IQP Report: Predict the price of a stock

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

Karl Brzoska

Submitted to Professor Mayer Humi, Project Advisor

Abstract:

This IQP looked at different methods to predict stock prices. 20 stocks were chosen from a sector in the stock market to make short term predictions. The first model was a simple linear model, which did not prove to have a good prediction rate. After adding a fourier series, the prediction accuracy increased. These predictions stayed within the error bounds with an average of 0.6 days. Error bounds were created by taking the average deviation between the actual stock price and the predicted stock price from model one.. With market influence, the prediction accuracy increased even further with predictions staying within the error bounds with an average of 3 days. The final method used was a Savitsky Golay method, which was inconsistent.

Table of Contents

Executive Summary:

The purpose of this report is to determine if we can make short term predictions for the price of a stock. The first method we used is a simple line of best fit. By coming up with an equation in the form y=mx+b, we were able to determine a basic trend of the stock price. This provided us with a trendline to determine the direction in which the stock prices were going, but other than that it was not very effective in predicting the prices. This model does not take into account the deviations between the trend line and actual stock prices. In order to address this problem, we performed a fourier series on the residuals of the regression line and actual stock prices. A graph of the residuals shows us that it somewhat resembles a periodic graph. The fourier series is a way to approximate a periodic graph by giving it a function. By modeling the residuals, we hoped that we could determine the behavior of the deviations of the regression line and actual prices. By combining the fourier series and linear model, our predictions became more accurate than with just a linear model. However, because of the volatility of the stock market, our predictions are still not great. The main issue with this model is that it is not taking into account the volatility or state of the stock market. As mentioned earlier, the Dow Jones Industrial Average is a measure of how the top 30 companies in the stock market are doing. So if we can somehow incorporate this and Treat the Dow Jones Industrial Average as a stock, can we improve these predictions? We found the answer to be true for our stocks. The predictions that took into account the state of the market were more accurate than the predictions that did not.

1. Introduction

Background

When we think of money and wealth, what is one thing that usually comes up? We usually think of those wall street brokers in fancy suits that try to get people on board to buy a stock. Or we think of those analysts with their multiple math phds that use everything from probability to linear algebra to predict the stock market. But for those of us without math phds, can we predict stock prices? Stock prices are volatile. And with the effects of covid and inflation on the stock market, prices are even more volatile. How can we create basic mathematical models to mimic the stock market and use these to make short term predictions into the future?

The answer to how we can make predictions lies in algebra and differential equations. In grade school we learned about the formula y=mx+b. How can we use this formula to help make predictions? In high school we learned about sine and cosine waves. How can we use these topics to make predictions about the stock market? And can we somehow combine these two topics to create an even better model? Can we take into account the state of the market? During covid, the role of health news as played a role forming better predictions of the stock market.[1]

It is hard to detail the previous work that has been accomplished in this field. The quantitative analysts, or quants, use math and statistics beyond the scope of this project that most people like myself don't understand. So instead, we will be using more basic math concepts to form our models and make our predictions.

The purpose of this project is to come up with different methods to predict stock prices and test them. The first method we will use is the basic linear model. By fitting the data to the formula, y=mx+b, we should be able to find a trend of the stock prices. By finding the trend, we will have the tendency of which direction the stock price is going. The next method we will use is the fourier series, which will be explained later. The gist of the fourier series is that we will be able to incorporate the difference between our predicted prices and the actual stock prices. The next method that will be used is the Savitsky-Golay method. Can we increase the precision of our predictions by changing the exponent on our linear model? And what about market influence? The Dow Jones Industrial Average is a stock market index of 30 prominent companies listed in stock exchanges here in the USA. Can we somehow incorporate the trend of the Dow Jones Industrial Average into our precision models for a single stock? These are all questions that we will attempt to answer throughout this project.

These models are implemented using python and matlab. Because of simplicity and a limit in a specific area of python, matlab takes over as the main program when incorporating the fourier series. The hope is that we can create a model that will be able to predict stock prices in the short term.

