PHYS 512 - PS3

Damien Pinto (260687121)

October 2019

1)

Code for this question: python3 wmap_camb_example.py
First, I modified line 37 such that the get_params() function provided by Prof. Sievers only output a fit (*cmb*) that is the same length as our data (+2 for the monopole and dipole terms), which . We then use *wmap[:,1]* as our x_i, *cmb* (with the two first terms excluded) as our μ_i, and *wmap[:,2]* as σ_i in

$$\chi^2 = \Sigma_{i=1}^{N} \left(\frac{x_i - \mu_i}{\sigma_i} \right)^2 \tag{1}$$

, which returns a value of $\chi^2 = 1588.4366....$

2)

Code for this question: python3 levenberg_marquadt.py As indicated by the question, I implemented a Levenberg-Marquardt fitting algorithm. I constructed the matrix of derivatives ($A' = Ap$) of the data using a two-sided derivative. The function *(*get_Ap) gets a function and a set of parameters to input into that function. One by one, for each parameter individually, the function augments the parameter it's on by a 200th of its initial value, evaluates the function there, subtracts one 200th of its value, and computes the derivative there. It then takes the difference and divides by one 100th of the parameter's initial value to construct the two-sided derivative. I acknowledge that the factor of one 200th seems a little arbitrary. I tried to find the optimal δp given our parameters and what we saw in class, that the optimal increment with respect to a parameter x is:

$$\delta p^x_{optimal} = \sqrt{\frac{\epsilon f(x)}{f''(x)}}. \tag{2}$$

where $\epsilon = 10^{-12}$ is the machine precision for doubles. I then substituted :

$$\begin{aligned} f''(x) &\approx \frac{f'(p + \delta p^x) - f'(x - \delta p)}{2\delta p^x} \\ &\approx \frac{(f(x + 2\delta p^x) - f(x)) - (f(x) - f(x - 2\delta p^x))}{2(\delta p^x)^2} \\ &\approx \frac{f(x + 2\delta p^x) - f(x - \delta p^x)}{2(p^x)^2} \end{aligned}$$

which gives:

$$\delta p^x_{optimal} = \sqrt{\sqrt{\frac{2\epsilon f(x)}{f(x + 2\delta p^x) - f(x - \delta p^x)}}},$$

and then tried to compute the right-hand side numerically for various δp^x to see which one produced results closest to δp^x. This would have to be done for every parameter, without guarenty of finding the right δp^x for each one. Furthermore it always failed because it produced infinite values... so I abandoned that and tried just checking the condition number (ratio of the largest and lowest eigenvalues) of my Ap matrices for various fractions. This also didn't produce the greatest results. Here are the condition numbers for a fraction number n from 2 to 10 (as in I would add and subtract p/n to the parameters to compute the derivatives): Then I tried denominators of 10 to 150 in steps of 10: Then from 150 to 200: And finally from

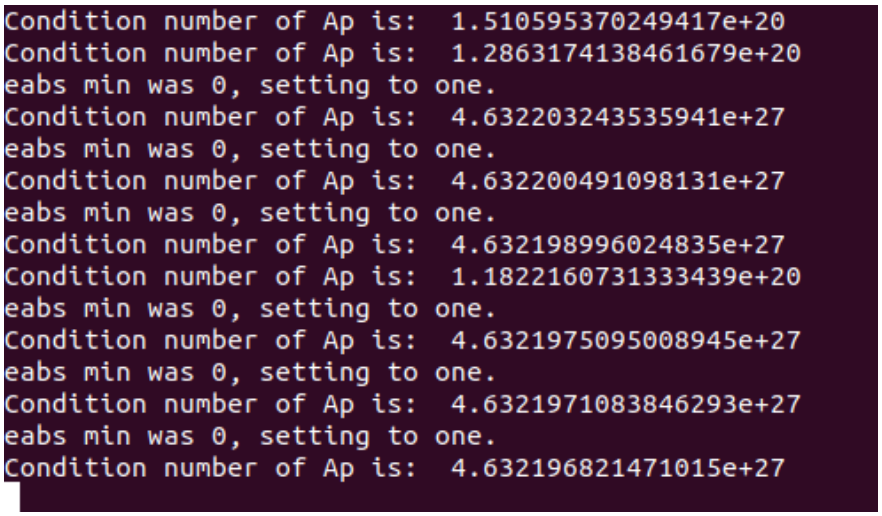

Figure 1: *Condition numbers for the gradient matrix Ap when trying out a denominator of 2 (top) to 10 (bottom) in steps of 1.*

