Prepared for submission to JHEP

Flavour Symmetry Embedded - GLoBES

Jian Tang,[a] **Tse-Chun Wang**[a]

[a] *School of Physics, Sun Yat-Sen University, Guangzhou 510275, China*

E-mail: tangjian5@mail.sysu.edu.cn, wangzejun@mail.sysu.edu.cn

Abstract: Abstract...

Contents

1 Introduction **1**

2 Overview FASE-GloBES **2**
- 2.1 Download and Installation 4
- 2.2 Compile and Run 4
- 2.3 Model setting 5
- 2.4 Prior setting 6

3 Examples **6**
- 3.1 Setup for a model 6
- 3.2 Initialise the code 9
- 3.3 Prior setup 10
- 3.4 Constraint of model parameters 10
- 3.5 Model testing 11

4 Conclusion **14**

1 Introduction

The discovery of neutrino oscillations points out the fact that neutrinos have mass, and provides evidence beyond the Standard Model (BSM). This phenomenon is successfully described by a theoretical framework with the help of three neutrino mixing angles (θ_{12}, θ_{13}, θ_{23}), two mass-square splittings (Δm^2_{21}, Δm^2_{31}), and one Dirac CP phase (δ) [1–4]. Thanks to the great efforts in the past two decades, we almost have a complete understanding of such a neutrino oscillation framework. More data in the neutrino oscillation experiments is needed to determine the sign of Δm^2_{31}, to measure the value of $\sin\theta_{23}$, to discover the potential CP violation in the leptonic sector and even to constrain the size of δ [4]. For these purposes, the on-going long baseline experiments (LBLs), such as the NuMI Off-axis ν_e Appearance experiment (NOνA) [5] and the Tokai-to-Kamioka experiment (T2K) [6], can answer these questions with the statistical significance $\gtrsim 3\sigma$ in most of the parameter space. Based on the analysis with their data, the normal mass ordering ($\Delta m^2_{31} > 0$), the higher θ_{23} octant ($\theta_{23} > 45^\circ$), and $\delta \sim 270^\circ$ are preferred so far [4]. The future LBLs, Deep Underground Neutrino Experiment (DUNE) [7], Tokai to Hyper-Kamiokande (T2HK) [8], and the medium baseline reactor experiment, the Jiangmen Underground Neutrino Observatory (JUNO) [9, 10] will further complete our knowledge of neutrino oscillations.

Flavour symmetry models are used to explain the origin of the neutrino mixing, and to predict the value of oscillation parameters (some of useful review articles are [11–17]). These models are motivated by some interesting features, *such as* $\theta_{12} \sim 33^\circ$, and $\theta_{23} \sim 45^\circ$.

Before the discovery of non-zero θ_{13} measurement by Daya Bay experiment [18], the 'tri-bi-maximal' neutrino mixing (TBM) ansatz, which was proposed in 2002 by Horrison, Perkins, and Scott [19], fitted with the experimental data in a good agreement:

$$U_{\text{TBM}} = \begin{pmatrix} 2/\sqrt{6} & 1/\sqrt{3} & 0 \\ -1/\sqrt{6} & 1/\sqrt{3} & 1/\sqrt{2} \\ 1/\sqrt{6} & -1/\sqrt{3} & 1/\sqrt{2} \end{pmatrix}.$$

With the fact that $\theta_{13} \approx 8°$, several ways to obtain such non-zero value of θ_{13} are proposed. One of popular proposals is to correct the tri-bi-maximal neutrino mixing such that

$$\sin\theta_{12} = (1+s)/\sqrt{3}, \ \sin\theta_{13} = r/\sqrt{2}, \ \text{and} \ \sin\theta_{23} = (1+a)/\sqrt{2}.$$

Th neutrino mixing ansatz can be realised by high-energy symmetries G_f. The symmetry of discrete groups G_f, preserved at the high energy but slightly broken at the lower energy, predicts the neutrino mixing, mass-square splittings, and the CP violation phase (Dirac and Majorana phases), with reduced degrees of freedom. The symmetries need to be broken at the low energy. Otherwise, the flavour of leptons cannot be distinguished. There are several approaches for the symmetry breaking, including the direct, indirect, and semi approaches. Direct approach preserves the residual symmetries of G_f in the charged-lepton or neutrino sector. On the other hand, there is no residual symmetry preserved in neither charged-lepton nor neutrino sector in the indirect approach. In the semi approach, the charged-lepton and neutrino sectors preserve different residual symmetries, respectively. This symmetry is broken by extending the Higgs sector or introducing the flavons. To achieve the δ prediction, many models are based on a discrete family symmetry G_f together with a non-commuting CP symmetry H_{CP}. Broken in different approaches, the symmetry $G_f \otimes H_{\text{CP}}$ can predict different patterns for the neutrino mixing. For example, in the semi-direct approach, S_4 is preserved in the leading order, and leads the bimaximal (MB) or tri-bimaximal (TB) neutrino mixing, while the higher order terms bring the correction to the neutrino mixing.

Currently, some works discuss on how the future experiments can be used for testing these flavour symmetry models in the phenomenological point of view, *e.g.* [20–23]. A large number of these works are based on the c-library – **G**eneral **L**ong **B**aseline **E**xperiment **S**imulator (**GLoBES**) [24, 25], which is a convenient simulation tool to simulate neutrino oscillation experiments via the Abstract Experiment Definition Language (AEDL). Some AEDL files for experiments are also available on **GLoBES** website. In addition, the working group of DUNE experiment also releases their AEDL files [26]. **GLoBES** can be taken as one of popular and useful tools in the community of neutrino oscillation physics. However, it has not yet to be extended for the purpose of analysing flavour symmetry models.

In this work, we will present our simulation tool **Fla**vour **S**ymmetry **E**mbedded - **GLoBES** (**FaSE-GloBES**) that is a simulation code based on GLoBES library and specific for studying the flavour symmetry.

