Here is the report of PHYS512 problemset 3 written by Jiaxing MA.

problem1

a

To find a linear fit, we can expand the equation, the result looks like:

$$z = a(x^2 + y^2) - 2ax_0x - 2ay_0y + a(x_0^2 + y_0^2) - z_0$$

As we can see, the we can choose another group of parameters, and the equation looks like:

$$z = a(x^2 + y^2) + bx + cy + d$$

Where, the relation between the new parameters and the old ones is

$$\begin{aligned} a &= a \\ x_0 &= \frac{-b}{2a} \\ y_0 &= \frac{-c}{2a} \\ z_0 &= \frac{b^2 + c^2}{4a} - d \end{aligned}$$

b

To do the linear fit, I set the function to get A as:

```
# Get A for x and y,
def get_A(x,y):
    A = np.zeros([len(x),4])
    # d
    A[:,0]=1
    A[:,1]=y
    A[:,2]=x
    A[:,3]=x**2+y**2
    return A
```

Figure 1

After the fit I got the fit parameter as:

$$[a, b, c, d] = [1.66704455e^{-4}, -1.51231182e^{3}, -1.94115589e^{-2}, 4.53599028e^{-4}]$$

c

To calculate the error in the data, I used the standard divination of the predict z and the data z, here is the code:

```
e = np.std(pred-z)
# focal length in meter
f = (1/(4*fitp[3]))*1e-3
```

Figure 2

and the noise in data I got is 3.7683. To calculate the focal length, I use the parameter a:

$$f = \frac{1}{4a} \times 10^{-3}$$
$$f = 1.4996599841252247m$$

problem2

Here is the code I used to calculate spectrum and the χ^2:

```
cmb=get_spectrum(pars)
sig  = wmap[:,2]
pred = cmb[2:len(wmap[:,0])+2,0]
chisq = np.sum(((wmap[:,1]-pred)/sig)**2)
ax1.plot(pred)
plt.legend()
plt.show()
print("chisquare = ", chisq)
```

Figure 3

Here is the result I have,

```
pars are  [6.5e+01 2.0e-02 1.0e-01 5.0e-02 2.0e-09 9.6e-01]
No handles with labels found to put in legend.
chisquare =  1588.2376532931526
```

Figure 4

$$\chi^2 = 1588.23765$$

problem3

I ran a Newton's method to find the best-fit values. The parameter and error I find is

$$H_0 = 6.93e^{1} \pm 2.40$$
$$w_b h^2 = 2.25e^{-2} \pm 5.36e^{-4}$$
$$w_c h^2 = 1.14e^{-1} \pm 5.21e^{-3}$$
$$A_s = 2.04e^{-9} \pm 3.91e^{-11}$$
$$slope = 9.70e^{1} \pm 1.34e^{-2}$$

Here is the code result.

```
[6.72317873e+01 2.24515469e-02 1.16316689e-01 2.07051060e-09
 9.66109953e-01] 1588.2376532931526
[6.92521896e+01 2.24910377e-02 1.14102305e-01 2.04389814e-09
 9.69632351e-01] 1235.6869942522808
[6.93038325e+01 2.24911081e-02 1.13980136e-01 2.04299964e-09
 9.69719064e-01] 1227.9373195762569
[6.93035849e+01 2.24912615e-02 1.13980906e-01 2.04301170e-09
 9.69721420e-01] 1227.9359998384614
[6.93035941e+01 2.24912630e-02 1.13980888e-01 2.04301151e-09
 9.69721447e-01] 1227.9360029416498
get the good result
final parameters are  [6.93035941e+01 2.24912630e-02 1.13980888e-01 2.04301151e-09
 9.69721447e-01]  with errors  [2.40264754e+00 5.35871828e-04 5.21471741e-03 3.90921374e-11
 1.34219106e-02]
```

Figure 5

The optimized χ^2 I got is 1227.936. I stop the Newton's method when the change in χ^2 is small enough.
When I change the tau, I cannot get newton's method running without get tau to negative, so I just take the derivative of tau and put the derivative into lhs to calculate the new error in the parameter, and the error and parameter became:

$$H_0 = 6.93e^{1} \pm 3.69$$
$$w_b h^2 = 2.25e^{-2} \pm 8.52e^{-4}$$
$$w_c h^2 = 1.14e^{-1} \pm 7.05e^{-3}$$
$$A_s = 2.04e^{-9} \pm 6.02e^{-10}$$
$$slope = 9.70e^{1} \pm 2.62e^{-2}$$

```
final parameters are  [6.93035941e+01 2.24912630e-02 1.13980888e-01 5.00000000e-02
 2.04301151e-09 9.69721447e-01]  with errors  [3.69342094e+00 8.51578673e-04 7.04616746e-03
1.51520661e-01
 6.02484135e-10 2.62448728e-02]
```

