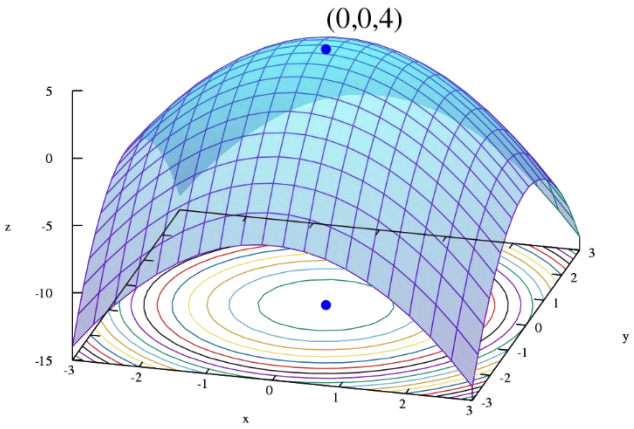
TIF285: Learning From Data

PROJECT 1



BROU-BONI Joël

SANOGO Ibrahim Bechir

Goal of the project

The goal of this project is to replicate the results of a paper conducted by S.Wesolowki , N.Klco or R.J Furnstahl. The authors are using a toy model to simulate an effective field theory namely a Taylor series of some specified function and are presenting the procedure to do parameter estimations when you know what type of model can be used for the Bayesian approach. This project. The project is divided into 2 parts, the first one where we have to represent and interpret Figure 1 3 and 4 and a second one where we have to represent the Table III and after that redo everything we had to do with a different dataset generated by ourselves.

Basic Part

Part A

To reproduce figure 3 results, we had to use the emcee library to sample the Log posteriors. These functions are calculated thanks to the likely hood and a flat or gaussian prior given in the equation below:

$$\mathrm{pr}(\mathbf{a}|\bar{a}, I) = \left(\frac{1}{\sqrt{2\pi}\bar{a}}\right)^{k+1} \exp\left(-\frac{\mathbf{a}^2}{2\bar{a}^2}\right)$$

Figure 1 Gaussian Prior

$$\mathrm{pr}(D|\mathbf{a}, I) = \prod_{j=1}^{N_d}\left(\frac{1}{\sqrt{2\pi}\sigma_{j,\mathrm{exp}}}\right) e^{-\chi^2/2} \quad \text{With} \quad \chi^2 = \sum_{i=1}^{N_d}\left(\frac{d_i - g_{\mathrm{th}}(x_i)}{\sigma_i}\right)^2$$

Figure 3 Likely Hood

Figure 2 Chi2 Function

The gaussian Prior can be implemented in a different way, indeed it is also equal to:

$$GaussPrior = \frac{1}{2} * \boldsymbol{a}.\mathrm{T@inv(CovarianceMatrix)@}\ \boldsymbol{a}$$

$$GaussPrior = \frac{1}{2} * \frac{\boldsymbol{a}.\boldsymbol{T@a}}{\mathrm{abar}^2}$$

where @ is a dot product in python and with the normalization omitted.

Finally, we use emcee to sample the posteriors and corner to create plots exactly as in the figure 3 of the paper.

