km3flux

Release

Nov 02, 2017

Contents

KM3Flux is a collection of neutrino flux models + assorted utilities for computing event weights.

CHAPTER 1

Install

This is developed on python 3.6. Lower versions (especially py2) might or might not work.

Install the dependencies

- numpy
- pandas
- h5py
- pytables

In your python env, do

```
pip install km3flux
```

CHAPTER 2

Contents

2.1 Example Gallery

orphan

2.1.1 Plot AllFlavor Flux

Demonstrate AllFlavorFlux

```
# Author: Moritz Lotze <mlotze@km3net.de>
# License: BSD-3

from km3flux.flux import AllFlavorFlux
```

use a single function call for mixed flavors

```
flavor = ['nu_mu', 'anu_mu', 'anu_mu']
ene = [10, 20, 30]
zen = [1, 1.5, 2]

flux = AllFlavorFlux()
print(flux(ene, zen, flavor))
```

Out:

```
[0 0 0]
```

Total running time of the script: (0 minutes 0.020 seconds)

2.1.2 Plot Honda Flux

Demonstrate the flux API.

```
# Author: Moritz Lotze <mlotze@km3net.de>
# License: BSD-3

import numpy as np
import matplotlib.pyplot as plt

from km3flux.flux import Honda2015
import km3pipe.style
km3pipe.style.use('moritz')
```

Out:

```
Loading style definitions from '/home/docs/checkouts/readthedocs.org/user_builds/
↪km3flux/conda/latest/lib/python3.6/site-packages/km3pipe/kp-data/stylelib/moritz.
↪mplstyle'
```

define energy + zenith range

```
n_points = 100

zen = np.linspace(0, np.pi, n_points)
ene = np.logspace(0, 2, n_points)
```

look at numu flux

```
numu_flux = Honda2015('nu_mu')
print(
    numu_flux(ene)
)
```

