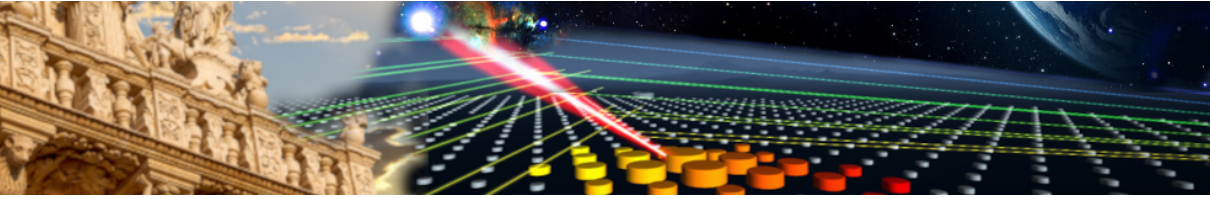
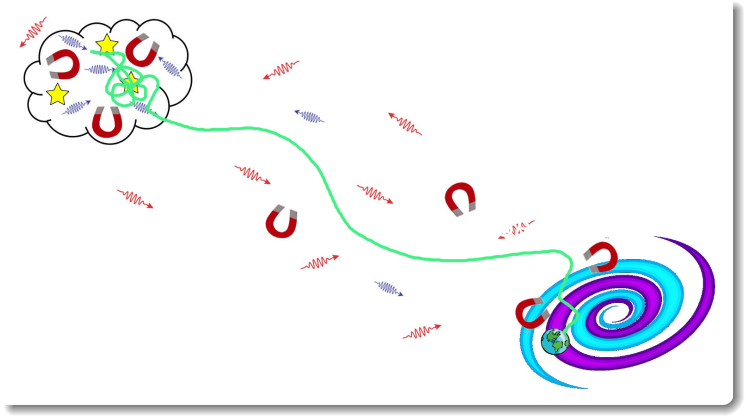
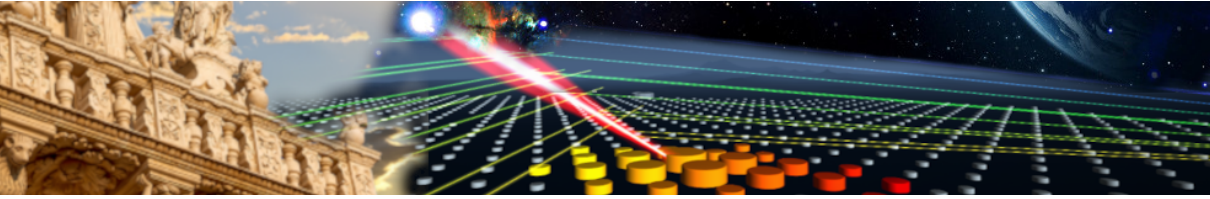


Hands-on Session: UHECR Propagation for Experimentalists

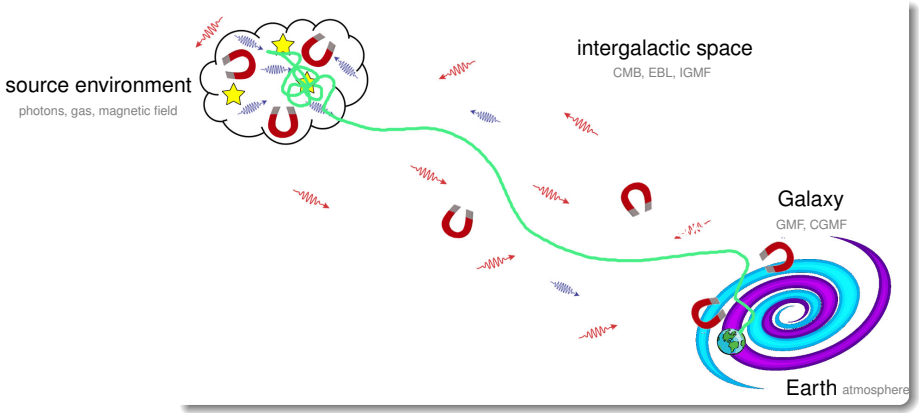
M. Unger (KIT)





