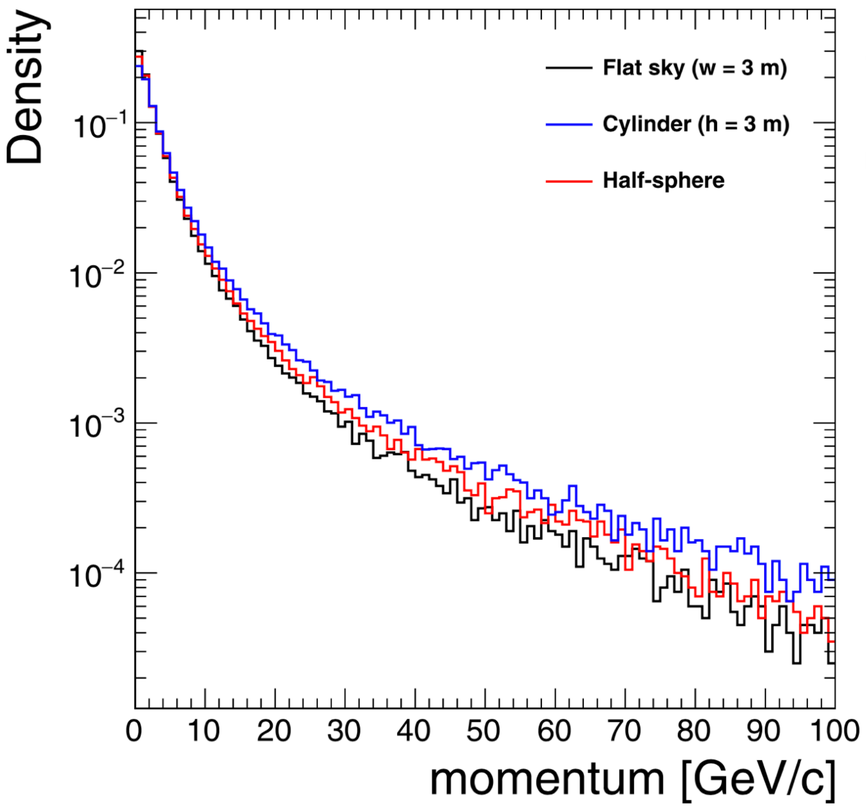


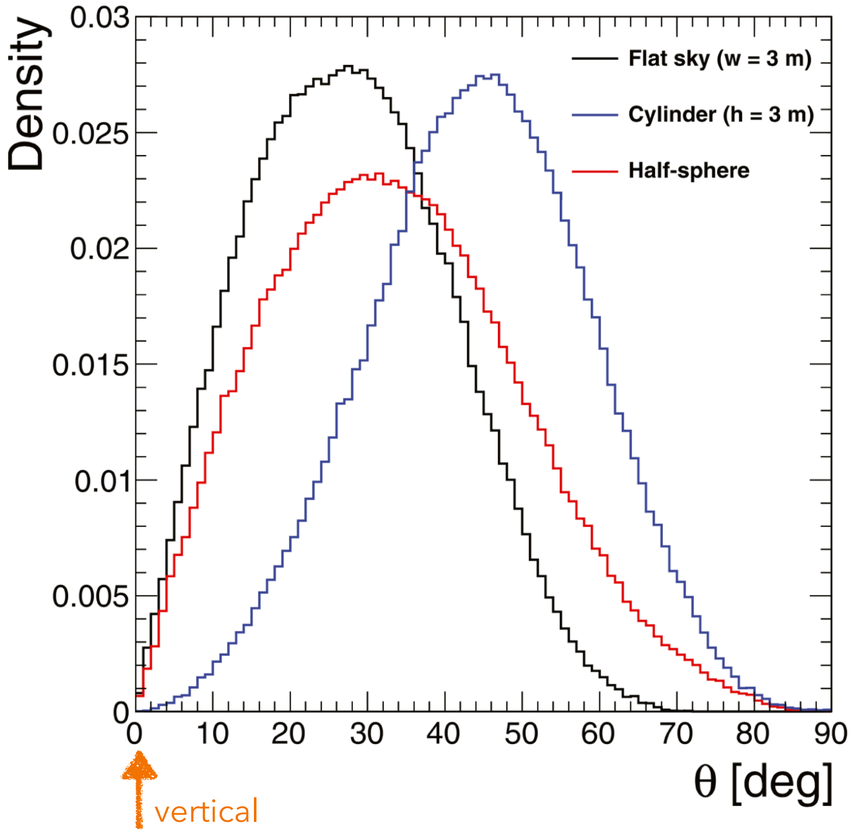


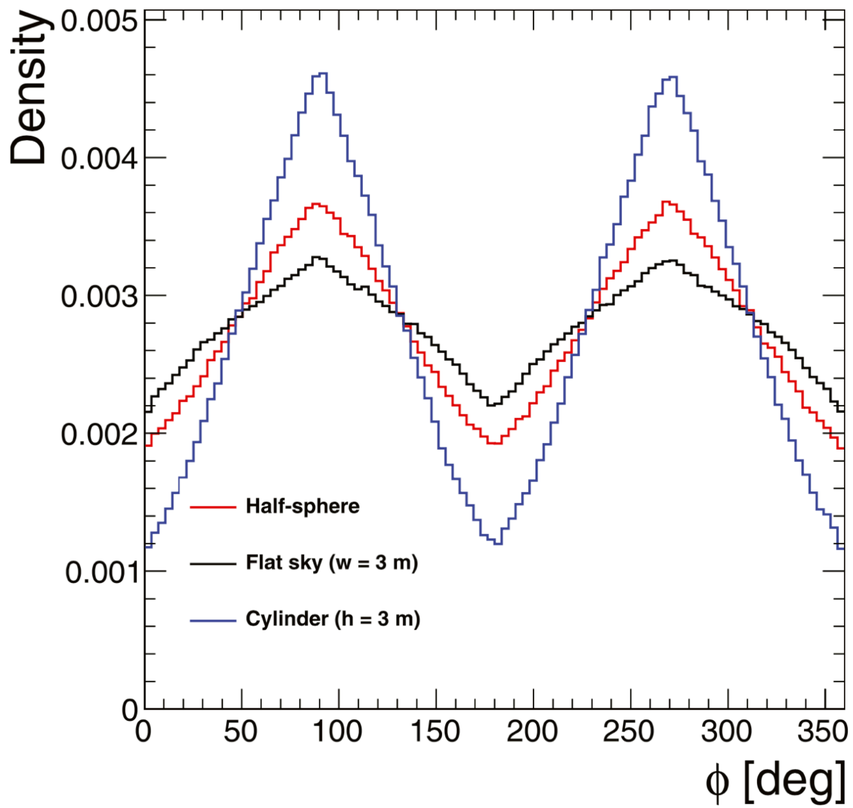


The three generations are not equivalent due do a different geometrical coverage

The half-sphere have full coverage, the cylindrical and the flat sky surfaces have only a partial coverage