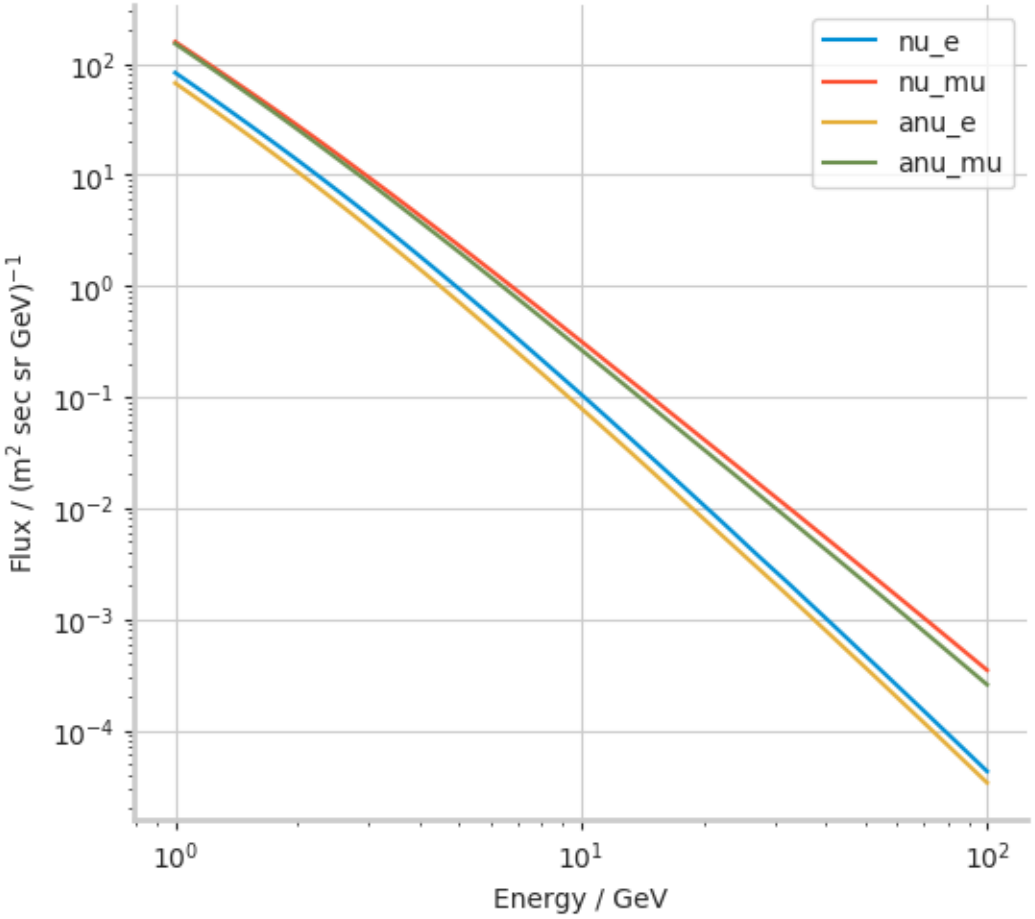
Out:

```
[  1.57511702e+02   1.40805786e+02   1.25795576e+02   1.12316559e+02
   1.00219031e+02   8.93900826e+01   7.96797888e+01   7.09984000e+01
   6.32340637e+01   5.62887906e+01   5.00835425e+01   4.45199189e+01
   3.95451221e+01   3.50945576e+01   3.11179301e+01   2.75776457e+01
   2.44246439e+01   2.16235953e+01   1.91270247e+01   1.69018435e+01
   1.49249012e+01   1.31683489e+01   1.16155118e+01   1.02435344e+01
   9.03300924e+00   7.96393262e+00   7.01492636e+00   6.17573148e+00
   5.43332560e+00   4.77819088e+00   4.20110151e+00   3.69189985e+00
   3.24402000e+00   2.84875206e+00   2.50035978e+00   2.19347598e+00
   1.92268010e+00   1.68489571e+00   1.47598439e+00   1.29295198e+00
   1.13250322e+00   9.91471728e-01   8.67979826e-01   7.59566700e-01
   6.64595330e-01   5.81278077e-01   5.08030294e-01   4.43981439e-01
   3.87965869e-01   3.39107059e-01   2.96335949e-01   2.58766972e-01
   2.25913190e-01   1.97150981e-01   1.72084528e-01   1.50239087e-01
   1.31186389e-01   1.14582963e-01   1.00015893e-01   8.72806098e-02
   7.61524692e-02   6.64388153e-02   5.79810103e-02   5.05870742e-02
   4.41452009e-02   3.85195385e-02   3.36050613e-02   2.93202090e-02
   2.55696673e-02   2.22979010e-02   1.94359310e-02   1.69367028e-02
   1.47662771e-02   1.28816055e-02   1.12454584e-02   9.80914811e-03
   8.54625655e-03   7.44326914e-03   6.48291980e-03   5.64949261e-03
   4.92167267e-03   4.28624029e-03   3.73253126e-03   3.24920246e-03
   2.82894557e-03   2.46221055e-03   2.14280804e-03   1.86484463e-03
   1.62238475e-03   1.41163399e-03   1.22768864e-03   1.06752965e-03
   9.28168588e-04   8.06721721e-04   7.01285215e-04   6.09404031e-04
   5.29577691e-04   4.60217128e-04   3.99848958e-04   3.47446816e-04]
```

Plot vs energy, for multiple flavors

```
flavors = {'nu_mu', 'anu_mu', 'nu_e', 'anu_e'}
for flav in flavors:
    flux = Honda2015(flavor=flav)
    plt.plot(ene, flux(ene), label=flav)

plt.yscale('log')
plt.xscale('log')
plt.xlabel('Energy / GeV')
plt.ylabel(r'Flux / (m$^2$ sec sr GeV)$^{-1}$')
plt.legend()
```



Total running time of the script: (0 minutes 0.094 seconds)

2.2 API Reference

- *km3flux.flux*
- *km3flux.weights*
- *km3flux.aeff*

2.2.1 km3flux.flux

Assorted Fluxes, in (m^2 sec sr GeV)^-1

BaseFlux(**kwargs)	Base class for fluxes.
AllFlavorFlux([fluxclass])	Get mixed-flavor fluxes.
Honda2015([flavor])	Get Honda 2015 atmospheric neutrino fluxes.
HondaSarcevic([flavor])	Get Honda + Sarcevic atmospheric neutrino fluxes.
PowerlawFlux([gamma, scale])	E^-gamma flux.
DarkMatterFlux([source, flavor, channel, mass])	Get Dark Matter spectra (taken from M.
dmflux(energy, **kwargs)	Get the DM flux for your energies.
e2flux(energy, **kwargs)	
all_dmfluxes([source])	Get all dark matter fluxes from all channels, masses, flavors.
all_dmfluxes_samplesd	

km3flux.flux.BaseFlux

class km3flux.flux.**BaseFlux**(***kwargs*)
Base class for fluxes.

Methods

__call__(energy, zenith=None)	Return the flux on energy, optionally on zenith.
integrate(zenith=None, emin=1, emax=100, **integargs)	Integrate the flux via romberg integration.
integrate_samples(energy, zenith=None, emin=1, emax=100)	Integrate the flux from given samples, via simpson integration.

