technische universität dortmund

experimentelle physik 5 astroteilchenphysik

Particle propagation for CORSIKA in PROPOSAL

Speaker: Jean-Marco Alameddine

19.06.2019

Technische Universität Dortmund

Introduction

- **PROPOSAL**: Tool to propagate particles through media
  - $\rightarrow$ MC simulations, multivariate statistics
- **Requirements:** Accuracy, performance
- **Processes:** Energy losses, scattering, decays
- C++ library with Python bindings
- GitHub, UnitTests (Travis CI), ...

https://github.com/tudo-astroparticlephysics/PROPOSAL

jean-marco.alameddine@udo.edu

MMC — JAVA

PROPOSAL v1 — C++

PROPOSAL v2 — modern C++ , Polymorphism, ...

PROPOSAL code structure

```
Propagator(const ParticleDef&,
           const std::vector<Sector::Definition>&,
           const Geometry&,
           const InterpolationDef&)
```

- `Propagator` as base class to propagate a particle
- Objects owns all information necessary for propagation

```
Propagator(const ParticleDef&,
           const std::vector<Sector::Definition>&,
           const Geometry&,
           const InterpolationDef&)
```

- `ParticleDef` includes static information about particle

→ Wide range of predefined particles available

→ Simple creation of additional particles:

```
ParticleDef new_particle = ParticleDef::Builder().SetMass(1000).build();
```

```
Propagator(const ParticleDef&,
           const std::vector<Sector::Definition>&,
           const Geometry&,
           const InterpolationDef&)
```

- List of `Sector::Defintion` objects

→ *Chain of resposibility*: Propagation of our particle through several sectors

→ Each `Sector` object is responsible for the propagation within its borders

Parameter	Description
`Medium`	Medium of the sector
`EnergyCutsSettings`	Stores e_{cut} and v_{cut}
`Geometry`	Geometry of the sector
`stopping_decay`	Whether to force a final decay of the particle if its energy is $\leq e_{\text{low}}$
`cont_rand`	Whether to use continuous randomization
`exact_time`	Whether to calculate the time exactly out of the tracking integral or to use $v = c$ as an approximation
`scattering_model`	Choice of the multiple scattering model
`particle_location`	Location of the particle
`utility_def`	Definition of cross section parameters

`Sector::Defintion` properties, adapted from arXiv:1809.07740

```
Propagator(const ParticleDef&,
           const std::vector<Sector::Definition>&,
           const Geometry&,
           const InterpolationDef&)
```

- Geometry describing the detector
- Different options for particle infront / inside / behind the detector volume

```
Propagator(const ParticleDef&,
           const std::vector<Sector::Definition>&,
           const Geometry&,
           const InterpolationDef&)
```

- `InterpolationDef` as an optional parameter

→ When used, calculated crosssections (and several derived values) are saved in interpolation tables

→ Error of interpolation compared to integration: $\leq 10^{-5}$

→ Performance increased by several orders of magnitude

```
Propagator::Propagator(const ParticleDef& particle_def,
                       const std::string& config_file)
```

- Simple usage of a configuration (json) file is possible

```
{
  "global":
  {
    "seed" : 1,
    "continous_loss_output" : false,
    "only_loss_inside_detector" : false,

    "interpolation":
    {
      "do_interpolation" : true,
      "path_to_tables" : ["resources/tables"],
    },

    "exact_time" : true,
    "stopping_decay" : true,
    "scattering" : "Highland",

    "brems" : "BremsAndreevBezrukovBugaev",
    "photo" : "PhotoButkevichMikhailov",
    "lpm" : false,
    "photo_hard_component" : true,
    "photo_shadow" : "ShadowButkevichMikhailov",
  },

  "sectors": [
    {
      "hierarchy": 0,
      "medium": "ice",
      "density_correction": 1,

      "geometry":
      {
        "shape": "sphere",
        "origin": [0, 0, 0],
        "outer_radius": 6374134000000,
        "inner_radius": 0
      },
      ...
```

Propagation algorithm

```
std::vector<DynamicData*> Propagator::Propagate(double MaxDistance_cm)
```

Choose next `Sector`

Propagate in chosen `Sector`
until `MaxDistance` reached
or sector border reached

Particle did not reach `MaxDistance`, did not decay, energy not below e_{low}

```
double Sector::Propagate(double distance)
```

Remember: Differentiate between continuous losses and stochastic losses !

