Signal Monte-Carlo generation for supernova relic neutrinos

Sonia El Hedri

June 6, 2019

We need to generate signals involving a low energy positron (according to a signal-dependent spectrum) and a 2.2 MeV photon (resulting from neutron capture). The general procedure is the following:

- **Input spectra:** We first need to generate random values of the positron energy according to a signal-dependent spectrum. These energies will then be given to code that generates ZBS files. Here I describe how to generate positron energies for relic supernova neutrinos.
- **ZBS files:** Generate ZBS files containing time information and run numbers. These files contain a fixed number of events for each subrun. The ZBS generation code is interfaced with the model-dependent code that gives the energy of each event.
- **SKDetSim:** Run SKDetSim using the ZBS files of the previous step as an input. SKDetSim will output either ZBS or ROOT files depending on whether the ROOT files contain enough information for the lowe reconstruction.
- **Lowe reconstruction:** Run the standard event reconstruction algorithms for low energy neutrinos on the SKDetSim output. Outputs ROOT files.
- **Random trigger background:** Superimpose random trigger background events for the AFT trigger window.

The main codes are in

/home/elhedri/SK2p2MeV/mc/generate/vector/updated_mc/

1 IBD relic neutrino spectrum

The code is the C++ file snrate.cpp. It computes the SRN spectrum according to the procedure described in [1] and uses the Strumia-Vissani IBD cross-section [2]. The spectral parameters are given in an input file. The parameters common to all models are:

- `type`: the spectrum type (blackbody, pinched, pinched with different spectra for neutron star and black hole forming supernovae).
- `lumi`: luminosity for electron antineutrinos (before flavor oscillations) and neutron-star-forming supernovae. If the model does not take the flavor dependency and the NS/BH discrepancy into account, this is just the electron antineutrino luminosity.

Right now, four models are implemented:

- **Blackbody spectrum**: the spectrum of the electron antineutrinos is a Fermi-Dirac spectrum with temperature T_ν. This model therefore requires an additional parameter

  `tnu`: electron antineutrino temperature

- **Pinched spectrum**: this parameterization takes into account the fact that the neutrinos with higher energies take longer to travel out of the proto-neutron-star. The spectrum can be approximated analytically:

$$\Phi(E_\nu) = \frac{L.(1+\alpha)^{1+\alpha}}{\Gamma(1+\alpha)\langle E_\nu\rangle^2}\left(\frac{E_\nu}{\langle E_\nu\rangle}\right)^2 \exp\left[-(1+\alpha)\frac{E_\nu}{\langle E_\nu\rangle}\right]$$

  and we have two specific parameters

  `emean`: the mean energy $\langle E_{\bar{\nu}_e}\rangle$

  `alpha`: the pinching parameter α

- **Pinched spectrum with MSW effect**: for the DSNB, it is a good approximation to consider that neutrino oscillations inside the star are dominated by the MSW effect. These oscillations can therefore be simply parameterized by a survival probability:

$$\Phi_{\bar{\nu}_e} \to \bar{p}\Phi_{\bar{\nu}_e} + (1-\bar{p})\Phi_{\bar{\nu}_x}.$$

  This probability $\bar{p}$ can in fact take two values: 0 for the inverted hierarchy, 0.68 for the normal hierarchy. So, aside from the pinched spectra parameters, we need

  `pbar`: survival probability $\bar{p}$

  `lumX`: ratio of heavy flavor antineutrino luminosity over electron neutrino luminosity

- **Pinched spectrum with MSW effect and BH/NS discrepancy**: supernovae that form a black hole typically have higher accretion rates than supernovae that form neutron stars. In simulations, this leads to higher luminosities and higher energies for electron (anti)neutrinos for black hole forming supernovae. So now, we need

  `lumE_BH`: ratio of the electron antineutrino luminosity for BH forming supernovae over the one for NH forming supernovae

`lumX_NS`: ratio of the heavy flavor antineutrino luminosity for NS forming supernovae over the one for electron antineutrinos and NS forming supernovae

`lumX_BH`: ratio of the heavy flavor antineutrino luminosity for BH forming supernovae over the one for electron antineutrinos and NS forming supernovae

`emeanE_NS`: mean energy for electron antineutrinos and NS forming supernovae

`emeanE_BH`: mean energy for electron antineutrinos and BH forming supernovae

`emeanX_NS`: mean energy for heavy flavor antineutrinos and NS forming supernovae

`emeanX_BH`: mean energy for heavy flavor antineutrinos and BH forming supernovae

`fBH`: fraction of supernovae that end up forming black holes

In practice, for the more complicated models, some of these parameters will be equal to each other or fixed according to results from simulations.

