Documentation for sntools 1.4*

Jost Migenda†

May 3, 2024

Contents

*See `https://github.com/SNEWS2/sntools` for the most recent version.

†King's College London, `jost.migenda@kcl.ac.uk`

1 Introduction

sntools is a Monte Carlo event generator for supernova neutrino interactions.

It is a command line application that uses detailed time- and energy-dependent neutrino fluxes (provided by various supernova models) to generate interactions within the detector volume and write them to event files that can be used as an input for a full detector simulation. sntools was originally developed for Hyper-Kamiokande [1] and later extended to support different detectors and detector materials.

Additionally, sntools can be used as a Python library that implements neutrino cross sections. See section 2.3.

1.1 Installation Instructions

First, make sure you have Python 3.8 or higher installed on your computer. (If you don't, you can get it from python.org or as part of the Anaconda distribution.)

In a terminal, run `pip install sntools` to install the latest version of sntools and all dependencies. Then, run `python -c 'import sntools; sntools.setup()'` to check whether sntools is working and to download the sample input file used below.

1.2 Quick Start Guide

To generate your first supernova neutrino events with sntools, open a terminal window and type in the following command:

```
sntools fluxes/intp2001.data --format nakazato --endtime 50
```

This generates events for the first 50 ms of a supernova based on the neutrino flux given in the input file `fluxes/intp2001.data`[1] (which is in the `nakazato` format) and writes them to a file named `outfile.kin` in the current directory.

A more realistic usage that demonstrates more of sntools' capabilities looks like this:

```
sntools fluxes/intp2001.data --format nakazato --channel es --detector
SuperK --distance 20 --verbose --output intp2001es.kin
```

This uses the neutrino flux given in the same input file to generate neutrino-electron elastic scattering events in Super-Kamiokande for a supernova at 20 kpc distance. It also produces more verbose output, which lets you see that sntools generates events separately for ν_e, $\bar{\nu}_e$, ν_x and $\bar{\nu}_x$ (which have different fluxes and cross sections), before merging them into an output file named `intp2001es.kin`.

The input file is a required argument. The following optional arguments are available:

`--format <value>` Format of input file(s). See section 2.5.

`--output <value>` Name of the output file.

[1] This sample file, which is for a 20 $M_\odot$ progenitor from [2], is included as part of sntools. See `http://asphwww.ph.noda.tus.ac.jp/snn/` for more information.

`--mcformat <value>` Format of output file. Can be `NUANCE` (used e. g. by Hyper-Kamiokande), `RATPAC` (used e. g. by SNO+ and WATCHMAN) or `JUNO_ROOT` (used by the JUNO internal software).[2]

`--detector <value>` Detector configuration. See section 2.2.

`--channel <value>` Interaction channel to generate events for. See section 2.3.

`--transformation <value>` Transformation between initial flux (inside the supernova) and detected flux on Earth. See section 2.4.

`--distance <value>` Distance to supernova (in kpc).

`--starttime <value>` **and** `--endtime <value>` Generate events in a certain time range (in milliseconds). If these aren't specified, events are generated for the full time range for which fluxes are given in the input file.

`--randomseed <value>` Set an integer as a seed for the random number generator, to generate events reproducibly.

`--maxworkers <value>` Maximum number of parallel processes. sntools will generate events for up to this many channels in parallel, which may improve performance on multi-core CPUs.

`--verbose` Print more detailed output.

`--version` Print the current version number and exit.

The command `sntools -h` displays a more detailed overview over all arguments, including their default values.

2 More Details

An overview over the structure of sntools is given in figure 1.

The main user interface is provided by the file `genevts.py`, which parses command line arguments and then calls code in `channel.py` to generate events. Events are generated separately for each combination of interaction channel and input species.[3] The code first calculates the total number of events expected in each 1 ms bin, which is given by

$$N(t) = \iint \frac{\mathrm{d}\,\Phi(t, E_\nu)}{\mathrm{d}E_\nu} \frac{\mathrm{d}\,\sigma(E_\nu, E_e)}{\mathrm{d}E_e} \,\mathrm{d}E_e \,\mathrm{d}E_\nu, \tag{1}$$

[2] For a description of these formats see `http://neutrino.phy.duke.edu/nuance-format/` and `https://rat.readthedocs.io/en/latest/generators.html#external`. For more details on the JUNO SN-Event internal structure, contact `marta.colomer@ulb.be`.

[3] sntools uses the species ν_e, $\bar{\nu}_e$, ν_x and $\bar{\nu}_x$, with ν_x ($\bar{\nu}_x$) representing any one of ν_μ and ν_τ ($\bar{\nu}_\mu$ and $\bar{\nu}_\tau$). Since the energy of supernova neutrinos is too low to produce $\mu^\pm$ or $\tau^\pm$ in a detector, neutrinos with those two flavours interact in the same ways.