Hands-on Session: UHECR Propagation for Experimentalists

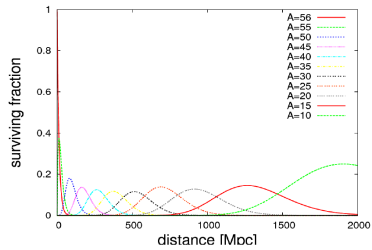
M. Unger (KIT)



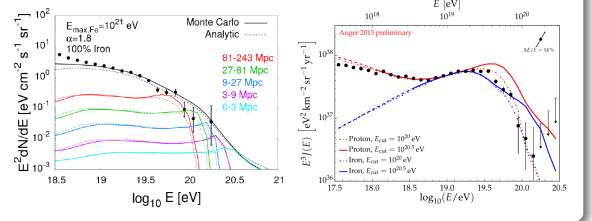
Goal: Explore Some of the Concepts Learned at this School

- #0 Getting Started (all)
- #1 Propagation of Nuclei
- #2 UHECR Flux at Earth
- #3 Cosmogenic Neutrinos

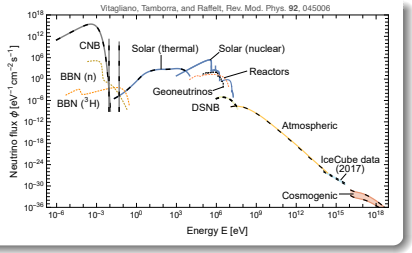
A. Taylor, lecture 3



A. Taylor, lecture 2; R. Engel, today's lecture



A. Marrone. D. Paesani

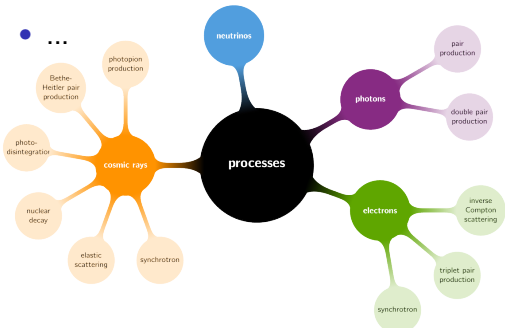


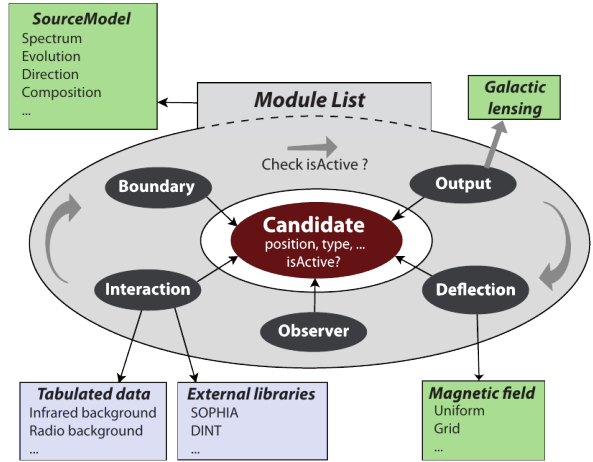
Hands-on: UHECR Propagation with CRPropa

(see also: SimProp, PriNCe)

CRPropa: versatile open-source library to

- propagate UHECRs (p,..., Fe) in intergalactic photon fields
- simulate deflections in the Galactic magnetic field and IGMF
- predict cosmogenic photon and neutrino fluxes
- ...