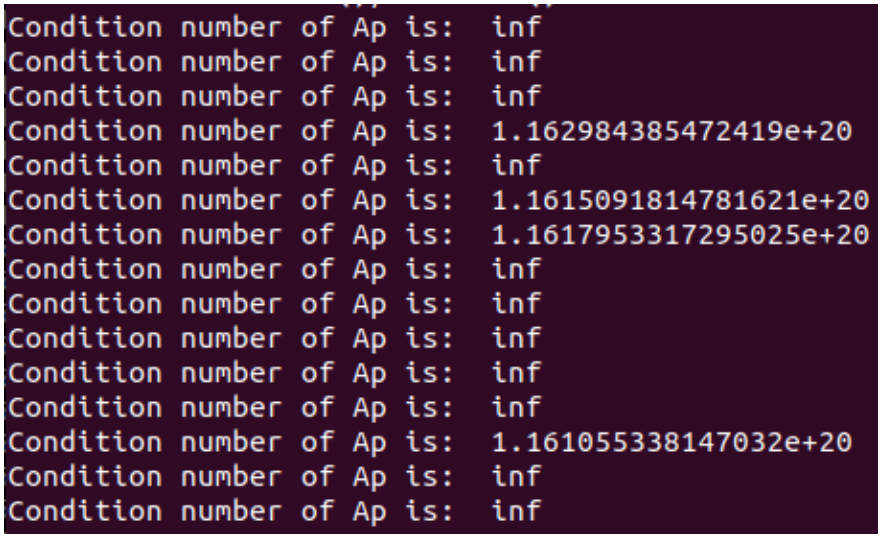

Figure 2: *Condition numbers for the gradient matrix Ap when trying out a denominator of 10 (top) to 150 (bottom) in steps of 10.*

200 to 260: This took a bit of time, and didn't seem to be returning great results, so I just used $n = 200$ for my denominator in my get_Ap function as it produced the best condition number and seemed potentially reasonable.

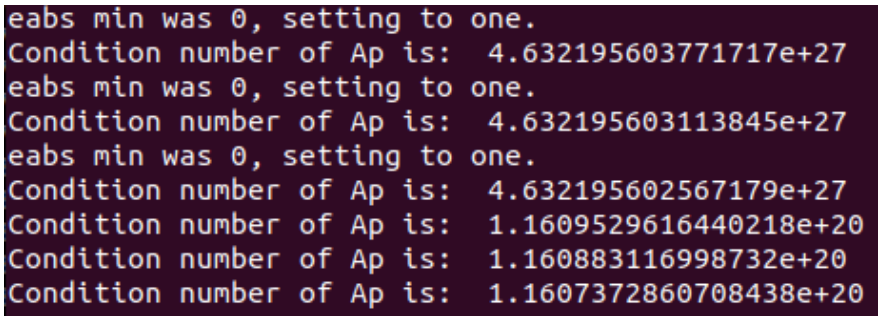

Figure 3: *Condition numbers for the gradient matrix Ap when trying out a denominator of 150 (top) to 200 (bottom) in steps of 10.*

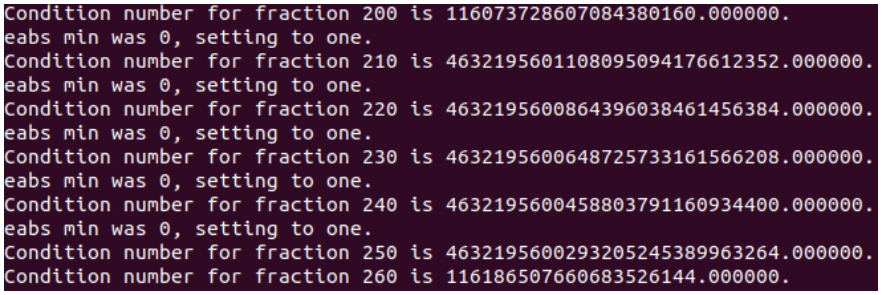

Figure 4: *Condition numbers for the gradient matrix Ap when trying out a denominator of 200 (top) to 6 (bottom) in steps of 10.*

