Sterile neutrinos and non-standard neutrino interactions in GLoBES

Joachim Kopp

v1.0 (Nov 2010)

Sterile neutrinos and non-standard neutrino interactions (NSI) are two of the most generic types of new physics possible in the neutrino sector. For an introduction to the formalism of NSI, see e.g. [3].

There are several implementations of NSI in GLoBES (e.g. the MonteCubes plugin [1], for instance, can handle NSI), but to the best of my knowledge, there is no publicly available implementation of sterile neutrinos. The code discussed here can handle both: It works with up to 9 neutrino flavors, and can handle NSI in the production and detection processes as well as non-standard matter effects (NSI in propagation). The non-standard operators can affect the active neutrino species as well as the sterile ones.

To use the code, include `snu.c` in your project by modifying your `Makefile` accordingly, and `#include` the header file `snu.h` in your source code.

The first step is to initialize the sterile neutrino/NSI engine. If you only need NSI, but no sterile neutrinos, you may use the simple command

```
snu_init_probability_engine();
```

On the other hand, if you need sterile neutrinos, you need to specify how many neutrino species you would like to include, and how the mixing matrix is defined. The syntax for this is

```
int snu_init_probability_engine(int n_flavors,
  int rotation_order\left[2], int phase_order[]);
```

The arguments are

`n_flavors`	The number of neutrino flavors. Must be ≥ 3. Note that for `n_flavors` $= 3$, the optimized algorithms from [2] are used to diagonalize the Hamiltonian, while for more than four flavors, the GNU Scientific Library is used.

`rotation_order`, `phase_order`	Specifies how the neutrino mixing matrix U is composed of individual rotation matrices, and which of the rotation matrices carry the complex phases. A definition of the form

`rotation_order` $= \{\{i_1,\, j_1\},\, \{i_2,\, j_2\},\, \dots\}$
`phase_order` $= \{k_1, k_2, \dots\}$

means $U = R(\theta_{i_1 j_1}, \delta_{k_1}) R(\theta_{i_2 j_2}, \delta_{k_2}) \cdots$, where

$$R(\theta_{ij}, \delta_k) = \begin{pmatrix} 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos\theta_{ij} & \cdots & e^{i\delta_k}\sin\theta_{ij} & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & -e^{-i\delta_k}\sin\theta_{ij} & \cdots & \cos\theta_{ij} & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \end{pmatrix}$$

is a rotation matrix in the ij-sector with complex phase δ_k. The indices i and j run from 1 to `n_flavors`. Both `rotation_order` and `phase_order` should have `n_flavors`(`n_flavors` $- 1)/2$ entries. Since there are only (`n_flavors` $- 1$)(`n_flavors` $- 2)/2$ phases, some of the entries of `phase_order` should be -1, indicating that the respective rotation matrices will be treated as real. In the above syntax, the standard three-flavor mixing matrix $U_{3\times3} = R(\theta_{23}, 0) R(\theta_{13}, \delta) R(\theta_{12}, 0)$ would be given by

`rotation_order` = {{2, 3}, {1, 3}, {1, 2} }
`phase_order` = {-1, 0, -1} .

After initializing the sterile neutrino/NSI engine, you have to make it known to GLoBES by calling

```
glbRegisterProbabilityEngine(6*SQR(n_flavors) - n_flavors,
                             &snu_probability_matrix,
                             &snu_set_oscillation_parameters,
                             &snu_get_oscillation_parameters,
                             NULL);
```

Note the expression for the number of oscillation parameters in the first line. As with any new probability engine, the call to `glbRegisterProbabilityEngine` has to occur before any calls to `glbAllocParams` or `glbAllocProjection` to make sure that all parameter and projection vectors have the correct length.

It is recommended to use `glbSetParamName` to assign human-readable names to the oscillation parameters by which they can be referred to in subsequent calls to `glbSetOscParamByName` and `glbSetProjectionFlagByName`. A list of suitable names is provided in the global array `char snu_param_strings[][64]`. The syntax is `TH12`, `TH13`, `TH14`, etc. for the mixing angles, `DELTA_0`, `DELTA_1`, ... for the CP phases, `ABS_EPS_`c`_`$\alpha\beta$ for the absolute values of the non-standard parameters, and `ARG_EPS_`c`_`$\alpha\beta$ for their phases. Here, $c = \texttt{S}, \texttt{M}, \texttt{D}$ to identify NSI in the source, in propagation (non-standard matter effects), and in detection. α and β are neutrino flavors, where the standard flavors are denoted by `E`, `MU`, `TAU`, and the sterile flavors are `S1`, `S2`, `S3`, ... For example, `ABS_EPS_M_ETAU` refers to $|\varepsilon^m_{e\tau}|$ in the notation of [3].

Having initialized and registered the non-standard probability engine, you can use all GLoBES functions in the same way as you would for standard oscillations. Note, however that at the moment the initial and final flavors in the definition of an oscillation channel can only involve the three active flavors. It is for instance not possible to compute sterile neutrino appearance.

After using the sterile neutrino/NSI engine, you may want to release the (small) amount of memory allocated by it by calling

```
snu_free_probability_engine();
```

References

[1] Mattias Blennow and Enrique Fernandez-Martinez. Neutrino oscillation parameter sampling with MonteCUBES. *Comput. Phys. Commun.*, 181:227–231, 2010.

[2] Joachim Kopp. Efficient numerical diagonalization of hermitian 3×3 matrices. *Int. J. Mod. Phys.*, C19:523–548, 2008. Erratum ibid. **C19** (2008) 845.

[3] Joachim Kopp, Manfred Lindner, Toshihiko Ota, and Joe Sato. Non-standard neutrino interactions in reactor and superbeam experiments. *Phys. Rev.*, D77:013007, 2008.