SNEWS
SUPERNOVA EARLY WARNING SYSTEM

Karlsruher Institut für Technologie

# multimessenger/supernova*

**Tool for calculating CEvNS interactions**

Melih Kara
kara@kit.edu

## SNEWS Collaboration Meeting
## 05.08.2022 Thursday

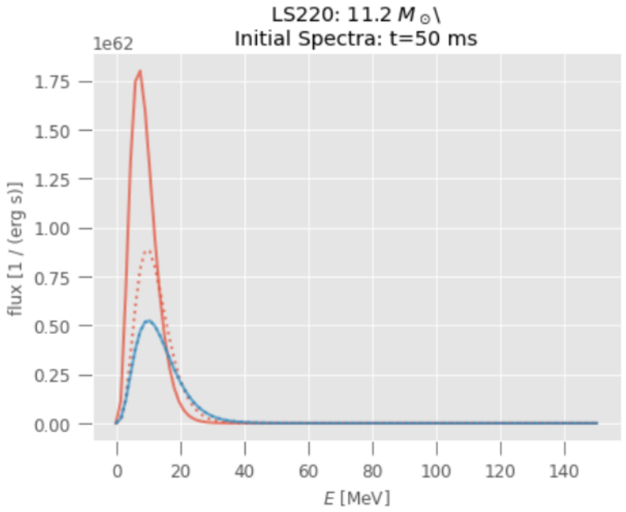
* cause we are running out of abbreviations

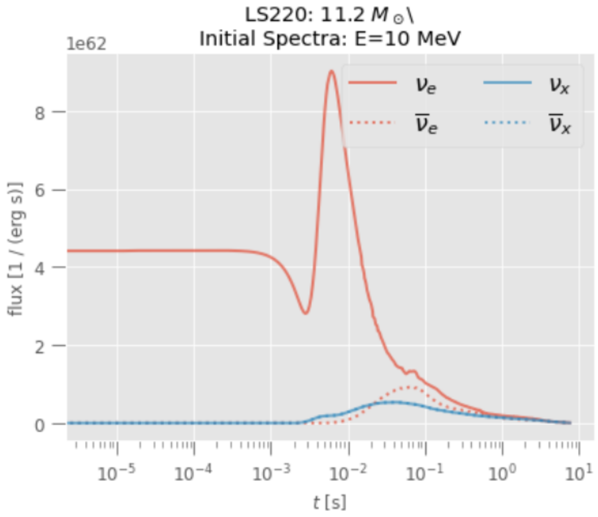
GitHub
**click me**-*ow*

# CEvNS interaction Rates

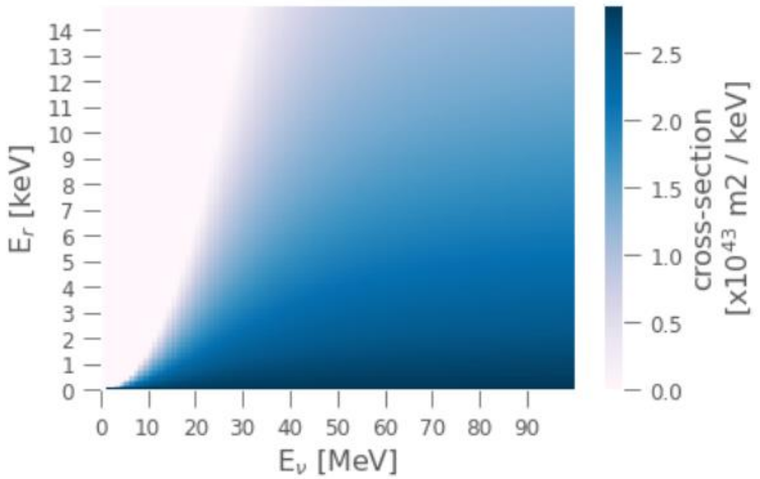
$$\frac{d^2R}{dE_R dt_{pb}} = \sum_{\nu_\beta} N_{Xe} \int_{E^{\nu}_{min}} dE_\nu \; f_\nu(E_\nu, t, d) \; \frac{d\sigma}{dE_R}(E_\nu, E_r)$$

