

# PHYS 512: Problem Set 4

Jade Ducharme

October 2021

## Problem 1

In order to determine if the Chi-squared value obtained for the given list of parameters is good, I need to find the p-value, which is “the probability of obtaining test results at least as extreme as the results actually observed, under the assumption that the null hypothesis is correct”. In our case, the null hypothesis is that the given parameters are a good fit to the data. Therefore, a very small p-value means the probability that our data can occur/be generated using the given parameters is very small. In other words, a small p-value means the fit is bad.

Using *scipy.stats.chi2*, I compute the p-values for the Chi-squared values obtained using both given sets of parameters and present them in Table 2.

Parameters	$\chi^2$	dof	p-value
[60, 0.02, 0.1, 0.05, $2 \times 10^{-9}$ , 1.0]	15265	2501	0.0
[69, 0.022, 0.12, 0.06, $2.1 \times 10^{-9}$ , 0.95]	3273	2501	$1.1 \times 10^{-23}$

Table 1: Chi-squared value and p-value obtained for different sets of parameters.

In both cases, the p-value is incredibly small. A usual rejection region for the p-value is anywhere between 0 and 5%, and here we are nowhere near 5%. In fact, the computer doesn’t even have enough precision to calculate the p-value for the first set of parameters, which is why it spits out 0.0. Therefore, I would consider neither of these sets of parameters a good fit to the data.

## Problem 2

For Problem 2, I use the Levenberg-Marquadt (LM) method to find the best-fit parameters. For this method, I first need to find the local gradient of the power spectrum function for different parameter values. I did this by adapting a function I wrote for Problem Set 1 – Problem 1. In order to achieve the best possible  $\chi^2$ , I tried running my LM iterator for a very long time, but I found that every successive accepted LM step would improve the  $\chi^2$  by a smaller and smaller value each time, as illustrated in Figure 1.

I present my LM best-fit parameters and their uncertainties in *planck\_fit\_params.txt*. I also present, in Figure 2, the best-fit curve for my parameters as well as the residuals.

The  $\chi^2$  value for the best-fit parameters after 25 iterations of the LM chain is 2585 with p-value 0.12. This is still pretty bad, but it is a significant improvement on the results from Problem 1. A p-value of 0.12

```

Initial chisq 3272.5723646011556
Accepting step with chisq improvement 672.7127754216408
Accepting step with chisq improvement 12.952885780036922
Accepting step with chisq improvement 1.8071430359905207
Accepting step with chisq improvement 0.3176100916566611
Accepting step with chisq improvement 0.1565575759609601
Accepting step with chisq improvement 0.05084517267823685
Accepting step with chisq improvement 0.029677353542410856
Accepting step with chisq improvement 0.010374445004799782
Accepting step with chisq improvement 0.0021652689856637153
Accepting step with chisq improvement 0.0001671431332397333
Accepting step with chisq improvement 2.9931457447673893e-05
Accepting step with chisq improvement 0.00012089631945855217
Final chisq 2584.532019470198

```

Figure 1: Python log for my LM fit using 25 iterations and passing the second set of parameters given in the problem set as the initial guess. The improvement gets smaller and smaller at each step. (Note – here only the *accepted* steps are being printed.)

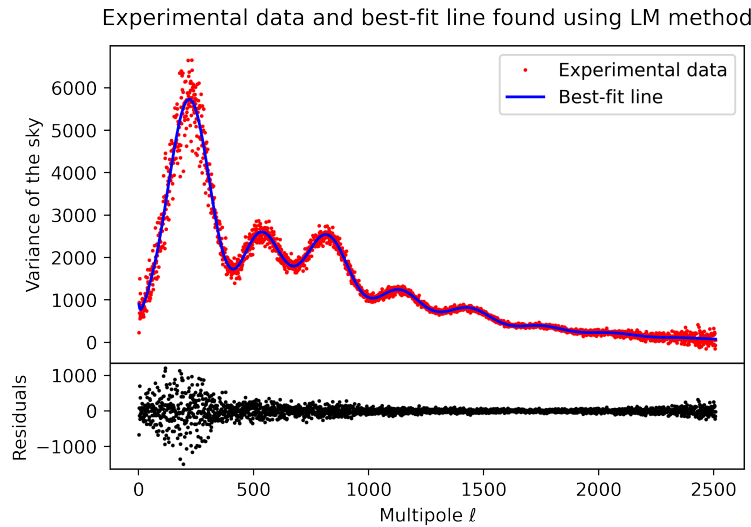


Figure 2: Best-fit curve and experimental data points as a function of the multipole  $\ell$ . Looks pretty good!

means that there is a 12% chance that the experimental data can be retrieved using the best-fit parameters. Pretty bad, but not bad enough to reject the null hypothesis!

