E0 259 Assignment-2

Aditya Gupta
SR No: 22205

In this module, we are given the Mars opposition data observed by Tycho Brahe and later studied by Kepler. We try to fit the data to Kepler's model of the solar system and find the best fit parameters. The data contains the time, latitude and longitude of Mars during 12 oppositions over the span of 22 years.

Kepler's model

Kepler initially assumed that the planets move in circular orbits around the Sun with a constant angular velocity. However, the data did not fit this model. He then proposed a model with two imaginary points: the center, around which the planets move in circular orbits, and the equant, around which the planets move with a constant angular velocity. Now we can try to fit the data to this model.

To represent the positions of these points, we imagine a coordinate system with Sun at the origin and the distance between the Sun and the center defined as 1 unit of distance. The line connecting the Sun and the center makes an angle c with the x-axis. Mars orbits in a circle around the center with radius r. The equant is at a distance e_1 from the Sun and makes an angle e_2 with the x-axis. Mars orbits with a constant angular velocity s around the equant. Also, at the first opposition in the data, Mars makes an angle z with the x-axis when viewed from the equant.

Implementation

We will find the best set of parameters using grid search techniques. To make this computationally feasible, we do this in two steps. First, we find a set of parameters which have a decent fit with the data. Then we refine the search around this set of parameters to find the best fit.

We have to optimize the parameters c, r, e_1, e_2, z, s to minimize the maximum error in the data. Here error is calculated by finding the angular difference (from the Sun) between the position of Mars on its orbit around the center as predicted from the Sun's longitude and the longitude calculated from the equant. Since we have 12 oppositions, we will have 12 errors. We have to find a fit where the maximum error is less than 4 arc minutes.

Since each data point is independent of the others, we can easily parallelize the search. Using the multiprocessing library in python, we can run the search on multiple cores to speed up the process. This reduces our running time by a factor of the number of cpu cores available.

Initial search

Out of the 6 parameters, we already have a good estimate of s since we know that Mars takes about 687 days to complete one orbit. For the initial search, we will use this value of s and search for the other parameters.

Now, c and e_2 are the angular position of the center and the equant respectively. Since these can range from 0 to 360 degrees, we will search for these parameters in steps of 10 degrees. These will be searched to a higher precision in the next step.

For e_1, since we do not have a good bound on this parameter, we will search for this parameter in steps of 0.1 units of distance, starting from 0.5 units. We do not start from 0 since we already know that the equant does not coincide with the Sun (based on the fact that Mars does not have a constant angular velocity around the Sun). We need to keep e_1 as the outermost loop and keep increasing it until we find a reasonable fit.

When it comes to z, it is also an angle that can range from 0 to 360 degrees. However, we can restrict the range of this parameter using some simple geometry. We know that Mars is somewhere along the longitude line drawn from the Sun. This means that the longitude line drawn from the equant should intersect with the longitude line drawn from the Sun at some point. Using this, we know that if the longitude of Mars from the sun is l, then z can't be more than l (otherwise they will intersect behind the Sun). z also can't be less than the angle between the Sun and the equant. We can use these bounds to restrict the search space for z.

Finally, we have r. We can restrict the search space for r by finding a rough estimate for it first using the other parameters described above. We can approximate r as the average distance between the intersection of the longitude lines drawn from the Sun and the equant. We can then search for r in a 10% interval around this value.

The initial search takes into account the above parameters and ranges to return a set of parameters that have a max error of less than 0.5 degrees (30 arc minutes). The running time of this search was about 30 minutes. This search returned 87 sets of parameters, where the lowest max error was 0.33 degrees. The full set of parameters is given in 'filter.txt'.

Refinement

Now that our search has been limited to only 87 points, we can search over smaller intervals to find the best fit. This time we can consider all 6 parameters for optimisation. However, trying to minimise the max error directly is not a good idea since the error is not a smooth function of the parameters. Instead, we can try to minimise the sum of the squares of all the 12 errors. This will indirectly minimise the max error as well. Since the function is smooth, we have a higher chance of avoiding local minima. It also makes the search faster.

