1. With uncorrelated guassian noise, χ^2 is simply given by

$$\chi^2 = \sum_i^N \left(\frac{d_i - y_i}{\sigma_i} \right)^2,$$

where d_i represents the true data, y_i a model fit, and σ_i the known measurement errors. Using the given parameters, the initial χ^2 is found to be 1588.426.

2. (a) Newtons fit produced an output of the fit parameters of:

$$\begin{aligned} H_0 &= 69 \pm 2 \\ \omega_b h^2 &= 0.0225 \pm 0.0005 \\ \omega_c h^2 &= 0.114 \pm 0.005 \\ A_s &= (2.04 \pm 0.04) \cdot 10^{-9} \\ power_{slope} &= 0.97 \pm 0.01 \end{aligned}$$

These parameters result in a final chi square of 1228.246 and produce the below plot.

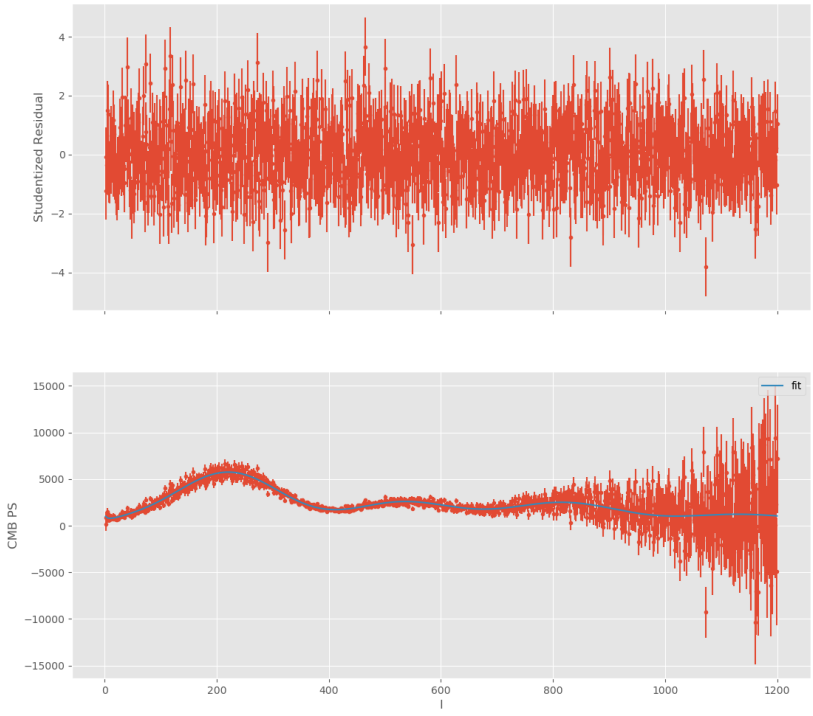

Figure 1: Levenberg-Marquardt Fitting results

(b) Since we are assuming uncorrelated errors, we would expect the new errors on the parameters to remain the same. However, considering τ when taking the gradient during the calculation of the final covariance matrix and recalculating the errors from this does in fact produce different errors, indicating that there is some correlation we haven't taken into account. New results are:

$$\begin{aligned} H_0 &= 69 \pm 4 \\ \omega_b h^2 &= 0.0225 \pm 0.0008 \\ \omega_c h^2 &= 0.114 \pm 0.007 \\ A_s &= (2.04 \pm 0.6) \cdot 10^{-9} \\ power_{slope} &= 0.97 \pm 0.03 \end{aligned}$$

Errors are found by taking the square-root of the diagonal matrix $C = J^T N^{-1} J$ where J is the gradient of our function and N is the noise matrix given by $1/e_i^2$ where e_i are the measurement errors.