**fluxes**




**Cross-sections**



# CEvNS interaction Rates

```
A = sn.Models(model_name='Fornax_2021')
```

```
> Available files for this model, please select an index

[0]     lum_spec_12M_r10000_dat.h5
[1]     lum_spec_13M_r10000_dat.h5
[2]     lum_spec_14M_r10000_dat.h5
[3]     lum_spec_15M_r10000_dat.h5
[4]     lum_spec_16M_r10000_dat.h5
[5]     lum_spec_17M_r10000_dat.h5
[6]     lum_spec_18M_r10000_dat.h5
[7]     lum_spec_19M_r10000_dat.h5
[8]     lum_spec_20M_r10000_dat.h5
[9]     lum_spec_21M_r10000_dat.h5
[10]    lum_spec_22M_r10000_dat.h5
[11]    lum_spec_23M_r10000_dat.h5
[12]    lum_spec_25M_r10000_dat.h5
[13]    lum_spec_26.99M_r10000_dat.h5
[14]    lum_spec_26M_r10000_dat.h5
```

- Object oriented
- Flexible (change the input composite)
- Scalable by the distance

- Computes rates per time,

Fluxes can be fetched from any snewpy model

$$\frac{d^2R}{dE_R\,dt_{pb}} = \sum_{\nu_\beta} N_{Xe} \int_{E^{\nu}_{min}} dE_\nu \, f_\nu(E_\nu, t, d) \, \frac{d\sigma}{dE_R}(E_\nu, E_r)$$

# CEvNS interaction Rates

```
A = sn.Models(model_name='Fornax_2021')
```

```
> Available files for this model, please select an index


[0]     lum_spec_12M_r10000_dat.h5
[1]     lum_spec_13M_r10000_dat.h5
[2]     lum_spec_14M_r10000_dat.h5
[3]     lum_spec_15M_r10000_dat.h5
[4]     lum_spec_16M_r10000_dat.h5
[5]     lum_spec_17M_r10000_dat.h5
[6]     lum_spec_18M_r10000_dat.h5
[7]     lum_spec_19M_r10000_dat.h5
[8]     lum_spec_20M_r10000_dat.h5
[9]     lum_spec_21M_r10000_dat.h5
[10]    lum_spec_22M_r10000_dat.h5
[11]    lum_spec_23M_r10000_dat.h5
[12]    lum_spec_25M_r10000_dat.h5
[13]    lum_spec_26.99M_r10000_dat.h5
[14]    lum_spec_26M_r10000_dat.h5
```

Fluxes can be fetched from any snewpy model

- Object oriented
- Flexible (change the input composite)
- Scalable by the distance

- Computes rates per time, rates per recoil energy

$$\frac{d^2R}{dE_R dt_{pb}} = \sum_{\nu_\beta} N_{Xe} \int_{E^{\nu}_{min}} dE_\nu \; f_\nu(E_\nu, t, d) \; \frac{d\sigma}{dE_R}(E_\nu, E_r)$$

# CEvNS interaction Rates

```
A = sn.Models(model_name='Fornax_2021')
```

```
> Available files for this model, please select an index


[0]     lum_spec_12M_r10000_dat.h5
[1]     lum_spec_13M_r10000_dat.h5
[2]     lum_spec_14M_r10000_dat.h5
[3]     lum_spec_15M_r10000_dat.h5
[4]     lum_spec_16M_r10000_dat.h5
[5]     lum_spec_17M_r10000_dat.h5
[6]     lum_spec_18M_r10000_dat.h5
[7]     lum_spec_19M_r10000_dat.h5
[8]     lum_spec_20M_r10000_dat.h5
[9]     lum_spec_21M_r10000_dat.h5
[10]    lum_spec_22M_r10000_dat.h5
[11]    lum_spec_23M_r10000_dat.h5
[12]    lum_spec_25M_r10000_dat.h5
[13]    lum_spec_26.99M_r10000_dat.h5
[14]    lum_spec_26M_r10000_dat.h5
```

Fluxes can be fetched from any snewpy model

- Object oriented
- Flexible (change the input composite)
- Scalable by the distance