1.1 Goal

The stock market can be an unpredictable place. No one knows whether a stock is going up, down, or in circles. Making long term predictions in the stock market is difficult because of its volatile nature. How can we determine the price of a stock one month from now? One year from now? How about one day though? Maybe two days? While we can’t be one hundred percent sure that our stock price prediction for tomorrow will be correct, we can surely be more confident than if we were to predict the price a month from now. The goal of this IQP is to use statistics, python programming, and matlab to formulate mathematical models and apply them to real time stock data to create short term predictions. The stock market is a volatile place. And right now in 2022, the stock market is even more volatile due to the effects of covid and politics. I chose this project because I’ve always been interested in the math that goes behind these predictions. While the math used here is nowhere near the level of professional quantitative analysts, it’s a good introduction to basic math we can use to make short term predictions. At the end of this project, we hope our short term predictions can come close to the actual price of the stocks.

2. Stock Selection

2.1 Researching Stocks

Before creating any models, we must pick our stocks to analyze. I picked twenty different stocks from the same field using Yahoo Finance. Because of the rise of technology and software in recent years, I picked stocks from the software application sector. This sector contains companies that develop, distribute, and maintain software for commercial or public use. I tried to keep most of the stock prices between 20 USD and 100 USD for consistency. By keeping the price on the lower end, the risk of losing a lot of money is lower. In addition smaller investors don't have much capital to start with.

2.2 Filtering

In order to maximize my predictions, I went through the process of filtering out five of my twenty stocks that had the quality of having the highest standard deviation. This was done using the yahoo_fin package in python. Using python, I created a program that takes in user input. The user input takes in a stock ticker symbol as well as the time period in which the stock data will be extracted. By using matplotlib, I was able to create a visualization of the stock price and other characteristics. For this project, I wanted stock data from the past two years. My program then calculated the mean and standard deviation of the daily stock prices of the past two years. I then filtered out five of my twenty stocks that had the greatest standard deviations.

2.3 Stock Selection

I chose these stocks based on price, what they do, and what sector they are in. I picked stocks from the software application sector. All of the statistics, such as mean and standard deviation were taken from the past two years. So from 5-18-2020 to 5-18-2022. 5-19-2022 was normalized as day one of our predictions.

Agilysys(AGYS)-Mean is $40.85 Standard deviation is $12.43
Agilysys Inc. develops proprietary enterprise software and other products for the hospitality industry. Agilysys Inc. specializes in point of sale, property management,

inventory and procurement, document management, workforce management, and mobile and wireless products.

Alarm.com(ALRM)-Mean is $75.86 Standard deviation is $13.35
American technology company that provides cloud based services for remote control, home automation and monitoring services.Monitoring services can include contracts through third-party contractors such as ADT. Services include interactive security, video monitoring, energy management and home automation, and are enabled through an ecosystem of integrated devices and hardware partnerships.

Alteryx Inc(AYX)- Mean is $95.59 Standard deviation is $32.69. Not used

Applovin Corp(APP)-Mean is $70.60 Standard deviation is $17.16
AppLovin enables developers of all sizes to market, monetize, analyze and publish their apps through its mobile advertising, marketing, and analytics platforms MAX, AppDiscovery, and SparkLabs. AppLovin operates Lion Studios, which works with game developers to promote and publish their mobile games. AppLovin also has large investments in various mobile game publishers. I chose this stock because what they do is interesting.

Blackblaud Inc(BLKB)-Mean is $65.80 Standard deviation is $8.24
Blackbaud Inc is a cloud computing provider that serves the social good community—nonprofits, foundations, corporations, education institutions, healthcare

organizations, religious organizations, and individual change agents. Its products focus on fundraising, website management, CRM, analytics, financial management, ticketing, and education administration. I chose this stoke because cloud computing is a pretty hot field now.

Braze Inc(BRZE)-Mean is $53.44 Standard deviation is $14.99
It is a customer engagement platform used by businesses for multichannel marketing. Multichannel marketing is the blending of different distribution and promotional channels for the purpose of marketing. Distribution channels include a retail storefront, a website, or a mail-order catalogue.
Braze provides customer engagement technology for companies such as HBO Max PureGym,Burger King, Babylon Health, Grubhub, NASCARvand CleanChoice Energy

Cerence inc(CRNC)-Mean is $76.39 Standard deviation is $30.23. Not used

Coinbase global inc(COIN)-Mean is $236.44 Standard deviation is $55.71
Coinbase Global Inc operates a cryptocurrency exchange platform. I chose this stock because it’s the medium in which many of these cryptocurrencies are sold and bought.

Domo INC(DOMO)-Mean is $56.21 Standard deviation is $19.83
Domo specializes in business intelligence tools and data visualization.