### Problem 3

For Problem 3, I wrote a Metropolis-Hastings algorithm for the MCMC chain and ran it for 100000 iterations. I picked my proposal jumps by randomly sampling the normal distribution with mean = mean of current parameter and std = twice the error on the parameter as obtained from the curvature matrix in Problem 2. The corner plot is presented in Figure 3.

As we can see from the corner plot, the histograms look nice and Gaussian, indicating that the chain is running as expected and picking up on high-likelihood values for the parameters. The best-fit value for each parameter is given by its histogram mean  $\pm$  its histogram standard deviation. These best-fit values can be found in *mcmc\_fit\_params.txt*.

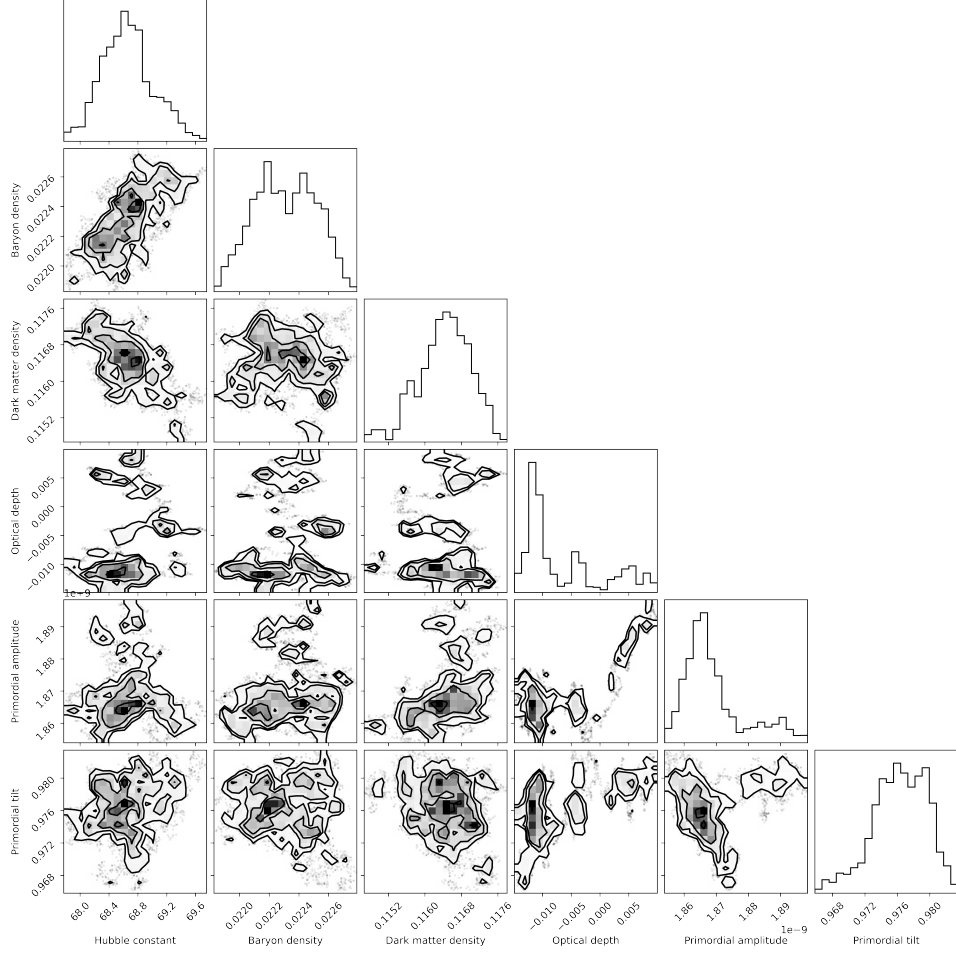


Figure 3: Corner plot after 100000 iterations.