Figure 6

Here is the fit and data:

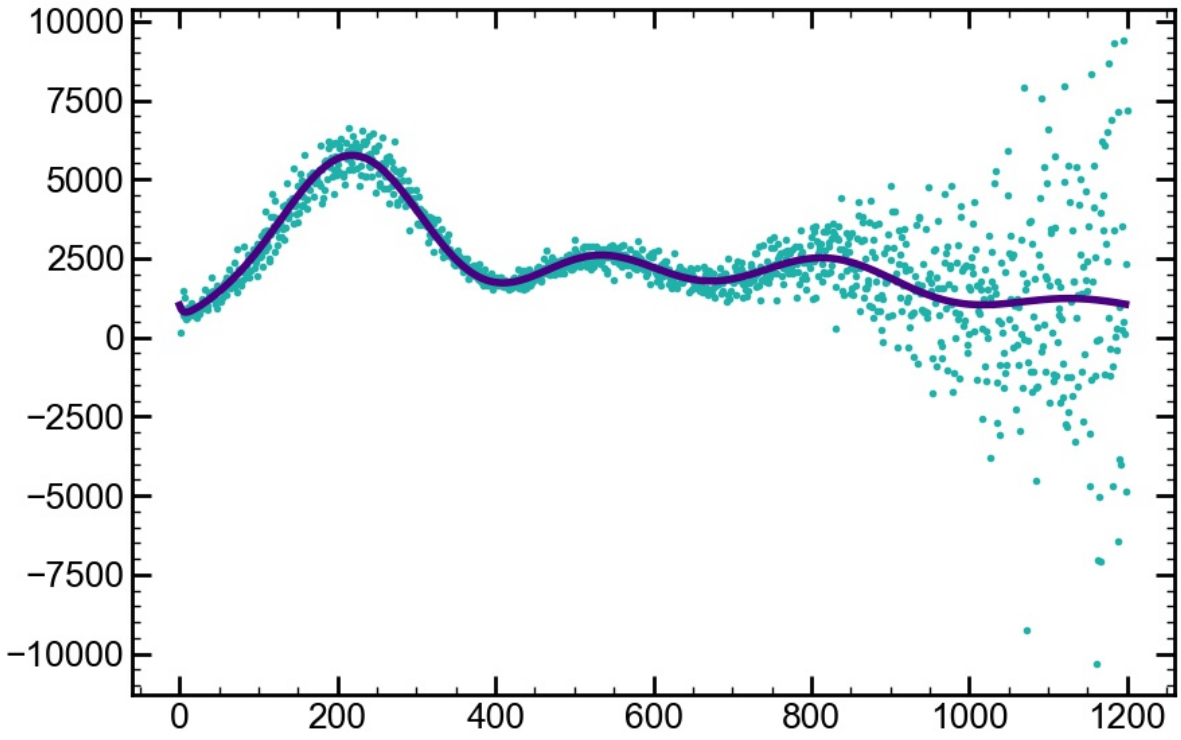
10000
7500
5000
2500
0
−2500
−5000
−7500
−10000
0
200
400
600
800
1000
1200

Figure 7

problem4

I use 10000 steps in mcmc, because my computer cannot get 20000. I use the covariance matrix to set the step size, and here is the code I use:

```
pars_best = pars
pars = np.insert(pars,3,tau)
f_r = get_spectrum(pars_best,tau+dtau)
f_l = get_spectrum(pars_best,tau-dtau)
derivs_tau = (f_r-f_l)/(2*dtau)
derivs = np.insert(derivs,3,derivs_tau,1)
lhs=derivs.T@Ninv@derivs
rhs=derivs.T@Ninv@resid
lhs_inv=np.linalg.inv(lhs)
#  with consider tau
par_sigs=np.sqrt(np.diag(lhs_inv))
par_errs=np.sqrt(np.diag(np.linalg.inv(lhs)))
```

Figure 8

The accept step is 897, and I plot the chain, also do the fft to see if they are converged. Here is the result:

```
onestep i= 9989
jump
jump
onestep i= 9992
onestep i= 9993
onestep i= 9994
walk step = 897
onestep i= 9995
onestep i= 9996
onestep i= 9997
onestep i= 9998
onestep i= 9999
```