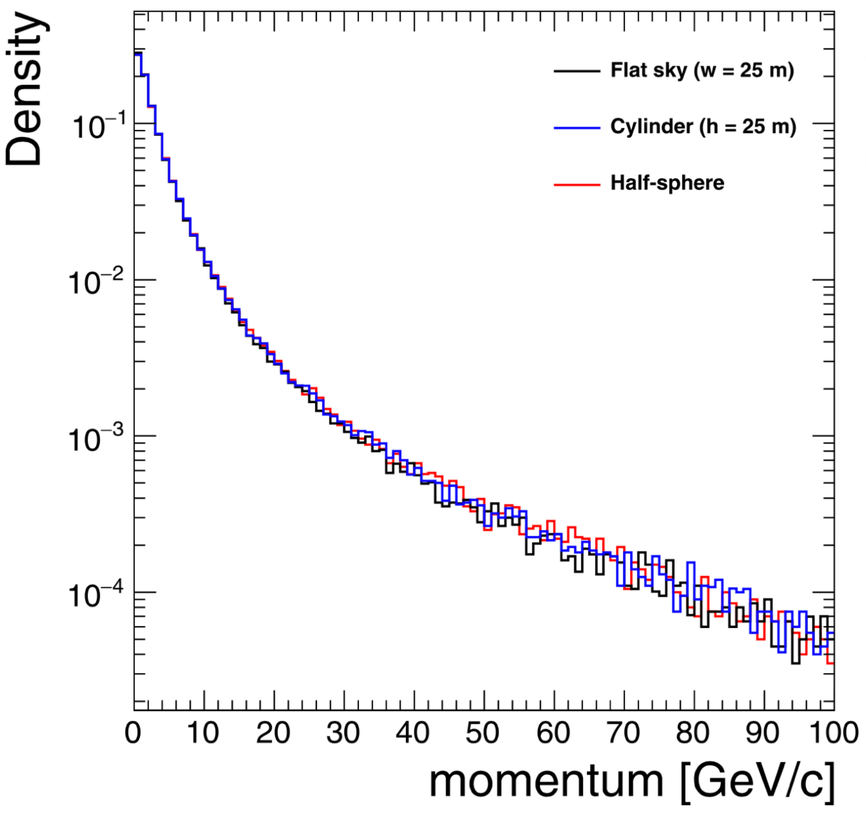


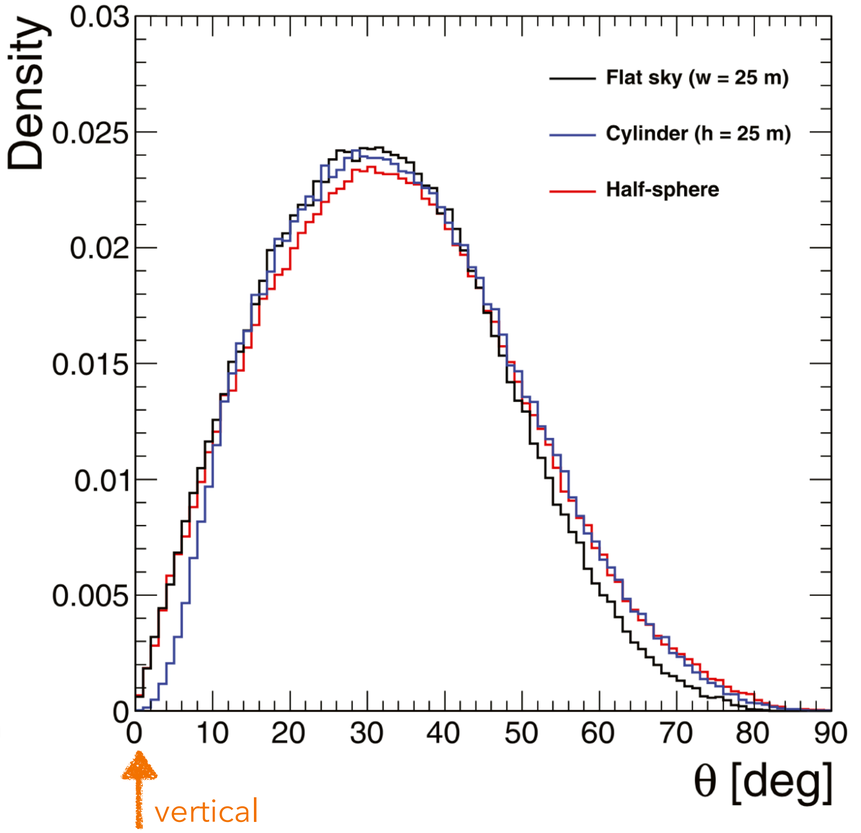


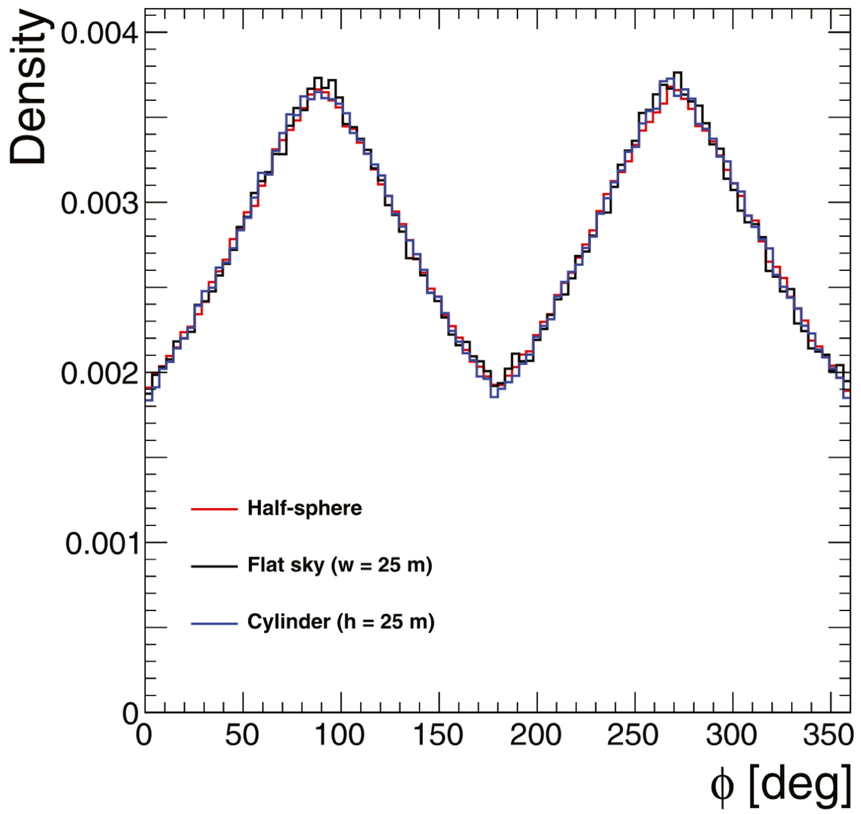


The three generations are equivalent (in terms of “physics”)

Increasing the size of the flat sky and of the cylinder the geometrical coverage increases and the distributions get closer







The half-spherical generation is 3/4 times more effective than the cylindrical/flat sky (in this geometrical configuration)

5.1. Basic usage

The use of the library requires the initialization of the EcoMug class, the choice of the generation method, and the definition of the size and position of the generation surface, as in the example codes 1, 2 and 3.

```
EcoMug gen; // initialization of the class
gen.SetUseSky(); // plane surface generation
gen.SetSkySize({{10., 10.}}); // x and y size of the plane
// (x,y,z) position of the center of the plane
gen.SetSkyCenterPosition({{0., 0., 20.}});
```

Example code 1: EcoMug setup for a flat surface generation.

```
EcoMug gen; // initialization of the class
gen.SetUseCylinder(); // cylindrical surface generation
gen.SetCylinderRadius(10.); // cylinder radius
gen.SetCylinderHeight(30.); // cylinder height
// (x,y,z) position of the center of the cylinder
gen.SetCylinderCenterPosition({{0., 0., 15.}});
```

