Tutorial 1: Sterile neutrino at reactor experiments – solutions

S. Gariazzo

March 24, 2021

The purpose of this first tutorial is to understand the phenomenology of sterile neutrinos at reactor experiments. The results are obtained with `GLoBESfit` and plotted using python. If you have problems in compiling `GLoBESfit`, you can download some output files from `http://personalpages.to.infn.it/~gariazzo/courses/2103_GGI_nus/globesfit.tar.gz`.

1 Compare spectrum and rate constraints

The first thing to learn is how to use `GLoBESfit` to compute the χ^2. Once `GLoBESfit` is properly installed and compiled, we can produce the profiles using the `source/glf_rate` and `source/glf_spectrum`, respectively to obtain the sum of the χ^2 from all the rate or spectrum experiments. The contours for the first figure 1 are obtained using the following commands:

```
cd path/to/GLoBESfit
mkdir -p output/spectrum output/rate plots/
source/glf_spectrum -o output/spectrum/all.dat
source/glf_rate -o output/rate/all.dat
```

In both cases, the output files (`output/spectrum/all.dat` and `output/rate/all.dat`) contain the values of $\log_{10}\sin^2\theta_{14}$, $\log_{10}\Delta m^2_{41}$ [eV^2] and the χ^2. Since the scan is performed, by default, on $N = 51$ points per variable, the files are 2601 lines long.

Here you can see an example where I used python to read the files and produce the figure:

```
import matplotlib.pyplot as plt #for plots
import numpy as np #numerical utilities

# chi2 levels for contours at (0,) 1, 2, 3 sigma with 2 d.o.f.
levels = [0, 2.30, 6.18, 11.83]

class Dataset: #I will use a class to read the files and store some useful quantities
    def __init__(self, filename, label="", c="k"): #initialize the class
        self.filename = filename
        self.color = c
        self.label = label
        self._read()

    def _read(self): #read the .dat file
        try: #check that the file exists and is readable
            self.data = np.loadtxt(self.filename)
        except (IOError, ValueError):
            print("File not found or corrupted: %s"%self.filename)
            return
        try: #check that it is possible to extract all the columns correctly
            self.s14 = 10** np.unique(self.data[:, 0])
            self.dm41 = 10** np.unique(self.data[:, 1])
            chi2 = self.data[:, 2]
        except IndexError:
```

```
            print("File content is not valid: wrong structure of indices")
            return
        try: #transform the last column in a 2D matrix
            self.chi2 = chi2.reshape(len(self.dm41), len(self.s14)).T
        except ValueError: #if it fails, the file is corrupted or incomplete
            print("Wrong number of lines, cannot reshape the chi2")
            return
        self.minchi2 = np.min(self.chi2) #extract minimum of the chi^2
        self.chi2 = self.chi2-self.minchi2 #replace chi^2 with Delta chi^2

plt.figure() #create an empty figure
#perform a loop, for each iteration we will have filename, color and label
for f, c, l in zip(
    ["output/spectrum/all.dat", "output/rate/all.dat"],
    ["r","b"],
    ["spectra", "rates"]
):
    d = Dataset(f) #read the file f and store the objects in the class
    plt.contour( #plot contours in a 2D plane
        d.s14,
        d.dm41,
        d.chi2,
        levels=levels, #define the levels using the standard Delta chi^2
        colors=c,
        linewidths=1,
        linestyles=["-","-","--", ":"], #each contour has a different style
    )
    plt.plot(np.nan, c=c, label=l)
#set logscales
plt.xscale("log")
plt.yscale("log")
#set axis labels
plt.xlabel(r"$\sin^2\theta_{14}$")
plt.ylabel(r"$\Delta m^2_{41}$ [eV$^2$]")
#set axis limits
plt.xlim([1e-3,1])
plt.ylim([1e-2,10])
#set legend, adjust the layout and save the plot
plt.legend(loc="upper left")
plt.tight_layout()
plt.savefig("plots/spe_rat.pdf")
```

The code will save in a new file `plots/spe_rat.pdf` something very similar to figure 1.