2 Overview FASE-GloBES

Fl**A**vor **S**ymmetry **E**mbedded (**FASE**) is a supplemental tool for General Long Baseline Experiment Simulator (**GLoBES**) in order to analysis how a leptonic flavour symmetry model can be tested in neutrino oscillation experiments. **FASE** is written in the `c/c++` language, and consist with three codes **FASE_GLoBES.c**, **model-input_diag.c** and **model-input.c**. The user defines the model in to **model-input_diag.c** or **model-input.c**, while **FASE_GLoBES.c** does not need to be touched.

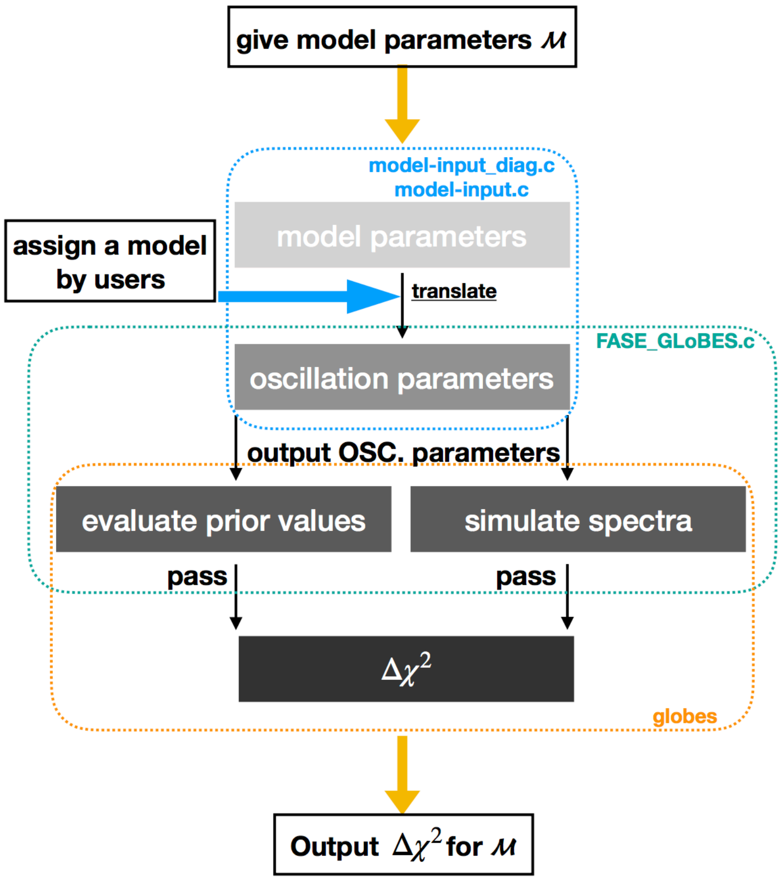

Figure 1. A scheme to correlate the model parameters with standard neutrino oscillation parameters. The error propagation is implemented in the simulation code up to the spectra analysis.

The **FASE-GLoBES** is **FASE** working with **GLoBES** as shown in Fig. 1, in which three parts are shown: 1. **the parameter translation** (the blue box), 2. **giving oscillation-parameter values** (the green box), and 3. **the χ^2-value calculation** (the orange box). The idea behind this flow chart Fig. 1 is that given a set of value for model parameter, the corresponding values for oscillation parameters are obtained by a translation, which is assigned by the user in **model-input_diag.c** or **model-input.c**. And then, through **FASE_GLoBES.c**, these oscillation-parameter values are passed in to **GLoBES** library to simulate the event spectrum for evaluating the χ^2 value.

API functions to use **FASE** are listed:

1. `MODEL_init(`N_{para}`)`,

2. `FASE_glb_probability_matrix`,

3. FASE_glb_set_oscillation_parameters,
4. FASE_glb_get_oscillation_parameters,
5. FASE_prior_OSC,
6. FASE_prior_model.

The first one is to initialise FASE with the parameter N_{para} the number of input parameters. The next three functions need to be included to replace the default GLoBES probability engine by the one that can read the output from **model-input_diag.c** or **model-input.c**, as follows

```
glbRegisterProbabilityEngine(6,
&FASE_glb_probability_matrix,
&FASE_glb_set_oscillation_parameters,
&FASE_glb_get_oscillation_parameters,
NULL);
```

This probability engine can work with the oscillation or model parameter. It can be set by the user with the parameter `PARA`. If `PARA=STAN` (`PARA=MODEL`) the probability engine works with oscillation (model) parameters. The final two functions are prior functions. Once the user gives the prior in oscillation (model) parameters, the user needs to call `FASE_prior_OSC` (`FASE_prior_model`) as follows.

```
glbRegisterPriorFunction(FASE_prior_OSC,NULL,NULL,NULL);
```
or
```
glbRegisterPriorFunction(FASE_prior_model,NULL,NULL,NULL);
```

We note that except for setting the probability engine and the prior function, the other parts in the main code should follow with the GLoBES manual.

In the following, we give more details about using **FASE-GLoBES**. In Sec. 2.1, we will give the instruction to download and install **GLoBES** and **FASE**. Then, in Sec. 2.2, we will introduce how to compile and run the code. More complicated is setting the model, and the way to do this will be described in Sec. 2.3. Finally, we will discuss how to set up the prior in Sec. 2.4.