Example code 2: EcoMug setup for a cylindrical surface generation.

```
EcoMug gen; // initialization of the class
gen.SetUseHSphere(); // half-spherical surface generation
gen.SetHSphereRadius(30.); // half-sphere radius
// (x,y,z) position of the center of the half-sphere
gen.SetHSphereCenterPosition({{0., 0., 0.}});
```

Example code 3: EcoMug setup for a half-spherical surface generation.

```
// Setup of the instance of the EcoMug class
// as in the example code 1
EcoMug gen;
gen.SetUseSky();
gen.SetSkySize({{10., 10.}});
gen.SetSkyCenterPosition({{0., 0., 20.}});

// The array storing muon generation position
std::array<double, 3> muon_position;

// Loop to generate 1000 cosmic-ray muons
for (auto event = 0; event < 1000; ++event) {
  gen.Generate(); // generate a cosmic-ray muons

  // access position, direction and momentum
  // pleas note that GetGenerationPosition()
  // returns a std::array<double, 3>
  muon_position = gen.GetGenerationPosition();
  double muon_p = gen.GetGenerationMomentum();
  double muon_theta = gen.GetGenerationTheta();
  double muon_phi = gen.GetGenerationPhi();
  double muon_charge = gen.GetCharge();

  ... // the code where generated CR muons are used
}
```

Example code 4: Accessing position, direction, momentum and charge of generated cosmicray muons in EcoMug.