$v < v_{\text{cut}}$
continuous losses

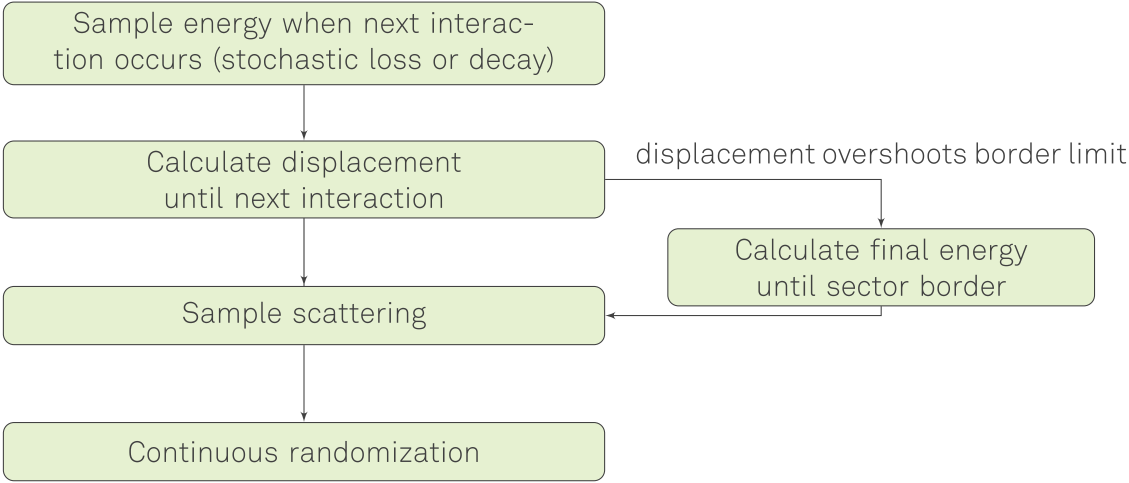
$v > v_{\text{cut}}$
stochastic losses

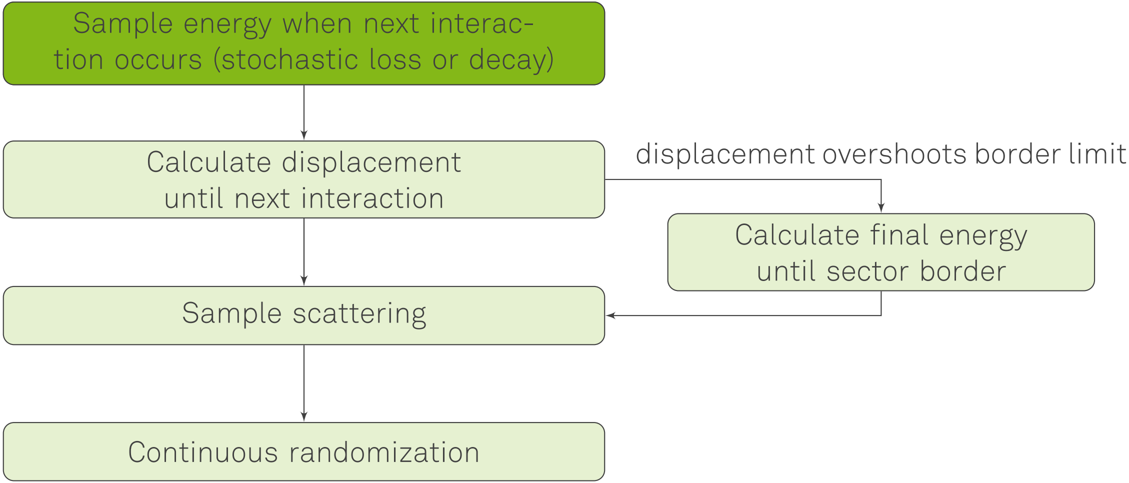
$$\text{with } v_{\text{cut}} = \min\left[{}^{e_{\text{cut}}}/E, {v'}_{\text{cut}}\right]$$

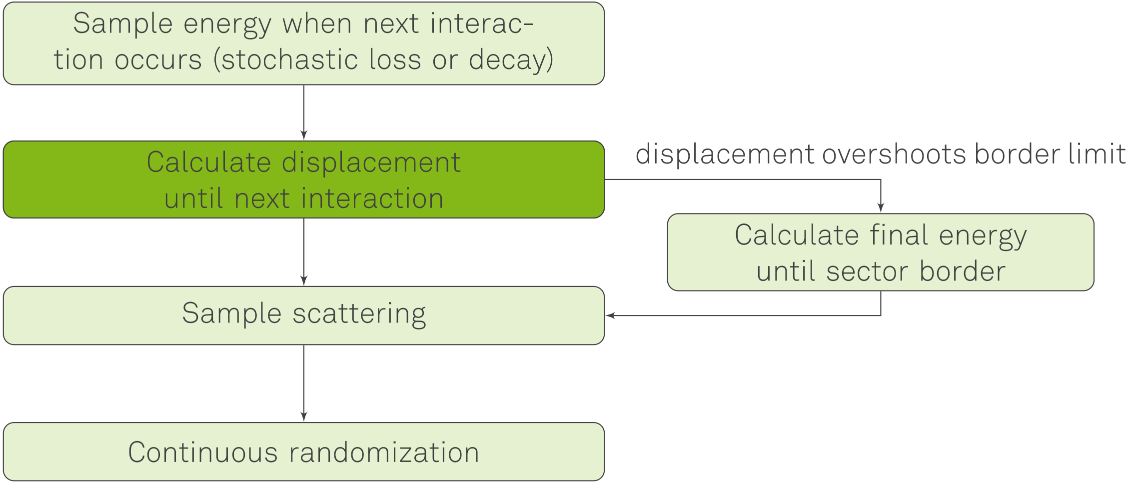
```cpp
double Sector::Propagate(double distance)
```