(c) To approximate the derivative, I used the double sided numerical approximation given by

$$\frac{\partial f(x; p_0, ..., p_i, ...p_n)}{\partial p_i} \approx \frac{f(x; p_0, ..., p_i + \delta p_i, ...p_n) - f(x; p_0, ..., p_i - \delta p_i, ...p_n)}{2\delta p_i}.$$

For the values of δp_i I used the initial parameter guesses divided by a factor A. In order to determine an ideal value for A, I plotted the derivatives of each parameter over a wide range, and took a value close to when the derivative no longer changed much from one value to an other. See Fig.2 for an example with H_0 being the parameter, this behaviour was qualitatively the same for each parameters, so I chose a value of 70 for A, as this was the smallest value within the band of lines overlapping even when zoomed in.

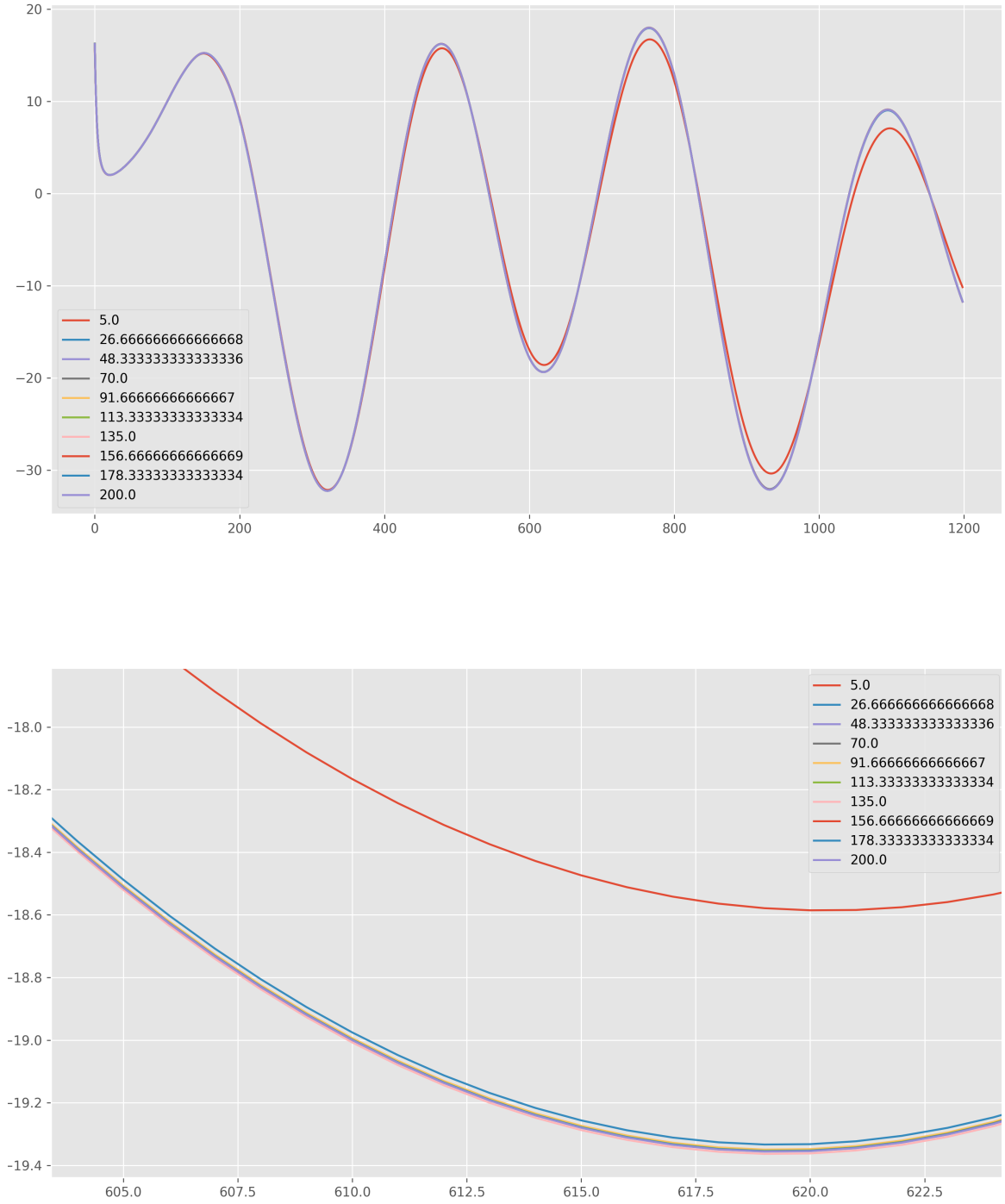

Figure 2: (Top) Gradient of spectrum with respect to H_0 for varying scaling values of δH_0. (Bottom) Zoomed in showing small differences.

3. (a) As can be seen in the following section, I was not able to get a properly converged chain. This resulted in not the best values of

