KAVLI IPMU INSTITUTE FOR THE PHYSICS AND MATHEMATICS OF THE UNIVERSE

東京大学 THE UNIVERSITY OF TOKYO

Validation p-theta – Release 2018

Benjamin Quilain (Kavli IPMU, The University of Tokyo)

T2K

OA forum meeting, 2018/11/15

Introduction

- The following slides proposes to learn how to run the basic parts of the p-theta code through some simple examples.

- A description of the different codes is provided in a README located in the apps/ folder of the code.

- The results shown here are produced using the 2018 Asimov A (see TN 367).

- The slides are divided in two sets :

  1. A quick validation with a limited number of MC throws, which is too limited to cover the phase space of nuisance parameters and therefore, to provide a proper marginalization.
  2. Retrieve a simple sensitivity result from 2018 with 20,000 throws

  → Check the « validation » set of slides.

Inputs

- All inputs necessary for this validation can be found in inputs_2018.tar.gz which is provided with the release.

- Here is a very quick summary of files needed to run the code :

1. inputs/Matrices/Ptheta_Matrix_postfit_run1-8_2017bxsec_CC1pi_Datafit.root
2. inputs/PDFNue/numode_nue_binning/*, numode_cc1pi_binning/* & antinu_nue_binning/*
3. inputs/PDFNumu/numode_numu_binning.root & antinu_numu_binning
4. inputs/RFNue/nubeam/* , nu_beam_cc1pi/* & nubar_beam/*
5. inputs/SplineFile/2017v4.root

- To run the 2nd part of these validations studies, additional inputs required :

1. inputs/Throws/Osc_RC_20k.root
2. inputs/Throws/Syst_20k.root
3. inputs/Validation_ToyXP_AsimovA_2018.root

Asimov data set

Parameters	Set A	Set B
Δm^2_{21}	7.53×10^{-5} eV^2	7.53×10^{-5} eV^2
Δm^2_{32}	2.509×10^{-3} eV^2	2.509×10^{-3} eV^2
$\sin^2\theta_{23}$	0.528	0.45
$\sin^2\theta_{12}$ ($\sin^2 2\theta_{12}$)	0.304 (0.846)	0.304 (0.846)
$\sin^2\theta_{13}$ ($\sin^2 2\theta_{13}$)	0.0219 (0.0857)	0.0219 (0.0857)
δ_{CP}	-1.601	0
Earth matter density	2.6 g/cm^3	2.6 g/cm^3
Baseline length	295 km	295 km
Mass hierarchy	Normal	Normal

- Only Asimov A is used in these validations slides.

- These values are used to generate the input inputs/Validation_ToyXP_AsimovA_2018.root , and are hardcoded in codes that generates toy XP (ex : GenerateXP, MakeBreakdown...).
  → Search for : OP->ComputeOscProb(0.0857, -1.601, 2.509e-3, 0.528, 2.509e-3, 0.528);

Asimov data set

- For all these studies, the assumed number of neutrino corresponds to the full run 1-9d analysis :
  1.4938×10^{21} POT in FHC & 1.6346×10^{21} POT in RHC.

- These values are given as inputs of the different codes through the command line :
  -tnu 1.4938
  -tnb 1.6346

- New from 2018 : There is now a shell/RunAtStart.sh file to help you with the different inputs. Please :

1. Update RunAtStart.sh giving it the location of the inputs from p3 (if you unzipped the inputs directly in the Minimal/ folder, nothing needs to be changed).

2. source shell/RunAtStart.sh

I. Validate the number of expected events

- You can first check if the number of predicted events in each sample assuming Asimov A corresponds to default.

- For this purpose, you may run MakeBreakdown. It is automated by entering shell/ repository, and running : ./RunMakeBreakdown.sh.

- Results are shown on next slide. If you have a different please check :

1. That the oscillation parameter value in apps/MakeBreakdown.cxx corresponds to Asimov A from p4 (OP->ComputeOscProb(0.0857, -1.601, 2.509e-3, 0.528, 2.509e-3, 0.528);). If you change it, don't forget to do : make clean before making again !

2. Your number of POT in FHC & RHC in RunAtStart.sh (which are used in RunMakeBreakdown.sh) corresponds to previous slides.