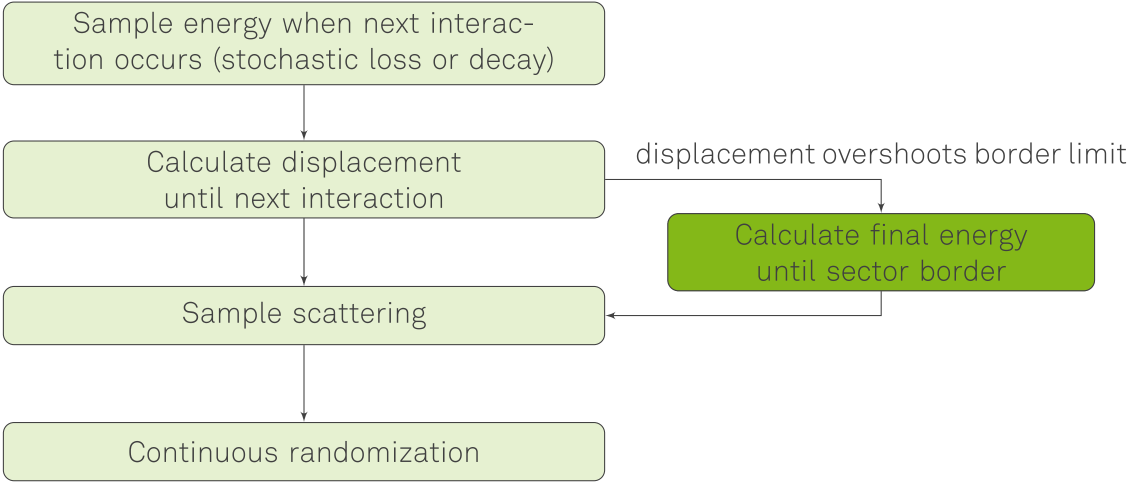
Remember: Differentiate between continuous losses and stochastic losses !

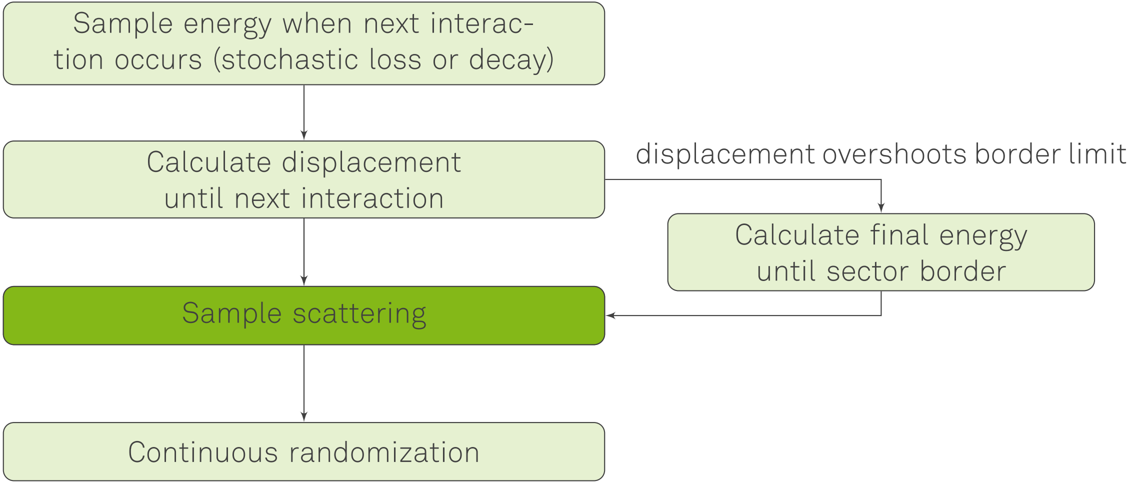
$$\dashrightarrow \overset{\text{stoch.}}{\ast} \xrightarrow[\text{continuous}]{\qquad\qquad} \overset{\text{stoch.}}{\ast} \xrightarrow[\text{continuous}]{\qquad\qquad} \overset{\text{stoch.}}{\ast} \dashrightarrow$$