2.1 Download and Installation

download and install GLoBES and FASE-GLoBES
[will fill in after the code ready on Github]

2.2 Compile and Run

As FASE works based on GLoBES, the user should use the **GLoBES** `Makefile`, but include the binary file for FASE. To do so, we include the script in the `Makefile`,

```
{makefile_execution}:  my_program.o FASE_GLoBES.o model-input_diag.o
gcc my_program.o FASE_GLoBES.o model-input_diag.o -o my_executable $(LDFLAGS)
```

```
$(local_LDFLAGS)
```
or
```
{makefile_execution}:  my_program.o FASE_GLoBES.o model-input.o
gcc my_program.o FASE_GLoBES.o model-input.o -o my_executable $(LDFLAGS) $(local_LDFLAGS)
```

where `{compile_execusion}` is the makefile commend for the program `my_program.o` and the execution `my_executable` is the output. After giving this script in the **GLoBES** `Makefile`, the user needs to makefile the program `my_program` on the terminal.

```
makefile {makefile_execution}
```

And, the user can execute the program by typing the following commend on the terminal.

```
./my_executable
```

The user can also compile the program without the **GLoBES** `Makefile` with the following script.

```
gcc my_program.o FASE_GLoBES.o model-input_diag.o -lglobes -LGLB_DIR/lib/ -o my_executable
```
or
```
gcc my_program.o FASE_GLoBES.o model-input.o -lglobes -LGLB_DIR/lib/ -o my_executable
```

And, the execution is also in the same way: `./my_executable`.

Finally we need to initialize **FASE** by giving N_{para} which is the number of model parameters. To do so, we need to include the following script in the main code.
`MODEL_init(`N_{para}`);`

2.3 Model setting

FASE includes three `c++` files: **model-input_diag.c**, **model-input.c**, and **FASE_GLoBES.c**. As shown in the previous paragraph, these files are for different service. In **model-input_diag.c** and **model-input.c**, the user can assign the relation between the oscillation and model parameter sets. The difference between these two code is that in **model-input.c**, the user give the relation between these two parameter sets directly, while in **model-input.c**, the mass matrix, which is provided by the user, can be diagonalised to obtain the corresponding oscillation-parameter values.

For different methods to include the model, we use different `c++` file. Adopting the mass matrix expressed by model parameters, we assign this model in **model-input_diag.c**. In function `MtoS`, we give the mass matrix in the 3×3 complex matrix `Mass_Matrix[]`. Passing the complex matrix `Mass_Matrix[]` into the function `STAN_OSC`, we will get the corresponding oscillation parameters in an array `OSC_PARAMS`, components of which are the value for θ_{12}, θ_{13}, θ_{23}, δ, Δm^2_{21}, and Δm^2_{31}. The expression of `Mass_Matrix[]` is given by the user. For the first four components, values are given in the unit of **rad**, while the other two are in $\mathbf{eV}^2$. We can also assign the relation between oscillation and model parameters in **model-input.c**. The relations are given in `MtoS` as well.

2.4 Prior setting

Given a set of values for model parameters, **FASE_GLoBES.c** will obtain the corresponding oscillation-parameter values from **model-input_diga.c** or **model-input.c**, and will pass these values to simulate the event spectrum and to give the prior value. Two gaussian prior functions are provided in **FASE** – `FASE_prior_OSC` and `FASE_prior_model`. These two functions are for different purposes. If the user give the prior in oscillation (model) parameters, the user should register `FASE_prior_OSC` (`FASE_prior_model`) for the prior with the **GLoBES** function `glbRegisterPriorFunction`, as we introduced in the beginning of this section. The Gaussian prior is

$$\chi^2_{prior} = \sum_i \frac{(\theta_i - \theta_i^c)^2}{\sigma_i^2}, \tag{2.1}$$

where θ_i is one of parameters constrained by prior, θ_i^c (σ_c) is the central value (Gaussian width) of the prior for θ_i. We note that θ_i can be either model or oscillation parameter. But, oscillation and model parameters can be mixed up together in the using of prior.

The values of θ_i^c and σ_i need to be given by the user through three arrays: `Central_prior`, `UPPER_prior`, and `LOWER_prior`, in which there are six components. To treat asymmetry of width for the upper and lower Gaussian widths, we give values in two arrays `UPPER_prior`, and `LOWER_prior`, respectively. If the user gives the prior in model parameters, the order of each component follow with the setup of input of the probability engine. While the user gives the prior in oscillation parameters, the six components in order are θ_{12}, θ_{13}, θ_{23}, δ, Δm^2_{21}, and Δm^2_{31}. The first four parameter are in **rad**, and the final two are in $\mathbf{eV}^2$.

Finally, some restrictions are imposed by the studied flavour symmetry model. We set up these restrictions in the function `model_restriction`, which is in `model-input_diga.c` and `model-input.c`. In the function `model_restriction`, the user needs to *return* 0 once the restriction is broken. For example, if the normal ordering is imposed, we give " `if (DMS31<0) { return 1;}` " in `model_restriction`, where `DMS31` is for Δm^2_{31}. Then, when the restriction is broken, `model_restriction` returns the value 1 to the prior function `FASE_prior_OSC` or `FASE_prior_model`. In the following, the prior function will give 10^6 for the χ^2 value, and it will be selected out when we study the statistically reasonable region in the parameter space.

3 Examples

3.1 Setup for a model

We now take the littlest seesaw model in the tri-direct approach as an example. In this model, the atmospheric and solar flavon vacuum alignments are $\langle\phi_{\text{atm}}\rangle \propto \left(1, \omega^2, \omega\right)^T$ and $\langle\phi_{\text{sol}}\rangle \propto (1, x, x)^T$, where stands for a cube root of unity and the parameter x is real because of the imposed CP symmetry. Under this model, the light left-handed Majorana neutrino mass matrix is given by

$$m_\nu = m_a \begin{pmatrix} 1 & \omega & \omega^2 \\ \omega & \omega^2 & 1 \\ \omega^2 & 1 & \omega \end{pmatrix} + e^{i\eta} m_s \begin{pmatrix} 1 & x & x \\ x & x^2 & x^2 \\ x & x^2 & x^2 \end{pmatrix}, \tag{3.1}$$

where x, η, m_a, and the ratio $r \equiv m_s/m_a$ are four parameters and will be constrained by experimental data. We note that from Eq. (3.1), $m_1 = 0$ and the normal mass ordering are imposed, and will need to be imposed in **FASE-GLoBES**. Therefore, the restrictions in this model are $m_a > 0$ and $r > 0$.

