GLoBES Tutorial: Simulating accelerator neutrino experiments

GLoBES workshop in Heidelberg, Germany, January 24-26, 2007

Joachim Kopp 25.01.07

This tutorial will introduce some of the basic concepts required to simulate accelerator neutrino experiments with GLoBES (General Long Baseline Experiment Simulator). We will start from a simple and incomplete implementation of the T2K superbeam experiment, which we will gradually improve. We will first consider the precision measurement of the leading atmospheric oscillation parameters θ_{23} and Δm^2_{31}, and later discuss the possible detection of non-zero θ_{13} and δ_{CP}.

Part 1: Precision measurement of θ_{23} and Δm^2_{31}

Problem 1: Warm-up

Consider first the program in the directory `globes-tutorials/th23dm31/`, which computes χ^2 as a function of the fit values of θ_{23} and Δm^2_{31}, but is still lacking several important features. Compile the program by typing `make` and run it with the command `./th23dm31`. Each line of the output file `th23dm31.dat` will contain three numbers: the fit value for θ_{23} in degrees, the fit value for Δm^2_{31} in eV^2, and the corresponding χ^2. Use the script `th23dm31.gnuplot` to view the results as an EPS plot, which should look like Fig. 1.
Now, familiarize yourself with the C code (`th23dm31.c`) and the AEDL experiment definition (`T2K-tutorial.glb`). In doing so, you can already watch out for the aforementioned shortcomings of the code. There are essentially three of them, which we will one by one discuss below in problems 2, 4, and 5.

Problem 2: Spectral analysis vs. total rates

Let us examine Fig. 1 more closely: There is a very strong correlation between θ_{23} and Δm^2_{31}: A mixing angle far from its assumed "true" value of 45° is still compatible with the data, if, at the same time, a larger Δm^2_{31} is assumed. The reason for this is, that increasing Δm^2_{31} means that larger parts of the neutrino energy spectrum are affected by oscillations, which compensates the smaller oscillation amplitude. This woul be a severe limitation to the sensitivity of the experiments, but luckily, it can be remedied by an improved data analysis: Instead of performing a total rates analysis, we should incorporate spectral information. Then, we can determine the energy at which the first oscillation maximum occurs, and thus pin down Δm^2_{31}.
Accordingly, we have to increase the number of analysis bins (`$bins`) from `1` to `20` and select a different χ^2 function in `T2K-tutorial.glb`. Change the command `@sys_on_function = "chiTotalRatesTilt"` to `@sys_on_function = "chiSpectrumTilt"` in the first two rules (`#NU_E_Appearance_QE` and `#NU_MU_Disappearance_QE`). Similarly, replace the χ^2 function `@sys_off_function`, which is used if systematical errors are switched off, to `chiNoSysSpectrum`. Leave the charged current ν_e appearance rule unchanged to reflect the fact that the Super-Kamiokande detector cannot reconstruct the energy of CC ν_e events very well, so that a spectral analysis is not possible for them. The plot you will obtain