- Computes rates per time, rates per recoil energy per neutrino flavor

$$\frac{d^2R}{dE_R dt_{pb}} = \sum_{\nu_\beta} N_{Xe} \int_{E^{\nu}_{min}} dE_\nu \, f_\nu(E_\nu, t, d) \, \frac{d\sigma}{dE_R}(E_\nu, E_r)$$

# CEvNS interaction Rates

```
A = sn.Models(model_name='Fornax_2021')
```

```
> Available files for this model, please select an index


[0]     lum_spec_12M_r10000_dat.h5
[1]     lum_spec_13M_r10000_dat.h5
[2]     lum_spec_14M_r10000_dat.h5
[3]     lum_spec_15M_r10000_dat.h5
[4]     lum_spec_16M_r10000_dat.h5
[5]     lum_spec_17M_r10000_dat.h5
[6]     lum_spec_18M_r10000_dat.h5
[7]     lum_spec_19M_r10000_dat.h5
[8]     lum_spec_20M_r10000_dat.h5
[9]     lum_spec_21M_r10000_dat.h5
[10]    lum_spec_22M_r10000_dat.h5
[11]    lum_spec_23M_r10000_dat.h5
[12]    lum_spec_25M_r10000_dat.h5
[13]    lum_spec_26.99M_r10000_dat.h5
[14]    lum_spec_26M_r10000_dat.h5
```

**Fluxes can be fetched from any snewpy model**

- **Object oriented**
- **Flexible (change the input composite)**
- **Scalable by the distance**

- **Computes rates per time, rates per recoil energy per neutrino flavor, per isotope**

$$\frac{d^2R}{dE_R dt_{pb}} = \sum_{\nu_\beta} N_{Xe} \int_{E^\nu_{min}} dE_\nu \; f_\nu(E_\nu, t, d) \; \frac{d\sigma}{dE_R}(E_\nu, E_r)$$

# CEvNS int

```
A = sn.Models(model_name='Forna
```

> Available files for this mode

```
[0]     lum_spec_12M_r10000_dat
[1]     lum_spec_13M_r10000_dat
[2]     lum_spec_14M_r10000_dat
[3]     lum_spec_15M_r10000_dat
[4]     lum_spec_16M_r10000_dat
[5]     lum_spec_17M_r10000_dat
[6]     lum_spec_18M_r10000_dat
[7]     lum_spec_19M_r10000_dat
[8]     lum_spec_20M_r10000_dat
[9]     lum_spec_21M_r10000_dat
[10]    lum_spec_22M_r10000_dat
[11]    lum_spec_23M_r10000_dat
[12]    lum_spec_25M_r10000_dat
[13]    lum_spec_26.99M_r10000_
[14]    lum_spec_26M_r10000_dat
```

Xenon_Atom.py | Supernova_Models.py | Recoil_calculations.py | Plotter.py

sktop\Ka

```python
Xe134 = {
    'Type'      : 'Xe134',
    'MassNum'   : 134,
    'AtomicNum' : 54,
    'Mass'      : 133.9053945,
    'Spin'      : 0,
    'Fraction'  : 0.104357
}

Xe136 = {
    'Type'      : 'Xe136',
    'MassNum'   : 136,
    'AtomicNum' : 54,
    'Mass'      : 135.907219,
    'Spin'      : 0,
    'Fraction'  : 0.088573
}

ATOM_TABLE = {
    'Xe124' : Xe124,
    'Xe126' : Xe126,
    'Xe128' : Xe128,
    'Xe129' : Xe129,
    'Xe130' : Xe130,
    'Xe131' : Xe131,
    'Xe132' : Xe132,
    'Xe134' : Xe134,
    'Xe136' : Xe136
}
```

ated_dat

composite)

rates per recoil energy

pe

Fluxes can be fetched fron

$$\int_{E^{\nu}_{min}} dE_{\nu}\ f_{\nu}(E_{\nu}, t, d)\ \frac{d\sigma}{dE_R}(E_{\nu}, E_r)$$