$$\begin{aligned} H_0 &= 79 \pm 3 \\ \omega_b h^2 &= 0.0248 \pm 0.0007 \\ \omega_c h^2 &= 0.099 \pm 0.006 \\ \tau &= 0.26 \pm 0.03 \\ A_s &= (3.0 \pm 0.1) \cdot 10^{-9} \\ power_{slope} &= 1.05 \pm 0.02 \end{aligned}$$

   (b) Unfortunately, I seemingly had a memory leak that would cause the amount of RAM that python used to slowly increase until my computer crashed, this limited me to chain lengths of roughly 7000. The chains would also seemingly converge and stay that way for a while, and then jump to a different set of parameters for a bit, only to return to being converged. Fig. 3 & 4 shows the parameters of a chain and their Fourier transform, each normalized and truncate from 1000 steps to ignore the burn in period. The Fourier transform with no prior is mostly angled indicating that it didn't really converge. With a prior, which the results will be given for later, the Fourier transform is roughly flat over the first two orders of magnitude, indicating that the chain did converge. This is also noticeable in how the chains themselves look a lot more like white noise than in the no prior case.

Figure 3: Chain with no prior parameter input. Legend indicates parameter index.

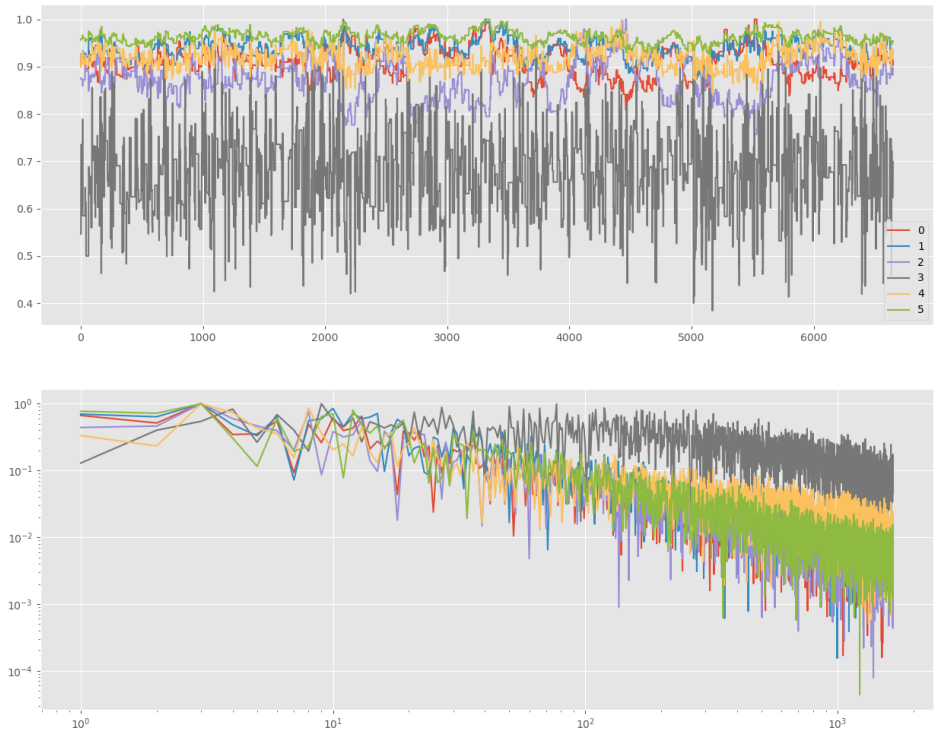

Figure 4: Chain with prior parameter input.

