EECS 189 Project S Final

A Modern Spin on Ptolemy's Geocentric Universe Model

https://github.com/normankarr/CS189-Final-Project.git

Norman Karr, Matthew Martinez, Rodrigo Palmaka
Alex Gao, Matthew Nemeth, Kristof Pusztai,

December 1, 2020

Abstract

The formal subject known as astronomy in our modern day was arguably pioneered by Greek astronomers about two thousand years ago. Equipped with only the basic geometric knowledge of the time, they were able to produce relatively precise representations of our solar system. As such, we attempt to recreate these elegant predictions using modern day machine learning methods. In particular we focused on replicating Ptolemy's geocentric model of our solar system. This method incorporates complex notions of geometry, centered around the belief that planetary motion is governed by epicycloids. We begin our analysis by exploring the basic properties of the data, followed by current machine learning techniques which take into account favorable geometric and trigonometric properties in the data. As a supplementary analysis, we explain and show that we can predict the lunar position and cycles using a trigonometric kernel regression.

1 Introduction

For this project we have decided to place ourselves in the shoes of the ancient Greek astronomer Ptolemy. Ptolemy was one of the first Greek astronomers to attempt to construct a geometric model of the celestial bodies in our solar system, in particular he built upon the geocentric model of the universe. Although Greek astronomy may be predated by ancient China, developments in astronomy made by the Greek astronomers such as Ptolemy cannot be understated. As such we have conducted our analysis on the premise that the geocentric model is, in fact, the true model governing the celestial bodies in the solar system. The geocentric model functions under the belief that the Earth is at the center of the solar system and that all other bodies we observe rotate around us. Under this model, Ptolemy believed the motion of the rotating bodies were governed by circles with epicycles [1]. These epicycles were introduced with the goal of explaining the apparent "backwards motion" of planets orbits when viewed from Earth. While we know today that this model is physically incorrect, the predictive power of this model garnered it's acceptance for over 1,000 years of human history. To the best of our team's knowledge there is no straight forward or canonical method to fit this model in 3 dimensions consequently creating a problem that we believe is ripe for machine learning.

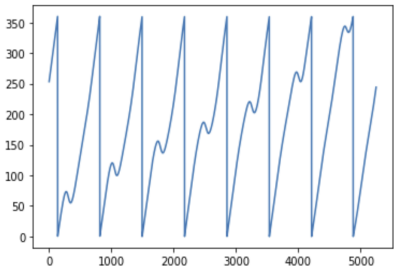
2 Data Collection and EDA (Data_Generation Fourier Predictons.ipynb)

For our data source, we used the AstroPy Python library to generate right ascension and declination for each body using the Jet Propulsion Laboratory (JPL) ephemeris dataset. We generated 10,500 data points for each body with the goal of including at least 2 orbit "cycles" within our data. The purpose of this was to ensure that we had enough data to

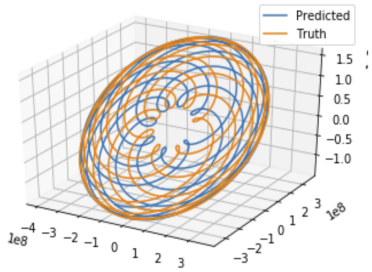
split into training, validation, and test sets while still capturing learn-able patterns within our training set. Additionally, we converted the standard angular astronomic measurements of right ascension and declination into X, Y, Z coordinates relative to Earth [2]. This allowed us to explore and compare two different avenues of prediction, focusing either on X, Y, Z or ascension/declination measurements. The data generation process can be seen in our Data_Generation.ipynb jupyter notebook in our project github repository.

2.1 Initial Exploration

We began our analysis by conducting standard EDA on our gathered data. In particular we made sure to plot the ascension and declination data that we had gathered in order to see if any clear trends existed. When plotting this, what we saw was immediately familiar. we had noticed an almost saw-tooth like sinusoid (See Figure below). We felt this problem was prime for Fourier based analysis and as such constructed a method to allow us to predict the ascension and declination using the dominant frequencies within the data. We were able to use the Fourier and inverse-Fourier transforms in such a way that we could retain only the top-k frequencies for prediction purposes. For each planet we set up a validation technique where we would extract the top k frequencies from one full cycle and then use them to predict on the next full cycle, recording the MSE for each k. What we noticed was that after running this on each planet, the k with the lowest MSE was when k equaled the full set of frequencies. This meant, that if available, we should use a full training cycle to predict future cycles around the earth. However, the one issues we encountered was that, each ellipse would not end in the same place it started, meaning that although the cycles had the same shape, there was a slight angle of rotation between each cycle (see Figure below).



