Comp Methods Homework 7

Aasim Zahoor Jan (aj3008@g.rit.edu)

October 27, 2020

`https://github.com/AasimZahoor/Comp_methods.git`

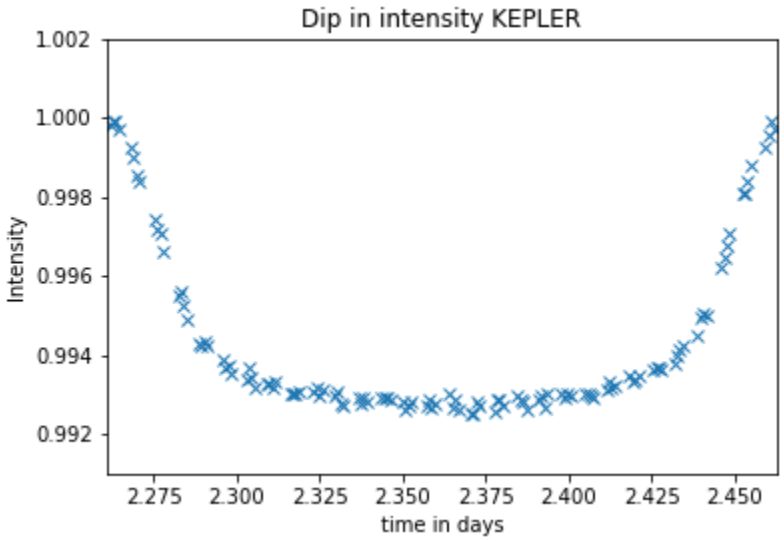
Question 1.

This problem has three parts. **First** it asks us to find the parameters of a box car function while will best fit the data using MCMC. **Second** it asks to find the radius of the planet Kepler-5b. **Third** it asks us to find the uncertainty.

Approach
First I folded the data that was given. I did this by using the below equation.

$$t_{folded} = \frac{t(data) - t_0(data)}{Period}$$

Here, the period is given to be 3.5485 days. Then I plotted the folded data to see if it matched what we expected. Here is the plot.



Then I created two functions, one which gave out the box car model values for given values of dip, tref and width and other which gave the log likelihood for the given box car model.

- **Box(s,tn)**:

  Parameters:
  s : [dip, tref, width of dip] (in that order)
  tn: folded time

Returns:
The data which resembles a Box when plotted
g=1 always(I had made this to include priors and we can add a few lines for it to give priors for the posteriors)

- **Like(I,q)**:

  Parameters:
  I : Intensity (observed)
  q : [Intensity(model), prior(useless as of now)]

  Returns:
  Log Likelihood of the given model

After making these functions I created my MCMC using the Metropolis Hasting Algorithm. First, I define my best guess in the arrays y and xt. Then there is a while loop which controls the number of values you want the MCMC to go through. Here that is 1000. In the while loop, first the code gets a value of dip depth from a normal distribution and calculates the likelihood ratio. If it is greater then 1 then it accepts it, but if it is less than 1 then it accepts or rejects it depending on whether a random value obtained from a uniform distribution is lesser or great the likelihood ratio respectively. Then it does the same for tref and dip width. This goes on till MCMC has gone through 1000 values for dip depth, tref and dip width. Mathematically,

$$If(r >= 1); \quad x_t = y$$

$$If(r < 1); \quad x_t = y \quad if \quad U(0,1) <= r \quad OR \quad x_t = x_t \quad if \quad U(0,1) > r$$