JCAP 09 (2022) 035, JCAP 05 (2016) 038

Getting Started: Installation

Installation:
Follow instructions at https://crpropa.github.io/CRPropa3/pages/Installation.html

- latest stable release 3.2.1 from 2023
- latest snapshot contains newest features (e.g. new GMF models)

Works for me: (Ubuntu Linux)

```
sudo apt install g++ gfortran python3 cmake swig      # see webpage above for other OS
export CRPDIR=$HOME/crp                               # where to install
mkdir -p $CRPDIR && cd $CRPDIR
git clone https://github.com/CRPropa/CRPropa3.git src # download source code
mkdir install && mkdir build && cd build
cmake ../src -DCMAKE_INSTALL_PREFIX=../install -DPython_INSTALL_PACKAGE_DIR=../install
make install -j8                                      # install with 8 CPUs
```

Getting Started: Check Installation

set environment:

```
export LD_LIBRARY_PATH=$CRPDIR/install/lib:$LD_LIBRARY_PATH
export PYTHONPATH=$CRPDIR/install/crpropa:$PYTHONPATH
```

simple check: start python and load crpropa library

```
python3
>>> import crpropa
```

There should be no output if installation was successful!

First Calculation – Setup

Simulation setup

We start with a `ModuleList`, which is a container for simulation modules, and represents the simulation.

[4]:
```
from crpropa import *

# simulation: a sequence of simulation modules
sim = ModuleList()

# add propagator for rectalinear propagation
sim.add(SimplePropagation())

# add interaction modules
sim.add(PhotoPionProduction(CMB()))
sim.add(ElectronPairProduction(CMB()))
sim.add(NuclearDecay())
sim.add(MinimumEnergy(1 * EeV))
```

Please note that all input, output and internal calculations are done using SI-units to enforce expressive statements such as `E = 1 * EeV` or `D = 100 * Mpc`.

https://crpropa.github.io/CRPropa3/pages/example_notebooks/basics/basics.html

First Calculation – Single Particle

Propagating a single particle

The simulation can now be used to propagate a cosmic ray, which is called candidate. We create a 100 EeV proton and propagate it using the simulation. The propagation stops when the energy drops below the minimum energy requirement that was specified. The possible propagation distances are rather long since we are neglecting cosmology in this example.

[5]:
```
cosmicray = Candidate(nucleusId(1, 1), 200 * EeV, Vector3d(100 * Mpc, 0, 0))

sim.run(cosmicray)
print(cosmicray)
print('Propagated distance', cosmicray.getTrajectoryLength() / Mpc, 'Mpc')
```

```
CosmicRay at z = 0
  source:  Particle 1000010010, E = 200 EeV, x = 100 0 0 Mpc, p = -1 0 0
  current: Particle 1000010010, E = 0.970856 EeV, x = -13950.9 0 0 Mpc, p = -1 0 0
Propagated distance 14050.929516298673 Mpc
```

https://crpropa.github.io/CRPropa3/pages/example_notebooks/basics/basics.html

First Calculation – Observer

Defining an observer

To define an observer within the simulation we create a `Observer` object. The convention of 1D simulations is that cosmic rays, starting from positive coordinates, propagate in the negative direction until the reach the observer at 0. Only the x-coordinate is used in the three-vectors that represent position and momentum.

[6]:
```
# add an observer
obs = Observer()
obs.add(Observer1D())  # observer at x = 0
sim.add(obs)
print(obs)
```

```
Observer
    ObserverPoint: observer at x = 0
    Flag: '' -> ''
    MakeInactive: yes
```

https://crpropa.github.io/CRPropa3/pages/example_notebooks/basics/basics.html

First Calculation – Output

Defining the output file

We want to save the propagated cosmic rays to an output file. Plain text output is provided by the TextOutput module. To save only cosmic rays that reach our observer, we add an output to the observer that we previously defined. This time we are satisfied with the output type Event1D.

[8]:
```
# event output
output2 = TextOutput('events.txt', Output.Event1D)
obs.onDetection(output2)

#sim.run(cosmicray)
#output2.close()
```

Note: If we want to use the CRPropa output file from within the same script that runs the simulation, the output module should be explicitly closed after the simulation run in order to get all events flushed to the file.

https://crpropa.github.io/CRPropa3/pages/example_notebooks/basics/basics.html

First Calculation – Source

Defining the source

To avoid setting each individual cosmic ray by hand we defince a cosmic ray source. The source is located at a distance of 100 Mpc and accelerates protons with a power law spectrum and energies between 1 - 200 EeV.