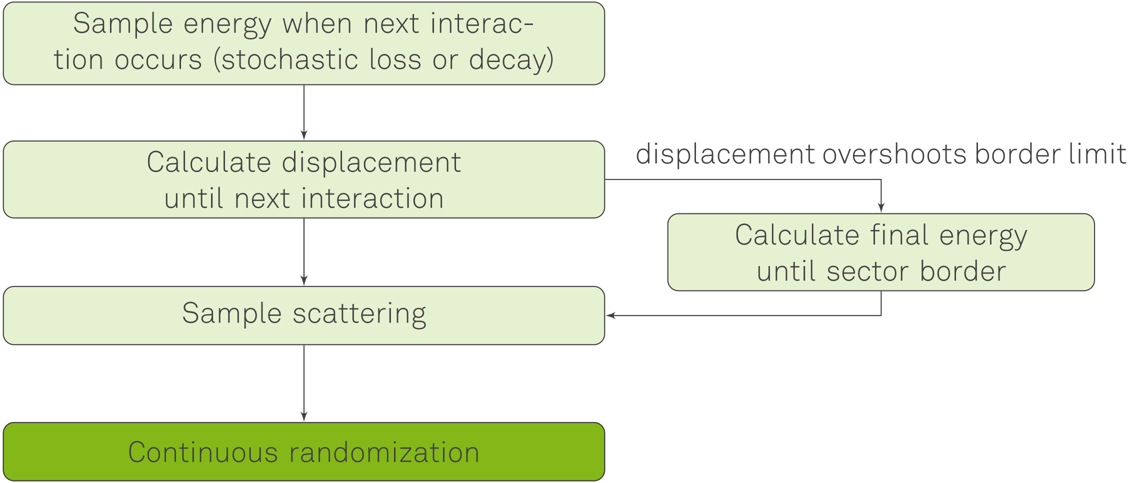

Sample energy when next interaction occurs (stochastic loss or decay)
Calculate displacement until next interaction
displacement overshoots border limit
Calculate final energy until sector border
Sample scattering
Continuous randomization

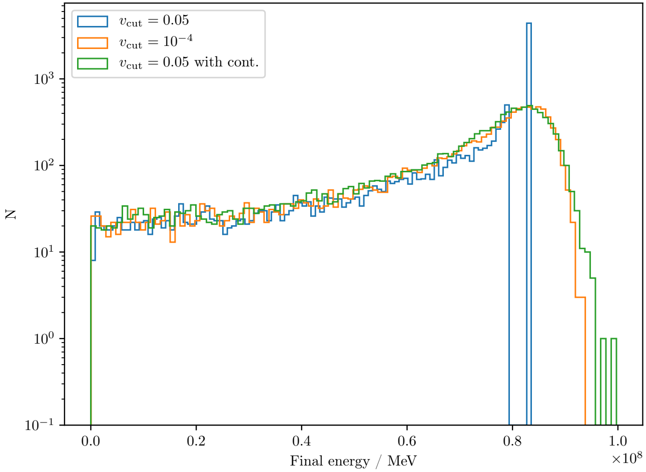

Sample energy when next interaction occurs (stochastic loss or decay)
Calculate displacement until next interaction
displacement overshoots border limit
Calculate final energy until sector border
Sample scattering
Continuous randomization


Sample energy when next interaction occurs (stochastic loss or decay)
Calculate displacement until next interaction
displacement overshoots border limit
Calculate final energy until sector border
Sample scattering
Continuous randomization

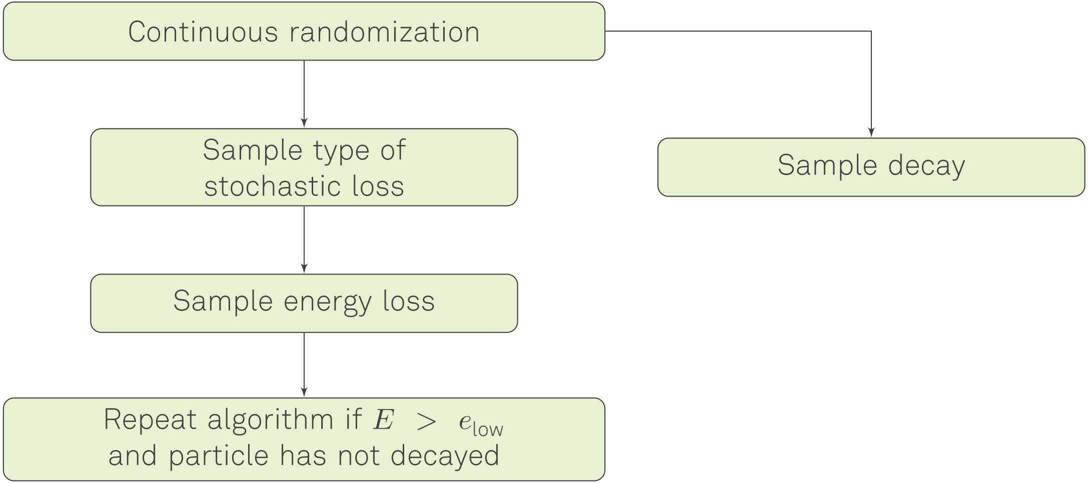

Sample energy when next interaction occurs (stochastic loss or decay)
Calculate displacement until next interaction
displacement overshoots border limit
Calculate final energy until sector border
Sample scattering
Continuous randomization


Sample energy when next interaction occurs (stochastic loss or decay)
Calculate displacement until next interaction
displacement overshoots border limit
Calculate final energy until sector border
Sample scattering
Continuous randomization


Sample energy when next interaction occurs (stochastic loss or decay)
Calculate displacement until next interaction
displacement overshoots border limit
Calculate final energy until sector border
Sample scattering
Continuous randomization



Propagation of 10^4 muons with energy 10^8 MeV through 300 m of standard rock.