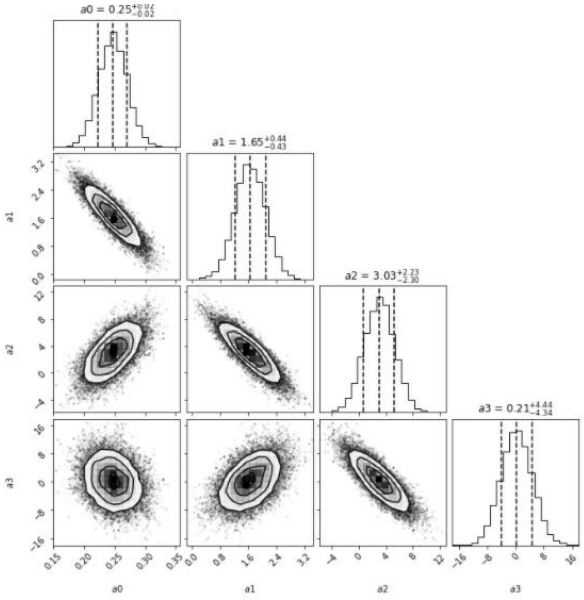


Figure 4 Projected Posteriors Plot Gaussian Prior

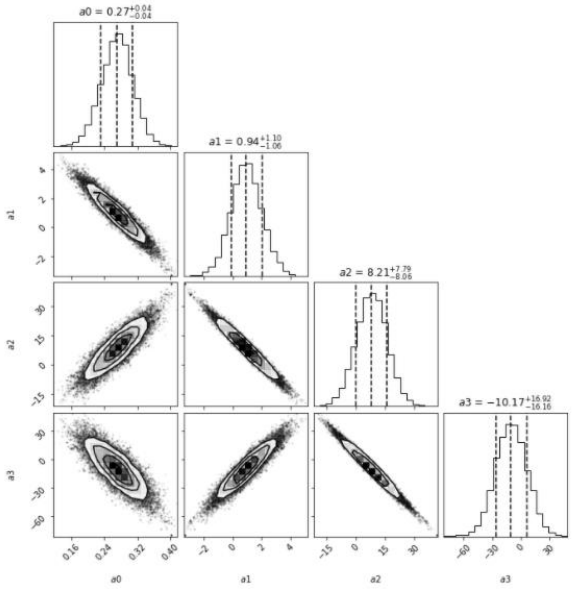
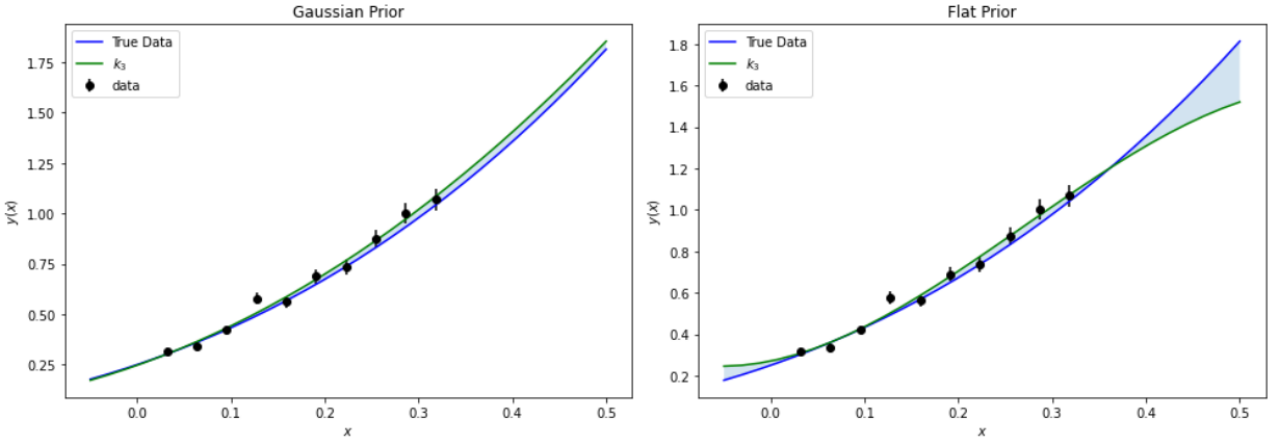


Figure 5 Projected Posteriors Plot Flat Prior

We have the results for the Gaussian prior on the right and for a uniform prior on the left. As we can see the result that we have are close to the one in the papers. The plots on the diagonal are marginalized pdfs for each coefficients ai and the lower triangle plots are two-dimensional posterior plots. They allow us to see the regions of high joint probabilities for the ais or LECs wrote at the top of the marginalized pdfs and to see the correlations among the parameters that we got.