[9]:
```
# cosmic ray source
source = Source()
source.add(SourcePosition(100 * Mpc))
source.add(SourceParticleType(nucleusId(1, 1)))
source.add(SourcePowerLawSpectrum(1 * EeV, 200 * EeV, -1))
print(source)
```

```
Cosmic ray source
    SourcePosition: 100 0 0 Mpc
    SourceParticleType: 1000010010
    SourcePowerLawSpectrum: Random energy E = 1 - 200 EeV, dN/dE ~ E^-1
```

https://crpropa.github.io/CRPropa3/pages/example_notebooks/basics/basics.html

First Calculation – Run

Running the simulation

Finally we run the simulation to inject and propagate 10000 cosmic rays. An optional progress bar can show the progress of the simulation.

[10]:
```
sim.setShowProgress(True)  # switch on the progress bar
sim.run(source, 10000)
```

https://crpropa.github.io/CRPropa3/pages/example_notebooks/basics/basics.html

First Calculation: Putting it all together (cut and paste this)

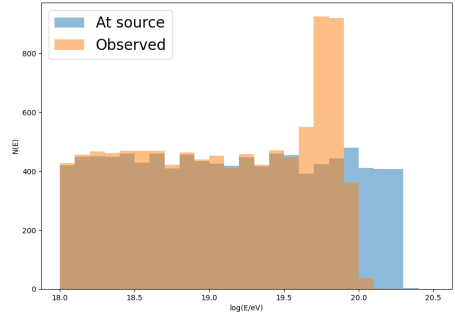
```
from crpropa import *

# simulation: a sequence of simulation modules
sim = ModuleList()

# add propagator for rectilinear propagation
sim.add(SimplePropagation())

# add interaction modules
sim.add(PhotoPionProduction(CMB()))
sim.add(ElectronPairProduction(CMB()))
sim.add(NuclearDecay())
sim.add(MinimumEnergy(1 * EeV))

# add an observer
obs = Observer()
obs.add(Observer1D())  # observer at x = 0
sim.add(obs)

# event output
output2 = TextOutput('events.txt', Output.Event1D)
obs.onDetection(output2)

# cosmic ray source
source = Source()
source.add(SourcePosition(100 * Mpc))
source.add(SourceParticleType(nucleusId(1, 1)))
source.add(SourcePowerLawSpectrum(1 * EeV, 200 * EeV, -1))

sim.setShowProgress(True)
sim.run(source, 10000)
```

First Calculation: Peeking into the output file

```
less events.txt
#         D         ID         E         ID0         E0
#
# D              Trajectory length [1 Mpc]
# ID/ID0/ID1     Particle type (PDG MC numbering scheme)
# E/E0/E1        Energy [1 EeV]
# no index = current, 0 = at source, 1 = at point of creation
#
# CRPropa version: 3.2.1-225-gd39003b5
#
1.00000E+02         1000010010         1.39309E+00         1000010010         1.40619E+00
1.00000E+02         1000010010         1.82626E+00         1000010010         1.85608E+00
1.00000E+02         1000010010         2.19376E+00         1000010010         2.24265E+00
1.00000E+02         1000010010         3.37704E+00         1000010010         3.51014E+00
1.00000E+02         1000010010         1.71551E+01         1000010010         1.87162E+01
1.00000E+02         1000010010         2.32895E+00         1000010010         2.38587E+00
1.00000E+02         1000010010         2.15826E+01         1000010010         2.35728E+01
1.00000E+02         1000010010         2.53034E+00         1000010010         2.60009E+00
1.00000E+02         1000010010         1.25770E+01         1000010010         1.36664E+01
1.00000E+02         1000010010         3.48896E+00         1000010010         3.63145E+00
1.00000E+02         1000010010         5.67061E+01         1000010010         6.66694E+01
1.00000E+02         1000010010         1.60153E+01         1000010010         1.74607E+01
1.00000E+02         1000010010         6.24159E+01         1000010010         8.93326E+01
```