Table 1. A summary of the relation between oscillation parameters and TDLS model parameters [27]. Two requirements are imposed by TDLS: the smallest mass state $m_1 = 0$ and the normal mass ordering. The sign of $\sin\delta$ depends on the sign of $x\cos\psi$: "+" ("−") is for $x\cos\psi > 0$ (< 0).

model parameters	x, η, r, m_a
combinations of model parameters	$y = \frac{5x^2+2x+2}{2(x^2+x+1)}(m_a + e^{i\eta}m_s)$, $z = -\frac{\sqrt{5x^2+2x+2}}{2(x^2+x+1)}\left[(x+2)m_a - x(2x+1)e^{i\eta}m_s\right]$, $w = \frac{1}{2(x^2+x+1)}\left[(x+2)^2 m_a + x^2\left(2x+1\right)^2 e^{i\eta}m_s\right]$, $\sin\psi = \frac{\Im(y^*z+wz^*)}{\lvert y^*z+wz^*\rvert}, \quad \cos\psi = \frac{\Re(y^*z+wz^*)}{\lvert y^*z+wz^*\rvert}$. $\sin 2\theta = \frac{2\lvert y^*z+wz^*\rvert}{\sqrt{(\lvert w\rvert^2-\lvert y\rvert^2)^2+4\lvert y^*z+wz^*\rvert^2}}$, $\cos 2\theta = \frac{\lvert w\rvert^2-\lvert y\rvert^2}{\sqrt{(\lvert w\rvert^2-\lvert y\rvert^2)^2+4\lvert y^*z+wz^*\rvert^2}}$.
oscillation parameters	$\Delta m^2_{21} = m^2_2 = \frac{1}{2}\left[\lvert y\rvert^2 + \lvert w\rvert^2 + 2\lvert z\rvert^2 - \frac{\lvert w\rvert^2-\lvert y\rvert^2}{\cos\theta}\right]$, $\Delta m^2_{31} = m^2_3 = \frac{1}{2}\left[\lvert y\rvert^2 + \lvert w\rvert^2 + 2\lvert z\rvert^2 + \frac{\lvert w\rvert^2-\lvert y\rvert^2}{\cos\theta}\right]$, $\sin^2\theta_{12} = 1 - \frac{3x^2}{3x^2+2(x^2+x+1)\cos^2\theta}$, $\sin^2\theta_{13} = \frac{2\left(x^2+x+1\right)\sin^2\theta}{5x^2+2x+2}$, $\sin^2\theta_{23} = \frac{1}{2} + \frac{x\sqrt{3(5x^2+2x+2)}\sin 2\theta\sin\psi}{2[3x^2+2(x^2+x+1)\cos^2\theta]}$, $\cos\delta = \frac{\cot 2\theta_{23}\left[3x^2-\left(4x^2+x+1\right)\cos^2\theta_{13}\right]}{\sqrt{3}\lvert x\rvert\sin\theta_{13}\sqrt{(5x^2+2x+2)\cos^2\theta_{13}-3x^2}}$, $\sin\delta = \pm\csc 2\theta_{23}\sqrt{1 + \frac{(x^2+x+1)^2\cot^2\theta_{13}\cos^2 2\theta_{23}}{3x^2[3x^2\tan^2\theta_{13}-2(x^2+x+1)]}}$.

```
    int MtoS(double OSC_PARAMS[6], double M_para[])
{
double x=M_para[0];
double eta=M_para[1];
double r=M_para[2];
double ma=M_para[3];
double ms=ma*r;

    double complex Mass_Matrix[] = {ma+ms*(cos(eta) + I*sin(eta)), ma*(cos(6.6666e-1*M_PI)
+ I*sin(6.6666e-1*M_PI))+x*ms*(cos(eta) + I*sin(eta)), ma*(cos(6.6666e-1*M_PI)
+ I*sin(6.6666e-1*M_PI))*(cos(6.6666e-1*M_PI) + I*sin(6.6666e-1*M_PI))+x*ms*(cos(eta)
+ I*sin(eta)), ma*(cos(6.6666e-1*M_PI) + I*sin(6.6666e-1*M_PI))+x*ms*(cos(eta)
+ I*sin(eta)), ma*(cos(6.6666e-1*M_PI) + I*sin(6.6666e-1*M_PI))*(cos(6.6666e-1*M_PI)
+ I*sin(6.6666e-1*M_PI))+x*x*ms*(cos(eta) + I*sin(eta)), ma+x*x*ms*(cos(eta) +
I*sin(eta)), ma*(cos(6.6666e-1*M_PI) + I*sin(6.6666e-1*M_PI))*(cos(6.6666e-1*M_PI)
+ I*sin(6.6666e-1*M_PI))+x*ms*(cos(eta) + I*sin(eta)), ma+x*x*ms*(cos(eta) + I*sin(eta)),
ma*(cos(6.6666e-1*M_PI) + I*sin(6.6666e-1*M_PI))+x*x*ms*(cos(eta) + I*sin(eta))};

    STAN_OSC(Mass_Matrix,OSC_PARAMS);

    return 0;
}
    double MtoS(double osc_para[6], double M_para[])
{
/*example:  tri-direct*/

double x_in = M_para[0];
double eta_in = M_para[1];
double r_in = M_para[2];
double ma_in = M_para[3];

osc_para[0]=TDth12(x_in,eta_in,r_in, ma_in);
osc_para[1]=TDth13(x_in,eta_in,r_in, ma_in);
osc_para[2]=TDth23(x_in,eta_in,r_in, ma_in);
osc_para[3]=TDdCP(x_in,eta_in,r_in, ma_in);
osc_para[4]=TDdm21(x_in,eta_in,r_in, ma_in);
osc_para[5]=TDdm31(x_in,eta_in,r_in, ma_in);

return 0;
}
```

The functions TDth12, TDth13, TDth23, TDdCP, TDdm21, and TDdm31 are included for obtaining the value of θ_{12}, θ_{13}, θ_{23}, δ, Δm^2_{21}, and Δm^2_{31} according to Table 1, respectively.