By looking at the curves it seems that there is an overfitting problem in the uniform case. Indeed, starting from a2 the coefficients of our model increase greatly and see their confidence interval degrade. Moreover the *ai* coefficients after a2 are getting bigger but this time of opposite sign (ai = Const*-ai-1) as to correct the too big value coefficients and to push the maximum of the likelihood to unnatural regions of parameter space which preceded it which has the consequence of degrade the confidence interval even more. There is also a bigger 2 parameter correlation between the ais coefficient in this case compared to the gaussian prior one which is normal in view of the things explained above. A

The 2nd step consisted in representing the 2 curves below




```
ais_G =np.percentile(samples_Gauss,[16,50,84],axis=0)
ais_F =np.percentile(samples_Flat,[16,50,84],axis=0)
```

We used the command **np.percentile(samples, [16, 50, 84],axis=0)** to extract numerical values for the credibility region and the mean from a python array

We chose to represent the data(from the data set),the true data(0.25+1.57*x+2.47**x²+1.29*x**3) and a0+a1*x+a2**x²+a3*x**3 where the *ai* are the coefficients we computed thanks to the Gaussian and uniform posteriors.

We have the same results as in the paper; however, we chose to represent the data before 0 and after 0.3 to emphasize the overfitting problem in the uniform case. It fits the data but it is not working outside of the data's range.

On the other hand, Gaussian prior fits accurately the data.

Extra Tasks

Part A

To perform the expected tasks in the extra part, we had to update our uniform prior and posterior. In the uniform case, we take the value of 1 000 000 so it will be possible to compute all the excepted values of the *ai* until order 6 otherwise the limit value of the prior is not big large enough to cover the entire range of the data.

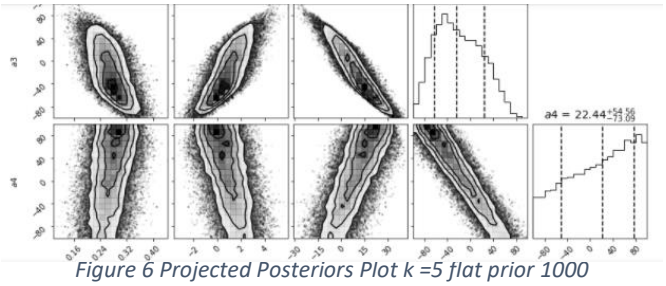


Figure 6 Projected Posteriors Plot k =5 flat prior 1000

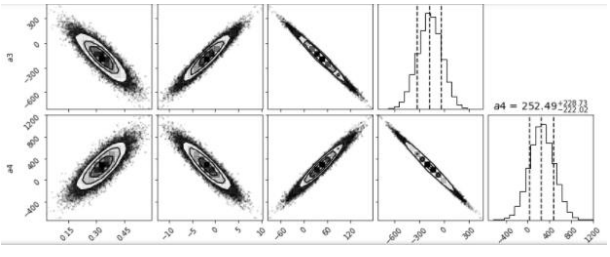


Figure 7 Projected Posteriors Plot k = 5 flat prior = 1000000

We then compute the same algorithm from the basic part, but we are performing it in a loop from order 1 to 7 to compare the results for both uniform and Gaussian prior.

The results can be observed in the notebook because they are to big to plot here. However, the main conclusion is that the error (68% dob) and the value of *ai* increase with the order and rises very strongly in the uniform case. The overfitting problem that we noticed in the previous part is getting worse when the order k increases.

Table 1 Coefficient estimates from sampling Part 1

	k	kmax	khi2/DOF	a0UniForm	a1UniForm	a2UniForm	a0Gauss	a1Gauss	a2Gauss
0	0.0	0.0	66.56	0.48	nan	nan	0.48	nan	nan
1	1.0	1.0	2.23	0.2	2.56	nan	0.2	2.55	nan
2	2.0	2.0	1.64	0.25	1.57	3.34	0.25	1.62	3.18
3	2.0	3.0	1.92	0.27	0.96	8.03	0.25	1.63	3.01
4	2.0	4.0	2.31	0.34	-2.01	45.83	0.25	1.66	2.87
5	2.0	5.0	2.88	0.56	-14.55	271.33	0.25	1.64	3.04
6	2.0	6.0	3.84	0.59	-16.62	317.35	0.25	1.64	3.06