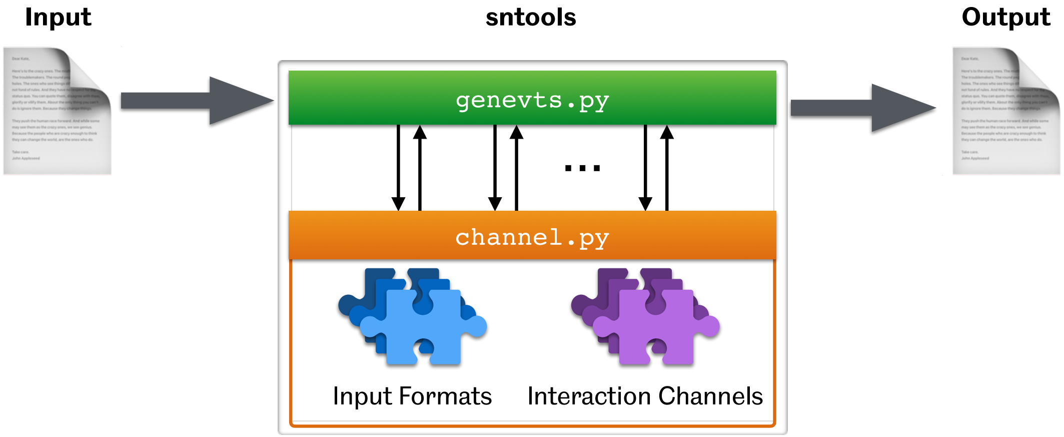

Figure 1: Overview over the structure of sntools. See text for a detailed description.

where $\Phi(t, E_\nu)$ is the neutrino flux and $\sigma(E_\nu, E_e)$ is the cross section of the current interaction channel. It then picks the actual number of events to generate within that time bin from a Poisson distribution with expectation value $N(t)$. Finally, it generates events by rejection sampling from the energy spectrum of neutrino interactions at that time and the distribution of outgoing particle directions.

The event generation code relies on a plug-in architecture to support various different input formats and interaction channels. Input format plug-ins provide functions that read in the data from an input file and return the number luminosity as a function of time and energy. Interaction channel plug-ins specify properties of the interaction channel, like the neutrino species that undergo this interaction, and provide functions to calculate quantities like the differential cross section $d\sigma(E_\nu, E_e)/dE_e$ or the kinematically allowed energy range. This modular design makes sntools easily extensible, with roughly 100 lines of code required to add a new input format or interaction channel.

Finally, `genevts.py` collects the events generated in all interaction channels and writes them to an output file, which can be used as input for a full detector simulation.

The rest of this chapter describes the various components of the code and the physics behind them in more detail.

2.1 Detector Materials

sntools currently supports three detector materials: water, liquid scintillator and water-based liquid scintillator. Water is assumed to consist of an oxygen-16 nucleus and two hydrogen nuclei (free protons). Generic liquid scintillator is assumed to consist of n carbon-12 nuclei and $2n$ hydrogen nuclei (free protons). In addition, two specific liquid scintillators[4] are implemented, which match the materials used in SNO+ [3] and JUNO. Water-based liquid scintillator is assumed to be a mixture of $x\,\%$ generic liquid scintillator and $(100 - x)\,\%$ water.

[4] both linear alkylbenzene (LAB), a mixture of $C_6H_5C_nH_{2n+1}$ molecules with $9 < n < 14$

The detector material doesn't need to be specified explicitly; it is determined by the detector configuration.

2.2 Detector Configurations

Select a detector configuration by using the `--detector <value>` command line option. The following detector configurations are currently implemented:

`HyperK` Inner detector of Hyper-Kamiokande [4]. Cylinder with a height of 65.8 m and a diameter of 64.8 m, filled with water.[5]

`SuperK` Inner detector of Super-Kamiokande [5]. Cylinder with a height of 36.2 m and a diameter of 33.7 m, filled with water.

`WATCHMAN` WATCHMAN detector [6]. Cylinder with a height and diameter of 12.8 m, filled with water.

`WATCHMAN-LS` Same as `WATCHMAN`, but filled with liquid scintillator.

`WATCHMAN-WbLS` Same as `WATCHMAN`, but filled with water-based liquid scintillator (3 % LS, 97 % water).

`THEIA25` THEIA-25 geometry [7]. Box with dimensions 20 m × 18 m × 70 m, filled with 25 kt of water-based liquid scintillator (10 % LS, 90 % water).

`THEIA100` THEIA-100 geometry [7]. Cylindrical detector filled with 100 kt of water-based liquid scintillator (10 % LS, 90 % water).

`SNOplusAV` Inner acrylic vessel of SNO+ [8]. Sphere with a radius of 6 m, filled with linear alkylbenzene (LAB) liquid scintillator.

`SNOplusEW` External water of SNO+ [8]. Two concentric spheres of radii 9 m and 6.05 m. The outer sphere is filled with water, the inner sphere is empty.

`JUNO` Central detector of JUNO [9]. Sphere with a radius of 17.7 m, filled with 20 kton of LAB liquid scintillator.

2.3 Interaction Channels

sntools supports multiple different interaction channels described in this section. By default, it will generate events across all channels that are available in the selected detector, but it can be restricted to a single channel by using the `--channel <value>` command line argument, where `<value>` can be one of `ibd`, `es`, `ps`, `o16e`, `o16eb`, `c12e`, `c12eb` and `c12nc`.

[5] In September 2019, the dimensions were changed slightly compared to the 2018 Design Report. The old configuration is available as `HyperKDR`.