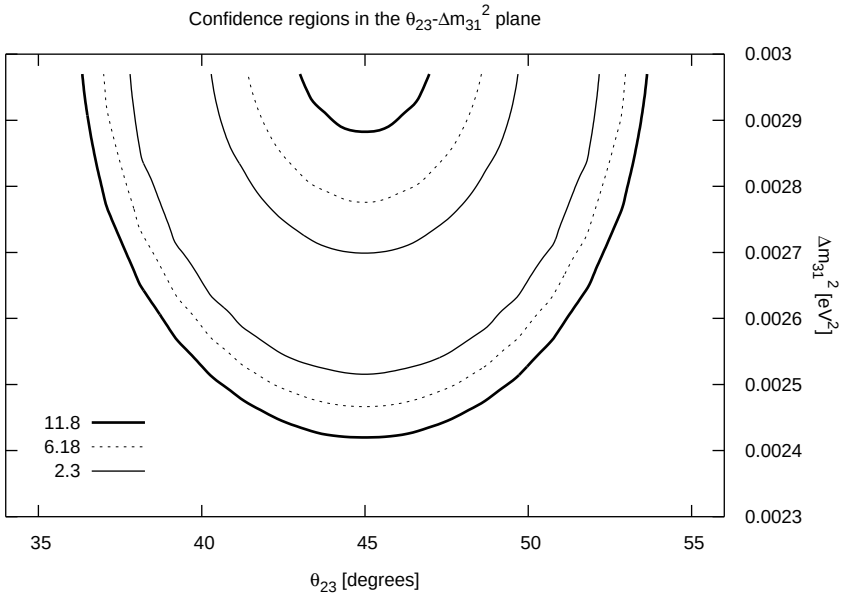


Figure 1: Output of `th23dm31`: Confidence regions in the θ_{23}–Δm^2_{31} plane for a (too) simple implementation of the T2K experiment.

after re-running the program, Fig. 2, looks much more like what we expect from T2K. The correlation between θ_{23} and Δm^2_{31} is still visible, but much less pronounced than before.

Problem 3: The octant degeneracy

Up to now, we have considered θ_{23} to be maximal. However, deviations Of about $\pm 5^\circ$ from this best fit value are still compatible with current data. Therefore, let us now consider the effect of $\theta_{23} = 40.0^\circ$. Modify `th23dm31.c` accordingly and compute the resulting confidence regions, which should resemble Fig. 3a. There are now two distinct regions in the parameter space, which are both compatible with the data at less than 1σ: The correct one around $\theta_{23} = 40.0^\circ$, and the degenerate one at $\theta_{23} = 90^\circ - 40.0^\circ$ (the so-called octant degeneracy). This looks correct, but let us now increase θ_{13} to a value close to the current upper bound, say, $\sin^2 2\theta_{13} = 0.1$. You will find that the degenerate solution is now excluded at 2σ (Fig. 3b). Try to understand this feature from the (approximate) analytical expressions for the neutrino oscillation probabilities:

$$\begin{aligned} P_{\mu\mu} = 1 &- \sin^2 2\theta_{23} \, \sin^2 \Delta + \alpha \, c_{12}^2 \, \sin^2 2\theta_{23} \, \Delta \, \sin 2\Delta \\ &- \alpha^2 \, \Delta^2 \left[\sin^2 2\theta_{12} \, c_{23}^2 + c_{12}^2 \, \sin^2 2\theta_{23} \left(\cos 2\Delta - s_{12}^2 \right) \right] + 4 \, s_{13}^2 \, s_{23}^2 \, \cos 2\theta_{23} \, \sin^2 \Delta \\ &- 2 \, \alpha \, s_{13} \, \sin 2\theta_{12} \, s_{23}^2 \, \sin 2\theta_{23} \, \cos \delta_{\rm CP} \, \Delta \, \sin 2\Delta \, , \end{aligned}$$

$$P_{\mu e}^{\rm vac} = \alpha^2 \, \sin^2 2\theta_{12} \, c_{23}^2 \, \Delta^2 + 4 \, s_{13}^2 \, s_{23}^2 \, \sin^2 \Delta + 2 \, \alpha \, s_{13} \, \sin 2\theta_{12} \sin 2\theta_{23} \, \cos(\Delta + \delta_{\rm CP}) \, \Delta \, \sin \Delta \, ,$$

where $\alpha = \Delta m^2_{21} / \Delta m^2_{31}$, $\Delta = \Delta m^2_{31} L / 4E$, $s_{ij} = \sin \theta_{ij}$, and $c_{ij} = \cos \theta_{ij}$.

