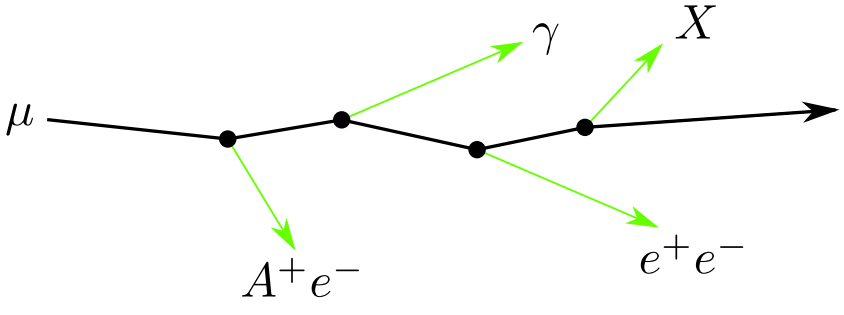
technische universität dortmund

e5 experimentelle physik 5
astroteilchenphysik



Recent developments for the high-energy lepton and photon propagator PROPOSAL

Jean-Marco Alameddine, Jan Soedingrekso, Alexander Sandrock, Maximilian Sackel

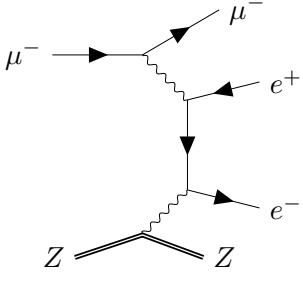
March 18, 2021

TU Dortmund University

Motivation behind PROPOSAL

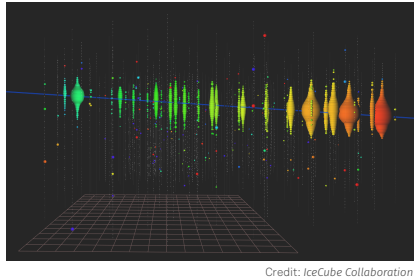
(and many other Monte Carlo Simulation tools...)

Cross sections:



→

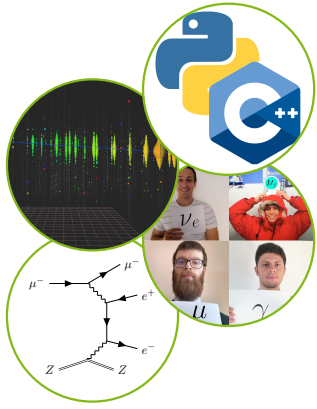
Experimental particle behaviour:



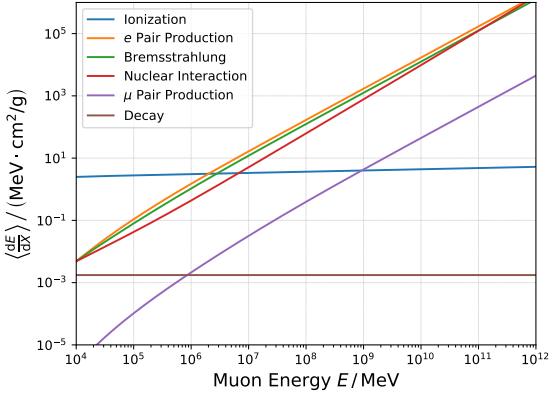
Credit: *IceCube Collaboration*

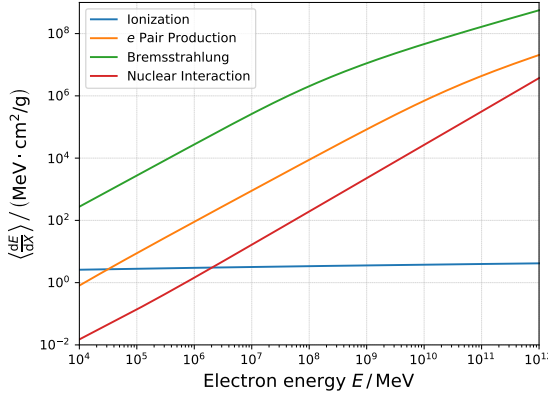
What is PROPOSAL?

- **PROPOSAL**: Software library to propagate high-energy leptons and photons
- Written in C++14, callable from Python as well
  - → Try: `pip install proposal`
- Customizable for advanced applications
  - → Selection of different parametrizations for each physical process
- Actively maintained
  - → Visit our GitHub: `https://github.com/tudo-astroparticlephysics/PROPOSAL`