For all supported channels, sntools includes code to calculate the differential cross sections, outgoing particle energy as a function of neutrino energy and other quantities. These could be used as a library from other Python code, e. g. as follows:

```
>> from sntools.interaction_channels import ibd
>> c = ibd.Channel(’eb’)
>> c.bounds_eE(eNu=20) # min/max energy of outgoing positron in MeV
[17.941220926704954, 18.70577898514169]
```

Type `help(ibd)` on the Python command line for documentation or see this SNOwGLoBES pull request for a full usage example.

2.3.1 `ibd`: Inverse Beta Decay

In both water and liquid scintillator, inverse beta decay (IBD; $\bar{\nu}_e + p \rightarrow n + e^+$) is the dominant interaction channel for supernova neutrinos due to its relatively high cross section and low energy threshold of $E_\nu^{\text{thr}} \approx 1.8\,\text{MeV}$, as well as the large number of free protons in the detector. The observed energy of IBD events is closely related to the neutrino energy, making this an excellent channel to reconstruct the $\bar{\nu}_e$ spectrum.

In sntools, IBD is implemented using the full tree-level cross section calculated in reference [10] and including radiative corrections based on the approximation from reference [11].[6]

2.3.2 `es`: Neutrino-Electron Scattering

In both water and liquid scintillator, elastic neutrino-electron scattering ($\nu + e^- \rightarrow \nu + e^-$) is a subdominant interaction channel due to its low cross section, which is only partially compensated by the large number of electrons in the detector. Elastic scattering events make up only a few per cent of all events but their angular distribution is strongly peaked into a forward direction, pointing away from the supernova. In detectors that can reconstruct the direction of scattered electrons, this channel can therefore be used to determine the direction of a supernova [12, 4].

Elastic scattering is sensitive to all neutrino flavours. However, the cross section of ν_e and $\bar{\nu}_e$, which can interact through both neutral and charged currents, is higher than that of ν_x and $\bar{\nu}_x$, which can interact only through neutral currents.

In sntools, elastic scattering is implemented using the tree-level cross sections from standard electroweak theory calculated by ’t Hooft [13] and including one-loop electroweak and QCD corrections as well as QED radiative corrections as calculated in reference [14].

2.3.3 `ps`: Neutrino-Proton Scattering

In liquid scintillator, neutrino-proton scattering ($\nu + p \rightarrow \nu + p$) is an additional available subdominant interaction channel. The recoiling proton energy spectrum is soft meaning

[6] That calculation uses the limit $m_e \rightarrow 0$. This approximation is accurate to better than 0.1 % above $E_e = 1\,\text{MeV}$.

the deposited energy in the detector is relatively low (< 5 MeV). These protons are invisible in water as they're below the Cherenkov threshold. Furthermore, the slow heavily ionizing protons lose energy very quickly, quenching the effective signal. Despite this, over a realistic detection threshold ($\mathcal{O}(100)$s keV), the neutrino-proton scattering yield can be the second largest available to liquid scintillator detectors below IBD.

Neutrino-proton scattering is a neutral-current interaction and is available to all (anti-)neutrino flavours. The proton recoil spectrum can also provide spectral information about the incoming neutrino, which could solve a long-standing problem of how to separately measure the total energy and temperature of ν_μ, ν_τ, $\bar{\nu}_\mu$, and $\bar{\nu}_\tau$ [15].

In sntools, the implementation of neutrino-proton scattering is based a prediction by Beacom, Farr, and Vogel [15]. Here the cross-section is calculated directly from the Standard Model, which has been extensively verified by GeV-scale experiments [16].

2.3.4 `o16e` and `o16eb`: Charged-Current Interactions on ^{16}O

In water, charged-current interactions of ν_e and $\bar{\nu}_e$ on ^{16}O nuclei,

$$\nu_e +^{16} \mathrm{O} \rightarrow e^- + X \tag{2}$$

$$\bar{\nu}_e +^{16} \mathrm{O} \rightarrow e^+ + X, \tag{3}$$

are a subdominant interaction channel.[7] Due to the high energy threshold of both interactions of approximately 15 MeV and 11 MeV, respectively, as well as the steep energy dependence of the cross sections, the number of events in each channel is a very sensitive probe of the high-energy tail of the supernova neutrino flux. It may vary by more than two orders of magnitude depending on the supernova models and oscillation scenario.

In sntools, the cross section for both interaction channels is based on a new shell model calculation [17] which selected 42 different nuclear states and calculated their respective partial cross sections. To simplify the evaluation of the cross section, the implementation in sntools uses the four-group fit presented in reference [18]. This fit matches the cross sections calculated from the full set of nuclear states to within a few per cent at neutrino energies of up to 100 MeV. For a typical supernova neutrino flux, the difference in the resulting event spectra when using the four groups instead of all 42 nuclear states is very small.

2.3.5 `c12e` and `c12eb`: Charged-Current Interactions on ^{12}C

In liquid scintillator, charged-current interactions of ν_e and $\bar{\nu}_e$ on ^{12}C nuclei,

$$\nu_e +^{12} \mathrm{C} \rightarrow e^- + X \tag{4}$$

$$\bar{\nu}_e +^{12} \mathrm{C} \rightarrow e^+ + X, \tag{5}$$

[7] Charged-current interactions of other neutrino species do not occur, since the energy of supernova neutrinos is too small to produce muons or τ leptons.

are a subdominant interaction channel.[8] Due to the high energy threshold of both interactions of approximately 17 MeV and 14 MeV, respectively, as well as the steep energy dependence of the cross sections, the number of events in each channel is a very sensitive probe of the high-energy tail of the supernova neutrino flux.