4. The two methods can be done in the following ways.

   (a) For re-analyzing an old chain, we simply give each point a weight corresponding to the chi square value of this specific parameter. That is, the parameters are calculated via

$$\bar{p_i} = \frac{\sum_{converged}^{N} p_i \exp\{-\frac{1}{2}\chi^2_{p_i}\}}{\sum_{converged}^{N} \exp\{-\frac{1}{2}\chi^2_{p_i}\}}$$

   Where

$$\chi^2_{p_i} = \left(\frac{p_i - p_{true}}{\sigma_{true}}\right)^2$$

Re analzying the same chain as in part 3, results in:

$$\begin{aligned} H_0 &= 70 \pm 3 \\ \omega_b h^2 &= 0.0231 \pm 0.0007 \\ \omega_c h^2 &= 0.114 \pm 0.006 \\ \tau &= 0.18 \pm 0.03 \\ A_s &= (2.7 \pm 0.1) \cdot 10^{-9} \\ power_{slope} &= 1.00 \pm 0.02 \end{aligned}$$

Indicating some amount of improvement over the not great initial work...

(b) When running a new chain, we simply modify the χ^2 value calculated for each new set of parameters by adding the $\chi^2_{p_i}$ as above, this results in fit parameters of:

$$\begin{aligned} H_0 &= 69 \pm 2 \\ \omega_b h^2 &= 0.0249 \pm 0.0006 \\ \omega_c h^2 &= 0.115 \pm 0.005 \\ \tau &= 0.055 \pm 0.008 \\ A_s &= (2.07 \pm 0.05) \cdot 10^{-9} \\ power_{slope} &= 0.97 \pm 0.01 \end{aligned}$$

Showing much better results, which makes sense since this chain actually converged. These parameters result in a final chi square of 1330.084 and results in the below plot. While slightly worse of a fit, the errors on these parameters are tighter. and include fitting tau. With a properly converging chain, these errors should work out to be better as they can more properly capture the landscape of the parameter minima.