The restrictions $m_a > 0$ and $r > 0$ need to be setup in model_restriction as follows.

```
double model_restriction(double model [])
{
double x=model[0];
double eta=model[1];
double r=model[2];
double ma=model[3];

if(ma<0) {return 1;}
if(r<0) {return 1;}

return 0;
}
```

If there is no any restrictions, we simply *return* 0 in model_restriction as follows.

```
double model_restriction(double model []){ return 0;}
```

3.2 Initialise the code

In the beginning of main code, we need to initialise GLoBES and FASE, and include the considered experiment (here we consider MOMENT experiment) as follows.

```
glbInit(argv[0]);
glbInitExperiment("expMOMENT_FIX_FLUX_150KM_addATM_NC.glb",&glb_experiment_list[0],
&glb_num_of_exps);
MODEL_init(4);
```

Then we register the probability engine.

```
glb_init_probability_engine();
glbRegisterProbabilityEngine(6,
&FASE_glb_probability_matrix,
&FASE_glb_set_oscillation_parameters,
&FASE_glb_get_oscillation_parameters,
NULL);
```

In the following we define the model-parameter values, and set the true value in model parameters.

```
float degree = M_PI/180;
float x_true,eta_true,r_true,ma_true;
x_true=-3.65029; eta_true=1.13067*M_PI; r_true=0.511325; ma_true=3.71199e-3;
glb_params true_values = glbAllocParams();
glb_params test_values = glbAllocParams();
glb_params input_errors = glbAllocParams();
glb_params centers = glbAllocParams();
glbDefineParams(true_values,x_true,eta_true,r_true,ma_true,0,0);
glbSetDensityParams(true_values,1.0,GLB_ALL);
glbCopyParams(true_values,test_values);
PARA=MODEL;
glbSetOscillationParameters(true_values);
glbSetRates();
```

We then finally set up the projection. The projection `free` is to set all four parameters free to find the minimal value of χ^2, which `project` is used for studying the constraint on x and η.

```
glb_projection projection = glbAllocProjection();
glb_projection free = glbAllocProjection();
glbDefineProjection(free,GLB_FREE,GLB_FREE,GLB_FREE,GLB_FREE,GLB_FIXED,GLB_FIXED);
glbSetDensityProjectionFlag(free, GLB_FREE, GLB_ALL);
glbDefineProjection(projection,GLB_FIXED,GLB_FIXED,GLB_FREE,GLB_FREE,GLB_FIXED,GLB_FIXED);
```

```
glbSetDensityProjectionFlag(projection, GLB_FREE, GLB_ALL);
```

3.3 Prior setup

To set up the prior, we need to register the prior function. As mentioned, when we use the prior in oscillation and model parameters, we register `FASE_prior_OSC` and `FASE_prior_model`, respectively. Here we present an example setting up priors in oscillation parameters according to NUFIT4.0 Table 2.

Table 2. The best fit and 3σ uncertainty, in the results of NuFit4.0 [4].

Parameter	$\theta_{12}/^\circ$	$\theta_{13}/^\circ$	$\theta_{23}/^\circ$	$\delta/^\circ$	$\Delta m^2_{21}/10^{-5}\mathrm{eV}^2$	$\Delta m^2_{31}/10^{-3}\mathrm{eV}^2$
best fit	33.82	8.61	49.6	215	7.39	2.525
3σ Range	$31.61 - 36.27$	$8.22 - 8.99$	$40.3 - 52.4$	$125 - 392$	$6.79 - 8.01$	$2.47 - 2.625$

We therefore register `FASE_prior_OSC` for the prior function as follows.

```
glbRegisterPriorFunction(FASE_prior_OSC,NULL,NULL,NULL);
```

And, then we give the value for the central values, and upper and lower Gaussian widths as follows.

```
UPPER_prior[0]=36.27*degree; LOWER_prior[0]=31.61*degree; Central_prior[0]=33.82*degree;
UPPER_prior[1]=8.99*degree; LOWER_prior[1]=8.22*degree; Central_prior[1]=8.61*degree;
UPPER_prior[2]=52.4*degree; LOWER_prior[2]=40.3*degree; Central_prior[2]=49.6*degree;
UPPER_prior[3]=392*degree; LOWER_prior[3]=125*degree; Central_prior[3]=215*degree;
UPPER_prior[4]=8.01e-5; LOWER_prior[4]=6.79e-5; Central_prior[4]=7.39e-5;
UPPER_prior[5]=2.625e-3; LOWER_prior[5]=2.427e-3; Central_prior[5]=2.525e-3;

int i;
for (i=0;i<6;i++) {UPPER_prior[i]=fabs(UPPER_prior[i]-Central_prior[i])/3;
LOWER_prior[i]=fabs(LOWER_prior[i]-Central_prior[i])/3;}
for (i=0; i<6; i++) glbSetOscParams(centers,0,i);
glbSetDensityParams(centers,1.0,GLB_ALL); glbCopyParams(centers,input_errors);
glbSetCentralValues(centers); glbSetInputErrors(input_errors);
```

The parameter `degree` is defined as $\pi/180$.

3.4 Constraint of model parameters

The user can use **FASE-GLoBES** to study the constraint of model parameters. Here we take tri-direct littlest seesaw as an example to present how we can adopt **FASE-GLoBES** to constrain two model parameters – x and η. The χ^2 function is constructed based on a log-likelihood ratio,

$$\chi^2(\vec{\theta},\xi_s,\xi_b) = 2\sum_i \left(\eta_i(\vec{\theta},\xi_s,\xi_b) - n_i + n_i \ln\frac{n_i}{\eta_i(\vec{\theta},\xi_s,\xi_b)}\right) + p(\xi_s,\sigma_s) + p(\xi_b,\sigma_b) + \chi^2_{prior}, \tag{3.2}$$