(a) Mars Ascension



(b) Fourier Prediction

Figure(a) shows raw data of Mars' degree of right ascension on the Y-axis plotted over time on the X-axis.
Figure(b) shows predicted values for right ascension in 3D space relative to Earth.

2.2 Reducing to a planar problem

Upon examination of data and plots, we noticed that each planet appeared to orbit on a plane. Therefore, we wanted to explore methods to reduce the data to 2-dimensions. We primarily used two strategies to do this. The first method was simply applying PCA analysis; the second method was a geometrical approach we developed ourselves which we describe in the following paragraph.

Given that the orbits lie in a plane, this means we simply need to rotate the data such that it is has a low variance along one axis. To find the proper rotations, we used a heavily geometrical approach. We began by finding a normal vector for the orbit. We did this by sampling a random set of triplets of points and then taking the cross product of the two vectors represented by the triplet. We then produce an orbital normal vector by averaging

and normalizing the results of our cross products. Given this normal vector, we then found its spherical coordinates: $\phi = cos^{-1}(z)$ and $\theta = tan^{-1}(\frac{y}{x})$ and then performed two subsequent rotations with the following 3D rotation matrices, $R_z(-\theta)$ and then $R_y(-\phi)$ to align the normal vector with the z-axis. Once the normal vector was aligned with the z-axis, the majority of the data will lie in the xy plane.

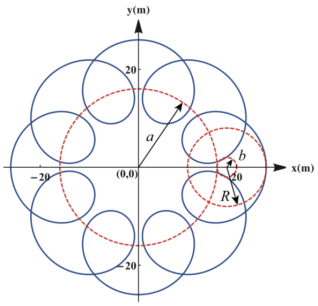
3 Methods for learning orbits

3.1 An Experimental Method: Fitting an Epicycloid (Epicycloid_Demo.ipynb)

One of our initial approaches was to try and to create a model that strictly followed Ptolemy's ideas. Ptolemy theorized that all planets orbited the earth following trajectories he referred to as epicycles which result in geometrical shapes known as prolate epicycloids or epitrochoids; which we will refer to as epicycloid. Our goal was to explicitly fit these shapes to each planet's orbits. We began by figuring out how to mathematically represent an epicycloid and found that it could be modelled with the following parametric equations [3]:

$$x(t) = (a+b)cos(t) - Rcos(\frac{a+b}{b}t)$$

$$y(t) = (a+b)sin(t) - Rsin(\frac{a+b}{b}t)$$



(a) Example Prolate Epicycloid

The main nuance that we have to account for is our parameter in the parametric equation. We want our equations X(t) and Y(t) to be parameterized about time. When we plotted X(t) and Y(t) from data we realized that we would need to explicitly tune the parameter (these plots can be seen in the *Epicycloid_Demo.ipynb*). Any given path had to be able to be slowed down, sped up, and offset. Thus we combined all of that information to create the following equations:

$$x_{epi}(t) = (a+b)cos(\frac{n\pi}{\Delta t}(t-\phi)) - Rcos(\frac{a+b}{b}\frac{n\pi}{\Delta t}(t-\phi))$$

$$y_{epi}(t) = (a+b)sin(\frac{n\pi}{\Delta t}(t-\phi)) - Rsin(\frac{a+b}{b}\frac{n\pi}{\Delta t}(t-\phi))$$

These equations would be able to express any epicycloid path in terms of time with only 5 degrees-of-freedom/parameters: $a, b, \frac{n}{\Delta t}, \phi, R$. The two new terms $\frac{n}{\Delta t}$ and ϕ represent the number of periods in a given span of time and the time offset respectively. With these equations, we can now fit x(t) and y(t) (both fits should output similar parameters). To perform the fit, we performed gradient descent on a squared loss function with the following equations:

$$L_x = \frac{1}{2}\sum_t (x_{epi}(t) - x_{true}(t))^2 \; L_y = \frac{1}{2}\sum_t (y_{epi}(t) - y_{true}(t))^2$$