In several scenarios, one could be interested in generating tracks from a subset of these parameters, saving space and computation time. EcoMug allows this by exposing to the user the following methods:

- `SetMinimumMomentum` - Set the minimum momentum for generated cosmic-ray muons;
- `SetMaximumMomentum` - Set the maximum momentum for generated cosmic-ray muons;
- `SetMinimumTheta` - Set the minimum zenith angle θ for generated cosmic-ray muons;
- `SetMaximumTheta` - Set the maximum zenith angle θ for generated cosmic-ray muons;
- `SetMinimumPhi` - Set the minimum azimuthal angle ϕ for generated cosmic-ray muons;
- `SetMaximumPhi` - Set the maximum azimuthal angle ϕ for generated cosmic-ray muons.

The full list of methods of the EcoMug class is available at the following link:
https://dr4kan.github.io/EcoMug/class_eco_mug.html.

5.4. *Using a user-defined function for the differential flux*

In those cases where the proposed parametrization of J does not provide an accurate description of the differential flux of CR muons, EcoMug gives the possibility to use a custom function for J, as shown in the example code 7.

```
double J(double p, double theta) {
  double A = 0.14*pow(p, -2.7);
  double B = 1. / (1. + 1.1*p*cos(theta)/115.);
  double C = 0.054 / (1. + 1.1*p*cos(theta)/850.);
  return A*(B+C);
}

EcoMug gen;
gen.SetUseSky();
gen.SetSkySize({{x, y}});
gen.SetSkyCenterPosition({0., 0., z});
gen.SetMinimumMomentum(150);
gen.SetDifferentialFlux(&J);

for (auto event = 0; event < nevents; ++event) {
  gen.GenerateFromCustomJ(); // generate from user-defined J
  ... // retrieve and use muon data
  gen.Generate(); // generate from J as in equation 2
  ... // retrieve and use muon data
}
```

Example code 7: Filtering the generation in EcoMug.

In the previous code, the same instance of the class `EcoMug` is used to generate according to a Gaisser-like parametrization [41] and to the default one (Eq. (2)). The use of a custom definition for J requires a function of both momentum and θ to be passed to the generator by means of the method `SetDifferentialFlux`. Afterwards, the user can invoke the generation of a CR muon, according to the specified J, with the method `GenerateFromCustomJ`.

EXTRAS - EcoMug

Appendix. Example of integration with GEANT4

EcoMug can be easily integrated with GEANT4, by modifying the (mandatory) user action class derived from `G4VUserPrimaryGeneratorAction`.

The relevant lines of codes to be included in the header file are reported below.

```
#include "EcoMug.h"
...
class PrimaryGeneratorAction :
      public G4VUserPrimaryGeneratorAction {
  public:
  ...
  G4ParticleGun *fParticleGun;
  G4ParticleDefinition *mu_plus, *mu_minus;
  EcoMug fMuonGen;
};
```