Gitlab inc(GTLB)-Mean is $73.84 Standard deviation is $25.87. Not used

Guidewire software inc(GWRE)-Mean is $109.16 Standard deviation is $11.46
Guidewire software inc offers an industry platform for property and casualty (P&C) insurance carriers in the U.S. and worldwide. Guidewire Software, Inc. is a provider of software products for property and casualty (P&C) insurers. The Company provides a platform for P&C insurance carriers.

CS disco inc(LAW)-Mean is $40.59 Standard deviation is $9.57
CS disco inc is the leading provider of software such as a service solutions developed by lawyers for lawyers,.DISCO is develops legal technology to automate and simplify complex and error-prone tasks that distract from practicing law. I chose this stock because what the company does is interesting.

Liveperson inc(LPSN)-Mean is $49.04 Standard deviation is $14.10
LivePerson is a global technology company that develops conversational commerce and AI software. For example if you are looking for a pair of shoes, you will type in you are looking for shoes and it will direct you to the shoe section on the website. I chose this stock because I have used this before and I’m interested in AI.

ncino(NCNO)-Mean is $65.12 Standard deviation is $13.33

Ncino builds financial software. Its cloud-based banking software is built on the Salesforce platform for financial institutions to streamline commercial and retail banking needs.

open text corporation(OTEX)-Mean is $46.07 Standard deviation is $3.90
OpenText software applications manage content or unstructured data for large companies, government agencies, and professional service firms. OpenText aims its products at addressing information management requirements, including management of large volumes of content, compliance with regulatory requirements, and mobile and online experience management

pegasystems inc(PEGA)-Mean is $115.67 Standard deviation is $19.48
Pegasystems develops software for customer relationship management and business process management.The company's key offering is the Pega Infinity platform, which combines business process automation with customer engagement applications.

procore technologies(PCOR)-Mean is $79.42 Standard deviation is $15.97
Procore's cloud-based construction management software allows teams of construction companies, property owners, project managers, contractors, and partners to collaborate on construction projects and share access to documents, planning systems and data, using an Internet-connected device. I chose this stock for the same reason as CS disco. It is software created for a very select group of people.

Smartsheet inc(SMAR)-Mean is $61.34 Standard deviation is $10.27

Smartsheet is a software as a service (SaaS) offering for collaboration and work management, developed and marketed by Smartsheet Inc. It is used to assign tasks, track project progress, manage calendars, share documents, and manage other work, using a tabular user interface.

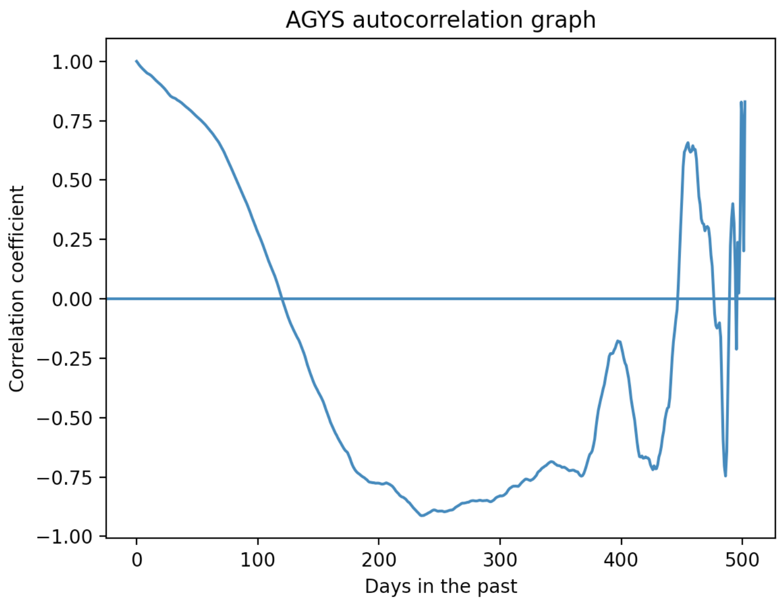
Sprout social(SPT)-Mean is $68.33 Standard deviation is $30.38. Not used