Continuous randomization
Sample type of stochastic loss
Sample decay
Sample energy loss
Repeat algorithm if $E > e_{\text{low}}$ and particle has not decayed

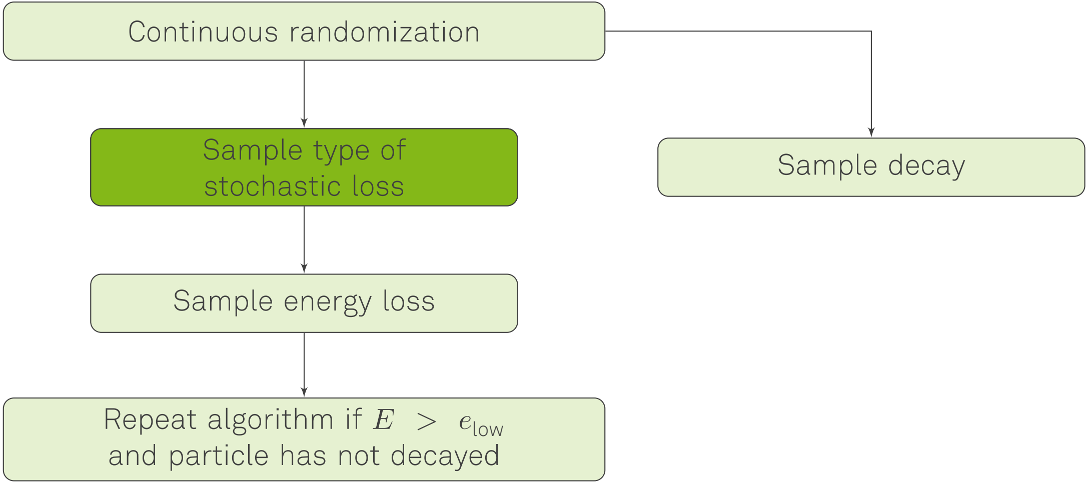
Continuous randomization

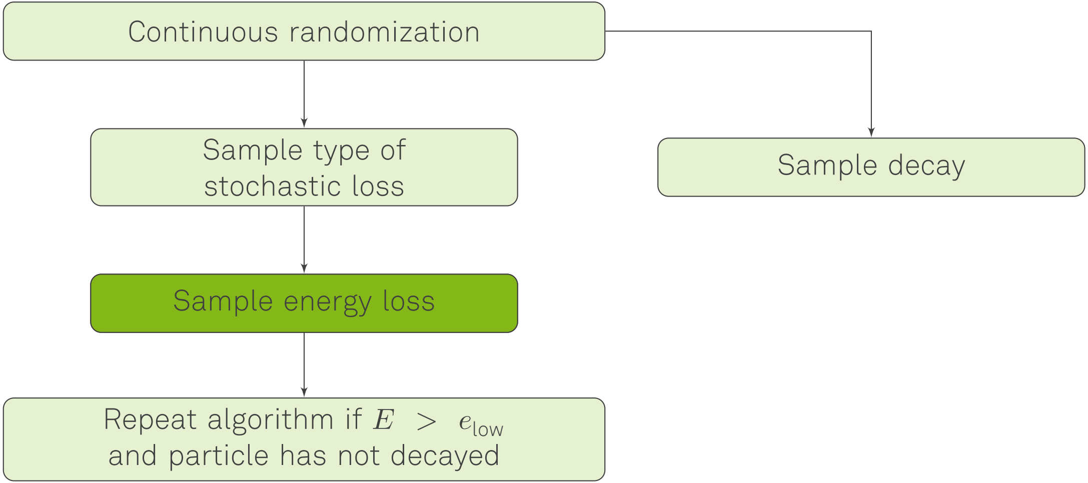
Sample type of stochastic loss

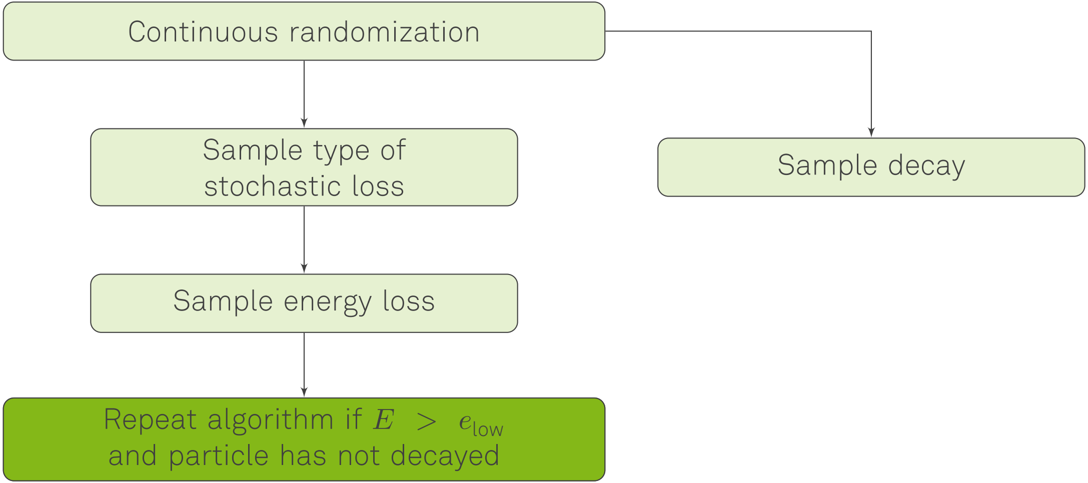
Sample energy loss

Repeat algorithm if $E > e_{\text{low}}$ and particle has not decayed

Sample decay


Continuous randomization
Sample type of stochastic loss
Sample energy loss
Repeat algorithm if $E > e_{\text{low}}$ and particle has not decayed
Sample decay


Continuous randomization
Sample type of stochastic loss
Sample decay
Sample energy loss
Repeat algorithm if $E > e_{\text{low}}$ and particle has not decayed


Continuous randomization
Sample type of stochastic loss
Sample energy loss
Repeat algorithm if $E > e_{\text{low}}$ and particle has not decayed
Sample decay

```
std::vector<DynamicData*> Propagator::Propagate(double MaxDistance_cm);
```

Return value: List of `DynamicData` objects, including

1. Stochastic losses (type, energy, position, time, ...)
2. Decay particles (no redundant static information!)
3. Produced particles (e.g. muon pair)
4. (Continuous losses)

C++ code example

```
Propagator prop(MuMinusDef::Get(), "resources/config.json");
Particle& mu = prop.GetParticle();
Particle mu_backup(mu);

mu_backup.SetEnergy(9e6); //energy in MeV
mu_backup.SetDirection(Vector3D(0, 0, -1));

std::vector<double> ranges;

for (int i = 0; i < 10; i++){
    mu.InjectState(mu_backup);
    prop.Propagate();