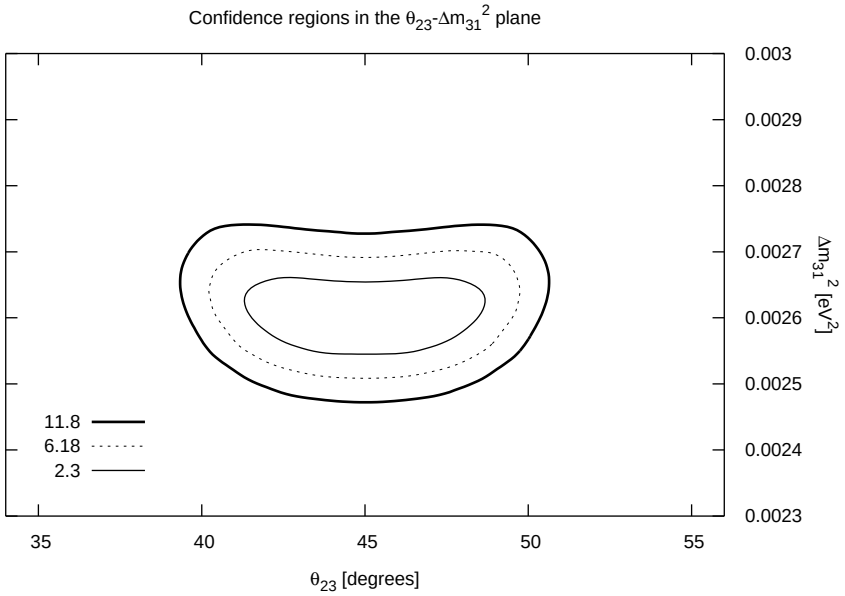
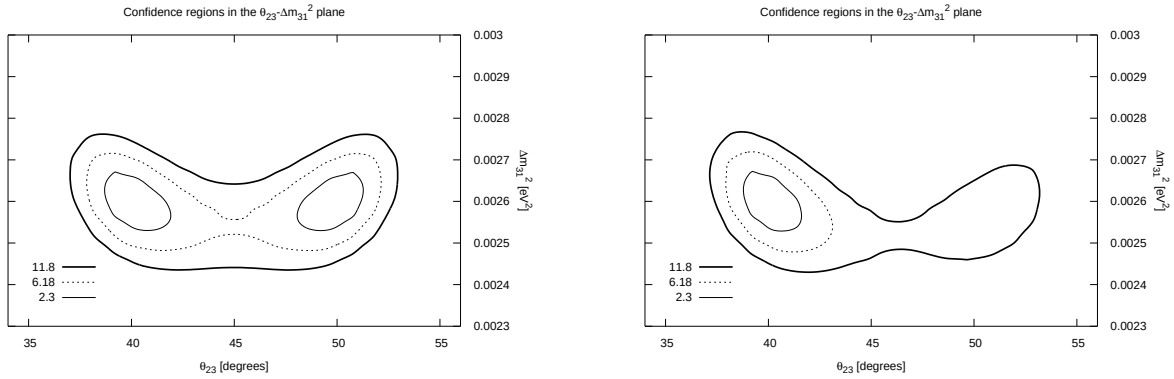


Figure 2: Solution of problem 2. Confidence regions in the θ_{23}–Δm^2_{31} plane with the inclusion of spectral information.



(a) (b)

Figure 3: Solution of problem 3: Confidence regions in the θ_{23}–Δm^2_{31} plane for non-maximal θ_{23} and (a) zero θ_{13}, (b) large θ_{13}.

Problem 4: Incorporation of correlations with θ_{13} and δ_{CP}

Fig. 3b should make us suspicious, since we do not expect T2K to have the capability to resolve the octant degeneracy. Indeed, we have so far assumed full knowledge about all oscillation parameters except θ_{23} and δ_{CP}, i.e. we have kept them fixed at their "true" values in our fits. Unless there were some theoretical argument (e.g. a flavor symmetry) pinning down these parameters, we should allow them to vary within their presently allowed ranges in the fit. This will yield smaller χ^2 values, i.e. the sensitivity will decrease.
The marginalization over multi-dimensional subspaces of the oscillation parameter space is one of the most powerful features of GLoBES. To enable it in our code, we use the API functions `glbAllocProjection`, `glbDefineProjection`, `glbSetDensityProjectionFlag`, and `glbSetProjection` to specify that θ_{12}, θ_{13}, Δm^2_{21}, and δ_{CP} should be marginalized over, while θ_{23}, Δm^2_{31}, and the matter density should be kept at their initial values. A short documentation of the required API functions is given in the appendix.
Furthermore, we have to specify the parameters of the prior terms

$$\chi^2_{\text{prior}} = \frac{(x - x_0)^2}{\sigma_x^2},$$

which can be added to χ^2 for all oscillation parameters x to include external information on these parameters. Set all x_0 to the `true_values` with the function `glbSetCentralValues`, and use `glbAllocParams`, `glbDefineParams`, `glbSetDensityParams`, and `glbSetInputErrors` to set

$$\sigma_{\theta_{12}} = 0.1 \cdot \theta_{12,\text{true}}$$
$$\sigma_{\Delta m^2_{21}} = 0.1 \cdot \Delta m^2_{21,\text{true}}.$$