Unity software inc(U)-Mean is $116.14Standard deviation is $27.26. Not used

3. First Model

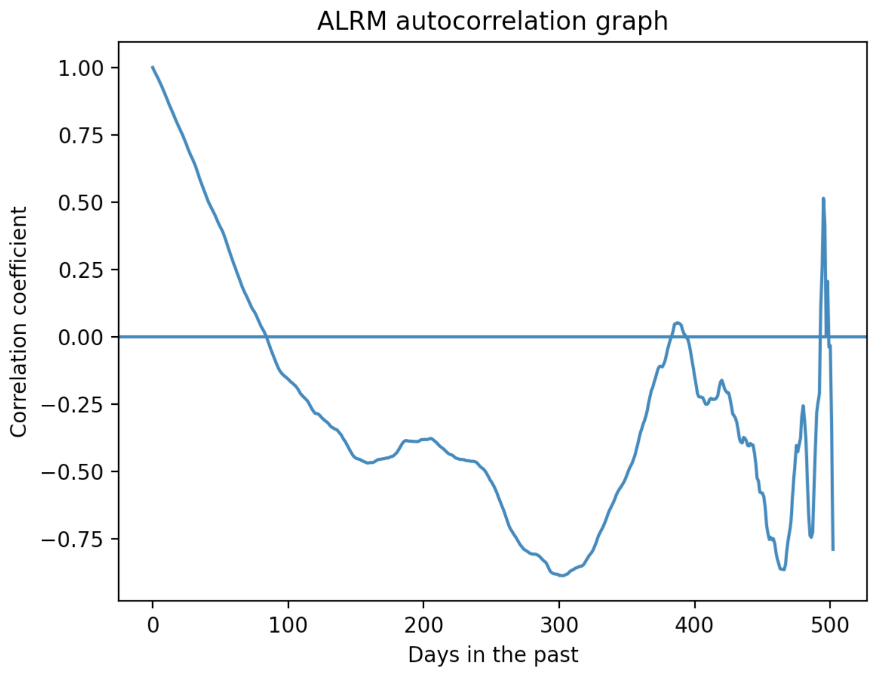
3.1 Autocorrelation

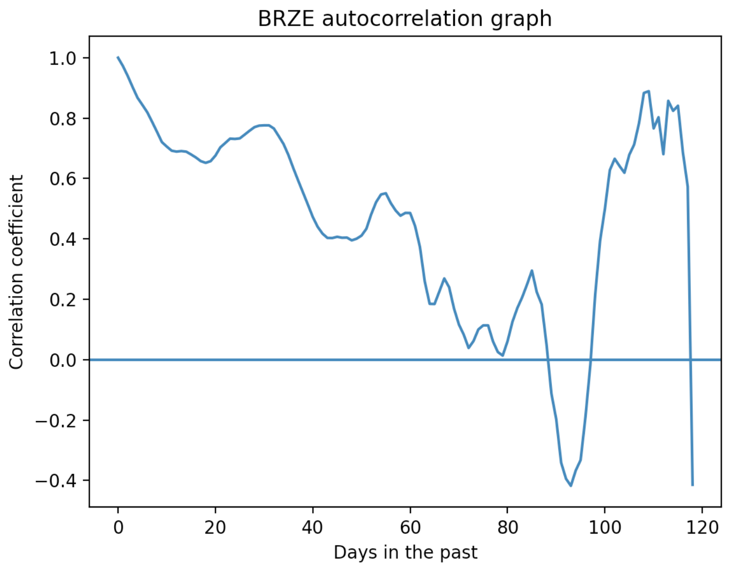
Autocorrelation represents the degree of similarity between a given time series and a delayed version of itself over consecutive time intervals. Autocorrelation measures the relationship between a variable's current value and its past values. This degree of similarity is defined by the correlation coefficient, which is the numerical measure of how correlated two variables are. However, since we measure autocorrelation, we are finding the correlation between a stock's price and that stocks' price earlier in time. So if we wanted to grab the autocorrelations, we would need to find the correlation coefficient between each stock price and the price the day before. Then we would do the same for every stock price and the price for two days before. And then three days before and etc.I created a python program to calculate the autocorrelations of a stock and graph it. I did this by using a built- in correlation coefficient function in python to calculate the correlations between autocorrelation, and its stock price the day before, and then two days before etc.



The closer the correlation coefficient is to one, the stronger the correlation between a stock and its previous prices are. If the correlation coefficient is zero, we can say the current stock price x days in the past has no influence on the current stock price. This represents the relevant period. The relevant period represents how many days in the past we can use a stock's previous price to predict its future price. For example, in the figure above, the correlation coefficient reaches zero 124 days in the past. This means we can only use the past 124 daily stock prices to predict the future price.