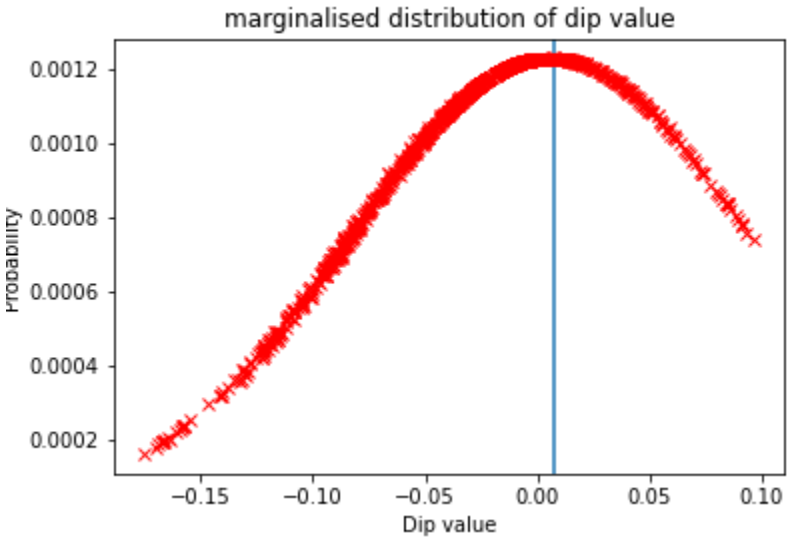
Here,

$$r = \frac{likelihood(model, y)}{likelihood(model, x_t)}$$

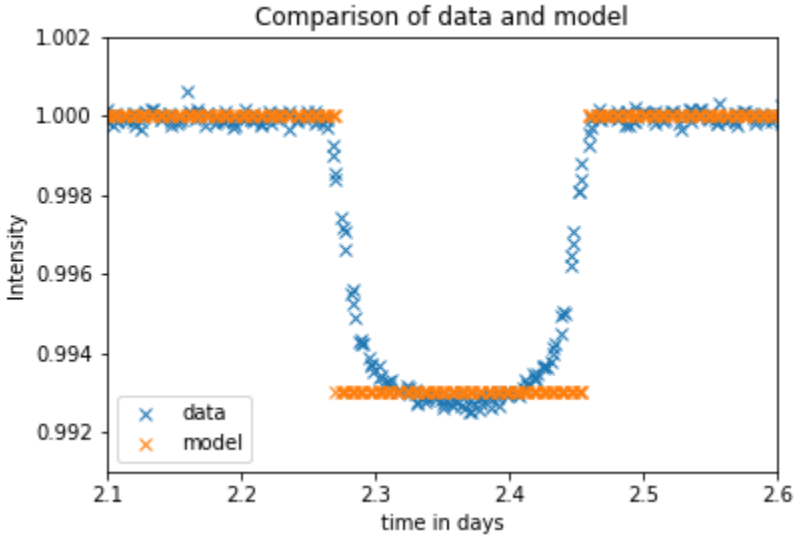
After MCMC my code calculates probability for all 1000 parameter values and finds the set of parameter values which give maximum probability. The Radius of the planet is obtained using the equation.(R_{star}=1.79*R_{sun})