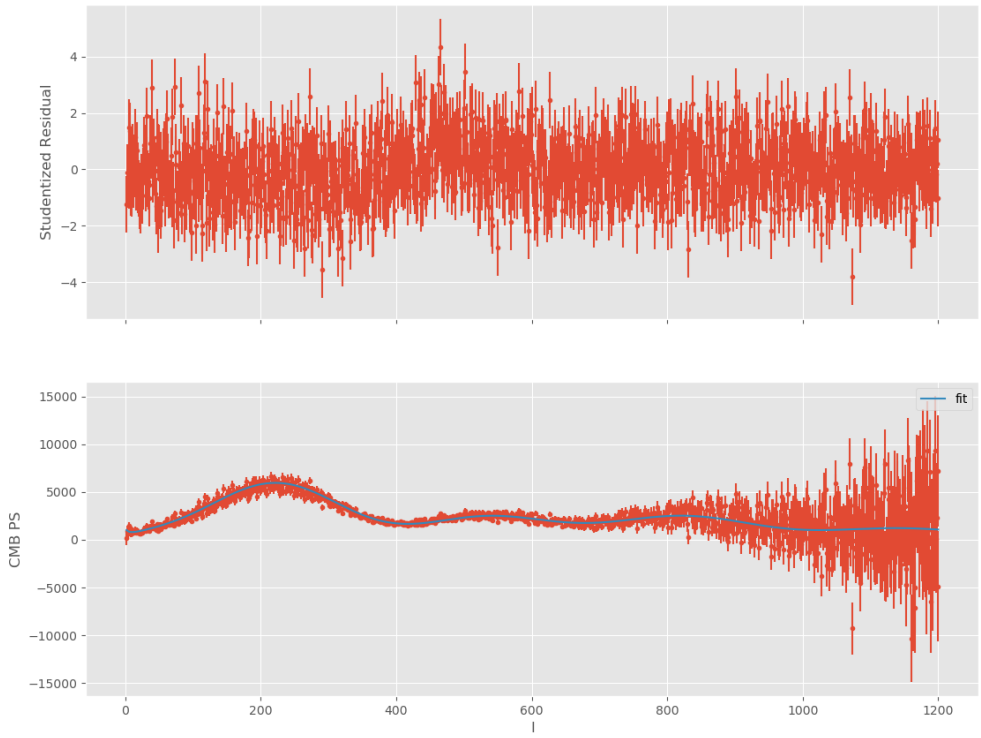

Figure 5: MCMC Fitting results

Raw output: LM Fitting:

```
>> python fitting_cmb.py
Initial guess: [6.5e+01 2.0e-02 1.0e-01 2.0e-09 9.6e-01]
Gives chi2 = 1588.426
Lambda set to: 1.000000e-03
 params : [2.33409307e+00 2.45453267e-03 1.62002608e-02 6.89932725e-11
 6.37758664e-03]
Parameters: [6.73340931e+01 2.24545327e-02 1.16200261e-01 2.06899327e-09
 9.66377587e-01]
Gives chi2= 1234.842
Lambda set to: 1.000000e-04
 params : [ 1.94976292e+00  3.72164960e-05 -2.21402209e-03 -2.60537698e-11
  3.28635183e-03]
Parameters: [6.92838560e+01 2.24917492e-02 1.13986239e-01 2.04293950e-09
 9.69663938e-01]
Gives chi2= 1227.922
Lambda set to: 1.000000e-05
 params : [ 4.68471026e-02 -1.32105003e-06 -8.83640502e-05 -6.73468662e-13
  5.30351749e-05]
Parameters: [6.93307031e+01 2.24904281e-02 1.13897875e-01 2.04226603e-09
 9.69716974e-01]
Gives chi2= 1227.921
Lambda set to: 1.000000e-06
 params : [ 1.62600042e-03  1.04314501e-07 -3.79835232e-06 -2.51285780e-14
  5.05992598e-06]
Parameters: [6.93323291e+01 2.24905324e-02 1.13894076e-01 2.04224091e-09
 9.69722034e-01]
Gives chi2= 1227.921
Lambda set to: 1.000000e-07
Lambda set to: 1.000000e-06
Converged!
 params : [1.11790623e-06 4.72948036e-09 2.94867319e-08 2.74793424e-16
 4.07362447e-08]
Parameters: [6.93323302e+01 2.24905372e-02 1.13894106e-01 2.04224118e-09
 9.69722074e-01]
Gives chi2= 1227.921
```

Analyzing chains:

```
>> python chanal.py chain27-21-09-39
Cutoff start? 1000
Parameter means are:
[7.93058317e+01 2.47683351e-02 9.86992540e-02 2.58401306e-01
 3.01740025e-09 1.05078415e+00]
Parameter errors are:
[3.09092215e+00 6.96747113e-04 5.71571234e-03 2.60649716e-02
 1.34223343e-10 1.86328256e-02]
Importance Sampled Parameters are:
[70.56938227098757, 0.023079353797447293, 0.11367958995125091,
0.18401608854210474, 2.708889670579451e-09, 1.0001440992143145]
>> python chanal.py prior_chain27-22-26-57
Cutoff start? 1000
Parameter means are:
[6.90661342e+01 2.24890784e-02 1.14880516e-01 5.52349999e-02
 2.06974270e-09 9.68497954e-01]
Parameter errors are:
[2.50934918e+00 5.91870949e-04 5.05638163e-03 7.65284298e-03
 4.73307562e-11 1.49978437e-02]
```