    ranges.push_back(mu.GetPropagatedDistance());
}
```

Python code example

```python
prop = pp.Propagator(particle_def=pp.particle.MuMinusDef.get(),
                     config_file="path/to/config.json")
mu = prop.particle
mu_backup = pp.particle.Particle(mu)

mu_backup.energy = 9e6 #energy in MeV
mu_backup.direction = pp.Vector3D(0, 0, -1)

ranges = []

for i in range(10):
    mu.inject_state(mu_backup)
    secondaries = prop.propagate()
    ranges.append(prop.particle.propagated_distance)
```

PROPOSAL changes for CORSIKA

Displacement calculation

$$-f(E) = \sum_{\text{crosssec.}} \sum_{\text{comp.}} \frac{\mathrm{d}E}{\mathrm{d}x}$$

Homogeneous medium:

$$-f(E)\,\rho_0 = \frac{\mathrm{d}E}{\mathrm{d}x}$$

$$\mathrm{d}x = -\frac{1}{\rho_0}\frac{\mathrm{d}E}{f(E)}$$

$$x_f = x_i - \frac{1}{\rho_0}\int_{E_i}^{E_f} \frac{\mathrm{d}E}{f(E)}$$

Non-homogeneous medium

$$-f(E)\,\rho(x) = \frac{\mathrm{d}E}{\mathrm{d}x}$$

$$\mathrm{d}x\,\rho(x) = -\frac{\mathrm{d}E}{f(E)}$$

$$\int_{x_f}^{x_i} \mathrm{d}x\,\rho(x) = -\int_{E_i}^{E_f} \frac{\mathrm{d}E}{f(E)}$$

solve for x_f

Future work

- Check / include cross sections for electron / positron propagation
- Photon propagation

→ Comparison with EGS4

- Magnetic field deflection

https://github.com/tudo-astroparticlephysics/PROPOSAL

https://arxiv.org/abs/1809.07740

PROPOSAL may be modified and distrubuted under terms of a modified LGPL license.
More information on our GitHub page.

Backup slides

Propagation

$$\frac{\mathrm{d}\sigma}{\mathrm{d}v} \underbrace{\longrightarrow}_{?} \text{energy losses}$$