where i runs over the number of bins, $\eta_i\vec{\theta}, \xi_s, \xi_b$ is the hypothesis event rate for bin i and E_i is the central bin energy. The vector $\vec{\theta}$ consist test model or oscillation parameters. Here, $\vec{\theta} = (x, \eta, r, m_a)$. The parameters ξ_s and ξ_b are introduced to account for the systematic uncertainty of normalisation for the signal (subscript $_s$) and background (subscript $_b$) components for the event rate, and are allowed to bary in the fit as nuisance parameters. For a given hypothesized set of parmaeters $\vec{\theta}$, the event rate for bin i is calculated as

$$\eta_i(\vec{\theta}, \xi_s, \xi_b) = (1 + \xi_s) \times n_i + (1 + \xi_b) \times b_i, \tag{3.3}$$

where n_i and b_i are the expected number of signal and background events in bin i, respectively. The nuisance parameters are constrained by the Gaussian prior $p(\xi, \sigma) = \xi^2/\sigma^2$ with corresponding uncertainties σ_s and σ_b for the signal and background, respectively. Finally, χ^2_{prior} is a set of Gaussian priors for hypothesis, and is expressed as Eq. 2.1. To get the constraint of x and η, we obtain the minimum of χ^2 value,

```
glbSetProjection(free);
float res0=glbChiNP(true_values,NULL,GLB_ALL);
```

Then, we set two loops to get the $\Delta\chi^2$ value for different x and η

```
float x,eta,r,ma,dx,deta;
float lower_x,upper_x,lower_eta,upper_eta;
FILE* File=fopen("data/constraint_x_eta.dat", "w");
lower_x=-9; upper_x=-3; lower_eta=0.8*M_PI; upper_eta=2*M_PI;
dx=(upper_x-lower_x)/100; deta=(upper_eta-lower_eta)/100;
glbSetProjection(projection);
for (x=lower_x;x<=upper_x;x=x+dx){
for (eta=lower_eta;eta<=upper_eta;eta=eta+deta){
glbSetOscParams(test_values,x,0); glbSetOscParams(test_values,eta,1);
float res=glbChiNP(test_values,NULL,GLB_ALL);
fprintf(File,"%f %f %f \n",x,eta/M_PI,res-res0);
} fprintf(File,"\n");}
```

We use the same code to obtain the constraint of any other combination of two parameters. Then, we can have the result as in Fig. 2.

3.5 Model testing

We can also study on excluding the model, assuming different true values for oscillation parameters. In this example, we present testing the tri-driect littlest seesaw model in various θ_{23} and δ. To do so, we set the true value in oscillation parameters. We change values of θ_{23} and δ of the true theory, and compute the minimal χ^2 value for the tested model with all four model parameters, free to be varied. And, the studied statistics function is exactly Eq. 3.2, but the true event rate n_i is predicted by a set of oscillation parameters, which will

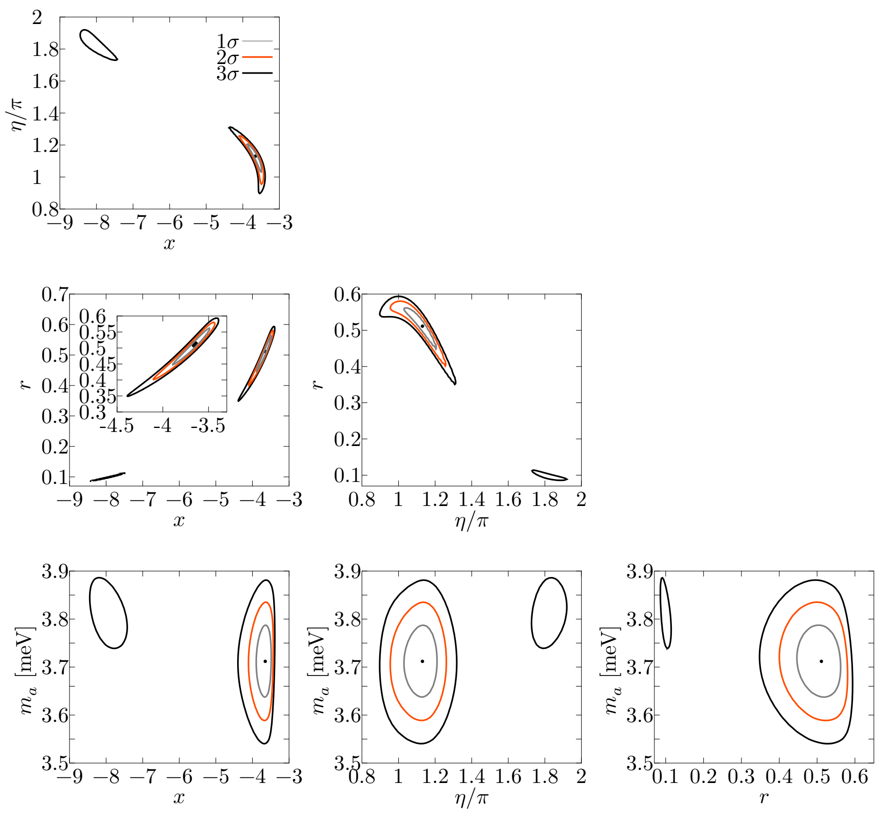

Figure 2. Precision measurements of any two model parameters at 3σ confidence level in the framework of three neutrino oscillations taking uncertainties of the current global fit results, for MOMENT, at 1σ (gray), 2σ (orange), 3σ (black). True values for the model parameters are used $(x,\ \eta,\ r,\ M_a) = (-3.65,\ 1.13\pi,\ 0.511,\ 3.71\ \text{meV})$.

be varied in the code. And all four model parameter can be varied with the prior given by Eq. 2.1.