$$\Delta I/I = R^2_{planet}/R^2_{star}$$

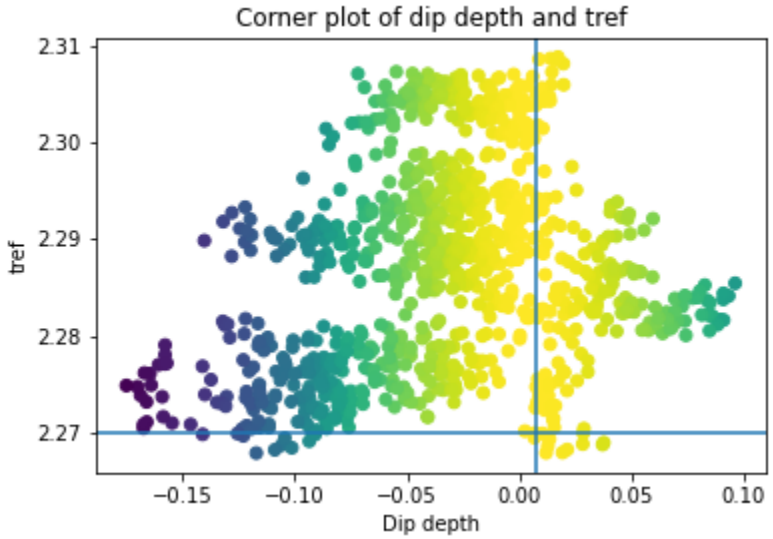
Results

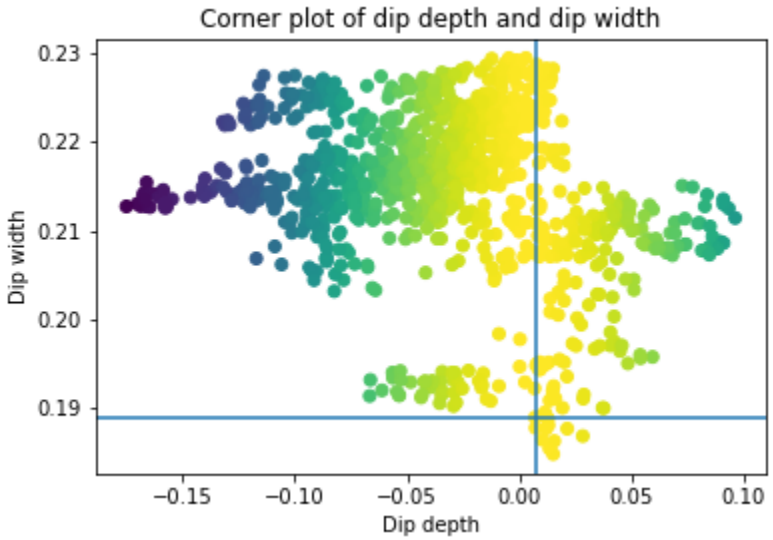


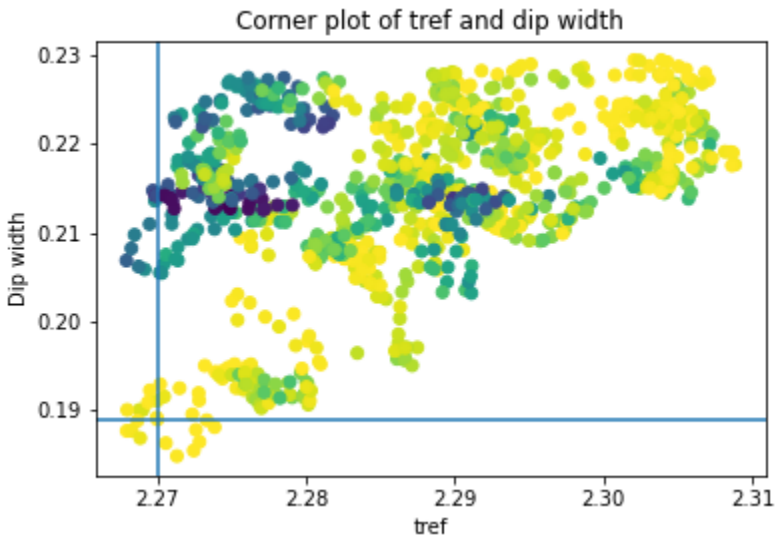
This plot shows the marginal distribution of dip depth. Even though the negative values are nonphysical I allowed them because it gave it a good shape. Blue line is the value of dip depth at maximum probability.



This plot compares the model with data


Corner plot of dip depth and tref
2.31
2.30
2.29
2.28
2.27
tref
−0.15
−0.10
−0.05
0.00
0.05
0.10
Dip depth


Corner plot of dip depth and dip width
0.23
0.22
0.21
0.20
0.19
Dip width
−0.15
−0.10
−0.05
0.00
0.05
0.10
Dip depth



These three are all corner plots with the cross hairs representing the most probable value. Brighter the points, more probable they are

```
the value of parameters is [0.006968087322077456, 2.2700049579707535, 0.1889992964941653]
the error associated with dip is = 0.05059115003256312
radius of the planet is= 0.14942037541335645

In [10]:
```

The parameter values are of form [dip depth, tref, dip width]. The error(one sigma) is for dip depth and radius is in terms of radius of Sun.