The chi-square value and p-value found using this MCMC are:

$$\chi^2 = [\text{INSERTVALUEHERE}]$$

$$p = [\text{INSERTVALUEHERE}]$$

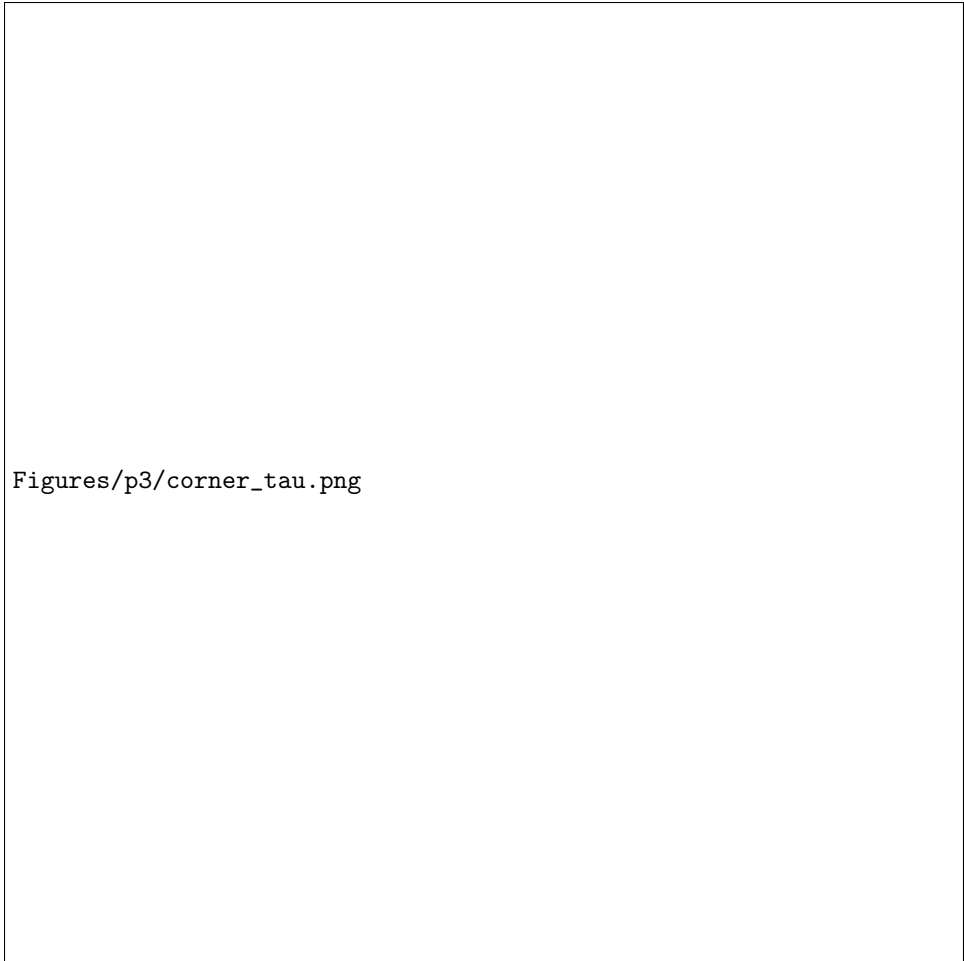
#### Problem 4

For Problem 4, I first re-ran the chain but only proposing jumps in  $\tau$  that were centered on 0.054 with standard deviation 0.0074. The corner plot for this run is presented in Figure 4.

The best-fit parameters for this run can be found in *mcmc-tau-fit-params.txt*. The chi-square and p-value for this run are given by:

$$\chi^2 = [\text{INSERTVALUEHERE}]$$

$$p = [\text{INSERTVALUEHERE}]$$



Figures/p3/corner\_tau.png

Figure 4: Corner plot after 100000 iterations (using prior on  $\tau$ ).

Next, I used importance sampling to re-sample the chain from Problem 3. I present the results from importance sampling in *mcmc\_tau\_importance\_fit\_params.txt*.

The final chi-square and final p-value found using importance sampling are given by:

$$\chi^2 = [INSERTVALUEHERE]$$

$$p = [INSERTVALUEHERE]$$

Finally, I plot the best-fit curve using the importance sampling best-fit parameters on top of the experimental data and present my final plot in Figure 5.



Figure 5: Best-fit curve and experimental data points as a function of the multipole  $\ell$ . Best-fit parameters found using importance sampling.