Although we had calculated all the partial derivatives, we had opted to go with numerical gradient descent by numerically calculating derivatives. The reason for this decision was some

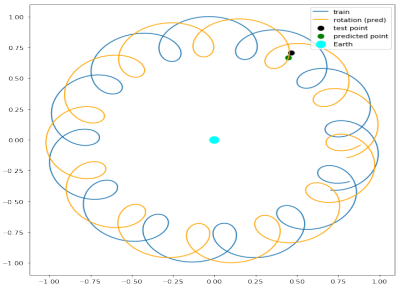
partial derivatives would result in overflow errors due to the numbers being on astronomical scales. Once we had the $x_{epi}(x)$ and $y_{epi}(t)$ fit parameters, we averaged the two to produce our final values for the 5 parameters.

3.2 Angle of Rotation via Gradient Descent (Rotation_GD.ipynb)

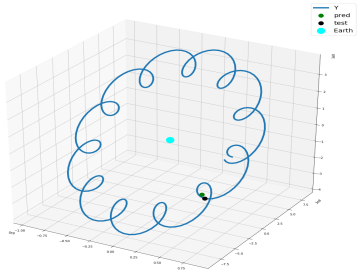
Once we turned this into a 2D problem, we saw that planets' orbits around the Earth were approximately cyclical so in the next orbit, it resembled the initial cycle rotated [4]. Given this, we devised an algorithm to learn the optimal angle of rotation between cycles using a bespoke implementation of Batch Gradient Descent. Specifically, we used exactly two cycles of a planets orbit and learned the rotation parameter θ between two consecutive cycles.

$$\hat{\theta} = argmin_\theta \frac{1}{n} \sum_{i=1}^{n} || \begin{bmatrix} x_i^{(2)} \\ y_i^{(2)} \end{bmatrix} - R(\theta) \begin{bmatrix} x_i^{(1)} \\ y_i^{(1)} \end{bmatrix} ||_2^2$$

We defined the above optimization problem where $R(\theta) = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix}$ is the standard 2D clockwise rotation matrix and applied it to the points in the first cycle (denoted by superscript). Furthermore, in order to keep the scale of the problem under control, we normalized the x, y coordinate vectors to have them be $\in [-1, 1]$.
Once the model is trained, we make predictions by computing how many cycles away a given test point is from our first train cycle and scaling theta by that amount. Simply put, we rotate the first train epicycloid to best match the epicycloid were the test point lies. Another consideration of our model included an "offset" term which accounts for a lag in data between the first and second cycle. This parameter was easily learned by observation and was a small workaround for data formatting. Although this method required some heuristics and extensive EDA to understand the cycles, it is a reminder that true ML problems aren't solved by simply plugging in models - there truly is some art behind the science of prediction.



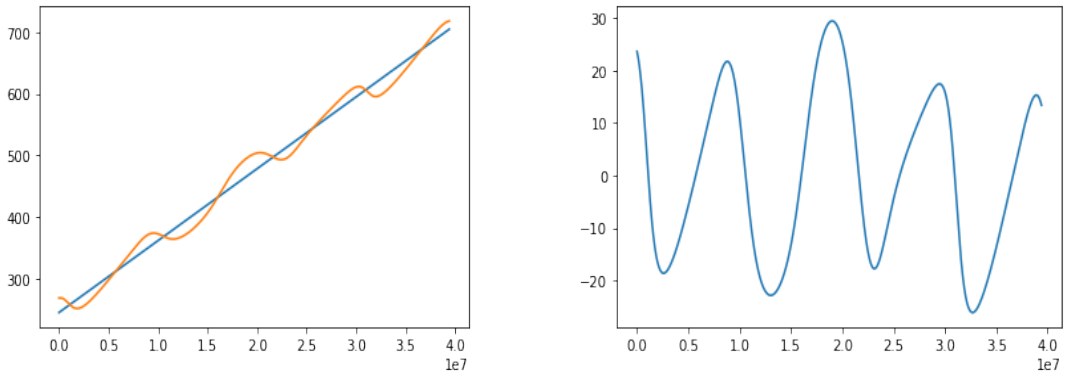
(a) Jupiter Prediction (2D)