Solutions to problems I faced

- Used sympy.exp instead of numpy.exp to store small likelihood values. numpy.exp would consider then zero causing problems while calculating likelihood ratio.
- Made the standard deviation of Gaussian distribution smaller so that it doesn't spit out some value which is far away from the true value and hence making it necessary for the code to go through more than 1000 parameter values to reach at the most probable value. This only works if you give a best initial guess and by that I mean as the same order as the answer.
- I had initially made the mistake of treating the parameters as arrays. What I mean is that I used normal.random.multivariate which gave three variables and I calculated likelihood ratio r only once. But later on I realised that even if the MCMC was moving in the right direction for tref but not for other parameters, it couldn't distinguish that. And that was the reason my code was never arriving at values remotely close to true value. But now my code checks the likelihood ratio for each of the parameters individually, allowing it to know that whether it is moving in right direction for each of the parameters.

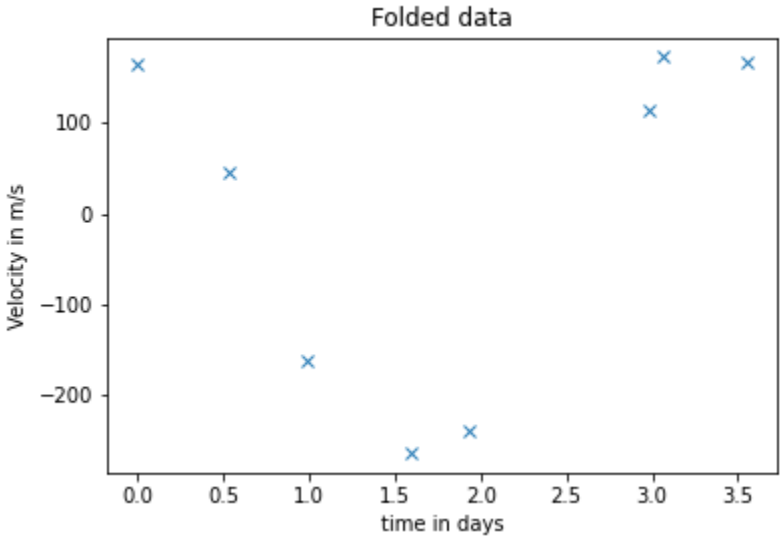
2. Bonus

This problem has three parts. **First** it asks us to find the parameters of a sine wave function while will best fit the data using MCMC. **Second** it asks to find the mass of the planet Kepler-5b.

Approach
First I folded the data that was given. I did this by using the below equation.

$$t_{folded} = \frac{t(data) - t_0(data)}{Period}$$

Here, the period is given to be 3.5485 days. Then I plotted the folded data to see if it matched what we expected. Here is the plot.



Then I created two functions, one which gave out the sine model values for given values of velocity and phase and other which gave the log likelihood for the given sine model.

- **sin(tn,q)**:

  Parameters:
  tn: folded time
  q=[velocity,phase]
  Returns:
  The data which resembles a sin wave fitting the observed data when plotted
  g=1 always (This was for priors to calculate the posteriors but for now it is useless)

- **Like(I,q)**:

  Parameters:
  I : observed data
  q : [model data, prior(useless as of now, give whatever value. Since this comes from sin function it gets a value of 1)]
  Returns:
  Log Likelihood of the given model

After making these functions I created my MCMC using the Metropolis Hasting Algorithm. First, I define my best guess in the arrays y and xt. Then there is a while loop which controls the number of values you want the MCMC to go through. Here that is 1000. In the while loop, first the code gets a value of velocity from a normal distribution and calculates the likelihood ratio. If it is greater then 1 then it accepts it, but if it is less than 1 then it accepts or rejects it depending on whether t a random value obtained from a uniform distribution is lesser or great the likelihood ratio respectively. It does the same for phase.

$$If(r >= 1); \quad x_t = y$$

$$If(r < 1); \quad x_t = y \quad if \quad U(0,1) <= r \quad OR \quad x_t = x_t \quad if \quad U(0,1) > r$$

Here,

$$r = \frac{likelihood(model, y)}{likelihood(model, x_t)}$$

After MCMC my code calculates probability for all 1000 parameter values and finds the set of parameter values which give maximum probability. We calculate the mass through the following steps:

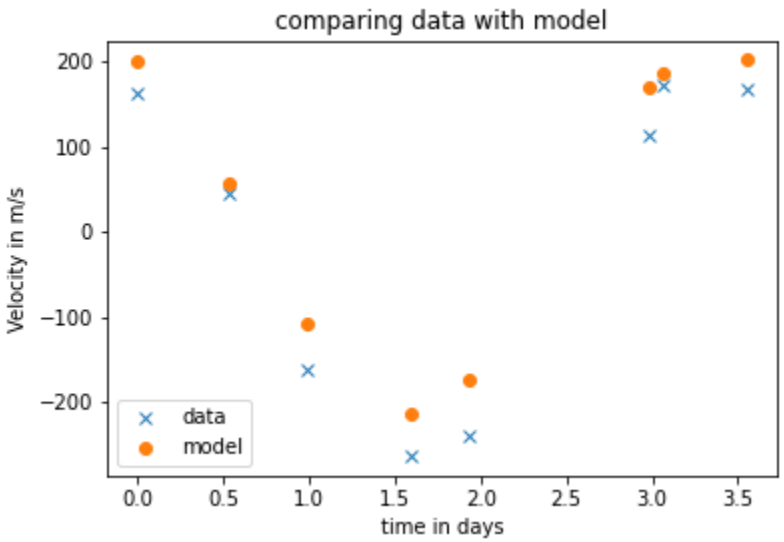
$$Step\ 1(Keplers\ Law) : R_{planets\ orbit} = (\frac{G * M_{star}}{4 * pi^2} * period^2)^{1/3}$$

$$Step\ 2 :\ V_{planet} = \frac{2 * pi * R_{planet's\ orbit}}{Period}$$

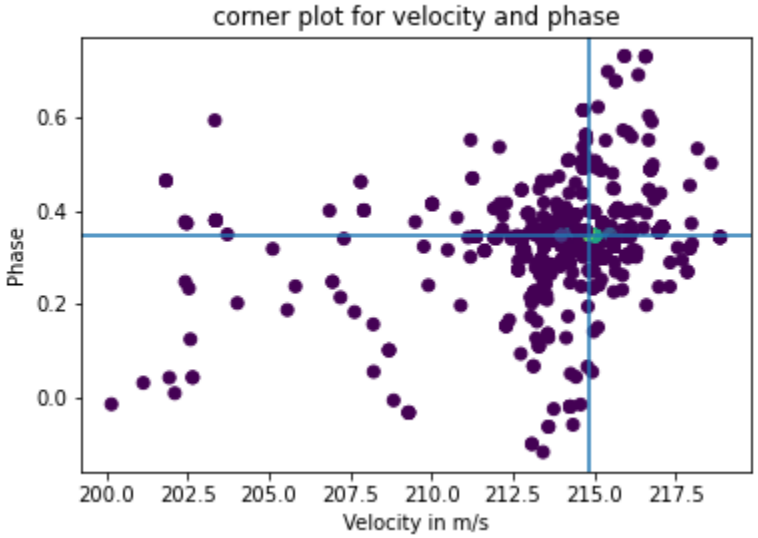
$$Step\ 3(Momentum\ conservation) :\ M_{planet} = \frac{M_{star} * V_{star}}{V_{planet}}$$

$$Density = M_{planet}/Volume_{planet}$$

Results



This figure compares the data with fit(sine).



Corner plot for Phase and velocity

```
In [18]: runcell(1, '/Users/aj3008/Desktop/MS_3rd_Sem/Comp_methods_in_AST/Comp_methods/
Hw_7/Bonus.py')
the velocity is= 214.53194306829604 m/s The phase is = 0.3500981722743863 radians
uncertainty associated with velocity is= 2.4408047406157527
mass of planet is= 0.0018740175202434266 M_sun
density is= 773.3344558724491 kg/m**3

In [19]:
```

I found that the radius of this planet is much larger than that of Earth's but it's density is lesser than that of water so I would say it is a GAS GIANT.