2 ZBS file generation

The main code for ZBS file generation is `vectgen.F` and should be ran using the following command:

```
./vectgen <run number> <random seed file> <ZBS output>
```

The random seed file is generated by

```
./make_random <random seed file>
```

and is just a text file with the seed. I assume we do this for reproducibility. The vector generation file iterates from the first run given to the last run given. For each run, it obtains the timing information (from a separate file) and stores it in the `IDATA` table, ultimately saved into the ZBS file and read by SKDetSim. For each subrun, it generates a fixed number of events, given by the variable `num`, set inside the code. At each iteration, the `spectrum` function is called to generate the following event parameters:

- A random positron energy, using the SRN code described in the previous section
- A random position for the positron in the fiducial volume
- A vertex for neutron capture, at most **5 cm** away from the positron vertex
- Random directions for the positron and the photon

The energy of the photon is set to 2.225 MeV and the neutron capture time is set to 200 μs. After setting the event parameters, `spectrum` fills in the corresponding tables for

the ZBS output. Then, `vectgen` fills in `IDATA` with the run and timing information for each event.

The command to generate ZBS files for all SK-IV runs is

```
scripts/qsub_vectgen.sh -f <first run> -l <last run> -s [spectral file]
```

that submits the jobs to the cluster for a range of runs. The `-s` option allows to specify an input file for the spectral parameters. This file is assumed to be in the `spectrum_input/` folder. Please do not put dots or slashes in the name!

3 SKDetSim, reconstruction, T2K dummy background

3.1 SKDetSim + lowe reconstruction

We use the most recent version of SKDetSim, with the `NEUTRON` compiler option turned on in the `GNUMakefile`. This option allows SKDetSim to use a larger time window (535 μs) to generate the positron and the 2.2 MeV gamma in the same event. It also generates the dark noise only up to 18 μs so that the random trigger background can be superimposed on the rest of the event without double counting. We take the input card from

```
/home/sklowe/solar_apr19/gen_mc_gain/supersim_root.card
```

In this card, only the random seeds need to be updated, as the run-dependency is already specified in the ZBS file.

The output of SKDetSim is a skroot file that is then passed to `lowfit_sk4_gain_corr_mc`, a code used for the solar neutrino reconstruction (see `/home/sklow/solar_apr19/gen_mc_gain/`) that calls the SKOFL code `lfallfit_sk4_gain_corr`. One potential problem: this code does not distinguish between positron and neutron hits...we might want to sort this out.

The command to run SKDetSim and the lowe reconstruction on the cluster after all the ZBS vector files have been generated is

```
scripts/qsub_skdetsim.sh <first run> <last run>
```

3.2 Random trigger background

The code to superimpose random trigger background on a skroot file is in

```
/home/elhedri/SK2p2MeV/mc/generate/combine/incorporate.C
```

and the command to run it is

```
./incorporate <output file> <input file> <run number>
```

For each run, the background injected corresponds to random trigger data taken from the same run. In order to use these Monte-Carlo simulations for neutron tagging later we add a tag for each hit after 18 μs to specify whether it is signal or background. The array of tags is contained in the `issignal` branch of the skroot tree and has to be matched with the `TQREAL` branch.

References

[1] S. Horiuchi, J. F. Beacom and E. Dwek, Phys. Rev. D **79**, 083013 (2009) doi:10.1103/PhysRevD.79.083013 [arXiv:0812.3157 [astro-ph]].

[2] A. Strumia and F. Vissani, Phys. Lett. B **564**, 42 (2003) doi:10.1016/S0370-2693(03)00616-6 [astro-ph/0302055].