2 Compare single rate or spectrum experiments

The plots comparing some selected rate (figure 2) or spectrum (figure 3) experiments are obtained in a very similar way with respect to the previous case. The only difference is that we need to use `GLoBESfit` with some options, in order to compute the χ^2 for some subset of experiments. While you are encouraged to try all the combinations, here you will find the commands required to obtain the curves shown in the figures. Notice that some of the curves were already obtained in the previous exercise.

```
# groups of experiments excluded by the -b option for glf_rate:
# 0: Bugey-4 + Rovno 91
# 1: Bugey-3 (15 m+ 40 m + 95 m)
# 2: Gösgen (38 m + 46 m + 65 m) + ILL
```

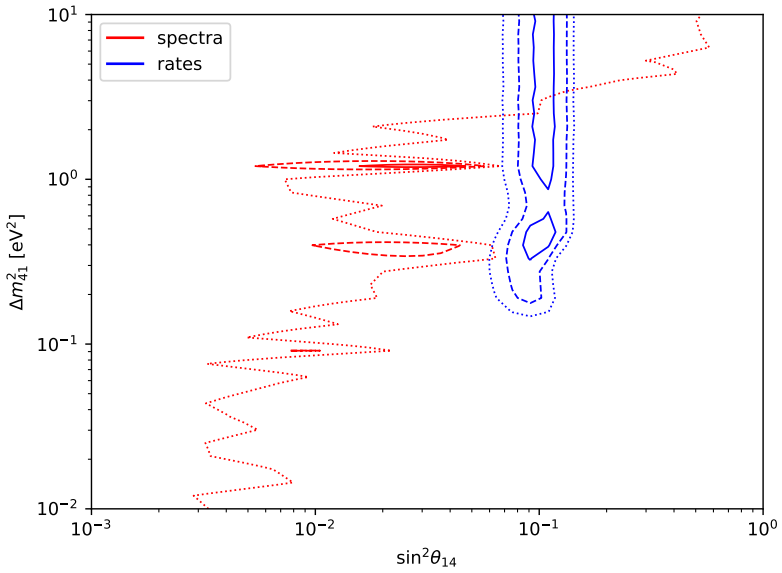

Figure 1: Comparison of 1, 2, 3σ constraints from rate and spectrum analyses of reactor antineutrino data.

```
# 3: Krasnoyarsk 87 (33 m + 92 m)
# 4: Krasnoyarsk 94
# 5: Krasnoyarsk 99
# 6: Savannah River (18 m)
# 7: Savannah River (24 m)
# 8: Rovno 88 (1I + 2I + 1S + 2S + 3S)
# 9: Nucifer
# 10: Palo Verde (750 m + 890 m)
# 11: Double Chooz (355 m + 469 m)
# 12: Chooz (998 m + 1115 m)
# 13: Daya Bay (EH1 + EH2)
# 14: RENO
source/glf_rate -b0 -b1 -b2 -b3 -b4 -b5 -b7 -b8 -b9 \
    -b10 -b11 -b12 -b13 -b14 -o output/rate/sav18.dat
source/glf_rate -b0 -b1 -b2 -b3 -b4 -b5 -b6 -b8 -b9 \
    -b10 -b11 -b12 -b13 -b14 -o output/rate/sav24.dat
source/glf_rate -b0 -b1 -b3 -b3 -b4 -b5 -b6 -b7 -b9 \
    -b10 -b11 -b12 -b13 -b14 -o output/rate/rov88.dat
source/glf_rate -b0 -b1 -b2 -b3 -b4 -b5 -b6 -b7 -b8 \
    -b9 -b10 -b12 -b13 -b14 -o output/rate/dc.dat
source/glf_rate -b0 -b1 -b2 -b3 -b4 -b5 -b6 -b7 -b8 \
    -b9 -b10 -b11 -b13 -b14 -o output/rate/chooz.dat
source/glf_rate -b0 -b1 -b2 -b3 -b4 -b5 -b6 -b7 -b8 \
    -b9 -b10 -b11 -b12 -b13 -o output/rate/reno.dat

# groups of experiments excluded by the -b option for glf_spectrum:
#0: DANSS
#1: DayaBay
#2: NEOS
#3: DoubleChooz
#4: Bugey
#5: Reno
source/glf_spectrum -b1 -b3 -b4 -b5 -o output/spectrum/danss_neos.dat
source/glf_spectrum -b0 -b2 -o output/spectrum/no_danss_neos.dat
```