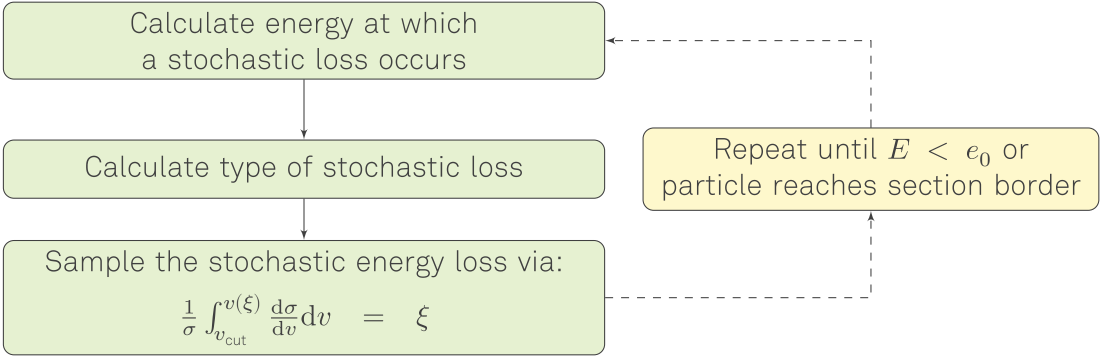
Propagation

$v < v_{\text{cut}}$
continuous losses

$v > v_{\text{cut}}$
stochastic losses

with $v_{\text{cut}} = \min\left[{}^{e_{\text{cut}}}/_{E}, v'_{\text{cut}}\right]$

Propagation



Standard interactions:

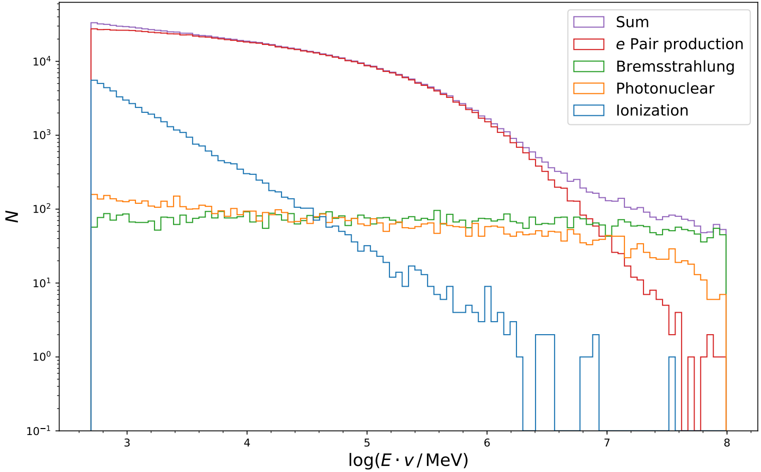
- e pair production
- Bremsstrahlung
- Photonuclear
- Ionization

Rare interactions:

- μ pair production
- Weak interaction

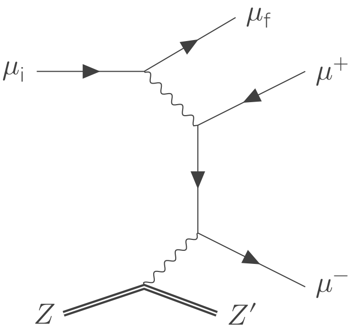
$\rightarrow$ Negligible contribution to overall energy loss

$\rightarrow$ Observable, interesting signature



Propagation of 10^4 muons with energy 10^8 MeV through 100 m of standard rock.

Direct Production of Muon Pairs



Energy fraction transferred to the muon pair:

$$v = \frac{(\epsilon_+ + \epsilon_-)}{E}$$

Asymmetry parameter:

$$\rho = \frac{(\epsilon_+ - \epsilon_-)}{(\epsilon_+ + \epsilon_-)}$$

E: Initial energy of the incoming muon μ_i
$\epsilon_\pm$: Energy of the produced (anti)muon

Double-differential cross section

For production of muon pairs [1]:

$$\frac{\mathrm{d}\sigma}{\mathrm{d}v\mathrm{d}\rho} = \frac{2}{3\pi}(Z\alpha r_\mu)^2 \frac{1-v}{v} \Phi(v,\rho) \ln(X)$$

For production of electron positron pairs [2]:

$$\frac{\mathrm{d}\sigma}{\mathrm{d}v\mathrm{d}\rho} = \frac{2}{3\pi} Z\left(Z+\xi\right)\left(\alpha r_e\right)^2 \frac{1-v}{v}\left(\Phi_e + \frac{m_e^2}{m_\mu^2}\Phi_\mu\right)$$

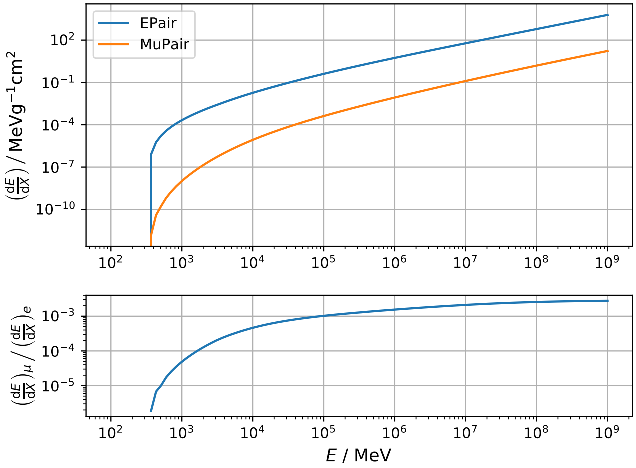
[1] Kelner, Kokoulin, Petrukhin: Phys. of Atomic Nuclei, Vol. 63, No. 9, 2000, pp. 1603-1611
[2] Kokoulin, Petrukhin: Proceedings of 12th ICCR, 1971, p. 2436