(b) Jupiter Prediction (3D)

3.3 Transformed Fourier Features (RA/DECL_REG.ipynb and RA/DECL_NN.ipynb)

In addition to the fourier transform approach, we noticed that the data in Figure(a) can be made continuous if we add back 360 at each "break". This is because we are dealing with angular data and so, after a full 360 degree rotation, a modulo operator is effectively applied due to the nature of angles, 361°= 1°. Reversing this modulo operation results in an effectively continuous graph which has a clear linear, cyclical nature to it as seen in the figure below.



(a) (b)

Figure(a) shows Mercury's transformed degree of right ascension with a regression line, with degrees on the Y-axis plotted over time on the X-axis.
Figure(b) shows Mercury's transformed degree of right ascension after subtracting the fitted line shown in Figure(a), degrees on the Y-axis, time on the X-axis.

We see that if we subtract out the fitted line, we get what resembles a series of sine and cosine waves. In fact, this is exactly the abstract definition of epicycloids [5]:

$$\sum_{j=0}^{N} a_j e^{ik_j t}$$

We can immediately see a resemblance to Fourier series of the form:

$$\sum_{n=-\infty}^{\infty} c_n e^{inwt} = a_0/2 + \sum_{n=1}^{\infty} a_n \cos(nwt) + \sum_{n=1}^{\infty} b_n \sin(nwt)$$

Running with this idea, we transformed our input, time, into "fourier features" and used simple linear regression to find the best fit for declination and the transformed right ascension. In addition, we treated the frequency parameter, w, as a hyperparameter and tuned this via a static validation set with an 80-20 split (train-validation) of our training data. Our work and results can be found in the RA/DECL.ipynb jupyter notebook in our github repository.

3.4 Baseline Method: Fitting an Ellipse

The moon was known to have a unique orbit compared to the planets and the sun, in that it did not have as large of characteristic loops that the epicycloids produced. We modeled the moon as a planar ellipse, flattening input xyz data from our data generation into a 2D plane using PCA, which also gave our unit major and minor axes. We used linear regression to get the sizes of these axes. Finally, we used Fast Fourier Transform (FFT) to get the period of the moon's orbit. To predict, we used $(x, y) = (acos(t), bsin(t))$ to get the coordinates on the 2D plane, and then converted it to 3D coordinates using the major and minor axes.

4 Lunar Phase

The moon's phase is determined by the relative angles of the moon and sun relative to the Earth. If we model space in 3D with the Earth at the origin, different phases of the moon will have unique angles between the moon's vector and the sun's vector. We used dot products between unit vectors of the Moon and Sun to represent their angle. We used the unit direction vectors of the bodies rather than the true position vectors, as the value of the dot

product should not changed based on how far away the moon and sun are - they should only be affected by their relative angle. This preprocessing leads to the convention that the lunar cycle is an approximate sinusoid where a -1 value is a Full moon, a 1 is a New moon, and the phases from -1 to 1 are the waxing gibbous, 3rd quarter, and waxing crescent, and the phases from 1 to -1 are the waning crescent, 1st quarter, and waning gibbous (*see Moon Prediction.ipynb*).

We modeled this data with two different methods. The first we used was Kernel Ridge Regression with the Exponential Sine Squared Kernel: $k(x_i, x_j) = exp(-\frac{2sin^2(\pi d(x_i,xj)/p)}{l^2})$. This kernel allows ridge regression to learn periodic (but not necessarily sinusoidal) functions. The second we used was a sine function, with parameters derived from a FFT search.

5 Results

5.1 Planetary Prediction

5.1.1 Prolate Epicycloid Fitting (Epicycloid_Demo.ipynb)

Using this model, we were only able to successfully train on Mercury, Venus, Mars, and Jupiter. It was unable to model farther away planets due these having epicycle radii much smaller than it's distance from Earth. As a result, the gradient descent algorithm would always generalize to a circle by pushing R→0. The other shortcoming of this model was that the gradients had too many local minima thus you could only achieve a good model with relatively good first guesses. This model was not applied to the sun or the moon because Ptolemy did not think that they followed Epicycloids.