Table 2 Dob of the ais

	ErrA0U	ErrA1U	ErrA2U	ErrA0G	ErrA1G	ErrA2G
0	+-0.01	+-nan	+-nan	+-0.47	+-nan	+-nan
1	+-0.01	+-0.1	+-nan	+-0.19	+-0.1	+-nan
2	+-0.02	+-0.4	+-1.33	+-0.23	+-0.39	+-1.24
3	+-0.04	+-1.07	+-8.04	+-0.22	+-0.45	+-2.31
4	+-0.07	+-2.69	+-32.3	+-0.22	+-0.47	+-2.37
5	+-0.13	+-6.58	+-113.69	+-0.22	+-0.46	+-2.33
6	+-0.29	+-18.05	+-391.63	+-0.22	+-0.45	+-2.42

The Khi2/Dof part indicate the quality of the fit. We were also supposed to compute the evidence, but we didn't manage to install Pymultinest, the library that can do it. Getting the evidence is also possible with a Laplace approximation but since we did not manage to have the good result through it, we chose to not represent it. However, this table is useful to compare the two different priors. Indeed, thanks to it we can now see that the gaussian prior is the best one for our application due to the overfitting problem in the flat prior.

Part B

In this part, the goal is to redo what we have done before but this time with a different dataset. The dataset is computed thanks to this formula:

$$d_j = g(x_j)(1 + c\eta_j) \quad \Longrightarrow \quad \sigma_j = c\, d_j$$

With C a constant value that we must choose ourselves. C is a specified relative error and allows us to maximize or minimize the importance of the gaussian Noise. As a result, the higher C is, the higher the order k must be to have a good model. To avoid us big computation time in the for loop given that we must increase k max a lot with big values of C and to also change the prior we chose to keep C<1. We have another parameter M which defines the size of our dataset. So, the bigger M is, the more data we will have. The influence of this parameter will notably be seen on the quality of our polynomial regression with the flat prior and the over fitting problem that we had before.