The implementation file initializes the class `EcoMug` and interfaces it to `G4ParticleGun`, as shown in the code below.

```
PrimaryGeneratorAction::PrimaryGeneratorAction() :
G4VUserPrimaryGeneratorAction(),
fParticleGun(0), mu_plus(0), mu_minus(0) {
  fMuonGen.SetUseCylinder();
  fMuonGen.SetCylinderRadius(2500*mm);
  fMuonGen.SetCylinderHeight(4170*mm);

  fParticleGun  = new G4ParticleGun(1);
  mu_minus  = G4ParticleTable::GetParticleTable()
              ->FindParticle("mu-");
  mu_plus  = G4ParticleTable::GetParticleTable()
             ->FindParticle("mu+");
}
...
void PrimaryGeneratorAction::GeneratePrimaries(G4Event* ev) {
  fMuonGen.Generate();
  array<double, 3> muon_pos = fMuonGen.GetGenerationPosition();
  double muon_ptot = fMuonGen.GetGenerationMomentum();
  double muon_theta = fMuonGen.GetGenerationTheta();
  double muon_phi = fMuonGen.GetGenerationPhi();
  fParticleGun->SetParticlePosition(G4ThreeVector(
    muon_pos[0]*mm,
    muon_pos[1]*mm,
    muon_pos[2]*mm
  ));

  fParticleGun->SetParticleMomentum(G4ParticleMomentum(
    muon_ptot*sin(muon_theta)*cos(muon_phi)*GeV,
    muon_ptot*sin(muon_theta)*sin(muon_phi)*GeV,
    muon_ptot*cos(muon_theta)*GeV
  ));

  // charge
  if (fMuonGen.GetCharge() < 0) {
    fParticleGun->SetParticleDefinition(mu_minus);
  } else {
    fParticleGun->SetParticleDefinition(mu_plus);
  }
  fParticleGun->GeneratePrimaryVertex(ev);
}
```

Conclusions - EcoMug

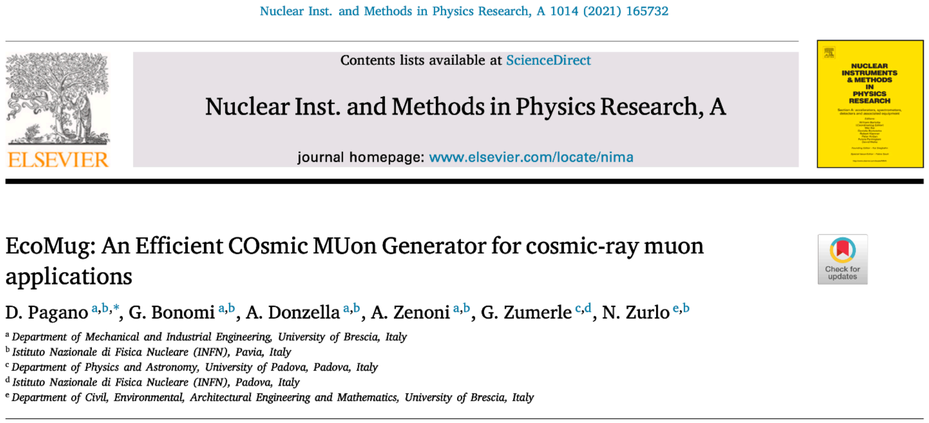
A new Monte Carlo generator of cosmic-ray muons, called EcoMug, has been presented:

- EcoMug gives the possibility of generating from different surfaces (plane, cylinder and half-sphere)
- it keeps the correct angular and momentum distribution of generated tracks
- it is based on a real data parametrization (*)
- it offers the possibility of restricting the angular variables momentum of muons at generation level
- it has a fast and optimized code

EcoMug is a header-only C++11 library (single .h file**)**

and it is freely available under the GPL-3.0 license at

https://github.com/dr4kan/EcoMug



Nuclear Inst. and Methods in Physics Research, A 1014 (2021) 165732

Contents lists available at ScienceDirect

Nuclear Inst. and Methods in Physics Research, A

journal homepage: www.elsevier.com/locate/nima

EcoMug: An Efficient COsmic MUon Generator for cosmic-ray muon applications

D. Pagano [a,b,*], G. Bonomi [a,b], A. Donzella [a,b], A. Zenoni [a,b], G. Zumerle [c,d], N. Zurlo [e,b]

[a] Department of Mechanical and Industrial Engineering, University of Brescia, Italy
[b] Istituto Nazionale di Fisica Nucleare (INFN), Pavia, Italy
[c] Department of Physics and Astronomy, University of Padova, Padova, Italy
[d] Istituto Nazionale di Fisica Nucleare (INFN), Padova, Italy
[e] Department of Civil, Environmental, Architectural Engineering and Mathematics, University of Brescia, Italy

(*) For those cases where the proposed parametrization does not provide an accurate description of the differential flux of muons, a custom definition can be used