5.1.2 Rotation Gradient Descent (Rotation_Results.ipynb)

Contrary to our initial hypothesis, there was no clear advantage of the Rotation model in modeling the outer planets. We believed at first that the simpler shape of the outer planets would facilitate the task of finding suitable local minima for the theta parameter. This wasn't necessarily the case since the best prediction was for Venus. Digging a little deeper, some planets like Saturn and Mercury had a hard time converging to the true theta parameter and would minimize the squared loss at objectively wrong values of theta. We believe this is a result of the complexity of the epicycloids. As mentioned previously, our model was trained on the 2D epicycloids so predictions were made by reversing the rotation we initially made to two dimensions. Therefore, all predictions are in the 3D space.

5.1.3 Transformed Fourier Feature Regression (RA/DECL_REG.ipynb)

We see that our models do a fairly good job on our test data with a few flaws presenting themselves. Visually, most of the planets orbits are captured seemingly well. In fact, 7 out of the 9 models had RMSE's lower then 15 degrees for both right ascension and declination. This implies that most of our models predicted, on average, within 15 degrees of the truth for both ascension and declination except for the 2 outlying cases. However, we must be careful as trying to predict farther into the future could cause the models to become further misaligned and compound the error due to the almost-cyclical nature of these planetary orbits.

Ultimately, we must realize that our data are governed by potentially almost-periodic functions of the form $\sum_{x=0}^{N} a_j e^{ik_j t}$, where the k_j's may be non-rationally related. In contrast, fourier series assume that the exponential terms are rational multiples of each other. As a result, we are trying to model potentially almost-periodic functions with a periodic function

which, of course, will lead to some issues in our prediction.

We see two cases where this model does especially bad, the Sun and Uranus, where the MSE's for declination are significantly larger. Comparing this to our neural network model, we can deduce that a majority of this error likely comes from our use of linear regression which was not able to handle the non-linearities for these two planets quite as well.

5.1.4 Transformed Fourier Feature with Neural Networks (RA/DECL_NN.ipynb)

Building off of the earlier section which explored transformed fourier feature regression, we replaced linear regression with a 2 layer neural network with 5 nodes in each using relu activation. We see that our neural network model performed significantly better on Mars, Uranus and the Sun, while the rest of the planets had similar performance compared to the regression.

The better performance for these bodies was likely a result of the networks ability to handle non-linearity's in the data with more ease compared to the regression which required significant tweaking to garner good performance. The similarities of the other planets was expected since we did not actually change the nature of our feature engineering process and so the assumptions of periodicity which plagued our regression model remain an issue here as well.

5.1.5 Baseline Ellipse (Moon Predictor.ipynb)

Although the moon's orbit is quite elliptical, there are small regular perturbations throughout each cycle, and each lunar cycle deviates a little as shown in the python notebook. Ptolemy's models capture this deviation with a moving deferent. The model had a RMSE of 4.941e+05. Adding a deferent would likely strengthen this model, but it would still not be able to capture the small perturbations around the ellipse.

Final RMSE (in km)				
Planet	Rotation Grad. Descent	Fourier Regression	Fourier NN	Epicycloid Fit
Sun	–	1.735e+08	1.818e+07	–
Moon	–	4.346e+04	4.323e+04	–
Mercury	1.482e+08	2.368e+07	2.395e+07	7.506e+7
Venus	1.489e+07	3.683e+07	3.695e+07	1.224e+8
Mars	4.818e+08	5.183e+07	4.404e+07	1.219e+8
Jupiter	4.115e+07	1.356e+08	1.298e+08	3.080e+8
Saturn	3.024e+08	1.7e+08	1.403e+08	–
Uranus	3.829e+07	1.042e+09	2.622e+08	–
Neptune	1.504e+08	1.191e+08	2.325e+08	–

5.2 Lunar Phase Prediction

For the exploration of these two models, two full years of moon and sun position data were generated. Training and testing both the ridge regression model and the sine function on this data returned almost identical results with small root mean squared errors. This confirmed that not only is the lunar cycle periodic, but the dot product of the positions is essentially sinusoidal. Thus, using the principle of Occam's Razor, we chose to use the sinusoidal model as a predictor, since it is simpler.