In sntools, the cross section for both interaction channels is based on the calculation in reference [19]. To interpolate between the tabulated cross sections, sntools uses the fit

$$\sigma(E_\nu) = \sigma_0 \cdot \left[a_1(E_\nu - E_{\text{thr}}) + a_2(E_\nu - E_{\text{thr}})^2 + a_3(E_\nu - E_{\text{thr}})^3\right], \tag{6}$$

where E_ν is the neutrino energy, E_{thr} is the energy threshold of the interaction and σ_0 and a_i are fit parameters. The energy of the outgoing lepton is approximated as $E_e = E_\nu - E_{\text{thr}}$.

2.3.6 `c12nc`: Neutral-Current Interactions on ^{12}C

In liquid scintillator, neutral-current interactions on ^{12}C nuclei,

$$\nu +^{12} \text{C} \rightarrow \nu' +^{12}\text{C}^* \tag{7}$$

$$^{12}\text{C}^* \rightarrow^{12} \text{C} + \gamma \tag{8}$$

are a subdominant interaction channel that neutrinos and antineutrinos of all flavours contribute to equally. In sntools, the cross section for this interaction channel is based on the theoretical calculation in [20]

$$\sigma(E_\nu) = 1.08 \times 10^{-38}\,\text{cm}^2 \left(\frac{E_\nu - E_{\text{thr}}}{m_N}\right)^2 \cdot \beta\kappa, \tag{9}$$

where E_ν is the neutrino energy, $E_{\text{thr}} \approx 15\,\text{MeV}$ is the energy threshold of the interaction, $m_N \approx 939\,\text{MeV}$ is the nucleon mass and $\beta\kappa = 1.11$ [21]. The energy of the outgoing γ is $E_\gamma = E_{\text{thr}}$.

2.4 Treatment of Neutrino Flavour Conversion

sntools implements different transformations between the original neutrino flux generated in the supernova and the flux detected on Earth, which can be selected by using the `--transformation <value>` command line argument.

In the simplest case, `NoTransformation`, no flavor transformation takes place. Apart from the geometrical factor $\frac{1}{4\pi d^2}$ (which depends only on the distance d of the supernova[9]), the flux of a neutrino species ν_i observed by a detector on Earth, Φ_{ν_i}, is identical to the flux originating within the supernova, $\Phi^0_{\nu_i}$, which is given by the input file(s).

[8] Footnote 7 applies here as well.

[9] This geometrical factor is always taken into account by sntools. For simplicity, I will omit it in the following.

2.4.1 Adiabatic MSW Effect

The two transformations `AdiabaticMSW_NMO` and `AdiabaticMSW_IMO` assume that neutrinos traverse a smoothly varying density profile[10] while exiting the star and undergo adiabatic flavour conversion via the MSW effect.

The resulting detected fluxes are linear combinations of the initial fluxes, which, for normal mass ordering (NMO), are given by [24]

$$\begin{aligned}
\Phi_{\nu_e} &= \sin^2\theta_{13}\cdot\Phi^0_{\nu_e} + \cos^2\theta_{13}\cdot\Phi^0_{\nu_x}\\
\Phi_{\bar\nu_e} &= \cos^2\theta_{12}\cos^2\theta_{13}\cdot\Phi^0_{\bar\nu_e} + (1-\cos^2\theta_{12}\cos^2\theta_{13})\cdot\Phi^0_{\bar\nu_x}\\
2\Phi_{\nu_x} &= \cos^2\theta_{13}\cdot\Phi^0_{\nu_e} + (1+\sin^2\theta_{13})\cdot\Phi^0_{\nu_x}\\
2\Phi_{\bar\nu_x} &= (1-\cos^2\theta_{12}\cos^2\theta_{13})\cdot\Phi^0_{\bar\nu_e} + (1+\cos^2\theta_{12}\cos^2\theta_{13})\cdot\Phi^0_{\bar\nu_x},
\end{aligned}\tag{10}$$

while for inverted mass ordering (IMO), they are

$$\begin{aligned}
\Phi_{\nu_e} &= \sin^2\theta_{12}\cos^2\theta_{13}\cdot\Phi^0_{\nu_e} + (1-\sin^2\theta_{12}\cos^2\theta_{13})\cdot\Phi^0_{\nu_x}\\
\Phi_{\bar\nu_e} &= \sin^2\theta_{13}\cdot\Phi^0_{\bar\nu_e} + \cos^2\theta_{13}\cdot\Phi^0_{\bar\nu_x}\\
2\Phi_{\nu_x} &= (1-\sin^2\theta_{12}\cos^2\theta_{13})\cdot\Phi^0_{\nu_e} + (1+\sin^2\theta_{12}\cos^2\theta_{13})\cdot\Phi^0_{\nu_x}\\
2\Phi_{\bar\nu_x} &= \cos^2\theta_{13}\cdot\Phi^0_{\bar\nu_e} + (1+\sin^2\theta_{13})\cdot\Phi^0_{\bar\nu_x}.
\end{aligned}\tag{11}$$

In both cases, the factor of 2 in the last two equations accounts for the fact that ν_x ($\bar\nu_x$) includes the fluxes of ν_μ and ν_τ ($\bar\nu_\mu$ and $\bar\nu_\tau$). These equations assume purely adiabatic transition (corresponding to $P_H = 0$ in [24, 23]) as explained below. Values for θ_{12} and θ_{13} are taken from the Particle Data Group [25].