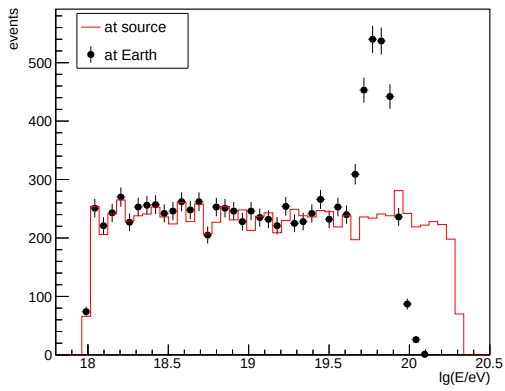
First Calculation: Plotting the result (matplotlib)

```
import matplotlib.pyplot as plt
import numpy as np

data = np.genfromtxt('events.txt', names=True)
print('Number of events', len(data))

logE0 = np.log10(data['E0']) + 18
logE  = np.log10(data['E']) + 18

plt.figure(figsize=(10, 7))
h1 = plt.hist(logE0, bins=25, range=(18, 20.5),
              histtype='stepfilled', alpha=0.5,
              label='At source')
h2 = plt.hist(logE,  bins=25, range=(18, 20.5),
              histtype='stepfilled', alpha=0.5,
              label='Observed')
plt.xlabel('log(E/eV)')
plt.ylabel('N(E)')
plt.legend(loc = 'upper left', fontsize=20)
plt.show()
```



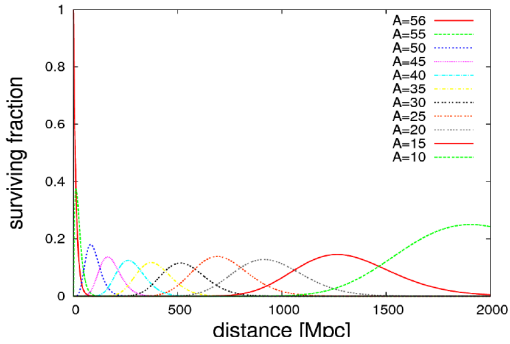
First Calculation: Plotting the result (ROOT)

```
TTree events;
events.ReadFile("events.txt", "D:ID:E:ID0:E0");

auto h1 = new TH1D("h1", "", 50, 17.8, 20.5);
auto h2 = new TH1D("h2", "", 50, 17.8, 20.5);
events.Project("h1", "log10(E0)+18");
events.Project("h2", "log10(E)+18");

auto canvas = new TCanvas("c", "c", 600, 500);
h1->SetTitle(";lg(E/eV);events");
h1->SetLineColor(kRed);
h2->SetLineColor(kBlack);
h2->Draw("E");
h1->Draw("HIST SAME");

auto leg = new TLegend(0.17, 0.81, 0.38, 0.94);
leg->AddEntry(h1, "at source", "L");
leg->AddEntry(h2, "at Earth", "PE");
leg->Draw();
```



Project #1: Extragalactic Propagation of Iron

Can you reproduce the figure from Andrew's lecture (#3, page 39)?



Hints:

- iron nuclues with $E_0 = 10^{20}$ eV

```
source.add(SourceParticleType(nucleusId(56, 26)))
```

- either run the example code repeatedly for different distances
- or inject particles at different distances

```
source.add(SourceUniform1D(0 * Mpc, 250 * Mpc))
```

- add photodisintegration and the EBL! (see next page)