Continous energy losses in ice





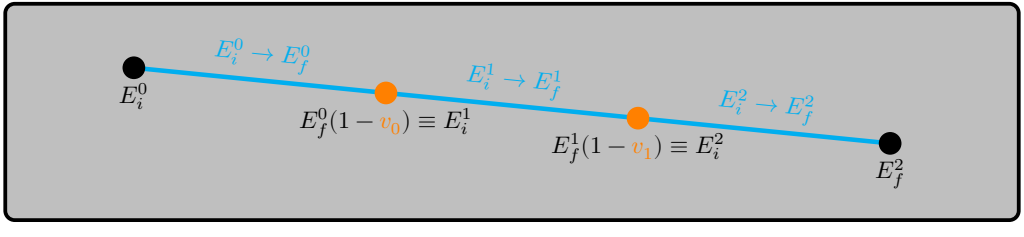
→ See more: Computer Phys. Comm., 184(9), 2070–2090 (2013)

Propagation principle

We define v as the relative energy loss, with:

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

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



$$\int_{E_i}^{E_f} \frac{\sigma(E)}{-f(E)} \cdot \mathrm{d}E = -\log(\xi)$$

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

Minimal PROPOSAL code example

Python Code

```python
import proposal as pp

# read properties from config file
particle = pp.particle.MuMinusDef()
prop = pp.Propagator(particle, "config.json")

# define initial particle state
init_state = pp.particle.ParticleState()
init_state.position = pp.Cartesian3D(0, 0, 0)
init_state.direction = pp.Cartesian3D(0, 0, 1)
init_state.energy = 1e6 # MeV

# propagation
prop_distances = []
for i in range(1000):
    output = prop.propagate(init_state,
                            max_distance = 1e5) # cm
    E_f = output.track()[-1].energy
    prop_distances.append(E_f)
```

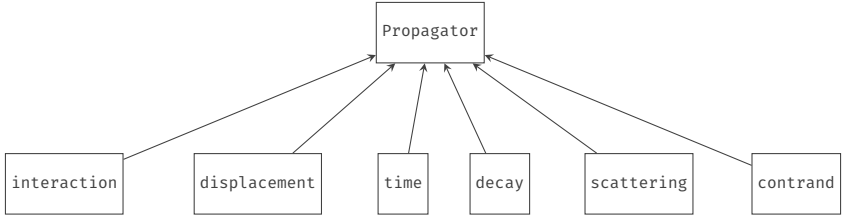
json file

```
"global":
{
  "cuts":
  {
        "e_cut": INF,
        "v_cut": 0.05,
        "cont_rand": false
  }
},

"sectors": [
{
  "medium": "ice",
  "geometries": [
        {
          "shape": "sphere",
          "origin": [0, 0, 0],
          "outer_radius": 6374134000000
        }
  ]
}
]
```

Modularization and further improvements

- Former versions of PROPOSAL: All calculations are hard-coded inside the Propagation routine
  - → Modularization: Turn calculations into individual modules



- Now, it is possible to use ...
  - ... the provided `Propagator` class as a "full particle simulation"
    - → For example: IceCube simulation chain (μ and τ)
  - ... only individual modules for external simulations/analyses
    - → For example: CORSIKA 8

Modules in PROPOSAL

Interaction

- Sample energy where next stochastic loss occurs
- Sample type and size of stochastic loss

Displacement

- Calculate displacement between two energies:

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

Time

- Calculate elapsed time between two energies:

$$t_f = t_i - \int_{E_i}^{E_f} \frac{\mathrm{d}E}{f(E)\, v(E)}$$

Scattering

- Sample multiple scattering between two energies
- Sample deflection angles for stochastic interactions

Continuous randomization

- Sample fluctuations on continuous energy losses:

$$\int_{E_i}^{E_f} \frac{E^2}{-f(E)} \left\langle \frac{\mathrm{d}^2 E}{\mathrm{d}x^2} \right\rangle \mathrm{d}E$$

Decay

- Sample energy when particle decays

→ See more: J. Phys. Conf. Ser. 1690, 012021 (2020)

Interpolation

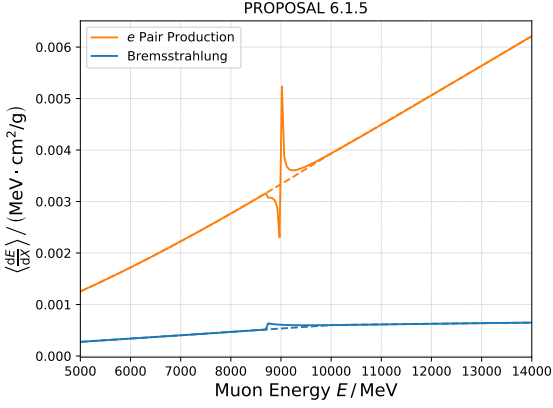
- Many integrals are calculated during propagation
  - → Usage of interpolation tables to increase performance
  - → Both cross section integrals (left) and integrals in modules (right) are interpolated