For all other parameters (including the matter density), we choose to omit the prior terms by setting the respective σ_x to 0.
Finally, we have to replace the call to `glbChiSys` by one to `glbChiNP`. Due to the multi-dimensional minimization in the oscillation parameter space, the calculation may now take several minutes. For debugging, you can speed things up by switching off systematical errors with the command

```
glbSwitchSystematics(GLB_ALL, GLB_ALL, GLB_OFF);
```

This tells GLoBES to calculate χ^2 according to the `@sys_off_function` directives instead of the default `@sys_on_function`. For T2K, this means using the χ^2 functions `chiNoSysSpectrum` and `chiNoSysTotalRates`, which do not include any free systematical biases. In contrast to this, the default functions `chiSpectrumTilt` and `chiTotalRatesTilt` leave the normalization and tilt of the neutrino spectrum variable within the ranges defined by `@signalerror` and `@backgrounderror`. Thus, GLoBES has to fit these systematics parameters in addition to the oscillation parameters, which greatly increases the computational effort.

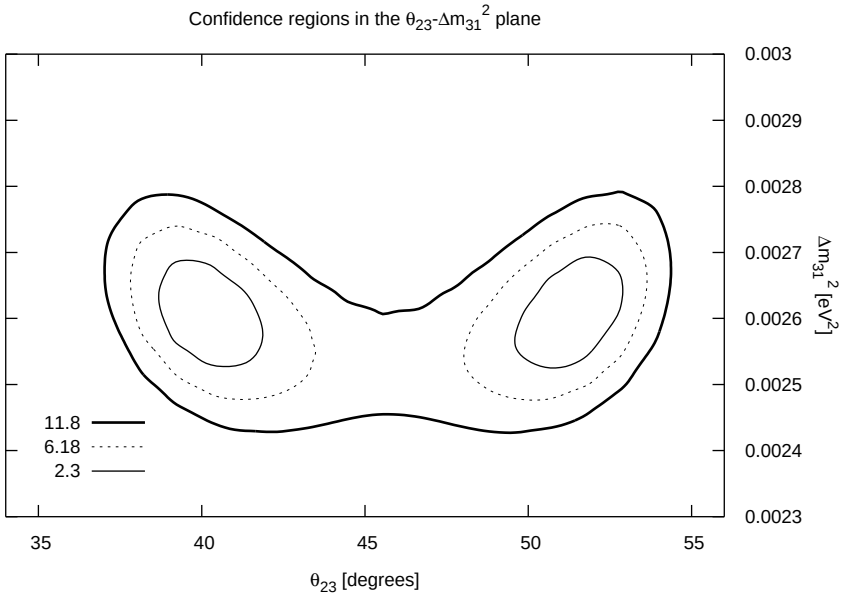


Figure 4: Solution of problem 4: Confidence regions in the θ_{23}–Δm^2_{31} plane for non-maximal θ_{23} and large θ_{13}, including parameter correlations.

Problem 5: The $\mathrm{sgn}(\Delta m^2_{31})$ degeneracy

Besides the octant degeneracy, there is another degeneracy in the three-flavor neutrino oscillation probabilities, which we have not considered ao far: The ambiguity in the sign of Δm^2_{31}. Indeed, if we considered the half-plane of negative Δm^2_{31}, we would find a mirror image of Fig. 4.
For the moment, however, we are only interested in the positions of the two solutions with $\Delta m^2_{31} < 0$. To find them, use the function `glbChiAll`, and choose the starting values such that the minimizer will converge to the appropriate local minima of χ^2. Create a new `glb_params` data structure and pass it to `glbChiAll` as the second argument to retrieve the position of the minimum.
Why is $|\Delta m^2_{31}|$ slightly smaller at the degenerate solutions than at the true one?