The plots can be performed using the same python code presented above, with few small variations:

```
import matplotlib.pyplot as plt #for plots
import numpy as np #numerical utilities

# chi2 levels for contours at (0,) 1, 2, 3 sigma with 2 d.o.f.
levels = [0, 2.30, 6.18, 11.83]

class Dataset: #I will use a class to read the files and store some useful quantities
    def __init__(self, filename, label="", c="k"): #initialize the class
        self.filename = filename
        self.color = c
        self.label = label
        self._read()

    def _read(self): #read the .dat file
        try: #check that the file exists and is readable
            self.data = np.loadtxt(self.filename)
        except (IOError, ValueError):
            print("File not found or corrupted: %s"%self.filename)
            return
        try: #check that it is possible to extract all the columns correctly
            self.s14 = 10** np.unique(self.data[:, 0])
            self.dm41 = 10** np.unique(self.data[:, 1])
            chi2 = self.data[:, 2]
        except IndexError:
            print("File content is not valid: wrong structure of indices")
            return
        try: #transform the last column in a 2D matrix
            self.chi2 = chi2.reshape(len(self.dm41), len(self.s14)).T
        except ValueError: #if it fails, the file is corrupted or incomplete
            print("Wrong number of lines, cannot reshape the chi2")
            return
        self.minchi2 = np.min(self.chi2) #extract minimum of the chi^2
        self.chi2 = self.chi2-self.minchi2 #replace chi^2 with Delta chi^2

#first figure
plt.figure() #create an empty figure
#perform a loop, for each iteration we will have filename, color and label
for f, c, l in zip(
    ["sav18.dat", "sav24.dat", "rov88.dat", "dc.dat", "chooz.dat", "reno.dat", "all.dat"],
    ["r","b","g","c","y","m","k"],
    ["sav18", "sav24", "rov88", "doublechooz", "chooz", "reno", "all"]
):
    d = Dataset("output/rate/" + f)
    plt.contour( #plot contours in a 2D plane
        d.s14,
        d.dm41,
        d.chi2,
        levels=levels, #define the levels using the standard Delta chi^2
        colors=c,
        linewidths=3 if "all.dat" in f else 1, #thicker lines for "all"
        linestyles=["-","-","--", ":"], #each contour has a different style
    )
    plt.plot(np.nan, c=c, label=l)
#set logscales
plt.xscale("log")
plt.yscale("log")
```

```
#set axis labels
plt.xlabel(r"$\sin^2\theta_{14}$")
plt.ylabel(r"$\Delta m^2_{41}$ [eV$^2$]")
#set axis limits
plt.xlim([1e-2,1])
plt.ylim([1e-2,10])
#set legend, adjust the layout and save the plot
plt.legend(loc="upper left")
plt.tight_layout()
plt.savefig("plots/rates.pdf")

#second figure
plt.figure() #create an empty figure
#perform a loop, for each iteration we will have filename and color
for f, c in zip(
    ["all", "danss_neos", "no_danss_neos"],
    ["k", "#ff9933", "#669900"],
):
    d = Dataset("output/spectrum/%s.dat" % f) #string formatting: replace %s with f
    plt.contour( #plot contours in a 2D plane
        d.s14,
        d.dm41,
        d.chi2,
        levels=levels, #define the levels using the standard Delta chi^2
        colors=c,
        linewidths=3 if "all.dat" in f else 1, #thicker lines for "all"
        linestyles=["-","-","--", ":"], #each contour has a different style
    )
    plt.plot(np.nan, c=c, label=f)
#set logscales
plt.xscale("log")
plt.yscale("log")
#set axis labels
plt.xlabel(r"$\sin^2\theta_{14}$")
plt.ylabel(r"$\Delta m^2_{41}$ [eV$^2$]")
#set axis limits
plt.xlim([1e-2,1])
plt.ylim([1e-2,10])
#set legend, adjust the layout and save the plot
plt.legend(loc="upper left")
plt.tight_layout()
plt.savefig("plots/spectra.pdf")
```

From the figure 2 we can see that some of the experiments provide an upper bound (Chooz, for example), while others have a closed preferred region around $\sin^2\theta_{14} \sim 0.1$. This is a consequence of the fact that some experiments measure a rate in full agreement with the theoretical expectation, while others observe a smaller rate (see e.g. slide 48 from the first part of the lectures). The fact that some of the experiments are unable to constrain small mass splittings is a consequence of the distance at which each of them measures oscillations: when $\Delta m^2_{41} L/E$ is too small, there can be no constraint on the suppression of the observed rate, because it cannot be generated by active-sterile oscillations.