I used this along with my lev_marq function to fit all the parameters defining the power spectrum output by CAMB while holding $\tau = 0.05$. The results can be seen in the first column of Table 1. The fact that this is producing a much better fit than what CAMB output, also that the χ^2 value decreases steadily during the process, indicate to me that it seems to be working somewhat. Another interesting fact is that when I made a mistake and failed to allow my loop to recognize a successful step, which made the function keep increasing λ (and thus prioritize gradient-based descent over curvature-based descent), the χ^2 value asymptotically approached the 1588.4366 that the fit made using Prof. Sievers' guess parameters produced. This means that relying purely on gradient decent and taking a large step in parameter space will produce a fit with exactly the same probability of producing the wmap data as the said fit with Prof. Sievers' guesses. If Prof. Sievers used a form of gradient descent to get his guesses, or something equivalent, this would be another indication that my derivatives seem to be working.

This produced the fit:

Table 1: Table of Fit Parameters

Parameter	Fixed τ	Variable τ
H_0	69 ± 2	69 ± 2
$\Omega_b h^2$	$(2.25 \pm 0.05) \cdot 10^{-2}$	$(2.23 \pm 0.05) \cdot 10^{-2}$
$\Omega_c h^2$	$(1.14 \pm 0.05) \cdot 10^{-1}$	$(1.15 \pm 0.05) \cdot 10^{-1}$
A_s	$(2.04 \pm 0.04) \cdot 10^{-9}$	$(1.87 \pm 0.09) \cdot 10^{-9}$
n_s	$(9.7 \pm 0.1) \cdot 10^{-1}$	$(9.7 \pm 0.1) \cdot 10^{-1}$
τ	0.05	$(4.1 \pm 0.2) \cdot 10^{-1}$
χ^2	1227.92	1227.80

Figure 5: *Fit of power spectrum of wmap cmb data produced using levenberg-marquardt method and holding the optical depth τ to 0.05.*

and the covariance matrix (actual values in "final_fit_hold_tau_flat_cov.txt"):

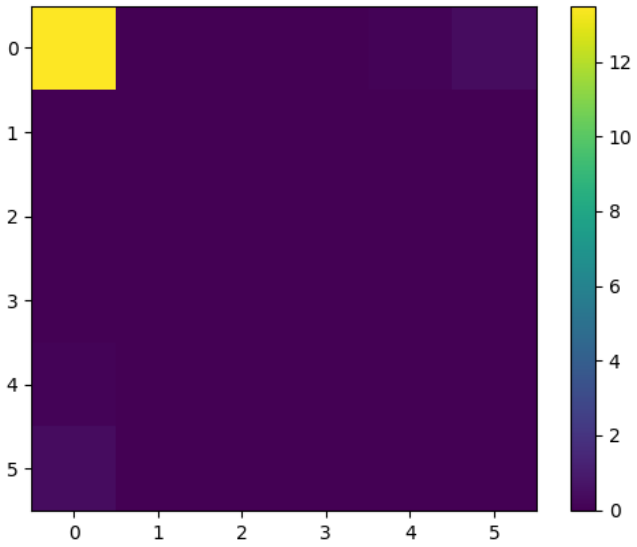

Figure 6: *Covariance matrix of the best fit parameters of the levenberg-marquardt fit produced by keeping τ at 0.05.*

As could be expected, the first cell overpowers the others given the magnitude of H0. If we replace the first column and the first row with the minimum value this might allow us to get a better view of the correlations between parameters:

We can see that the off-diagonal terms are of similar magnitude to the diagonal terms, and so we might expect the parameters of our fit to be correlated to a degree. This correlation might lead us to expect the values of our best fit parameters and their errors to shift when we float τ.

I then performed the task of fitting the data while floating τ. Since we were sort of "pinning" τ and forcing the fitting algorithm to adhere to that, I expected parameters to shift and errors to change, but as we can see in Table 1, the parameters did not shift that much, nor their errors. The freedom to float τ seems to have allowed the fitting function to obtain a "slightly" better fit.