Part 2: Generic three-flavor effects: θ_{13} and δ_{CP}

Problem 6: Confidence regions in the θ_{13}–δ_{CP} plane

One of the main aims of superbeam experiments is the detection of generic three-flavor effects, in particular of non-zero θ_{13} or δ_{CP}. To examine the capability of T2K to measure these parameters, we provide the program `th13delta` and the accompanying script `th13delta.gnuplot`, which produces a confidence plot in the θ_{13}–δ_{CP} plane. For simplic-

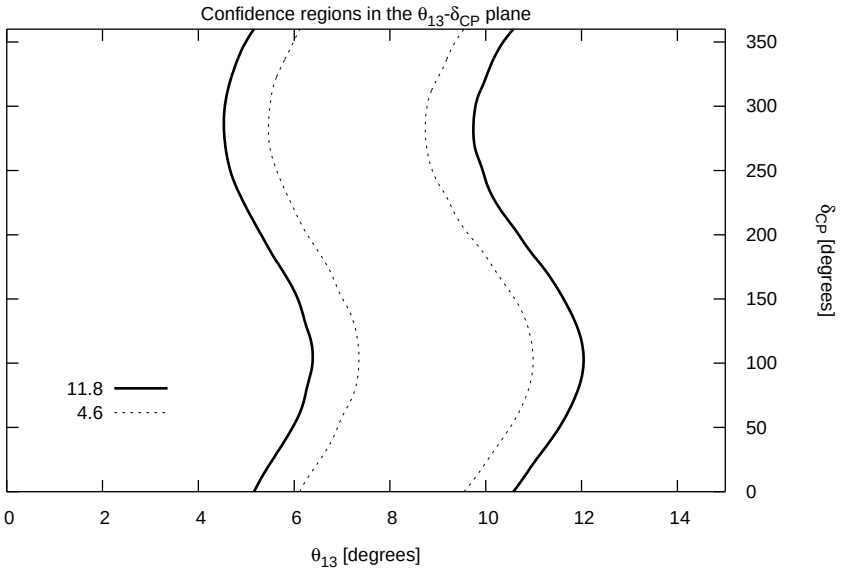


Figure 5: Output of `th13delta`: Confidence regions in the θ_{13}–δ_{CP} plane for a fit with the normal mass hierarchy. The "true" parameter values are $\theta_{23} = 45°$, $\Delta m^2_{31} = 2.6 \cdot 10^{-3}$ eV^2, $\sin^2 2\theta_{13} = 0.1$, and $\delta_{CP} = 90°$.

ity, the program only considers the normal mass hierarchy in the fit, i.e. we assume that the sgn(Δm^2_{31}) degeneracy has been resolved independently, e.g. with supernova neutrinos. Furthermore, we have set the "true" θ_{23} back to 45°, so that we can also neglect the octant degeneracy.
Fig. 5 shows the results of the simulation. As we can see from the plot, there is a strong correlation between θ_{13} and δ_{CP}, which makes it impossible to efficiently constrain the CP phase. In the remainder of this tutorial, we will consider two strategies for breaking this correlation.

<u>Problem 7:</u> Improving the sensitivity by anti-neutrino running

One idea for improving the sensitivity of superbeam experiments is to operate them in the anti-neutrino mode for several years after the neutrino running. Although the cross sections, and thus the event numbers, in the anti-neutrino mode are smaller by a factor of three than those for neutrinos, one can benefit from the fact that the dependence of the anti-neutrino oscillation probabilities on δ_{CP} is different. To see the effect of this, we need to incorporate a new flux definition, several new channels, and the appropriate rules in th AEDL file. The parameters of these new AEDL environments are the same as for the existing ones, so you can start by simply duplicating these. Give them new names, change the flux file to `JHFminus.dat`, and reverse the CP signs for the new channels.
As you can see from Fig. 6, the anti-neutrino running will exclude some δ_{CP} values at the