Another optimisation we can do is to optimise each parameter one by one instead of writing a nested loop for all of them. We search by dividing a 10% interval around the current value of the parameter into 1000 parts. For s, we use a 1% interval since we already have a pretty good estimate for it. If we tried to optimise all parameters at once, we would have to search over 1000^6 points. However, by optimising one parameter at a time, we only have to search over 1000×6 points. This makes the search much faster. This method of optimisation is a simpler version of an optimisation technique called 'Linesearch'. Linesearch is generally used for functions which are not smooth and differentiable, and hence can't be optimised using gradient descent. The scipy library in python also uses this method in its implementation of the 'Powell algorithm' for optimisation. However, we will stick to this simpler method of linesearch here.

For every point, we perform the refinement until the error reduction after an iteration becomes less than 1 arc second. We do this for all 87 points and find the set of parameters that result in the lowest max error. The running time of this search was about 4 minutes. The best set of parameters obtained had a max error of 0.0539 degrees (3.23 arc minutes). This is an acceptable fit for the data, and thus we can stop the search here.

Best fit

After running the refinement search, we find the best set of parameters to be:

$$\begin{aligned}
c &= 148.1624414062\\
r &= 8.6141152549\\
e_1 &= 1.5997084977\\
e_2 &= 148.9094921875\\
z &= 55.8961816406\\
s &= 0.5240725323
\end{aligned}$$

The errors for this set of parameters are:

```
[-0.0015521339,  0.0343470300,  0.0506838043,  0.0304692409,
  0.0539390389, -0.0066521782,  0.0345538086,  0.0172934821,
 -0.0155583705, -0.0138169837, -0.0416928534, -0.0165272041]
```

All angles mentioned here are in degrees. As we can see, the max error is 0.0539 degrees, which is less than 4 arc minutes. This is a good fit for the data. We can conclude that the data fits Kepler's model of the solar system with the above parameters, within the error bounds of the original data. Obviously we know today that this model is not correct, which was later shown by Kepler himself using triangulation of Mars' position from Earth.

The plot of the data and the fit is shown below.

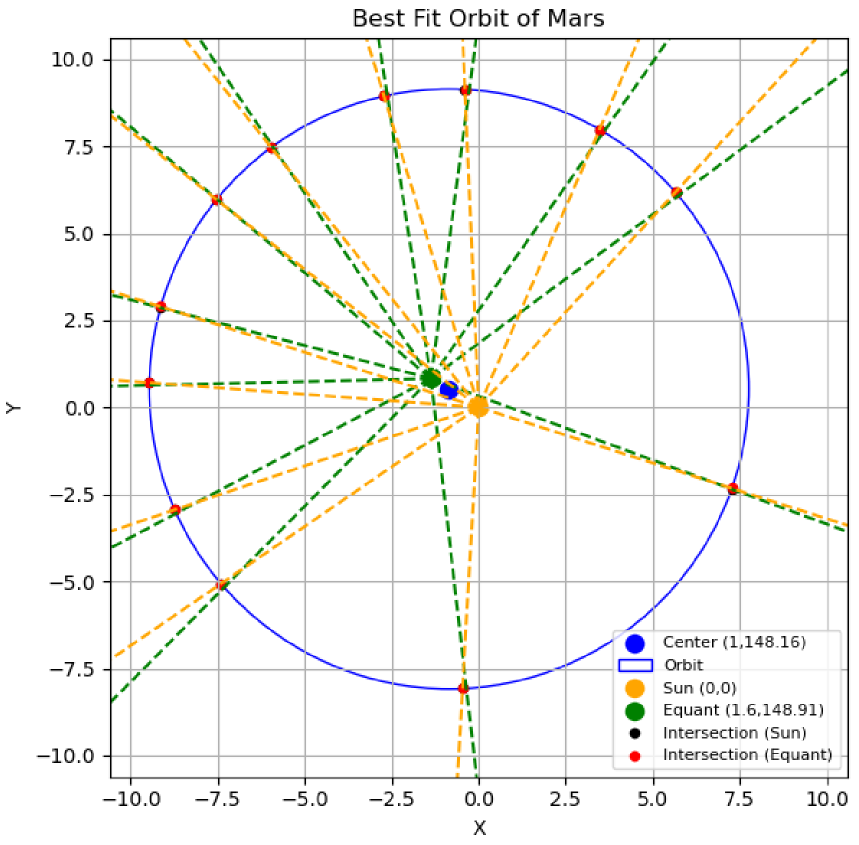

Figure 1: Best fit for Mars opposition data