3. The covariance matrix, which is used to re-tune your MC from pre-banff to post-banff nominal value, is the one given in the inputs (see also RunAtStart.sh).

4. Investigate the issue more seriously or contact the p-theta-convener

I. Validate the number of expected events

Reference expected number of nue events: 74.4645
Neutrino flavors
nue_sig:60.3397, numu:3.9823, numubar:0.202795, nue:9.17554, nuebar:0.389085, nuebar_sig:0.375129

Reference expected number of numu events: 272.341
Neutrino flavors
nue_sig:0.072924, numu:255.934, numubar:16.1107, nue:0.202191, nuebar:0.0209357, nuebar_sig:0.000271535,

Reference expected number of nuebar events: 17.1528
Neutrino flavors
nue_sig:3.05615, numu:0.815757, numubar:1.37931, nue:1.71484, nuebar:2.45093, nuebar_sig:7.73578,

Reference expected number of numubar events: 139.474
Neutrino flavors
nue_sig:0.00403337, numu:55.2081, numubar:84.1307, nue:0.0766752, nuebar:0.0498413, nuebar_sig:0.00489014,

Reference expected number of CC1Pi nue events: 7.02698
Neutrino flavors
nue_sig:5.54169, numu:0.480668, numubar:0.0241791, nue:0.962071, nuebar:0.0107258, nuebar_sig:0.00763716

II. Validate the systematic uncertainty

- You can then check if the systematic uncertainty calculated mostly using the covariant matrix and response functions applied to Asimov A corresponds to default.

- For this purpose, you may run OneSigma. It is automated by entering shell/ repository, and running : ./RunOneSigma.sh.

- Results are shown on next slide.

II. Validate the systematic uncertainty

- You can then check if the systematic uncertainty calculated mostly using the covariant matrix and response functions applied to Asimov A corresponds to default.

- For this purpose, you may run OneSigma. It is automated by entering shell/ repository, and running : ./RunOneSigma.sh.

- Results are :

```
Error source & nue1R      & numu1R     & nuebar1R  & numubar1R & nue1RD \\
Beam & 4.4 & 4.3 & 4.2 & 4.1 & 4.5 \\
Xsec & 8.6 & 5.6 & 6.3 & 4.3 & 5.2 \\
Xsec (constr. by ND) & 4.8 & 4.8 & 4.1 & 4.1 & 3.8 \\
Beam + xsec (all) & 7.8 & 4.4 & 5.7 & 3.3 & 5.3 \\
$E_{b}$ & 7.3 & 3.3 & 4.2 & 1.3 & 2.3 \\
Flux+Xsec only (constr. by ND) & 3.2 & 3.3 & 3.1 & 2.9 & 4.0 \\
SK+FSI+SI only & 4.0 & 3.3 & 4.3 & 2.9 & 16.8 \\
All & 8.8 & 5.5 & 7.2 & 4.3 & 17.7 \\
```

III. Perform the first fits

- The toy to be fitted is located in inputs/Validation_ToyXP_AsimovA_2018.root
- The characteristics of the fitter can be modulated using apps/FitterSettings.h
- Fixes : number of oscillation parameters, how to use parameters (fix them = profiling, vary them on a grid, marginalize over them etc.).
- In this validation study, we use the following assumptions :

```
const char OscParNames[N_OSC_PAR]
[64]={"sin2_2theta13","delta_CP","dm2_32","sin2_theta23","dm2_32bar","sin2_theta23_bar"};

int N_Bins_OscPar[N_OSC_PAR]={ 81, 51, 21,81,21,81};  // Data fit

// -- Default values of the oscillation parameters
double Ref_Values_OscPar[N_OSC_PAR]={0.0830, -1.601, 2.509e-3, 0.528, 2.509e-3, 0.528};  //Validation run 1-7, set 1

//Data fit VALOR
double StartValue[N_OSC_PAR]={5.95e-2, -3.14159265358979312, 2.2e-3, 0.3, 2.2e-3, 0.3}; //First value on the grid
double LastValue[N_OSC_PAR]={0.1405,3.14159265358979312,2.8e-3,0.7,2.8e-3, 0.7};
```

III. Perform the first fits

- We will focus on showing T2K sensitivity as a function of true delta CP value
  → vary deltaCP on a grid (51 points) & marginalize over all other parameters :

