Phys 512 Assignment 4

Ian Hendricksen

October 18, 2021

1. The test script prints "chisq is 15267.937150261654 for 2501 degrees of freedom" for the original set of parameters. Given that the mean and variance of χ^2 are n and $2n$ respectively, where n is the number of degrees of freedom, if our fit is acceptable, we should expect our value to be within 1σ of 2501, where $\sigma = \sqrt{2n}$ is the standard deviation. In fact, since we can write

$$\chi^2 = n + x\sqrt{2n}, \tag{1}$$

where x is the number of standard deviations χ^2 is from the mean, we can state that, for the above χ^2 and n, χ^2 is approximately 180.5σ from the mean, suggesting that the original set of parameters in the test script are not an acceptable fit.

If we change the parameters to [69, 0.022, 0.12, 0.06, 2.1e-9, 0.95], the test script prints "chisq is 3272.2053559202204 for 2501 degrees of freedom". In this case, we find that our χ^2 is approximately 10.9σ from the mean. This indicates that the fit has improved, but we are still not within a small enough range about the mean to conclude that these parameters are an acceptable fit.

2. To set up a Levenberg-Marquardt fitter, we first need to construct a function to take numerical derivatives w.r.t. the function `get_spectrum`. This is done with the function I have written called `get_spectrum_derivatives`, which implements the 4-point method seen in Assignment 1; it loops through each parameter and computes the derivative at each point w.r.t. that parameter. It then returns these derivatives as columns in a 2D matrix of derivatives.

These derivatives are then used in my Levenberg-Marquardt fitter `lvmq`. It begins by computing the value χ_0^2 (referred to as `chisq_0` in Python) for the data and an initial set of parameters m. The errors, as suggested by the introduction to the assignment, are the average of the upper and lower errors, and are assumed Gaussian and uncorrelated. These are then used to construct the noise matrix N^{-1} (referred to as `Ninv` in Python). For n iterations, we compute a model and its derivatives with the parameters m_{new} (which is equivalent to m for the first iteration). We then compute an adjustment to the parameters as

$$dm = \left(A'^T N^{-1} A' + \lambda \left[diag(A'^T N^{-1} A')\right]\right)^{-1} A'^T N^{-1} r, \tag{2}$$

where dm is the parameter adjustment, A' is our derivatives matrix, λ is a scalar, and $r = d - fun(m_{new})$, where d is our data. We then take a step by computing $fun(m_{new} + dm)$ and compute χ^2 w.r.t. the data and its errors. We then need to update λ for the next step. If $\chi^2 < \chi_0^2$ (i.e. successful step), we divide λ by 2 , and set it equal to 0 if $\lambda < 0.5$. We then update the set of parameters $m_{new} = m_{new} + dm$, and $\chi^2 < \chi_0^2$ for the next step, such that we are descending the gradient. If $\chi^2 > \chi_0^2$ (i.e. unsuccessful step), we set $\lambda = 1$ if $\lambda < 0.01$ and $\lambda = 2\lambda$ otherwise, ensuring that we take bigger steps to (hopefully) search for better parameters. After looping through n iterations, the user is returned the final set of parameters, their errors, and the curvature matrix associated with those parameters (as well as telling the user about runtime).

I ran `lvmq` using the second set of parameters provided in Q1 (found above) and 25 iterations. The best fit parameters and errors are written to `planck_fit_parameters.txt`; these are reported in Table 1. χ^2 for these parameters is 2583.713095273898, which is 1.1695059334208031σ from the mean.

Parameter	Best-Fit Value	Error
H_0	6.93889390e+01	1.70789285e+00
$\Omega_b h^2$	2.25063896e-02	3.23942059e-04
$\Omega_C h^2$	1.15179290e-01	3.74360623e-03
τ	1.59842341e-01	3.96554653e-02
A_s	2.55637789e-09	1.87749522e-10
n_s	9.80523797e-01	9.84564638e-03

Table 1: Levenberg-Marquardt best-fit parameters.

3. Now we run an MCMC chain using my function `mcmc`. Like `lvmq`, it begins by computing χ_0^2 for some set of guess parameters m w.r.t. the data. Before beginning, it computes the Cholesky decomposition matrix L of the inverse of the curvature matrix returned by `lvmq`. This is then used to create a set of parameters for each iteration n as

$$m_{trial} = m + k * L * np.random.randn(len(m)), \tag{3}$$

where k is some fractional value (nominally set to 0.1 to prevent the chain steps from going crazy). The random value allows our trial parameters m_{trial} to bounce around randomly, in analogy to a particle in some potential. I have also included an if statement to force $\tau = 0.01$ if $\tau < 0.01$, if nothing else but to prevent it from approaching negative nonphysical values, as it tended to do. Using the trial parameters m_{trial}, we compute χ^2 w.r.t. the data and its difference w.r.t. χ_0^2 as $\delta\chi^2 = \chi^2 - \chi_0^2$. If the condition

$$np.random.randn(1) < e^{-0.5\delta\chi^2} \tag{4}$$

is true, we update the original m such that now $m = m_{trial}$, and the original χ_0^2 such that now $\chi_0^2 = \chi^2$ from the trial step. These values are then used on the next iteration. No matter what, we record the trial χ^2 and the parameters m_{trial} for each iteration/step in the chain.