Figure 9

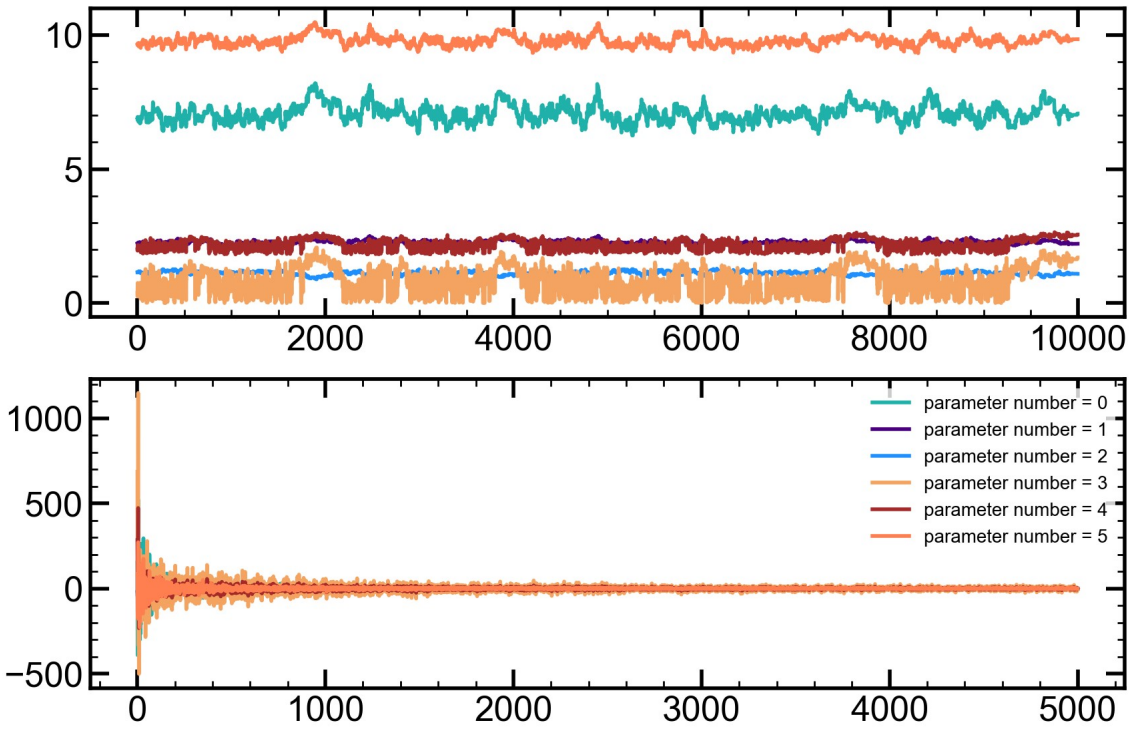

Figure 10: The upper figure is the chain after scale by their magnitude, and lower figure is the fft of the chains, the chains of different parameters are labeled.

From the fft, we can see that is not looks like white noise at lower frequency, and pretty good at higher, I think is not a good converge.
The result I got is:

$$
\begin{aligned}
H_0 &= 7.06e^{1} \pm 3.16 \\
w_b h^2 &= 2.27e^{-2} \pm 6.57e^{-4} \\
w_c h^2 &= 1.12e^{-1} \pm 6.06e^{-3} \\
\tau &= 8.35e^{-2} \pm 4.92e^{-2} \\
A_s &= 2.18e^{-9} \pm 2.00e^{-10} \\
slope &= 9.78e^{-1} \pm 1.84e^{-2}
\end{aligned}
$$

```
No prior tau, parameters = [7.06258029e+01 2.27384973e-02 1.11537612e-01 8.34823934e-02
 2.17913617e-09 9.78368267e-01]Error in parameters= [3.16529750e+00 6.57084763e-04
6.06020846e-03 4.92192146e-02
 1.99814128e-10 1.84410107e-02]
```

Figure 11

The χ^2 I got is 1228.807.

problem5

To set the τ prior, I forced the $\tau$to be in 3 σ in the code, and run the chain again, Here is the code and result I got.

```
if tau< 0.0544-3*0.0073 or tau > 0.0544+3*0.0073:
    print("jump")
```