In cases where the detected flux is a mixture of original fluxes of different species, sntools generates events for each original species separately with the appropriate weighting factor applied. For example, when generating inverse beta decay events using the `AdiabaticMSW_NMO` transformation, sntools will generate events first using the input flux $\Phi_{\bar\nu_e} = \cos^2\theta_{12}\cos^2\theta_{13}\cdot\Phi^0_{\bar\nu_e}$ and then using the input flux $\Phi_{\bar\nu_e} = (1-\cos^2\theta_{12}\cos^2\theta_{13})\cdot\Phi^0_{\bar\nu_x}$, before finally combining both sets of events into one output file.

2.4.2 Additional Transformations

sntools also supports the following flavour transformations, which are implemented by SNEWPY [26]: `CompleteExchange`, `NonAdiabaticMSWH`, `TwoFlavorDecoherence_NMO`, `TwoFlavorDecoherence_IMO`, `ThreeFlavorDecoherence`. See reference [27] for a full description. To use these transformations in sntools, prefix them with `SNEWPY-`. For

[10] After the accretion phase, the revived shock front travels outwards and passes through the layer within the star where the adiabatic flavour conversion takes place. This causes a sudden change in the matter and electron density and can severely impact the flavour conversion processes [22]. Since this occurs after the shock wave is revived, it mainly affects the late-time part of the supernova neutrino signal (e. g. for the Totani model, this effect is expected to become relevant more than 1 s after core-bounce [23]). Many simulations only cover the first few 100 ms and would thus likely not exhibit this effect; making the assumption of a smoothly varying density profile reasonable.

example, to apply three flavor decoherence use sntools with the command line argument `--transformation SNEWPY-ThreeFlavorDecoherence`.

Some additional flavour transformations, like `NeutrinoDecay`, are implemented in SNEWPY but require some more work to be supported by sntools. If you are interested in these, please join the discussion on GitHub!

2.5 Input Formats

sntools supports multiple different input formats for the neutrino fluxes from a simulation. To select one, use the `--format <value>` command line argument, where `<value>` can be one of `gamma`, `warren2020`, `nakazato`, `princeton` or `totani`. This section will briefly describe each format and the processing steps necessary to calculate the spectral number luminosity.

In addition, sntools supports the following input formats, which are implemented by SNEWPY [26, 27]: `Bollig_2016`, `Fornax_2021`, `Fornax_2022`, `Kuroda_2020`, `Mori_2023`, `Nakazato_2013`, `OConnor_2015`, `Sukhbold_2015`, `Tamborra_2014`, `Walk_2018`, `Walk_2019` and `Zha_2021`. SNEWPY also lets you download neutrino flux files from each of the corresponding simulations. See SNEWPY documentation for instructions. To use these flux files in sntools, prefix them with `SNEWPY-`. For example, to use an `OConnor_2015` input file, use sntools with the command line argument `--format SNEWPY-OConnor_2015`.

All formats contain separate information on the three species ν_e, $\bar{\nu}_e$ and ν_x. The fluxes of ν_x and $\bar{\nu}_x$ are typically assumed to be equal and most simulations don't provide separate fluxes for ν_x and $\bar{\nu}_x$. For simplicity, references to each species are omitted in the following.

2.5.1 Gamma Format

Files in this format contain, for each time step t_n, the luminosity L, mean energy $\langle E_\nu \rangle$ and mean squared energy $\langle E_\nu^2 \rangle$ of neutrinos. A sample file is provided under `fluxes/sample-gamma.txt`.

To reconstruct the neutrino spectrum from this, we assume that it is described by a normalized Gamma distribution [28, 29] given by

$$f(E_\nu) = \frac{E_\nu^\alpha}{\Gamma(\alpha+1)} \left(\frac{\alpha+1}{A} \right)^{\alpha+1} \exp\left[-\frac{(\alpha+1)E_\nu}{A} \right]. \tag{12}$$

In this formula, A is an energy scale, while α determines the shape of the distribution: $\alpha = 2$ corresponds to a Maxwell-Boltzmann distribution, while $\alpha > 2$ corresponds to a "pinched" spectrum, which is more typical for neutrino spectra from supernovae.

The first two energy moments of the distribution are

$$\langle E_\nu \rangle = \int_0^\infty \mathrm{d}E_\nu \, E_\nu f(E_\nu) = A \tag{13}$$

$$\langle E_\nu^2 \rangle = \int_0^\infty \mathrm{d}E_\nu \, E_\nu^2 f(E_\nu) = \frac{\alpha+2}{\alpha+1} A^2, \tag{14}$$

and therefore,

$$\alpha = \frac{\langle E_\nu^2 \rangle - 2\langle E_\nu \rangle^2}{\langle E_\nu \rangle^2 - \langle E_\nu^2 \rangle}. \tag{15}$$

Thus, the shape of the spectral number luminosity is uniquely determined by the mean energy $\langle E_\nu \rangle$ and the mean squared energy $\langle E_\nu^2 \rangle$, while the normalization is provided by $L/\langle E_\nu \rangle$. To determine the spectral number luminosity at arbitrary times, each of the three parameters is interpolated separately before calculating the spectral number luminosity using the interpolated values.