In FitterSettings.h :

```
static const Int_t N_OSC_PAR =  6;//number of used oscillation parameters
static const Bool_t DIFF_OSCPAR =  0;//oscillation parameters theta23 and Deltam23 in nu and nubar: 0->same, 1->different

enum OscPar{ s13=0,          //sin2(2theta13)
             delta,          //delta_CP
           dm32,           //Delta m2 32 in nu mode
           s23,            //sin2(theta23) in nu mode
           dm32_bar,       //Delta m2 32 in nubar mode
           s23_bar};       //sin2(theta23) in nubar mode


int UsedPar[N_OSC_PAR]={2,          //Whether parameters are varied on the grid
                        1,          //0: parameter is fixed
                        2,          //1: parameter is varied on the grid
                        2,          //2: parameter is marginalized over at each point of the grid
                        0,          //nue-> (1,*,2,2,2,2)
                        0};         //numu->(2,2,2,2,1,1)
```

III. Perform the first fits

- There are basically two families of fitters to fit Asimov or toy XP :

  1. MargAsimov / MargFitter → Calculate the marginal likelihood at each point of the GRID defined in FitterSetting separately.

  2. IntAsimov / IntFitter → Calculate the marginal likelihood for several points of the GRID in parallel.

- Remark : **Asimov family fits naturally the number of expected events, while **Fitter family fits the number of observed events.

- In all the fitters here, I will use 100 marginalization toys for this validation. This number is hardcoded in shell/RunAtStart.sh in the environment variable : number_of_throws

III. Perform the first fits

- For real sensitivity studies, we will use 20,000 marginalization throws. Note that 2018 analysis in TN367 was performed with 20,000 marginalization toys.

III. a. Run MargAsimov

- You can first fit the toys located in inputs/Validation_ToyXP_AsimovA_2018.root with MargFitter

- For this purpose, it is automated by entering shell/ repository, and running : ./RunMargAsimov.sh.

- Results will be generated for all the 51 points of different deltaCP values. You can compare your results for 4 points :

Grid point #0, Parameters bins:
delta_CP: 0 (-3.07999)
Default numbers of expected events, nue= 62.0263, numu= 272.789, nuebar= 18.8084, numubar= 139.835, nue CC1Pi= 5.79871
Default value of negative log likelihood: 1.5125
Value of marginal NLL for bin 0: 6.17442

Grid point #10, Parameters bins:
delta_CP: 10 (-1.848)
Default numbers of expected events, nue= 72.4255, numu= 272.484, nuebar= 16.8521, numubar= 139.575, nue CC1Pi= 6.80712
Default value of negative log likelihood: 0.0417285
Value of marginal NLL for bin 10: 5.41319

Grid point #20, Parameters bins:
delta_CP: 20 (-0.615999)
Default numbers of expected events, nue= 67.6372, numu= 272.088, nuebar= 17.8507, numubar= 139.211, nue CC1Pi= 6.53799
Default value of negative log likelihood: 0.47455
Value of marginal NLL for bin 20: 5.58672

Grid point #30, Parameters bins:
delta_CP: 30 (0.615999)
Default numbers of expected events, nue= 54.0552, numu= 272.071, nuebar= 20.4709, numubar= 139.212, nue CC1Pi= 5.35063
Default value of negative log likelihood: 4.44963
Value of marginal NLL for bin 30: 8.15429

III. b. Run IntAsimov

- You can then fit the toys located in inputs/Validation_ToyXP_AsimovA_2018.root with IntFitter

- Since the marginalization throws are generated using the same oscillation and systematic throws without randomness, MargAsimov and IntAsimov results should be exactly the same.

- Prior to run IntAsimov, the marginalization throws should be generated. It is automated by entering shell/ repository, and running : ./RunMakeTemplates.sh.

- In this example, I kept the number of maginalization throws = 100.

III. b. Run IntFitter

- When all throws are generated, you can then run IntAsimov. If you did not modified RunMakeTemplates.sh nor RunAtStart.sh, the throws should be automatically generated in the environment variable called mc_templates.

- You can run IntAsimov by entering shell/ repository, and running : ./RunIntAsimov.sh. Note that you should confirm in this file that the variable after « -t » contains the address of the throws generated by MakeTemplates (in the code, contained again in mc_tenplates).

- In this example, I kept the number of maginalization throws = 100.