As a predictor, the model was trained on 80% of the data of the dot product between the normalized vectors of the moon and sun, and then tested on the latter 20%. It produced a

small error of 0.00562 RMSE. While this is a good result, it is also testing results near in time to the original training data. Testing on classification (rather than the calculated phase angle) done with individual dates in different years also worked well, but future testing should be done with generated data from times farther out, to generate more specific thresholds for the specific phases (such as Waxing Crescent 1 vs Waxing Crescent 2). Additionally, the lunar cycle fluctuates over the year, so having a fixed lunar cycle in the model will not work for fine predictions [6]. Without the fine tuning of the cycle, predicting eclipses is nearly impossible, as eclipses occur when the Earth, Sun, and moon are collinear with minuscule variation, and therefore the accuracy of the predicted dot product must be incredibly large. The combination of more precise classification thresholds and compensating for this fluctuating lunar cycle will make the model more robust in handling farther times and more specific accuracy (as in the case of eclipses).

6 Conclusion

6.1 Final Thoughts

At the conclusion of this project, our team produced 5 different prediction models. For planetary orbit prediction we tried linear regression with Fourier features, Fourier Neural Networks, and two epicyloid based models. For moon phase prediction, we used a kernel ridge regression model as well as a simple sinusoidal model. From our data, we see that Fourier features are generally more versatile in its ability to train and predict all bodies. For the moon phases, we see that the simple sinusoidal model is adequate to model the lunar cycle, as it is almost perfectly sinusoidal.

Going back to Ptolemy, he theorized that all bodies orbited around the earth with epicycles. Ptolemy created his theory based off of Greek and Babylonian data. We wanted to see how well it would hold up with modern data and modern computing power. After applying this model and examining the results, we can conclude that epicycloids are primarily only good for modeling the general shape of an orbit. This conclusion is in line with reality since we know that celestial bodies actually follow elliptical orbits around the sun.

6.2 Areas for Improvement

Looking forward our team has identified several areas which we could further develop. In particular we would like to take into account the equant within our model. This addition is to account for the fact that the geocentric model has planetary rotations of non-constant speed, making modeling difficult, while the equant allows for the modeling of uniform motion during planetary cycle.

Additionally, another good avenue to explore for improved performance is the novel architecture first proposed in 2016 called "Equation Learner" (EQL) which brings a modified neural network approach to analytically learning equations that describe physical systems [7]. LSTM's and other implementations of Recurrent Neural Networks might also be promising due to the nature of the data we are dealing with here. It's been shown that LSTM's have a some nice properties when predicting cyclical functions such as the trigonemetric ones we are dealing with in this case [8].

References

[1] R. Fitzpatrick, "Ptolemy's model of the solar system." http://farside.ph.utexas.edu/Books/Syntaxis/Almagest/node3.html, 2010.

[2] A. Myers, "Coordinate transform." http://faraday.uwyo.edu/~admyers/ASTR5160/handouts/51605.pdf, 2017.

[3] X. Wang, "Fitting of particle trajectory in magnetic dipole equatorial field and prolate epicycloid," February 2020.

[4] H. Breitenbach, "Curves of planetary motion in geocentric perspective: Epitrochoids." http://gerdbreitenbach.de/planet/planet.html#:~:text=Venus, 2018.

[5] N. R. Hanson, "The mathematical power of epicyclical astronomy," *Chicago Journals History of Science Society*, vol. 51, p. 157, 6 1960.

[6] D. I. Bromberg, "The length of the lunar cycle." http://individual.utoronto.ca/kalendis/lunar/, 2018.

[7] G. Martius and C. H. Lampert, "Extrapolation and learning equations." https://arxiv.org/pdf/1610.02995.pdf, 2016.

[8] K. Balka, "Training keras lstm to generate sine function." https://medium.com/@krzysztofbalka/training-keras-lstm-to-generate-sine-function-2e3c0ca42c3b, 2019.