This fit produces the curve:

and the covariance matrix (with full values in "final_fit_with_tau_flat_cov_mat.txt"):

3)

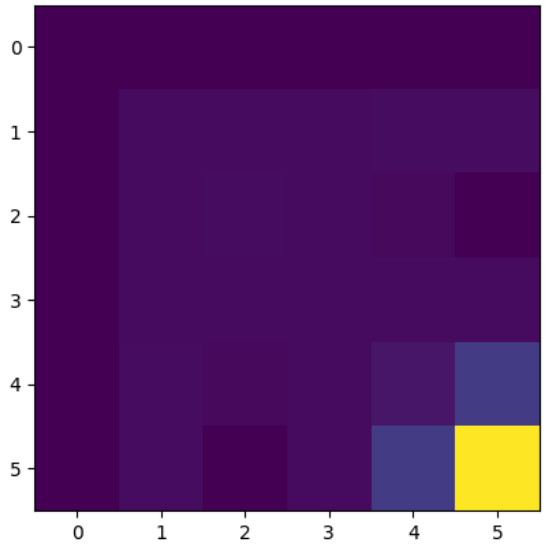

Figure 7: *Same as Figure 6 but the top row and first column have been replaced with the minimum value f the true matrix.*

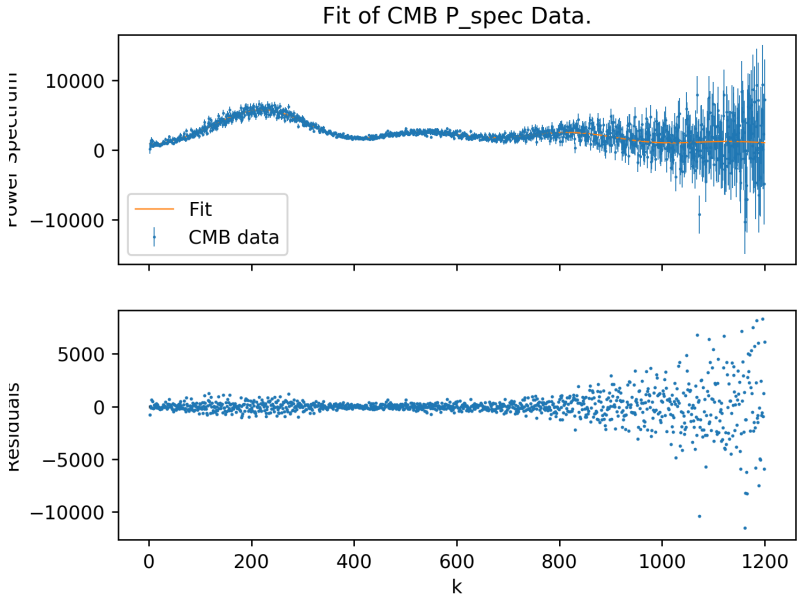

Figure 8: Fit of power spectrum of wmap cmb data produced using levenberg-marquardt method and floating the optical depth τ.

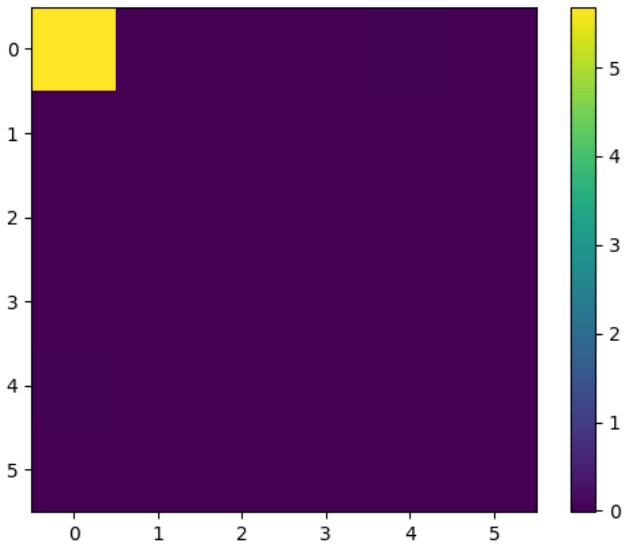

Figure 9: *Covariance matrix of the best fit parameters of the levenberg-marquardt fit produced by fitting all parameters of the camb power spectrum.*