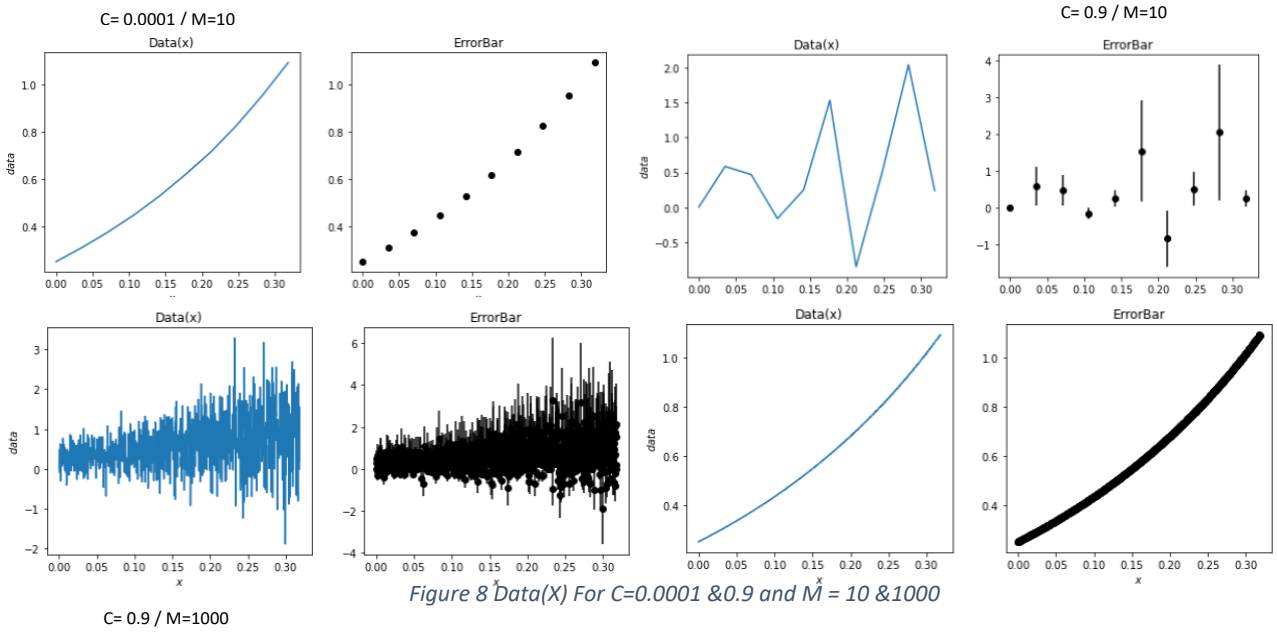


Figure 8 Data(X) For C=0.0001 &0.9 and M = 10 &1000

Therefore, we decided in this last part to study the influence of these two parameters in this part starting with C. As we said before C minimize or Maximize the importance of the Gaussian Noise in our system. This has the consequence of changing the dataset which is changing the optimal degree of the polynomial regression. The first step was then to put a big value here 0.4. We chose a fixed value of M for both cases: 10.

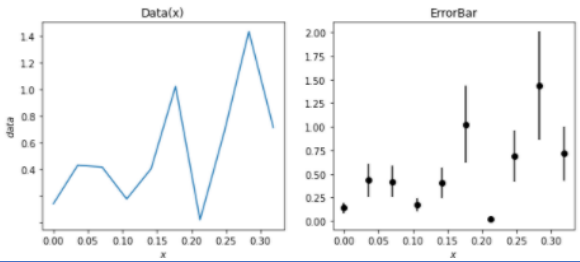


Figure 9 Curves for C=0.4

C= 0.0001 / M=1000

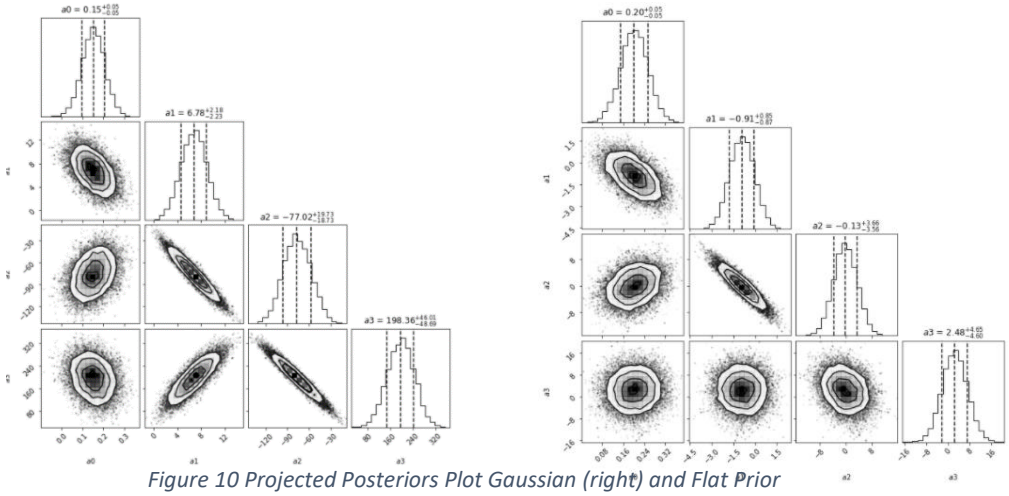
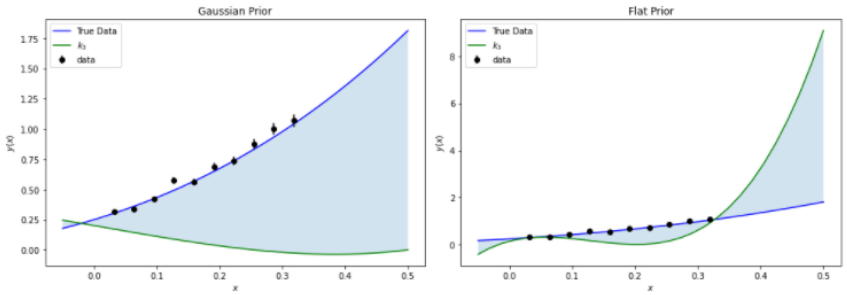


Figure 10 Projected Posteriors Plot Gaussian (right) and Flat Prior



The same phenomenon as the previous part appear here, but this time, it seems that we have an underfitting problem because. We do not have enough data to have a correct model and this can be seen on the curves below. Indeed, the error is much more important than in the first part of the project due to the poor quality of the data. The gaussian prior seems to not work correctly now but this is because the order is not big enough (the plot here is for k=3).