2.5.2 Warren2020 Format

Files in this format contain, for each time step t_n, the luminosity L, mean energy $\langle E_\nu \rangle$ and RMS energy $\sqrt{\langle E_\nu^2 \rangle}$ of neutrinos in the HDF5 format. After reading data from the input file, it is processed as described in section 2.5.1.

2.5.3 Nakazato Format

Files in this format[11] contain, for 20 energy bins E_k during each time step t_n, the quantities $\Delta N_k(t_n)/\Delta E_k$ and $\Delta L_k(t_n)/\Delta E_k$, which reflect the number luminosity and luminosity at those energies, respectively. For each energy bin, sntools calculates the mean energy within that bin, which is given by

$$\langle E_k \rangle = \frac{\frac{\Delta L_k(t_n)}{\Delta E_k}}{\frac{\Delta N_k(t_n)}{\Delta E_k}}, \tag{16}$$

and set the differential neutrino number flux at that energy to $\Delta N_k(t_n)/\Delta E_k$. Finally, a linear interpolation in time and cubic spline interpolation in energy are used to determine the spectral number luminosity at an arbitrary time and energy.

2.5.4 Princeton Format

Files in this format contain, for each time step t_n, the spectral luminosity $\mathrm{d}L/\mathrm{d}E$ for 20 logarithmically spaced energy bins E_k. I divide this by the central energy $\sqrt{E_k E_{k+1}}$ of the respective bin to get the spectral number luminosity at that energy. Finally, a linear interpolation in time and cubic interpolation in energy are used to determine the spectral number luminosity at an arbitrary time and energy.

This follows the procedure described in reference [30]. It is similar to that used for the Nakazato format described above, though with a different definition of the bin energy.

[11] See `http://asphwww.ph.noda.tus.ac.jp/snn/guide.pdf` for more details.

2.5.5 Totani Format

Files in this format contain, for each time step t_n, the total number of neutrinos emitted until that time, which makes it possible to calculate the number N_n of neutrinos emitted since the previous time step.

For 20 energy bins E_k per time step, a quantity X_k is provided, which is proportional to the number of neutrinos emitted during that time step and in that energy bin. I divide this by the width of each energy bin to get

$$X_k^{\text{spec}} = \frac{X_k}{E_{k+1} - E_k}, \tag{17}$$

which is proportional to the spectral number emission during that time step.

Integrating X_k^{spec} over all energy bins and dividing it by that integral gives the spectral number emission during that time step normalized to 1, X_k^{norm}. The spectral number luminosity at time t_n and energy E_k is given by

$$\frac{\mathrm{d}\,\mathrm{NL}(t_n, E_k)}{\mathrm{d}E} = \frac{N_n}{t_n - t_{n-1}} \cdot X_k^{\text{norm}}. \tag{18}$$

Finally, a linear interpolation in time and log cubic spline interpolation in energy are used to determine the spectral number luminosity at an arbitrary time and energy.[12]

3 Getting Help and Contributing

To report problems or ask for help, open an issue on GitHub or email the lead developer (see title page of this document).

sntools is designed to be versatile and support many different detectors and input formats. **Your contributions are welcome!** To help extend sntools, follow these steps:

- Clone the GitHub repository to your local computer.
- Edit the code—see the subsections below for help with the most common ways to extend sntools.
- Use `pip install .` to install the modified code; then try it out.
- To test your changes, run `python -m unittest discover` from the top-level directory of the repository. (Tests will also run automatically whenever you submit a pull request on GitHub.)
- Finally, please submit a pull request with your contributions!

sntools is released under a BSD license (see `LICENSE.md`) and with a Contributor Code of Conduct (see `CODE_OF_CONDUCT.md`). By participating in this project you agree to abide by its terms.

[12] This approach closely follows the one used in code provided by Totani. There is excellent agreement of the calculated fluxes between Totani's code and sntools, with differences of at most a few per mille due to slight differences in the numerical interpolation algorithms used.

3.1 Add a New Detector Configuration

In `detectors.py`:

- Add the name of your detector configuration to the list `supported_detectors`.
- Add an `elif` clause for your detector in the `__init__` function of the `Detector` class and set the detector dimensions and material there.

If your new detector configuration uses a detector material that is not yet implemented, see section 3.2.

3.2 Add a New Detector Material

In `detectors.py`, create a new dictionary for your detector material which contains three entries:

- `molecular_weight`: Molecular weight of the material in g/mol.
- `density`: Density of the material in g/cm^3.
- `channel_weight`: Dictionary with supported interaction channels as keys and their respective weighting factors as values. The weighting factor is the number of potential targets per molecule. For example, water has 2 protons and 10 electrons per molecule, so the weighting factor is 2 for `ibd` and 10 for `es`.

If necessary, add new interaction channels which may occur in this detector material (see section 3.3). Finally, add a new detector configuration which uses this detector material (see section 3.1).

