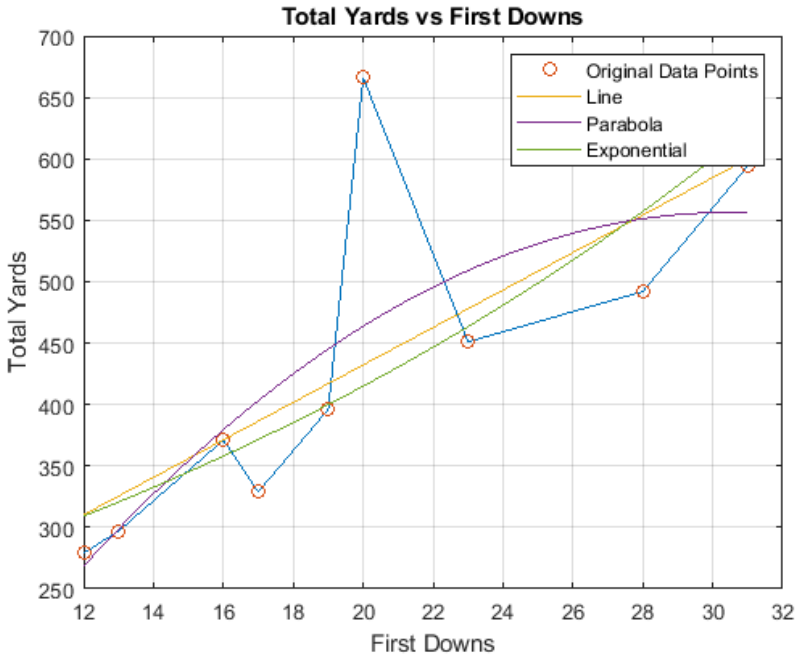
Introduction To Numerical Methods of Engineering Analysis
University of Florida
Mechanical and Aerospace Engineering

HW7 Solution, Angel Lopez Pol

Problem 1 :

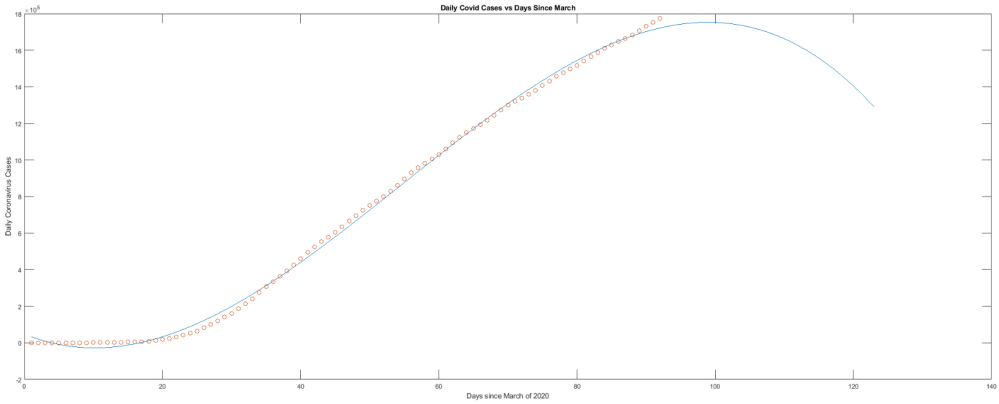
The code that I used to perform simple linear regression can be found on the hw7_p1.m file. I also had a bunch of extra functions that I made while figuring out how to perform simple linear regression, and I think I might be using some of those functions for the later problems.



I believe the line of best fit to be the exponential curve. At first my code only considered r^2 values, and the exponential curve had the highest r^2 out of the 3 but the difference was not massive. On the other hand, after adding standard error as an additional metric, I can see that the exponential function has a much smaller one. I did note that standard err. is a metric based on the y-values of the function, and by linearizing our exponential our y values change, but even accounting for this change the standard error is still smaller.

Problem 2 :

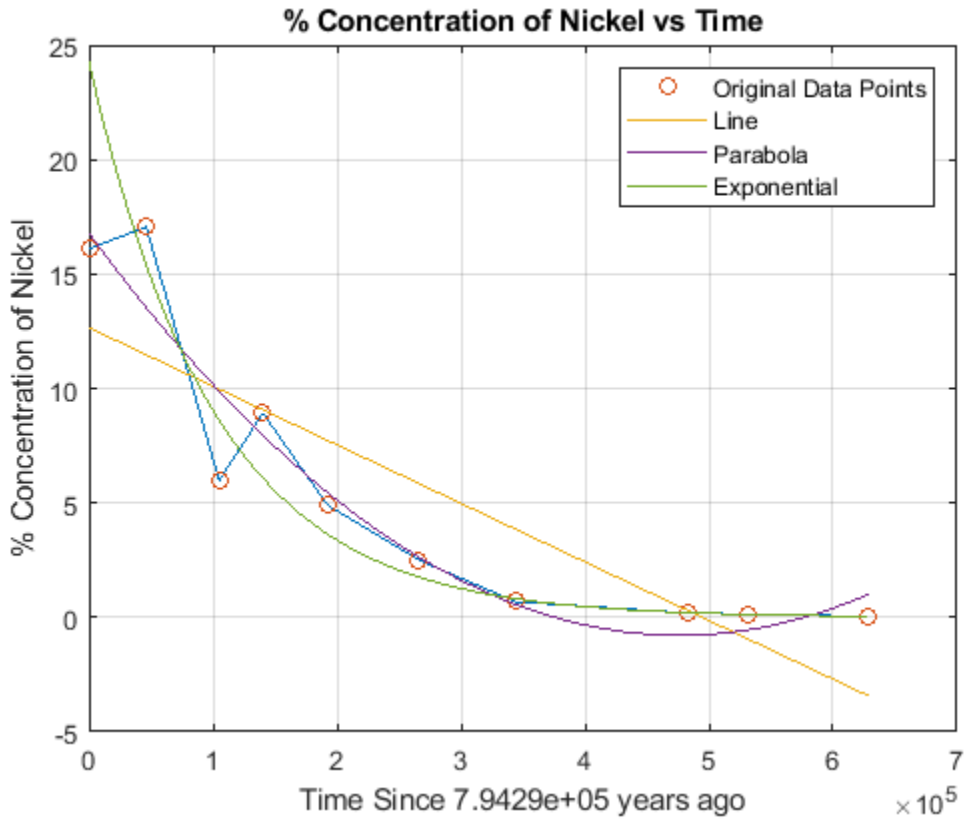
The code I used to analyze the Coronavirus data can be found in hw7_p2.m. The following is the graph that was generated from the code:



Based on my limited knowledge of epidemiology, I think that this cubic model might not be as reliable since there are multiple independent variables affecting the daily amount of covid cases. The curve of best-fit might be better modeled using multivariable general linear regression- assuming we know the independent variables. Otherwise, this model is good for conveying trends over the short term, but it loses credibility in the long term.

Problem 3:

The code I used to analyze the lunar soil samples can be found in hw7_p3.m. Here's a screenshot of the best-fit curves I obtained by using linear sum of least squares:



The Linear curve of best-fit is easy to implement, however it has a high standard error and low r^2 value. Although it can convey general trend, we can't use it to guess where additional data points might land with reasonable accuracy

On the other hand, the parabola models the system much better, but our matrix is close to singular/ill-conditioned, resulting in our QR Factorization yielding potentially inaccurate results if we were to try and calculate the best nth-degree polynomial of best fit (I experienced this first-hand when I first tried problem 5 using polynomial linear regression, before switching over to using cos since it modeled the system better). For the parabola this error is essentially negligible.

The power model does the best job if we use r^2 as our metric. However, we do have to make an "apples-to-apples" comparison. What I ended up doing is that I pushed the starting values of our data forward prior to the linearization process so that they were greater than 0. I have to admit though, I think that having your x values close to 0 also seems to skew the graph in certain directions, so I went ahead and made sure the distance from 0 was pretty sizable. I would say this is the big con with the power model.

Problem 4:

The code for this problem can be found in hw7_p4.m. My results for a_bar were as follows

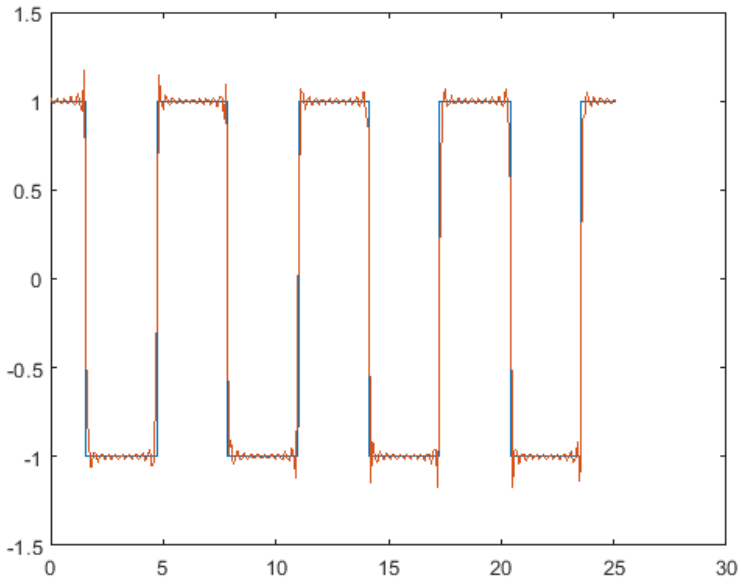
```
The coefficients obtained for our best-fit curve for the cookies can be found below:
    4.2035
    0.0255
    0.0144
   -0.0414
    0.0807
    0.0198
```

Where the first value corresponds to our constant, and the rest correspond to ingredients 1-5. Although I'm not sure if I calculated my values correctly, I do understand conceptually that these coefficients are tied to the amount of enjoyment that each ingredient contributes to the cookie recipe! For instance, as a baker I would want to add more of ingredient 4 based on these coefficients, because ingredient 4 has a much higher weight/tastiness factor than the rest of the ingredients. Additionally, I would want to add less of ingredient 3, as it seems that, historically, its presence in the recipe takes away from the enjoyment experience. Additionally, I interpret the constant as being the base level of enjoyment that an ingredient-less cookie would give a person- meaning that I see these 5 ingredients as "toppings" like chocolate, marshmallow, macadamia, etc. while a cookie with none of these ingredients is just a basic cookie.

Problem 5:

For problem 5, I approximated the curve of best fit using general linear regression with cosines (since the original data looks like a square version of cosine). Also I'm not super familiar with Fourier Series, but I believe my code exploits the same principle regarding how certain periodic patterns can be expressed as summations of sinusoidal functions with varying frequencies. I

think I learned about it on a Veritasium video about Fast Fourier Transforms. It's a really good video! The blue curve is the original function, while the orange curve is the 41st iteration of my cosine algorithm, including 41 cosine basis terms with frequencies 1-thru-n. My 41 coefficients can be found on the matlab code inside the a_bar_true variable, but for grading purposes, the first few can be found below




```
>> disp(a_bar_true)

    1.2732
   -0.0010
   -0.4244
    0.0010
    0.2546
   -0.0010
   -0.1819
    0.0010
    0.1415
   -0.0010
```

Theoretically, my code could keep adding cosines infinitely, but I chose to stop it at an r^2 value above 0.99, and a standard error value of 0.1. The graph outputted by my code can be found above