# CEvNS int

```
A = sn.Models(model_name='Forna
```

> Available files for this mode

```
[0]     lum_spec_12M_r10000_dat
[1]     lum_spec_13M_r10000_dat
[2]     lum_spec_14M_r10000_dat
[3]     lum_spec_15M_r10000_dat
[4]     lum_spec_16M_r10000_dat
[5]     lum_spec_17M_r10000_dat
[6]     lum_spec_18M_r10000_dat
[7]     lum_spec_19M_r10000_dat
[8]     lum_spec_20M_r10000_dat
[9]     lum_spec_21M_r10000_dat
[10]    lum_spec_22M_r10000_dat
[11]    lum_spec_23M_r10000_dat
[12]    lum_spec_25M_r10000_dat
[13]    lum_spec_26.99M_r10000_
[14]    lum_spec_26M_r10000_dat
```

Fluxes can be fetched fron

Xenon_Atom.py | Supernova_Models.py | Recoil_calculations.py | Plotter.py

```
Xe134 = {
    'Type'      : 'Xe134',
    'MassNum'   : 134,
    'AtomicNum' : 54,
    'Mass'      : 133.9053945,
    'Spin'      : 0,
    'Fraction'  : 0.104357
}

Xe134 = {
...
    'Xe126' : Xe126,
    'Xe128' : Xe128,
    'Xe129' : Xe129,
    'Xe130' : Xe130,
    'Xe131' : Xe131,
    'Xe132' : Xe132,
    'Xe134' : Xe134,
    'Xe136' : Xe136
}
```

**Once computed, can always be fetched in < 1s**

```
A = sn.Models(model_name='Fornax_2021', index=5)
```

```
A.compute_rates();
```

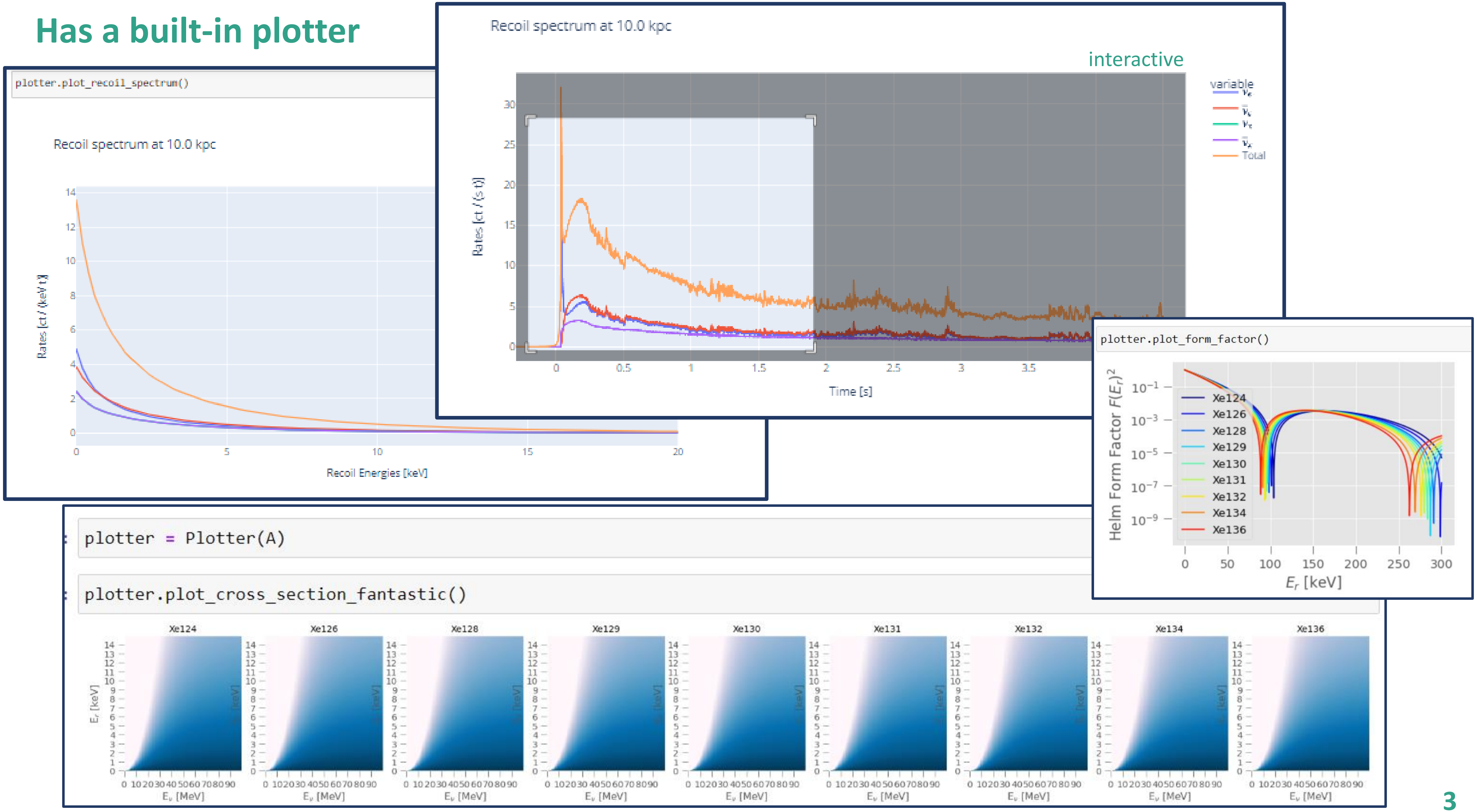
Computing for all isotopes: 100% 9/9 [00:00<00:00, 457.87it/s]