But when C is very low (close to 0) the data are clear of all the noise and the quality increases this underfitting problem then disappear for the prior and an order 3 curve is good enough to do a polynomial regression . This explains why the 2 priorities find the same result and to better it by adding more data.

Table 3 Results for C=0.00001

	k	kmax	khi2/DOF	a0UniForm	a1UniForm	a2UniForm	a0Gauss	a1Gauss	a2Gauss
0	0.0	0.0	2.05051e+07	0.4	nan	nan	0.4	nan	nan
1	1.0	1.0	4.72223e+05	0.23	2.33	nan	0.23	2.33	nan
2	2.0	2.0	6044.46	0.25	1.43	3.7	0.25	1.43	3.7
3	2.0	3.0	124.95	0.25	1.61	1.82	0.25	1.61	1.82
4	2.0	4.0	3.98	0.25	1.57	2.56	0.25	1.57	2.56
5	2.0	5.0	3.13	0.25	1.57	2.37	0.25	1.57	2.43
6	2.0	6.0	7.86	0.25	1.58	2.09	0.25	1.57	2.42

	ErrA0U	ErrA1U	ErrA2U	ErrA0G	ErrA1G	ErrA2G
0	+-0.0	+-nan	+-nan	+-0.4	+-nan	+-nan
1	+-0.0	+-0.0	+-nan	+-0.23	+-0.0	+-nan
2	+-0.0	+-0.0	+-0.0	+-0.25	+-0.0	+-0.0
3	+-0.0	+-0.0	+-0.01	+-0.25	+-0.0	+-0.01
4	+-0.0	+-0.0	+-0.03	+-0.25	+-0.0	+-0.03
5	+-0.0	+-0.0	+-0.08	+-0.25	+-0.0	+-0.06
6	+-0.0	+-0.0	+-0.18	+-0.25	+-0.0	+-0.06

As we said before M defines the size of our dataset. To study the importance of M we choose to fix a value of C that was giving a curve data(x) looking like the one in the first part and study to cases, M=10 and M=1000.

With M= 10 and c= 0.07, the observed results are like the one that we got before, which is giving us this overfitting problem that we observed previously. And as usual, the Gaussian error remains stable, it does not strongly increases compared to the uniform prior.

	k	kmax	khi2/DOF	a0UniForm	a1UniForm	a2UniForm	a0Gauss	a1Gauss	a2Gauss
0	0.0	0.0	43.19	0.39	nan	nan	0.39	nan	nan
1	1.0	1.0	2.97	0.22	2.23	nan	0.22	2.23	nan
2	2.0	2.0	2.39	0.24	1.4	3.43	0.24	1.45	3.23
3	2.0	3.0	2.7	0.24	2.22	-5.38	0.24	1.54	2.08
4	2.0	4.0	3.23	0.23	3.41	-28.5	0.24	1.57	1.97
5	2.0	5.0	4.04	0.23	3.33	-26.2	0.24	1.55	2.03
6	2.0	6.0	5.38	0.23	9.14	-279.51	0.24	1.56	1.97

	ErrA0U	ErrA1U	ErrA2U	ErrA0G	ErrA1G	ErrA2G
0	+-0.01	+-nan	+-nan	+-0.38	+-nan	+-nan
1	+-0.01	+-0.12	+-nan	+-0.21	+-0.12	+-nan
2	+-0.01	+-0.33	+-1.29	+-0.23	+-0.32	+-1.25
3	+-0.02	+-0.68	+-6.36	+-0.23	+-0.34	+-2.11
4	+-0.02	+-1.17	+-20.24	+-0.23	+-0.35	+-2.12
5	+-0.02	+-1.95	+-51.66	+-0.23	+-0.36	+-2.2
6	+-0.02	+-3.22	+-126.05	+-0.23	+-0.37	+-2.24