In order to answer the last two questions, let us open a slightly different version of the last plot, which you should now how to produce by now. Look at figure 4: you will notice that the most constraining experiment at high mass splittings is DayaBay. The line is independent of the mixing angle at such large mass splittings: oscillations are averaged out because of the smearing due to energy resolution of the experiment, and averaging of osillations in the detector. The limit by DayaBay here is stronger than those of other experiments thanks to the fact that DayaBay has three different detectors at different distances, and a very

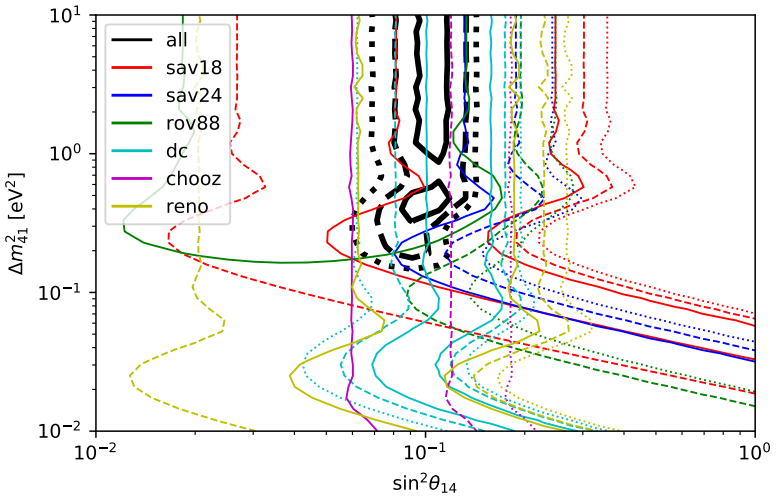

Figure 2: Comparison of the constraints from few different rate analyses.

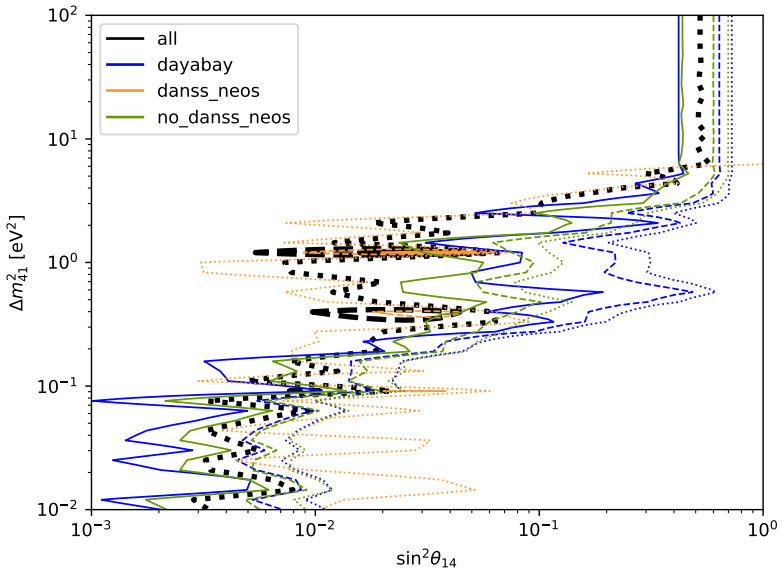

Figure 3: Comparison of the constraints from few different spectrum analyses.

high number of events. Oscillations are averaged out at such distances, so one can use the global suppression of the observed antineutrino spectrum at the different distances to decouple the effect of other oscillation channels (θ_{13}) from active-sterile oscillations, to finally obtain a constraint on θ_{14}. Experiments such as DANSS, operating at 10-12 m from the nuclear reactor, cannot do this because for the considered Δm^2_{41} it cannot measure the unoscillated flux (oscillations are already present at $\Delta m^2_{41} L/E \sim 10$ eV$^2 \times 10$ m/4 MeV for the nearest detector position), nor the global suppression of oscillations at large distances (they are not completely averaged out at the farthest detector position).