Project #1: Explore Options

- add photodisintegration

```
sim.add(PhotoDisintegration(CMB()))
```

- add interactions with the EBL

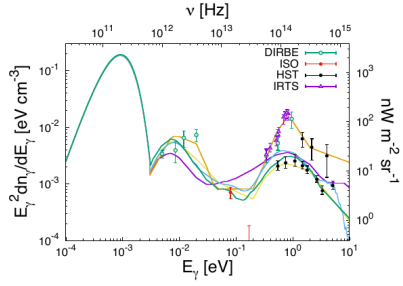
```
infrared = IRB_Gilmore12()
sim.add(PhotoPionProduction(infrared))
sim.add(PhotoDisintegration(infrared))
sim.add(ElectronPairProduction(infrared))
```

- change the energy at the source
- start with Si or N or He
- change the model of the EBL
  - what options are implemented? Find the documentation and refs!

```
IRB_Dominguez11, IRB_Finke10, IRB_Finke22,
IRB_Franceschini08, IRB_Gilmore12,
IRB_Kneiske04,IRB_Saldana21,
IRB_Saldana21_lower, IRB_Saldana21_upper,
IRB_Stecker05, IRB_Stecker16_lower,
IRB_Stecker16_upper
```

EBL Radiation Field Models

The EBL isn't actually known with very great accuracy (since it is difficult to measure directly)



This uncertainty should be propagated into theoretical calculations of interaction rates

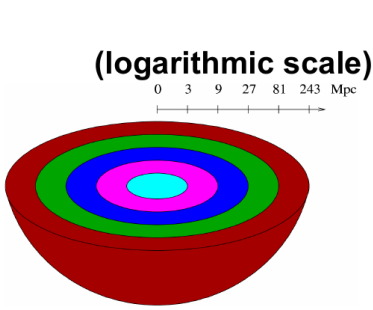
DESY.

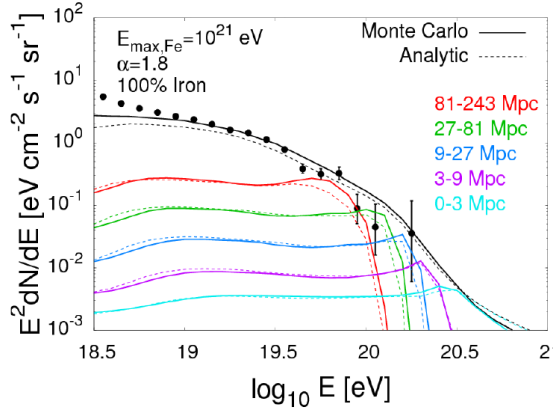
31

Andrew Taylor

Project #2: UHECR Flux at Earth

Can you reproduce colored curves from Andrew's lecture (#2, page 6)?





Hints:

- the volume of every shell grows $\propto R^2$ and the flux drops $\propto R^{-2}$
  $\rightarrow$ every shell of the same thickness contributes equally
  $\rightarrow$ sources can be simulated uniform in 1D distance
- note: Fe, power-law injection index is 1.8, $\lg(E_{\max}/\mathrm{eV}) = 12$

Project #2: UHECR Flux at Earth

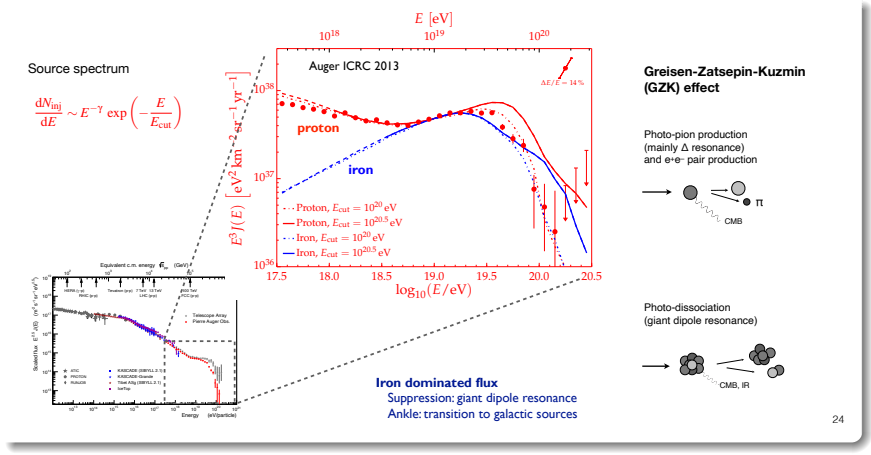
Note: For distances larger > 100 Mpc (black curve), cosmological effects need to be taken into account