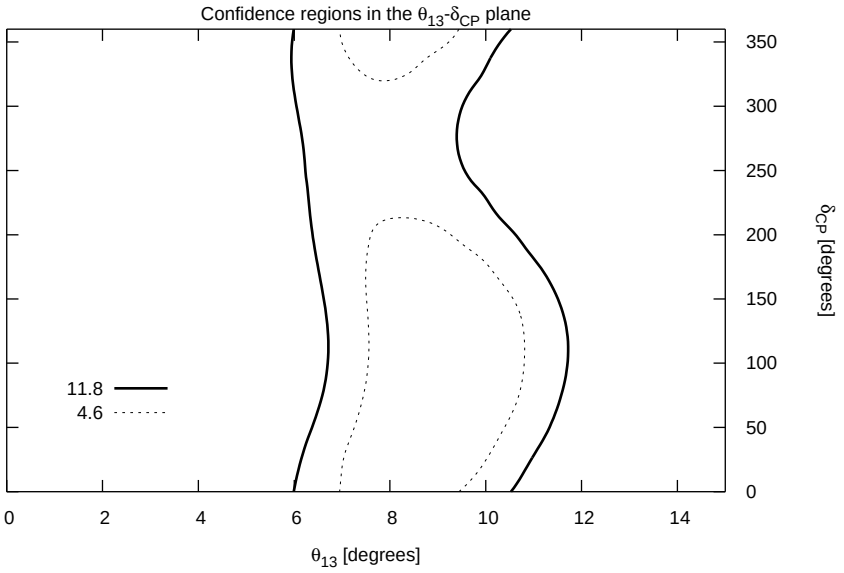


Figure 6: Solution of problem 7: Confidence regions in the θ_{13}–δ_{CP} plane for 3 years of neutrino running, followed by 3 years of anti-neutrino running, and assuming the normal mass hierarchy.

90% C.L., but fails to do so at 3σ.

Problem 8: Improving the sensitivity by incorporating reactor results

Another option for breaking some of the correlations in T2K is the combination with an advanced reactor neutrino experiment. To study this, incorporate `Reactor2.glb` into your simulation by adding an appropriate call to `glbInitExperiment`. Re-run the program to obtain Fig. 7, which shows that an advanced reactor experiment can measure θ_{13} with a great precision. Moreover, since the ν_e disappearance channel is independent of δ_{CP}, it does not suffer from parameter correlations. Therefore, the parameter constraints that can be set by a combination of T2K and reactor data, are similar to those obtainable with the much more expensive anti-neutrino running. Note, however, that this is only true as long is θ_{13} is large. For smaller values, one cannot so easily benefit from reactor data.

Appendix: Documentation of required API functions

`glb_params glbAllocParams()` *allocates the memory space needed for a parameter vector and returns a pointer to it.*

`glb_projection glbAllocProjection()` *allocates the memory space needed for a projection definition, and returns a pointer to it.*