3.3 Add a New Interaction Channel

- In the `interaction_channels/` folder, create a new file for your interaction channel. It must define two things:
  - A list of neutrino flavors that can interact in this channel (e.g. if all flavors interact in your channel: `possible_flavors = ["e", "eb", "x", "xb"]`)
  - A class named `Channel`, which inherits from the `BaseChannel` class defined in `interaction_channels/__init__.py`. Overwrite all functions and properties that are marked `@abstractmethod` in the `BaseChannel` class to implement the relevant physics. (See docstrings for explanations.)
- In `genevts.py`, add the channel in the `parse_command_line_options()` function.
- In `detectors.py`, add the channel to all relevant detector materials.

3.4 Add a New Input Format

First, check if it is straightforward to transform the input files into one of the supported formats described in section 2.5. If not:

- In the `formats/` folder, create a new file for your interaction channel. It must contain a class named `Flux`, which inherits from the `BaseFlux` class defined in `formats/__init__.py`. Overwrite all functions that are marked `@abstractmethod` in the `BaseFlux` class. (See docstrings for explanations.)

- In `genevts.py`, add the format in the `parse_command_line_options()` function.

References

[1] Jost Migenda. *Supernova Model Discrimination with Hyper-Kamiokande.* PhD thesis, University of Sheffield, 2019. `arXiv:2002.01649`.

[2] Ken'ichiro Nakazato, Kohsuke Sumiyoshi, Hideyuki Suzuki, Tomonori Totani, Hideyuki Umeda, and Shoichi Yamada. Supernova Neutrino Light Curves and Spectra for Various Progenitor Stars: From Core Collapse to Proto-neutron Star Cooling. *Astrophys. J. Suppl.*, 205:2, 2013.

[3] MR Anderson, S Andringa, L Anselmo, E Arushanova, S Asahi, M Askins, DJ Auty, AR Back, Z Barnard, N Barros, et al. Development, characterisation, and deployment of the sno+ liquid scintillator. *Journal of Instrumentation*, 16(05):P05009, 2021.

[4] K. Abe et al. Hyper-Kamiokande Design Report. 2018. `arXiv:1805.04163`.

[5] S. Fukuda et al. The Super-Kamiokande detector. *Nucl. Instrum. Meth.*, A501:418–462, 2003.

[6] M. Askins, M. Bergevin, A. Bernstein, S. Dazeley, S. T. Dye, T. Handler, A. Hatzikoutelis, D. Hellfeld, P. Jaffke, Y. Kamyshkov, B. Land, J. G. Learned, P. Marleau, C. Mauger, G. D. Orebi Gann, C. Roecker, S. D. Rountree, T. M. Shokair, M. Smy, R. Svoboda, M. Sweany, M. Vagins, K. A. van Bibber, R. B. Vogelaar, M. J. Wetstein, and M. Yeh. The Physics and Nuclear Nonproliferation Goals of WATCHMAN: A WAter CHerenkov Monitor for ANtineutrinos. 2 2015. `arXiv:1502.01132`.

[7] M. Askins, Z. Bagdasarian, N. Barros, E. W. Beier, E. Blucher, R. Bonventre, E. Bourret, E. J. Callaghan, J. Caravaca, M. Diwan, S. T. Dye, J. Eisch, A. Elagin, T. Enqvist, V. Fischer, K. Frankiewicz, C. Grant, D. Guffanti, C. Hagner, A. Hallin, C. M. Jackson, R. Jiang, T. Kaptanoglu, J. R. Klein, Yu. G. Kolomensky, C. Kraus, F. Krennrich, T. Kutter, T. Lachenmaier, B. Land, K. Lande, J. G. Learned, V. Lozza, L. Ludhova, M. Malek, S. Manecki, J. Maneira, J. Maricic, J. Martyn, A. Mastbaum, C. Mauger, F. Moretti, J. Napolitano, B. Naranjo, M. Nieslony, L. Oberauer, G. D. Orebi Gann, J. Ouellet, T. Pershing, S. T. Petcov, L. Pickard, R. Rosero, M. C. Sanchez, J. Sawatzki, S. H. Seo, M. Smiley, M. Smy, A. Stahl, H. Steiger, M. R. Stock, H. Sunej, R. Svoboda, E. Tiras, W. H. Trzaska, M. Tzanov, M. Vagins, C. Vilela, Z. Wang, J. Wang, M. Wetstein, M. J. Wilking, L. Winslow, P. Wittich, B. Wonsak, E. Worcester, M. Wurm, G. Yang, M. Yeh, E. D. Zimmerman, S. Zsoldos, and K. Zuber. THEIA: An advanced optical neutrino detector. *The European Physical Journal C*, 80(5):416, 2020.

[8] V Albanese, R Alves, MR Anderson, S Andringa, L Anselmo, E Arushanova, S Asahi, M Askins, DJ Auty, AR Back, et al. The sno+ experiment. *Journal of Instrumentation*, 16(08):P08059, 2021.

[9] Angel Abusleme et al. JUNO physics and detector. *Prog. Part. Nucl. Phys.*, 123:103927, 2022.

[10] A. Strumia and F. Vissani. Precise quasielastic neutrino/nucleon cross-section. *Phys. Lett.*, B564:42–54, 2003.