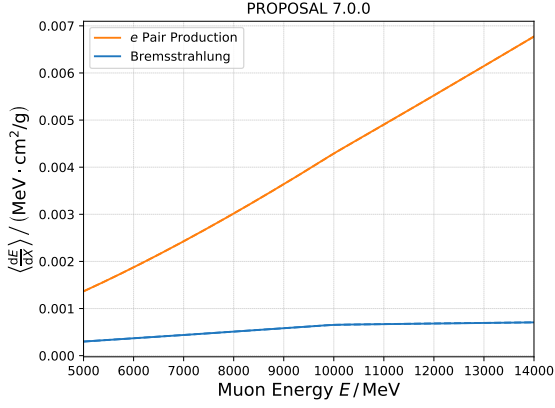
$$\int \frac{d\sigma}{d\nu}\, d\nu \longrightarrow \int_{E_i}^{E_f} \frac{\sigma(E)}{-f(E)} \cdot \mathrm{d}E = -\log(\xi)$$

$$\int \nu \frac{d\sigma}{d\nu}\, d\nu \longrightarrow x_f = x_i - \int_{E_i}^{E_f} \frac{\mathrm{d}E}{f(E)}$$

- Usage of an external interpolation routine based on cubic splines

→ Avoiding divergences when interpolated function has ”kinks” (e.g. due to energy cuts)



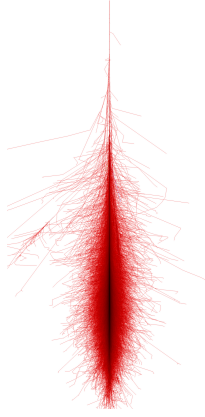


→ See more: github.com/MaxSac/cubic_interpolation

Application example: CORSIKA 8

CORSIKA 8

- CORSIKA: Monte Carlo Code to simulate Extensive Air Showers
- Up to CORSIKA7: Electromagnetic shower component simulated by EGS4
- CORSIKA 8: Construction of an EM shower model based on PROPOSAL modules (see CORSIKA GitLab)
  - → CORSIKA is interested in single propagation steps for electrons, positrons, muons, and photons



Credit: *https://www-zeuthen.desy.de/~jknapp/fs/showerimages.html*

PROPOSAL modules in CORSIKA 8

Simplified CORSIKA algorithm

put initial particle on stack

while *particle stack not empty* **do**

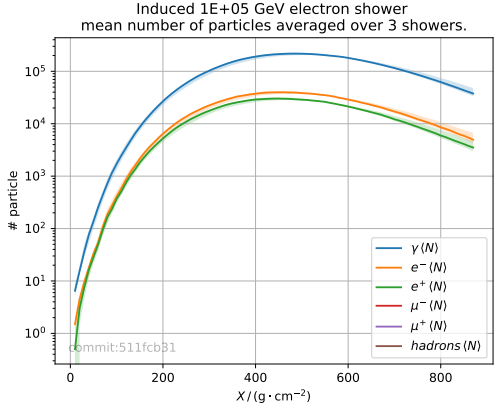
- extract particle from stack
- sample distance until next interaction ← interaction
- apply continuous processes on particle ← displacement | scattering
- sample interaction ← interaction
- put produced particles on particle stack ← secondaries

end

→ Usage of other models for particles that can not be treated by PROPOSAL (e.g. hadronic component)

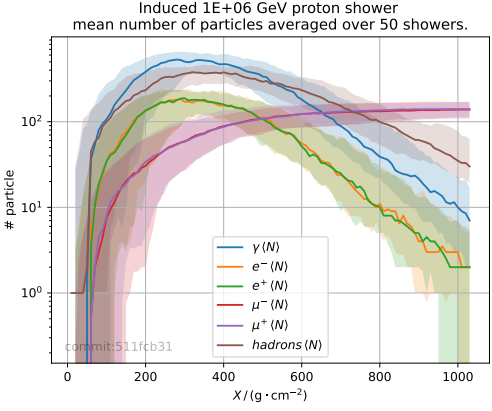
Example showers in CORSIKA 8

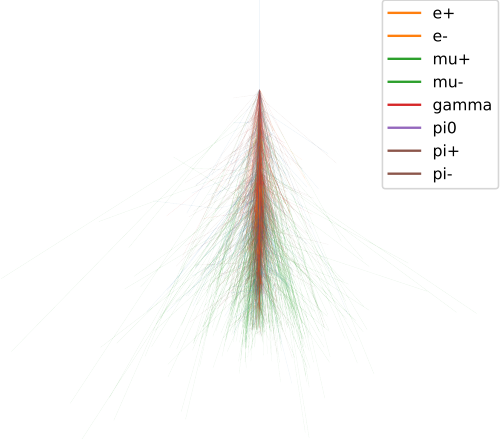
→ Production of first electromagnetic showers in CORSIKA 8 has been successful



Example showers in CORSIKA 8

→ Simulation of hadronic showers using PROPOSAL as an electromagnetic model





→ Careful analysis of obtained results necessary...

Summary

- PROPOSAL is a C++/Python software to propagate high-energy leptons and photons
  - → Customizable environment
- Calculations in PROPOSAL have been separated into individual modules
  - → Users can decide to either use a full particle simulation or only single modules
- Interpolation routine has been improved
- PROPOSAL is used in CORSIKA 8 to simulate the EM shower component

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