__init__(***kwargs*)

Methods

__init__(**kwargs)
integrate([zenith, emin, emax])
integrate_samples(energy[, zenith, emin, emax])

km3flux.flux.AllFlavorFlux

class km3flux.flux.**AllFlavorFlux**(*fluxclass='Honda2015'*)
Get mixed-flavor fluxes.

Methods

__init__(fluxclass='Honda2015')	
__call__(energy, zenith=None)	Return the flux on energy, optionally on zenith.

__init__ (*fluxclass='Honda2015'*)

Methods

__init__([fluxclass])

km3flux.flux.Honda2015

class km3flux.flux.**Honda2015** (*flavor='nu_mu'*)
Get Honda 2015 atmospheric neutrino fluxes.

Methods

__call__(energy[, zenith])
integrate([zenith, emin, emax])
integrate_samples(energy[, zenith, emin, emax])

__init__ (*flavor='nu_mu'*)

Methods

__init__([flavor])
integrate([zenith, emin, emax])
integrate_samples(energy[, zenith, emin, emax])

km3flux.flux.HondaSarcevic

class km3flux.flux.**HondaSarcevic** (*flavor='nu_mu'*)
Get Honda + Sarcevic atmospheric neutrino fluxes.

Methods

__call__(energy[, zenith])
integrate([zenith, emin, emax])
integrate_samples(energy[, zenith, emin, emax])

__init__ (*flavor='nu_mu'*)

Methods

__init__([flavor])
integrate([zenith, emin, emax])
integrate_samples(energy[, zenith, emin, emax])

km3flux.flux.PowerlawFlux

class km3flux.flux.**PowerlawFlux**(*gamma=2*, *scale=0.0001*)
E^-gamma flux.

Methods

__call__(energy[, zenith])	
integrate([zenith, emin, emax])	Compute analytic integral instead of numeric one.
integrate_samples(energy[, zenith, emin, emax])	

__init__(*gamma=2*, *scale=0.0001*)

Methods

__init__([gamma, scale])	
integrate([zenith, emin, emax])	Compute analytic integral instead of numeric one.
integrate_samples(energy[, zenith, emin, emax])	

km3flux.flux.DarkMatterFlux

class km3flux.flux.**DarkMatterFlux**(*source='gc'*, *flavor='nu_mu'*, *channel='w'*, *mass=1000*)
Get Dark Matter spectra (taken from M. Cirelli).

```
>>> from km3flux import DarkMatterFlux
>>> flux = DarkMatterFlux()
```

```
>>> ene = np.logspace(0, 2, 11)
```

```
>>> flux('anu_mu', ene)
array([  6.68440000e+01,   1.83370000e+01,   4.96390000e+00,
     1.61780000e+00,   5.05350000e-01,   2.29920000e-01,
     2.34160000e-02,   2.99460000e-03,   3.77690000e-04,
     6.87310000e-05,   1.42550000e-05])
```

Methods

__call__(energy[, zenith])	
integrate([zenith, emin, emax])	
integrate_samples(energy[, zenith, emin, emax])	

__init__(*source='gc'*, *flavor='nu_mu'*, *channel='w'*, *mass=1000*)

Methods

__init__([source, flavor, channel, mass])	
integrate([zenith, emin, emax])	
integrate_samples(energy[, zenith, emin, emax])	

km3flux.flux.dmflux

km3flux.flux.**dmflux**(*energy*, ***kwargs*)
Get the DM flux for your energies.

km3flux.flux.e2flux

km3flux.flux.**e2flux**(*energy*, ***kwargs*)

km3flux.flux.all_dmfluxes

km3flux.flux.**all_dmfluxes**(*source='gc'*)
Get all dark matter fluxes from all channels, masses, flavors.

2.2.2 km3flux.weights

nu_wgt	
atmu_wgt	
make_weights	
strange_flavor_to_mupage	
add_flavor	
add_weights_and_fluxes	

2.2.3 km3flux.aeff

Utilities for effective Area computation.

All energies in GeV.

integrated_energy	
integrated_zenith([zen_min, zen_max])	Integrate zenith.
aeff_scale_factor	

km3flux.aeff.integrated_zenith

km3flux.aeff.**integrated_zenith**(*zen_min=0*, *zen_max=3.141592653589793*)
Integrate zenith.

- genindex

Python Module Index

k

Index

Symbols

A

B

D

E

H

I

K

P