100% 9/9 [00:00<00:00, 46.67it/s]

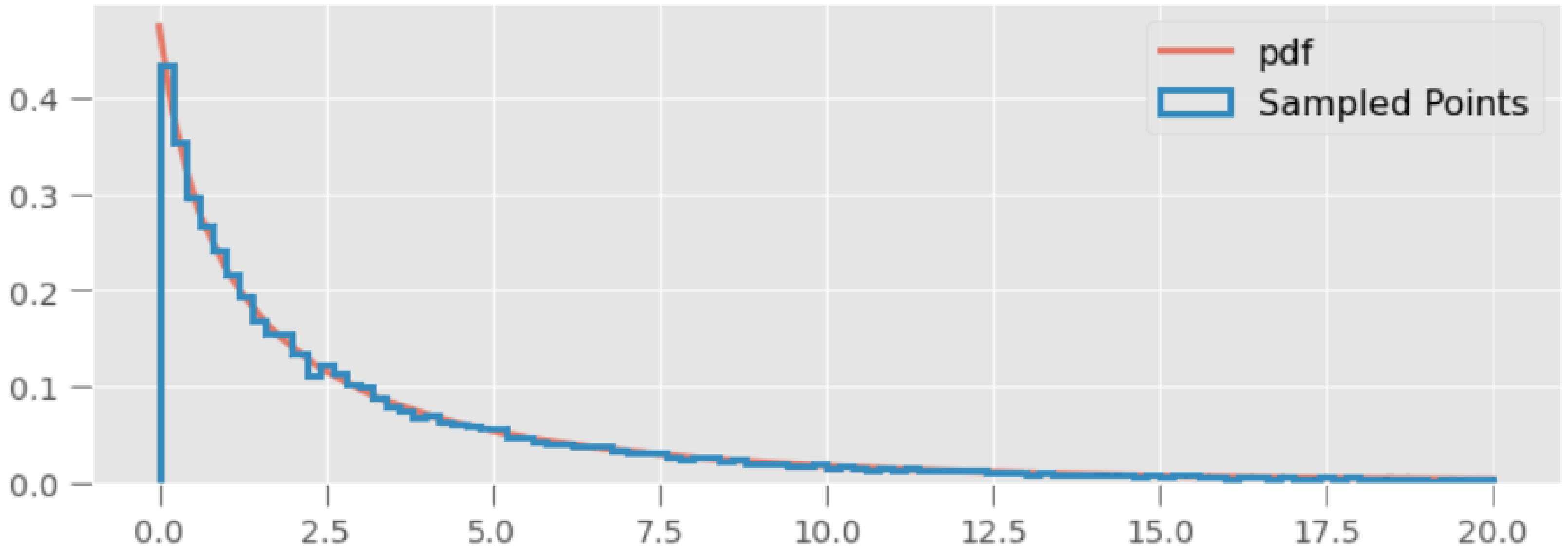
100% 9/9 [00:00<00:00, 44.24it/s]