Table 4 Result for M=10, C=0.07

With M= 1000 and c= 0.07, the over fitting problem disappear. Indeed, we do not have these big variations of values for the ais coefficients. Moreover, we can see that the DOB value is tighter. And this explain why the results below are better than the one with the given dataset.

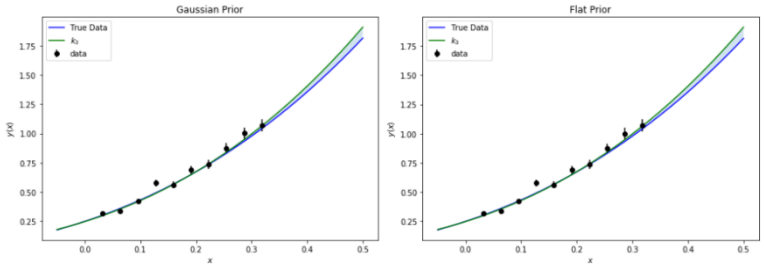


Figure 11 Comparison GtrueData(x) vs Model For M=100

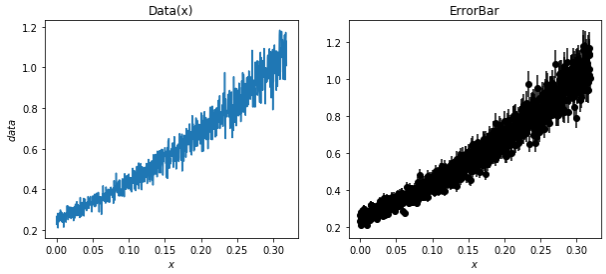


Figure 12 Data(x) and ErrorBar for M=1000

	k	kmax	khi2/DOF	a0UniForm	a1UniForm	a2UniForm	a0Gauss	a1Gauss	a2Gauss
0	0.0	0.0	3585.79	0.42	nan	nan	0.42	nan	nan
1	1.0	1.0	191.01	0.22	2.33	nan	0.22	2.33	nan
2	2.0	2.0	147.73	0.25	1.43	3.52	0.25	1.43	3.51
3	2.0	3.0	172.24	0.25	1.5	2.85	0.25	1.5	2.85
4	2.0	4.0	206.69	0.25	1.53	2.2	0.25	1.49	2.91
5	2.0	5.0	258.36	0.24	1.93	-8.32	0.25	1.49	2.91
6	2.0	6.0	344.52	0.24	2.14	-16.07	0.25	1.5	2.89

	ErrA0U	ErrA1U	ErrA2U	ErrA0G	ErrA1G	ErrA2G
0	+-0.0	+-nan	+-nan	+-0.42	+-nan	+-nan
1	+-0.0	+-0.01	+-nan	+-0.22	+-0.01	+-nan
2	+-0.0	+-0.04	+-0.16	+-0.25	+-0.04	+-0.16
3	+-0.0	+-0.09	+-0.82	+-0.25	+-0.09	+-0.77
4	+-0.0	+-0.17	+-2.58	+-0.25	+-0.09	+-0.82
5	+-0.0	+-0.29	+-6.56	+-0.25	+-0.09	+-0.84
6	+-0.0	+-0.44	+-13.64	+-0.25	+-0.09	+-0.82

Table 5 Results for M=100

As we saw these two parameters can have a big influence on the quality of the model we got. It is therefore necessary to choose them one according to the other (preferably having a large database when they are of poor quality for example).

Bibliography

Paper that inspired this work : https://arxiv.org/abs/1511.03618

PyMultinest Documentation : https://johannesbuchner.github.io/PyMultiNest/

Lecture notes : https://chalmers.instructure.com/courses/10188