The reason why we implement Equation 4 is to compare a random number to the ratio of new to old likelihoods, such that the step is sometimes taken and sometimes not, allowing for the parameters to bounce around randomly, and hopefully approach the global minimum. This also helps us to jump out of a local minimum after some period of time.

I ran `mcmc` for $n = 5000$ iterations/steps, providing it with the same initial parameters m as in Q2, as well as the curvature matrix returned by `lvmq` in Q2. Each step in the chain and its associated χ^2 value was written to `planck_chain.txt`. From the chain, we compute the parameter estimates as the means of the chain, and the errors as the standard deviations of the chain. These values are reported in Table 2. χ^2 for these values is 3152.7620191209962, which is 9.215463960286224σ from the mean (surprisingly, a worse fit than Levenberg-Marquardt).

Parameter	Best-Fit Value	Error
H_0	6.36150302e+01	4.60097970e+00
$\Omega_b h^2$	2.19522745e-02	3.96897223e-04
$\Omega_C h^2$	1.34268992e-01	1.18557824e-02
τ	1.08449706e-01	6.41561874e-02
A_s	2.40127555e-09	3.28960726e-10
n_s	9.21469124e-01	2.35802437e-02
Ω_Λ	0.6139703348532419	0.01212606757309287

Table 2: MCMC best-fit parameters.

In order to see if the chain has converged, I plotted H_0 and its Fourier transform (log-log scale), displayed in Figure 1.

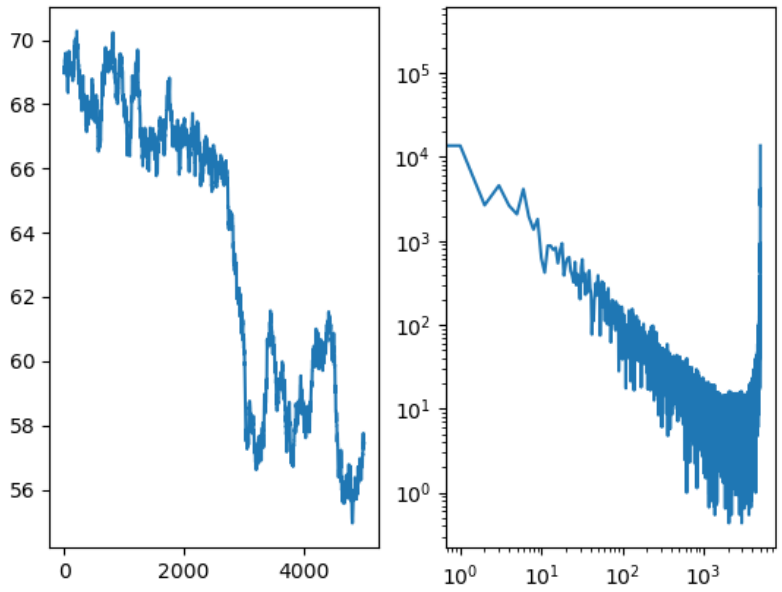

Figure 1: H_0 for each iteration of the chain and its FFT.

It is clear that the chain has not converged, as H_0 does not appear as noise, nor is its Fourier transform flat for large scales/low k. The same behavior is observed for the other parameters, suggesting the same.

From the estimated parameters, we compute Ω_Λ as

$$\Omega_\Lambda = 1 - \Omega_b - \Omega_C, \tag{5}$$

and include this in the Table 2 (note that this is also reported to the file `om_lambda.txt`). The error on this value is computed using first-order Taylor expansions for each step involving a mathematical operation between intermediate parameters.

4. For this problem, I went back and made some minor edits to `mcmc` such that it can accept priors. This essentially involved adding the new parameters m_p ("m priors") and $m_{p,err}$ ("m priors errors") to `mcmc`, where m_p is an array of zeros except for at τ's positions, which was set to be 0.0540, and $m_{p,err}$ is an array of 1e15 (i.e. enormous errors on non-prior parameters) except again at τ's position, which was set to be 0.0074, both according to polarization data.