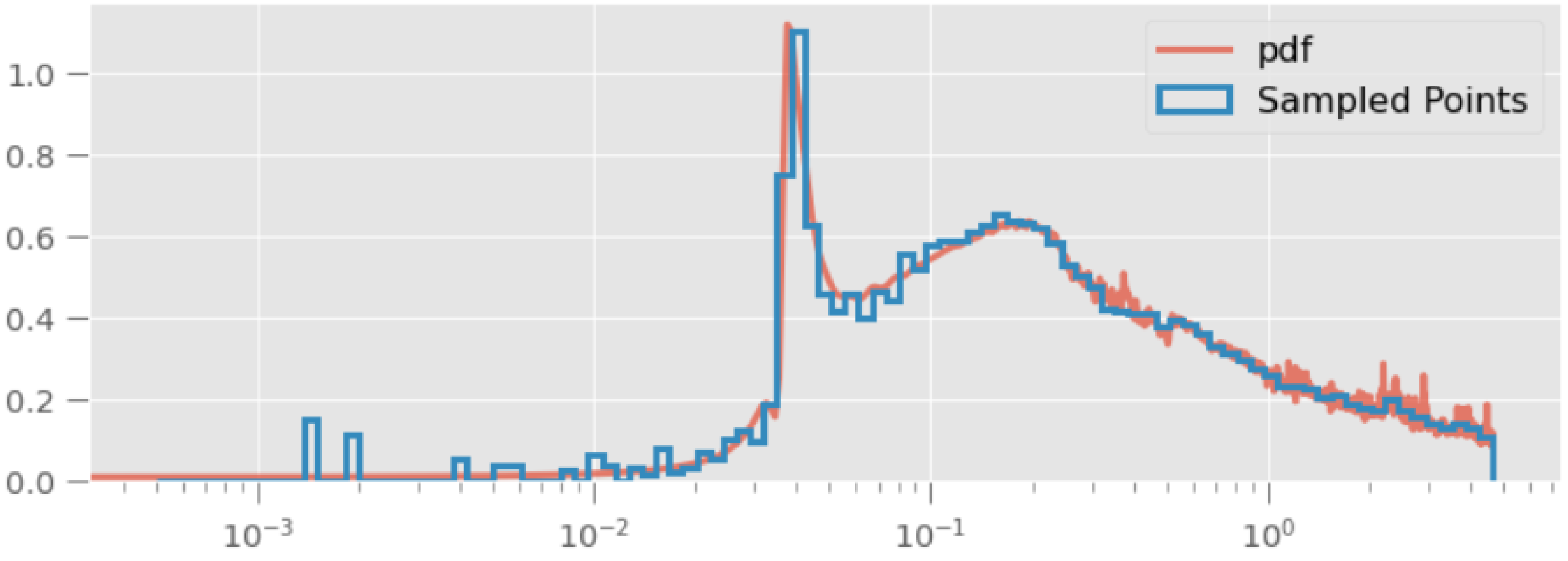
$$\int_{E^{\nu}_{min}} dE_{\nu}\ f_{\nu}(E_{\nu}, t, d)\ \frac{d\sigma}{dE_R}(E_{\nu}, E_r)$$