- energy loss due to redshift
- luminosity distance vs. proper distance
- cosmological time dilation
- cosmological source evolution

```
# add adiabatic energy loss
sim.add(Redshift())
# for time delays
source.add(SourceRedshift1D())
# uniform between z = [0, 3]
source.add(SourceUniform1D(0, redshift2ComovingDistance(3)))
# (1 + z)^3 between z = [0, 4]
source.add(SourceRedshiftEvolution(3, 0, 4))
```

Project #2: UHECR Flux at Earth (advanced)

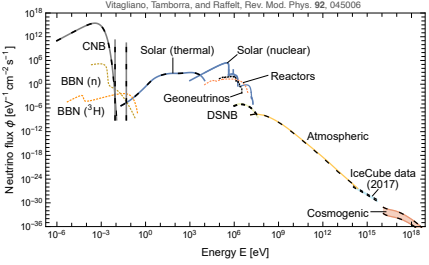
What about the p and Fe flux models from Ralph's lecture? (#3, page 24)?



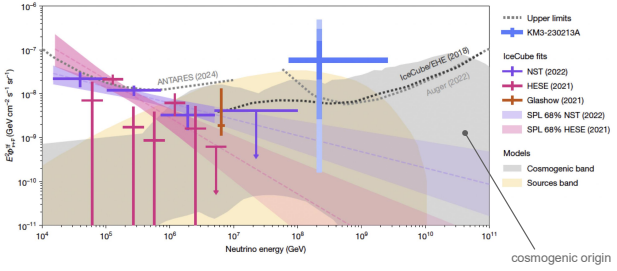
Hint: first calculate the spectra with low E_{cut} using the parameters given in Fig.6, page 30 of arXiv:1307.5059. Normalize the flux at 10 EeV to the data.

Project #3: Cosmogenic Neutrinos

Antonio Marrone

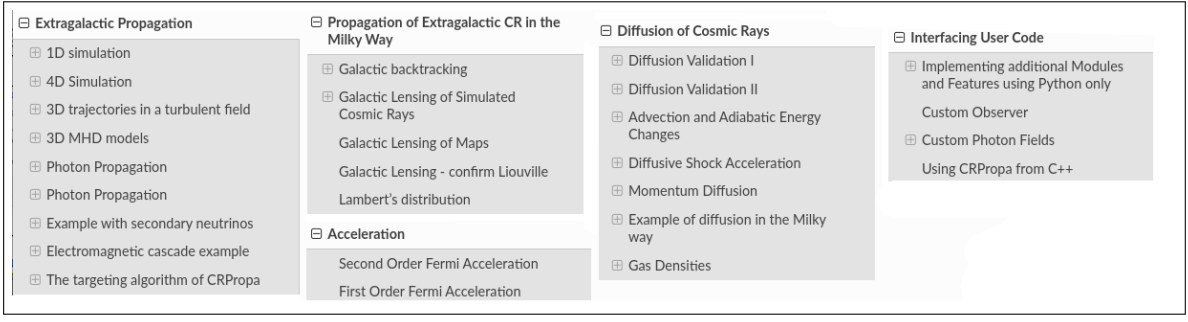


Daniele Paesani



- run the neutrino example at
  https://crpropa.github.io/CRPropa3/pages/example_notebooks/secondaries/neutrinos.html
- compare the cosmogenic neutrino fluxes for p and Fe injection
- how sensitive is the neutrino flux to the source evolution?
- hint: use the data-normalized spectra from project #2

Further Reading



https://crpropa.github.io/CRPropa3/