[11] A. Kurylov, M. J. Ramsey-Musolf, and P. Vogel. Radiative corrections to low-energy neutrino reactions. *Phys. Rev.*, C67:035502, 2003.

[12] K. Abe et al. Real-Time Supernova Neutrino Burst Monitor at Super-Kamiokande. *Astropart. Phys.*, 81:39–48, 2016.

[13] Gerard 't Hooft. Predictions for neutrino - electron cross-sections in Weinberg's model of weak interactions. *Phys. Lett.*, 37B:195–196, 1971.

[14] John N. Bahcall, Marc Kamionkowski, and Alberto Sirlin. Solar neutrinos: Radiative corrections in neutrino - electron scattering experiments. *Phys. Rev.*, D51:6146–6158, 1995.

[15] John F Beacom, Will M Farr, and Petr Vogel. Detection of supernova neutrinos by neutrino-proton elastic scattering. *Physical Review D*, 66(3):033001, 2002.

[16] LA Ahrens, SH Aronson, PL Connolly, BG Gibbard, MJ Murtagh, SJ Murtagh, S Terada, DH White, JL Callas, D Cutts, et al. Measurement of neutrino-proton and antineutrino-proton elastic scattering. *Physical Review D*, 35(3):785, 1987.

[17] Toshio Suzuki, Satoshi Chiba, Takashi Yoshida, Koh Takahashi, and Hideyuki Umeda. Neutrino-nucleus reactions on ^{16}O based on new shell-model Hamiltonians. *Phys. Rev.*, C98(3):034613, 2018.

[18] Ken'ichiro Nakazato, Toshio Suzuki, and Makoto Sakuda. Charged-current scattering off ^{16}O nucleus as a detection channel for supernova neutrinos. *PTEP*, 2018(12):123E02, 2018.

[19] E. Kolbe, K. Langanke, and P. Vogel. Weak reactions on C-12 within the continuum random phase approximation with partial occupancies. *Nucl. Phys. A*, 652:91–100, 1999.

[20] T.W. Donnelly and R.D. Peccei. Neutral Current Effects in Nuclei. *Phys. Rept.*, 50:1, 1979.

[21] B Armbruster et al. Measurement of the weak neutral current excitation C-12(nu(mu) nu'(mu))C*-12(1+,1,15.1-MeV) at E(nu(mu)) = 29.8-MeV. *Phys. Lett. B*, 423:15–20, 1998.

[22] Richard C. Schirato and George M. Fuller. Connection between supernova shocks, flavor transformation, and the neutrino signal. 2002. `arXiv:astro-ph/0205390`.

[23] G. L. Fogli, E. Lisi, A. Mirizzi, and D. Montanino. Probing supernova shock waves and neutrino flavor transitions in next-generation water-Cerenkov detectors. *JCAP*, 0504:002, 2005.

[24] Amol S. Dighe and Alexei Yu. Smirnov. Identifying the neutrino mass spectrum from the neutrino burst from a supernova. *Phys. Rev.*, D62:033007, 2000.

[25] P.A. Zyla et al. Review of particle physics. *PTEP*, 2020(08):083C01, 2020.

[26] Amanda L. Baxter, Segev BenZvi, Joahan Castaneda Jaimes, Alexis Coleiro, Marta Colomer Molla, Damien Dornic, Spencer Griswold, Tomer Goldhagen, Anne Graf, Alec Habig, Remington Hill, Shunsaku Horiuchi, James P. Kneller, Mathieu Lamoureux, Rafael F. Lang, Massimiliano Lincetto, Jost Migenda, McKenzie Myers, Evan O'Connor, Andrew Renshaw, Kate Scholberg, Andrey Sheshukov, Jeff Tseng, Christopher Tunnell, Navya Uberoi, and Arkin Worlikar. SNEWPY: A Data Pipeline from Supernova Simulations to Neutrino Signals. *Journal of Open Source Software*, 6(67):03772, 2021.

[27] Amanda L. Baxter, Segev BenZvi, Joahan Castaneda Jaimes, Alexis Coleiro, Marta Colomer Molla, Damien Dornic, Spencer Griswold, Tomer Goldhagen, Anne Graf, Alec Habig, Remington Hill, Shunsaku Horiuchi, James P. Kneller, Rafael F. Lang, Massimiliano Lincetto, Jost Migenda, Ko Nakamura, Evan O'Connor, Andrew Renshaw, Kate Scholberg, Navya Uberoi, and Arkin Worlikar. SNEWPY: A Data Pipeline from Supernova Simulations to Neutrino Signals. 2021. `arXiv:2109.08188`.

[28] M. T. Keil, Georg G. Raffelt, and H.-Th. Janka. Monte carlo study of supernova neutrino spectra formation. *Astrophys. J.*, 590:971–991, 2003.

[29] Irene Tamborra, B. Müller, L. Hüdepohl, H.-Th. Janka, and Georg G. Raffelt. High-resolution supernova neutrino spectra represented by a simple fit. *Phys. Rev.*, D86(125031), 2012.

[30] Shaquann Seadrow, Adam Burrows, David Vartanyan, David Radice, and M. Aaron Skinner. Neutrino Signals of Core-Collapse Supernovae in Underground Detectors. *Mon. Not. Roy. Astron. Soc.*, 480(4):4710–4731, 2018.