   I ran `mcmc` in exactly the same way as in Q3 (with the initial parameters being from Q2 and 5000 iterations), but with some minor changes to χ^2 values. If priors are present when the function is called, χ_0^2 is calculated in the same way, but we add another chi^2 value, which is the χ^2 value between the initial parameters m and m_p, with $m_{p,err}$ in the denominator. The same is done for each trial χ^2 computed (i.e. adding this "extra" χ^2 value from the priors). The chain is reported in the file `planck_chain_tauprior.txt`, and the parameter estimates and their errors are computed in the same manner as in Q3, which are reported in Table 3. χ^2 for these values is 2880.373903938942, which is 5.36408449197064σ from the mean (an improvement over our initial MCMC fit in Q3, but still not as good as Levenberg-Marquardt; strange).

   We check for convergence in the same manner as in Q3; the chain for H_0 and its Fourier transform are displayed in Figure 2. For the same reasons as in Q3, the chain does not appear to have converged (i.e. H_0 does not look like noise, and its FFT is not flat for large scales). This behavior is observed

Parameter	Best-Fit Value	Error
H_0	7.42424729e+01	2.36920487e+00
$\Omega_b h^2$	2.32714652e-02	4.88158041e-04
$\Omega_C h^2$	1.07738827e-01	5.56348398e-03
τ	1.09528451e-01	9.27421638e-02
A_s	2.27905600e-09	4.75238758e-10
n_s	9.70275828e-01	1.32684867e-02

Table 3: MCMC with priors best-fit parameters.

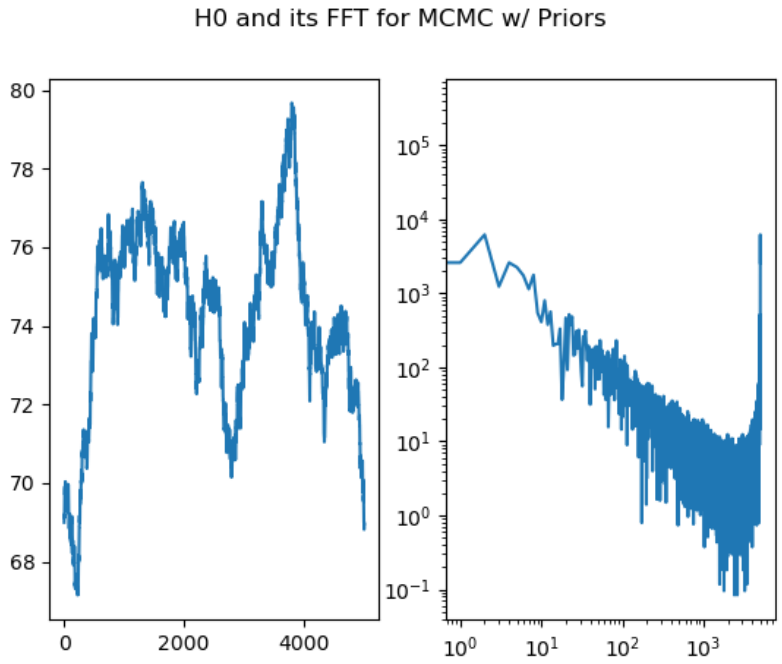

Figure 2: H_0 for each iteration of the chain and its FFT for MCMC with priors.

for all other parameters as well. I should mention that the entire script took about 10 hours to run; I had thought to increase the number of iterations to 10000 for both Q3 and Q4, but this would have ended up taking too long (20 hours or greater) for me to gather all the results in time to submit this assignment.

I then implemented importance sampling on the chain from Q3 by computing the weights for each step as

$$w = e^{-0.5\chi^2_{imp}}, \tag{6}$$

where χ^2_{imp} is the importance sampled χ^2, which compares the parameters at each step in the chain to m_p, divided by $m_{p,err}$. The importance sampled parameters are then computed as

$$m_{imp} = \frac{\sum w_i m_i}{\sum w_i}, \tag{7}$$

where w_i is the ith weight and m_i is the ith set of chain parameters. The importance sampled parameters are reported both in `planck_fit_params_imp_samp.txt` and in Table 4. χ^2 for these values is 2961.6039126171213, which is 6.512620607157386σ from the mean. Comparing the two χ^2 values

from MCMC with priors and importance sampling, we find that MCMC with priors is closer to the mean of the χ^2 distribution, and thus is a better fit.

Parameter	Best-Fit Value
H_0	6.19163431e+01
$\Omega_b h^2$	2.21653108e-02
$\Omega_C h^2$	1.39774664e-01
τ	5.92527370e-02
A_s	2.22639380e-09
n_s	9.11027855e-01

Table 4: Importance sampled best-fit parameters.