Continous energy loss per distance

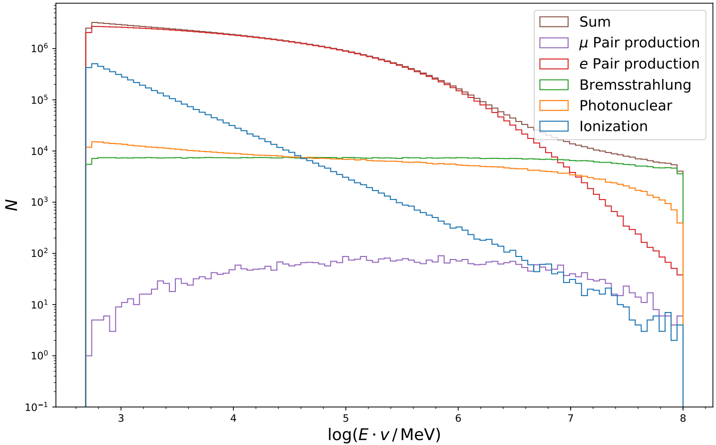
$$-\left\langle \frac{\mathrm{d}E}{\mathrm{d}x} \right\rangle = E \frac{N_\mathrm{A}}{A} \int_{v_\mathrm{min}}^{v_\mathrm{cut}} v \frac{\mathrm{d}\sigma}{\mathrm{d}v} \mathrm{d}v$$

with

$$v_\mathrm{min} = \frac{2m_\mu}{E},$$
$$v_\mathrm{max} = 1 - \frac{m_\mu}{E}.$$

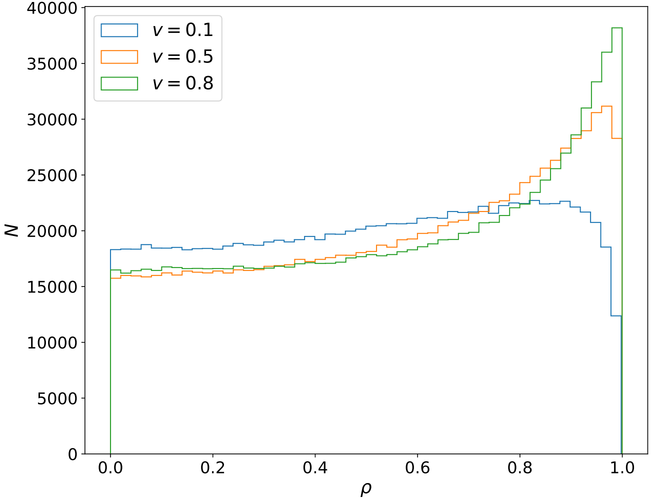


Comparion of e-pair and μ-pair production, only continous losses (i.e. $v_\mathrm{cut} = v_\mathrm{max}$).



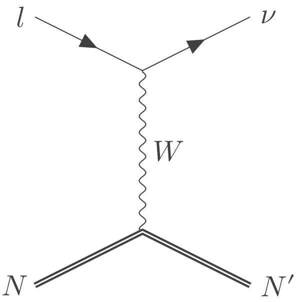
process	$N\,/\,N_{\mathrm{ges}}$	$E\,/\,E_{\mathrm{ges}}$
e pairp.	0,94	0,94
Ioniz.	$4 \cdot 10^{-2}$	$5 \cdot 10^{-2}$
Brems.	$1 \cdot 10^{-2}$	$7 \cdot 10^{-3}$
Photon.	$8 \cdot 10^{-3}$	$6 \cdot 10^{-3}$
μ pairp.	$6 \cdot 10^{-5}$	$5 \cdot 10^{-5}$

Stochastic losses, standard rock, 10^6 muons with $E = 10^8$ MeV, $e_{\mathrm{cut}} = 500$ MeV, $v_{\mathrm{cut}} = 5 \cdot 10^{-2}$.



Sampling of ρ for muons with $E = 1 \cdot 10^6$ MeV and different v in standard rock.

Weak interaction



- Highly suppressed process
- Similarities with "lollipop" signature in τ-events
- Crossing symmetry[3]:

$$\mathrm{d}\sigma\left(\mu Z \to \nu_\mu Z\right) = \frac{1}{2}\mathrm{d}\sigma\left(\nu_\mu Z \to \mu Z\right)$$

[3] Sandrock, Alexander: Higher-order corrections to the energy loss cross sections of high-energy muons, 2018, pp. 38-40

Future: Physical improvements in PROPOSAL

- Improvement of electron propagation
- Propagation of high-energy photons
- Deflection of particles in magnetic fields
- Propagation through media with non-homogenous density

https://github.com/tudo-astroparticlephysics/PROPOSAL

https://arxiv.org/abs/1809.07740

PROPOSAL may be modified and distrubuted under terms of a modified LGPL license.
More information on our GitHub page.