# Has a built-in plotter



# In XENONnT …




sample recoil energies, times
simulate and analyze signal
using WFSim

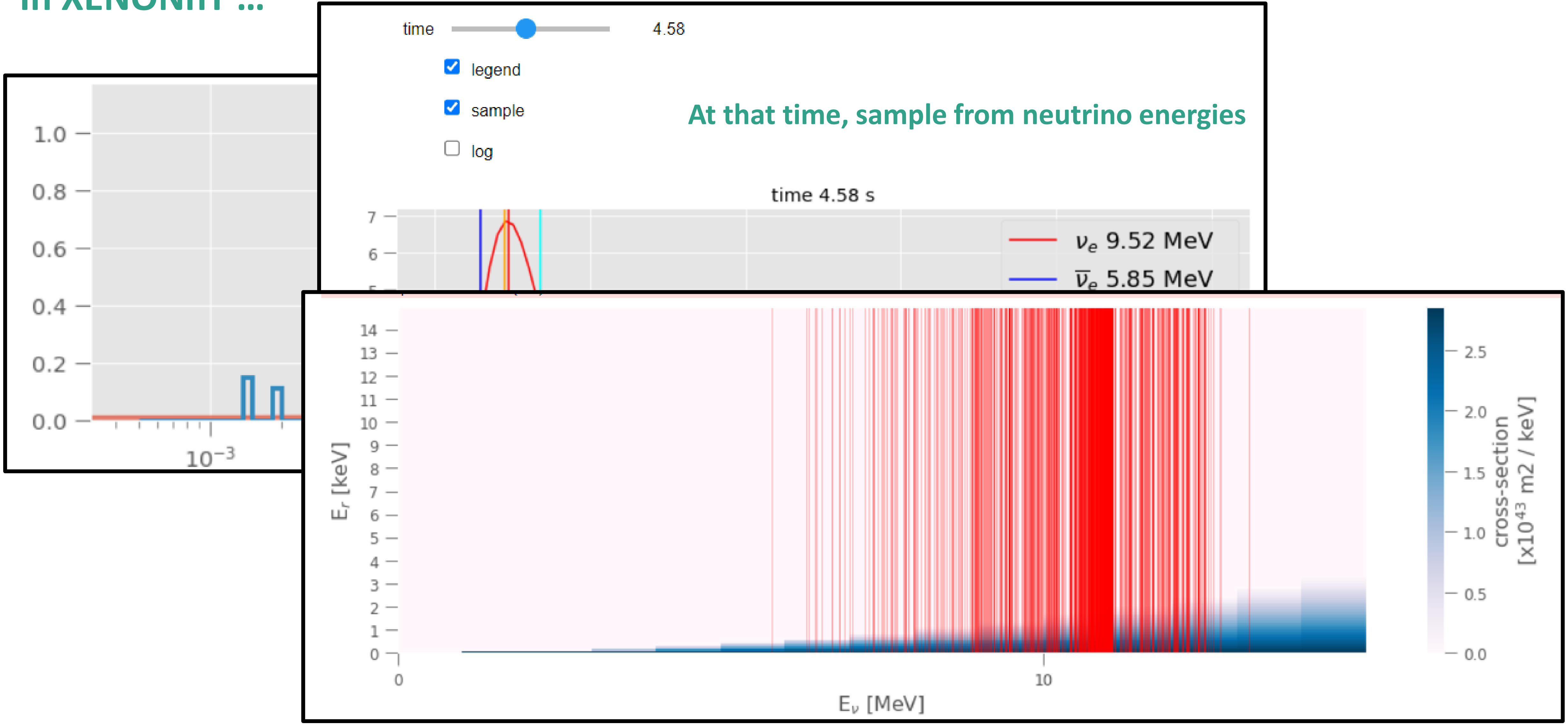
# Summary:

## - An efficient way of computing interaction rates

For any <u>snewpy</u> model
at any given <u>time</u>
for any given <u>recoil energy</u>
from any given <u>neutrino energy</u>
for any given <u>atom/isotope</u>
for all neutrino <u>flavors</u>

# Backup

# In XENONnT ...




At that time, sample from neutrino energies

At that neutrino energy sample a recoil energy based on cross-section probability

**This method actually gives you a terrible spectra**
**Regardless if you sample from time first (almost only returns first second)**
**Or sample time uniformly and look at the energy distributions.**

**Although you get those neutrinos more, they don't recoil energetic enough.**
**So you only see the high energy ones (and there are still a lot 10^12)**

**It makes more sense to study what we can see, and energy neutrino can detect those.**
**Then looking into flux distribution to see what models can generate enough of those events, and at what times**

**Sampling from time integrated recoil energy distribution is probably the best approximation to this.**