Figure 12

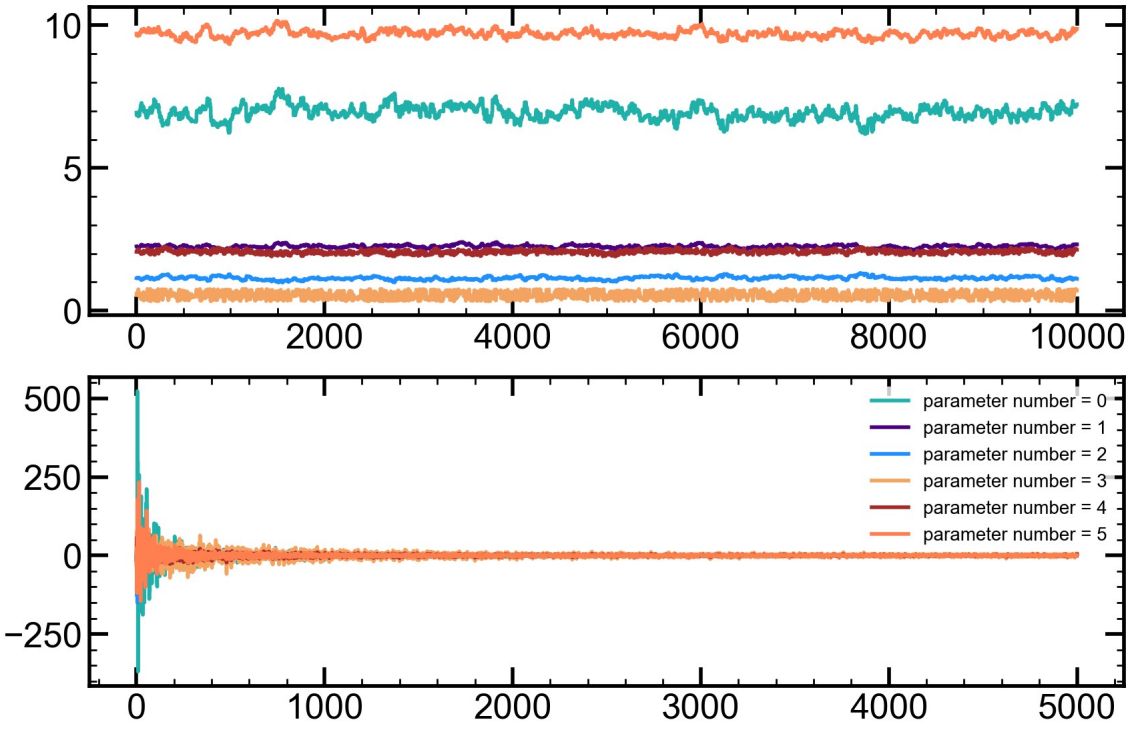

Figure 13: The upper figure is the chain after scale by their magnitude, and lower figure is the fft of the chains, the chains of different parameters are labeled.

As we can, is better converged. The parameters and limits are

$$
\begin{aligned}
H_0 &= 6.95e^{1} \pm 2.40 \\
w_b h^2 &= 2.25e^{-2} \pm 5.11e^{-4} \\
w_c h^2 &= 1.14e^{-1} \pm 5.45e^{-3} \\
\tau &= 5.44e^{-2} \pm 1.26e^{-2} \\
A_s &= 2.06e^{-9} \pm 6.34e^{-11} \\
slope &= 9.70e^{-1} \pm 1.26e^{-2}
\end{aligned}
$$

```
With prior tau, pararmeters = [6.94516813e+01 2.24613360e-02 1.13969231e-01
5.44473588e-02
 2.06081288e-09 9.69659518e-01]Error in parameters= [2.39765640e+00 5.11137570e-04
5.44901786e-03 1.26091523e-02
 6.33775695e-11 1.26376653e-02]
```

Figure 14

The χ^2 I got is 1228.05, is better than not set prior to τ.
Then I did the important sampling to the chains in problem4, and I weight the chains by χ^2 of τ, and here is code I used.

```
# Important Sampling
# get weight vector
wtvec=np.exp(-0.5*((chain1[:,3]-0.0544)/0.0073)**2)
chain1_scat=chain1.copy()
means=np.zeros(chain1.shape[1])
chain1_errs=np.zeros(chain1.shape[1])
for i in range(chain1.shape[1]):
    # weight the parameters
    means[i]=np.sum(wtvec*chain1[:,i])/np.sum(wtvec)
    #subtract the mean from the warm chain so we can calculate the
    #standard deviation
    chain1_scat[:,i]=chain1_scat[:,i]-means[i]
    chain1_errs[i]=np.sqrt(np.sum(chain1_scat[:,i]**2*wtvec)/np.sum(wtvec))

print("With important sample by tau, pararmeters = "+str(means)+ "Error in parameters= "+str(chain1_errs))
```

Figure 15

Here is the result I got:

```
With important sample by tau, pararmeters = [6.95546920e+01 2.25565686e-02 1.13687384e-01
5.45367169e-02
 2.05938723e-09 9.70650289e-01]Error in parameters= [2.54879117e+00 5.60446131e-04
5.14395847e-03 7.30636939e-03
 4.82797060e-11 1.37272205e-02]
```

Figure 16

$$
\begin{aligned}
H_0 &= 6.96e^{1} \pm 2.55 \\
w_b h^2 &= 2.26e^{-2} \pm 5.60e^{-4} \\
w_c h^2 &= 1.14e^{-1} \pm 5.14e^{-3} \\
\tau &= 5.45e^{-2} \pm 7.31e^{-2} \\
A_s &= 2.06e^{-9} \pm 4.83e^{-11} \\
slope &= 9.70e^{-1} \pm 1.37e^{-2}
\end{aligned}
$$

The χ^2 I got is 1227.98, is better than the other two.