The example code is given below.

```
{ ...  initalize the code ...}
{ ...  register the probability engine ...}

float degree = M_PI/180;
float x_true,eta_true,r_true,ma_true;
x_true=-3.65029; eta_true=1.13067*M_PI; r_true=0.511325; ma_true=3.71199e-3;
double M_para[4],OSC_para[6];
M_para[0]=x_true; M_para[1]=eta_true; M_para[2]=r_true; M_para[3]=ma_true;
MtoS(OSC_para, M_para);
double th12_true=OSC_para[0];
double th13_true=OSC_para[1];
double th23_true=OSC_para[2];
double dCP_true=OSC_para[3];
double DM21_true=OSC_para[4];
double DM31_true=OSC_para[5];
glb_params true_values = glbAllocParams();
glb_params test_values = glbAllocParams();
glb_params input_errors = glbAllocParams();
glb_params centers = glbAllocParams();

glbDefineParams(test_values,x_true,eta_true,r_true,ma_true,0,0);
glbSetDensityParams(test_values,1.0,GLB_ALL);
glbDefineParams(true_values,th12_true,th13_true,th23_true,dCP_true,DM21_true,DM31_true);
glbSetDensityParams(true_values,1.0,GLB_ALL);
```

```
{ ...  set up for prior ...}
```

```
float th23,dCP,dth23,ddCP,lower_th23,upper_th23,lower_dCP,upper_dCP;
FILE* File=fopen("data/ModelTest_th23_dCP_test.dat", "w");
lower_th23=40; upper_th23=50; lower_dCP=125; upper_dCP=392;
dth23=(upper_th23-lower_th23)/100; ddCP=(upper_dCP-lower_dCP)/100;
glbSetProjection(free);
for (th23=lower_th23;th23<=upper_th23;th23=th23+dth23){
for (dCP=lower_dCP;dCP<=upper_dCP;dCP=dCP+ddCP){
glbSetOscParams(true_values,th23*degree,GLB_THETA_23);
glbSetOscParams(true_values,dCP*degree,GLB_DELTA_CP);
PARA=STAN;
glbSetOscillationParameters(true_values);
glbSetRates();
PARA=MODEL;
float res=glbChiNP(test_values,NULL,GLB_ALL);
fprintf(File,"%f %f %f \n",th23,dCP,gsl_cdf_chisq_Qinv(gsl_cdf_chisq_Q(fabs(res),dof),1));
} fprintf(File,"\n");
}
```

```
{ ...  destroy the pointer ...}
```

Finally, in the code we adopt Wilk's theorem [28]. When comparing nested models, the $\Delta\chi^2$ test statistics is a random variable asymptotically distributed according to the χ^2-distribution with the number of degrees of freedom, which is equal to the difference in the number of free model parameters `dof`. Here the number `dof` is 2. And that is our output: `gsl_cdf_chisq_Qinv(gsl_cdf_chisq_Q(fabs(res),dof),1)`.

We use the same code to analysis the exclusion ability for other oscillation-parameter combinations. Then, we have the results shown in Fig. 3.

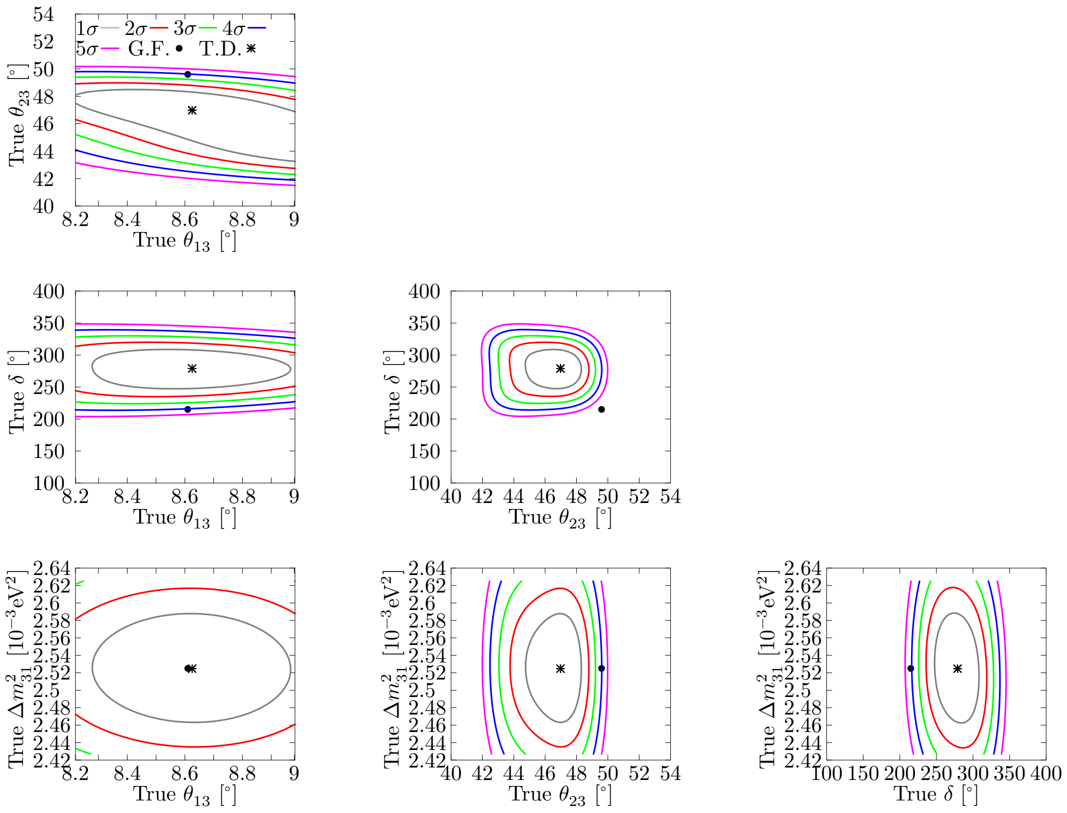

Figure 3. The 2-D exclusion contour for tri-direct littlest seesaw model on the plane of any two true standard parameters, from 1σ to 5σ. The range for each parameter is taken according to the 3σ uncertainty in NuFit4.0 results. The black dot denotes the best fit of NuFit4.0 results $((\theta_{12},\ \theta_{13},\ \theta_{23},\ \delta,\ \Delta m^2_{21},\ \Delta m^2_{31}) = (33.82^\circ,\ 8.61^\circ,\ 49.6^\circ,\ 215^\circ,\ 7.39\times10^{-5}\ \text{eV}^2,\ 2.525\times10^{-3}\ \text{eV}^2))$, while the star is the prediction by the tri-direct littlest seesaw model with NuFit4.0 results $((\theta_{12},\ \theta_{13},\ \theta_{23},\ \delta,\ \Delta m^2_{21},\ \Delta m^2_{31}) \sim (36.25^\circ,\ 8.63^\circ,\ 47^\circ,\ 279^\circ,\ 7.39\times10^{-5}\ \text{eV}^2,\ 2.525\times10^{-3}\ \text{eV}^2))$.