You will also be able to see that at the smallest Δm^2_{41}, DANSS is not competitive with NEOS, Double-Chooz, DayaBay and RENO. Again, the reason is that these experiments have detectors located at larger distances from the reactors providing the antineutrino flux. Oscillations due to $\Delta m^2_{41} \lesssim 0.1$ eV^2 in the DANSS detectors, even at the farthest position, cannot develop because the term $\Delta m^2_{41} L/E \lesssim 0.1$ eV$^2 \times 10$ m/4 MeV is too small. DayaBay, working at much larger distances and with higher statistics, provides instead strong constraints. This is not a surprise: remember that it is designed to be able to probe active neutrino oscillations at $\Delta m^2_{31} \sim 2.5 \times 10^{-3}$ eV^2 with very high precision.

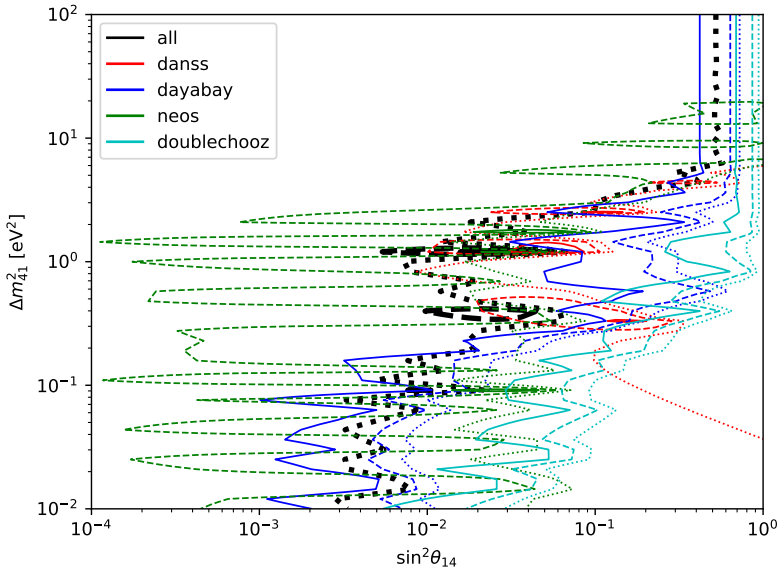

Figure 4: Comparison of the constraints from few different spectrum analyses.

3 Reactor flux models

For the final plot, shown in figure 5 we have to compute the contours with the different models for the reactor antineutrino flux:

```
source/glf_rate -S -o output/rate/all_S.dat
source/glf_rate -M -o output/rate/all_M.dat
source/glf_rate -H -o output/rate/all_H.dat
```

Again, the plot is performed using the same codes shown before, so I will not repeat the new one here. You can notice from the figure that the results can change significantly if a different antineutrino flux is considered, or if possible systematic errors are taken into account. The revisited constraints are more compatible with the results from spectrum experiments. Remember also that spectrum measurements are almost independent of the theoretical antineutrino flux.

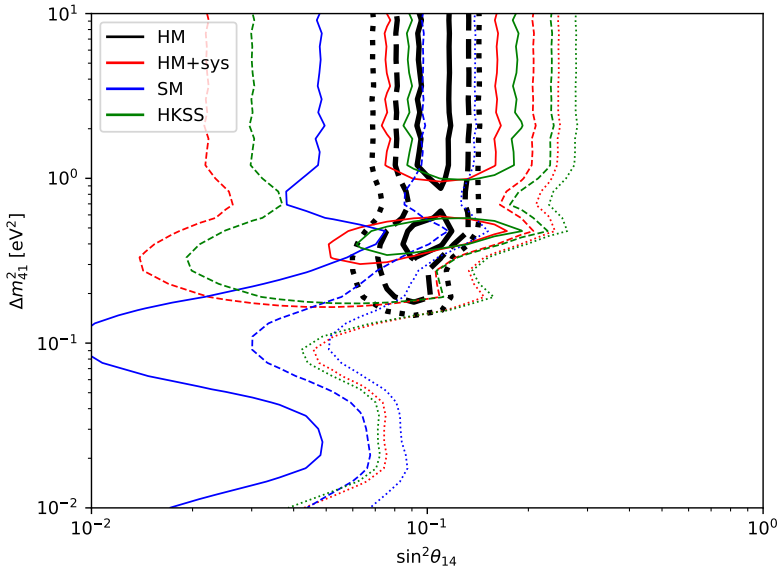

Figure 5: Comparison of the constraints from the rate experiments using different models for the reactor antineutrino flux.