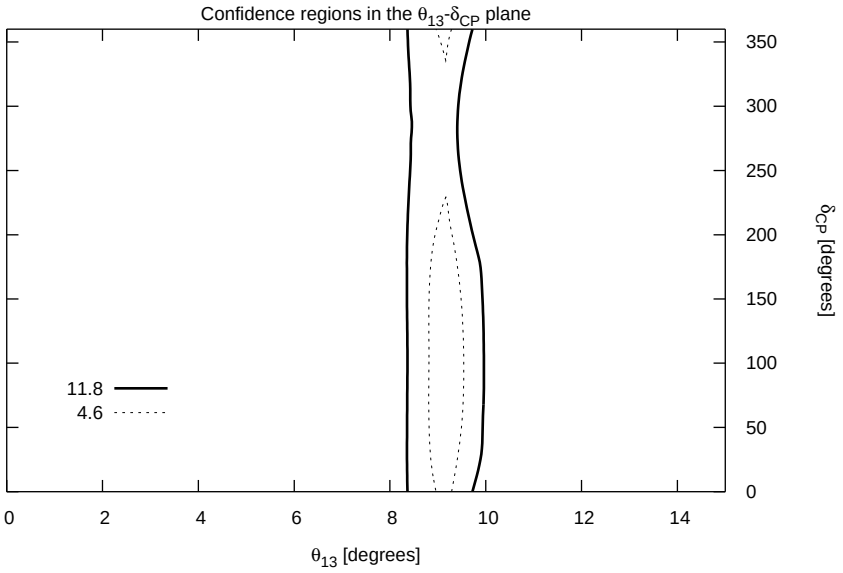


Figure 7: Solution of problem 8: Confidence regions in the θ_{13}–δ_{CP} plane for 3 years of T2K neutrino running, combined with `Reactor2` data (exposure of 8000 GW kt yrs), and assuming the normal mass hierarchy.

`double glbChiAll(const glb_params in, glb_params out, int exp)` *returns the minimized* χ^2 *over all parameters for the experiment* `exp`*. For the simulation of all initialized experiments, use* `GLB_ALL` *for* `exp`*. The values in* `in` *are the guessed fit values for the minimizer. The actually determined parameters at the minimum are returned in* `out`*. If* `out` *is set to* `NULL`*, this information will not be returned.*

`double glbChiNP(const glb_params in, glb_params out, int exp)` *returns the projected* χ^2 *onto the hyperplane specified by* `glbSetProjection` *for the experiment* `exp`*. For the simulation of all initialized experiments, use* `GLB_ALL` *for* `exp`*. The values in* `in` *are the guessed fit values for the minimizer (all free parameters) and the fit values on the hyperplane (all fixed parameters). The actually determined parameters at the minimum are returned in* `out`*, where the fixed parameters are still at their input values. If* `out` *is set to* `NULL`*, this information will not be returned.*

`glb_params glbDefineParams(glb_params in, double theta12, double theta13, double theta23, double delta, double dms, double dma)` *assigns a set of oscillation parameters to the vector* `in`*, which has to be allocated before. The return value is the pointer to* `in` *if the assignment was successful, and* `NULL` *otherwise.*

`glb_projection glbDefineProjection(glb_projection in, int theta12, int theta13,int theta23, int delta, int dm21, int dm31)` *defines a projection in the oscillation parameter space. For each parameter, the function expects one of the flags*

`GLB_FREE` *or* `GLB_FIXED`*, indicating whether the respective parameter is to me marginalized over or not. The return value is the pointer to* `in` *if the assignment was successful, and* `NULL` *otherwise.*

`void glbFreeParams(glb_params stale)` *frees the memory needed for a parameter vector* `stale`*.*

`void glbFreeProjection(glb_projection stale)` *frees the memory needed for the projection definition vector* `stale`*.*

`int glbInitExperiment(char *inf, glb_exp *in, int *counter)` *adds a single experiment with the filename* `inf` *to the list of currently loaded experiments. The* `counter` *is a pointer to the variable containing the number of experiments (normally* `&glb_num_of_exps`*), and* `in` *points to the beginning of the experiment list (normally* `&glb_experiment_list[0]`*). The function returns zero if it was successful.*

`int glbSetCentralValues(const glb_params in)` *sets the central values for the prior terms to* `in`*.*

`glb_params glbSetDensityParams(glb_params in, double dens, int which)` *sets the density parameter for experiment number* `which` *in the structure* `in` *to the value* `dens`*. If the assignment was unsuccessful, the function returns* `NULL`*. If* `GLB_ALL` *is used for* `which`*, the density parameters of all experiments will be set.*

`glb_projection glbSetDensityProjectionFlag(glb_projection in, int flag, int which)` *specifies whether the matter density in experiment number* `which` *should be marginalized over (*`flag` $=$ `GLB_FREE`*) or not (*`flag` $=$ `GLB_FIXED`*). The return value is the pointer to* `in` *if the assignment was successful, and* `NULL` *otherwise.*

`int glbSetInputErrors(const glb_params in)` *sets the input errors for all of the following minimizer calls to* `in`*. An input error of* 0 *corresponds to not taking into account the respective prior.*

`int glbSetProjection(const glb_projection in)` *sets the projection to* `in`*. The return value is* 0 *if successful, and* -1 *if unsuccessful.*