4 Conclusion

References

[1] B. Pontecorvo, *Neutrino Experiments and the Problem of Conservation of Leptonic Charge*, *Sov. Phys. JETP* **26** (1968) 984.

[2] Z. Maki, M. Nakagawa and S. Sakata, *Remarks on the unified model of elementary particles*, *Prog. Theor. Phys.* **28** (1962) 870.

[3] B. Pontecorvo, *Inverse beta processes and nonconservation of lepton charge*, *Sov. Phys. JETP* **7** (1958) 172.

[4] I. Esteban, M. C. Gonzalez-Garcia, A. Hernandez-Cabezudo, M. Maltoni and T. Schwetz, *Global analysis of three-flavour neutrino oscillations: synergies and tensions in the determination of θ_23, δ_CP, and the mass ordering*, *JHEP* **01** (2019) 106 [1811.05487].

[5] NOvA collaboration, *The NOvA Technical Design Report*, .

[6] T2K collaboration, *The T2K Experiment*, *Nucl. Instrum. Meth.* **A659** (2011) 106 [1106.1238].

[7] DUNE collaboration, *Long-Baseline Neutrino Facility (LBNF) and Deep Underground Neutrino Experiment (DUNE)*, 1512.06148.

[8] Hyper-Kamiokande Working Group collaboration, *A Long Baseline Neutrino Oscillation Experiment Using J-PARC Neutrino Beam and Hyper-Kamiokande*, 2014, 1412.4673.

[9] JUNO collaboration, *JUNO Conceptual Design Report*, 1508.07166.

[10] JUNO collaboration, *Neutrino Physics with JUNO*, *J. Phys.* **G43** (2016) 030401 [1507.05613].

[11] G. Altarelli and F. Feruglio, *Discrete Flavor Symmetries and Models of Neutrino Mixing*, *Rev. Mod. Phys.* **82** (2010) 2701 [1002.0211].

[12] H. Ishimori, T. Kobayashi, H. Ohki, Y. Shimizu, H. Okada and M. Tanimoto, *Non-Abelian Discrete Symmetries in Particle Physics*, *Prog. Theor. Phys. Suppl.* **183** (2010) 1 [1003.3552].

[13] S. F. King and C. Luhn, *Neutrino Mass and Mixing with Discrete Symmetry*, *Rept. Prog. Phys.* **76** (2013) 056201 [1301.1340].

[14] S. F. King, A. Merle, S. Morisi, Y. Shimizu and M. Tanimoto, *Neutrino Mass and Mixing: from Theory to Experiment*, *New J. Phys.* **16** (2014) 045018 [1402.4271].

[15] S. F. King, *Models of Neutrino Mass, Mixing and CP Violation*, *J. Phys.* **G42** (2015) 123001 [1510.02091].

[16] S. F. King, *Neutrino Mixing: from experiment to theory*, *Nucl. Part. Phys. Proc.* **265-266** (2015) 288.

[17] S. F. King, *Unified Models of Neutrinos, Flavour and CP Violation*, *Prog. Part. Nucl. Phys.* **94** (2017) 217 [1701.04413].

[18] Daya Bay collaboration, *Spectral measurement of electron antineutrino oscillation amplitude and frequency at Daya Bay*, *Phys. Rev. Lett.* **112** (2014) 061801 [1310.6732].

[19] P. F. Harrison, D. H. Perkins and W. G. Scott, *Tri-bimaximal mixing and the neutrino oscillation data*, *Phys. Lett.* **B530** (2002) 167 [hep-ph/0202074].

[20] P. Ballett, S. F. King, S. Pascoli, N. W. Prouse and T. Wang, *Precision neutrino experiments vs the Littlest Seesaw*, *JHEP* **03** (2017) 110 [1612.01999].

[21] S. S. Chatterjee, P. Pasquini and J. Valle, *Probing atmospheric mixing and leptonic CP violation in current and future long baseline oscillation experiments*, *Phys. Lett. B* **771** (2017) 524 [1702.03160].

[22] G.-J. Ding, Y.-F. Li, J. Tang and T.-C. Wang, *Confronting Tri-direct CP-symmetry models to neutrino oscillation experiments*, *Phys. Rev.* **D100** (2019) 055022 [1905.12939].

[23] J. Tang and T. Wang, *Study of a tri-direct littlest seesaw model at MOMENT*, *Nucl. Phys. B* **952** (2020) 114915 [1907.01371].

[24] P. Huber, M. Lindner and W. Winter, *Simulation of long-baseline neutrino oscillation experiments with GLoBES (General Long Baseline Experiment Simulator)*, *Comput. Phys. Commun.* **167** (2005) 195 [hep-ph/0407333].

[25] P. Huber, J. Kopp, M. Lindner, M. Rolinec and W. Winter, *New features in the simulation of neutrino oscillation experiments with GLoBES 3.0: General Long Baseline Experiment Simulator*, *Comput. Phys. Commun.* **177** (2007) 432 [hep-ph/0701187].

[26] DUNE collaboration, *Experiment Simulation Configurations Used in DUNE CDR*, 1606.09550.

[27] G.-J. Ding, S. F. King and C.-C. Li, *Tri-Direct CP in the Littlest Seesaw Playground*, *JHEP* **12** (2018) 003 [1807.07538].

[28] S. S. Wilks, *The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses*, *Annals Math. Statist.* **9** (1938) 60.