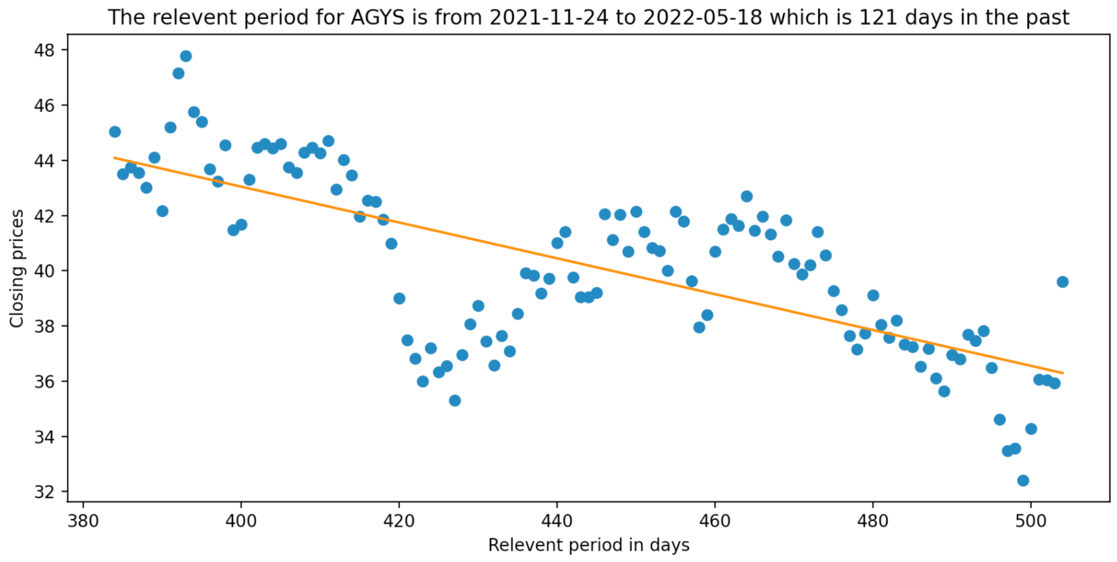
Here are the autocorrelation and relevant period graphs for a few more stocks.



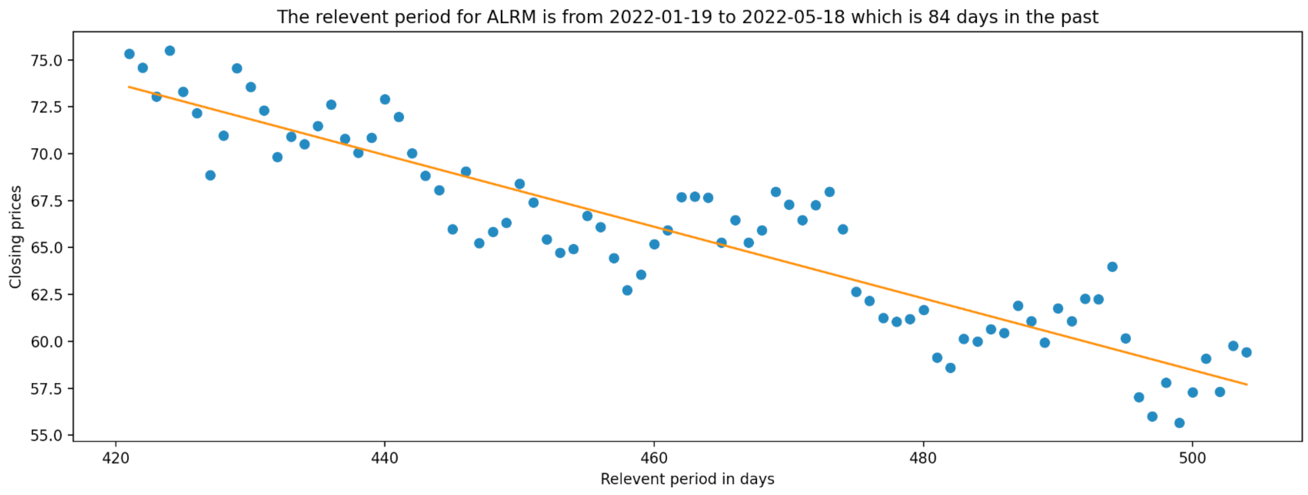


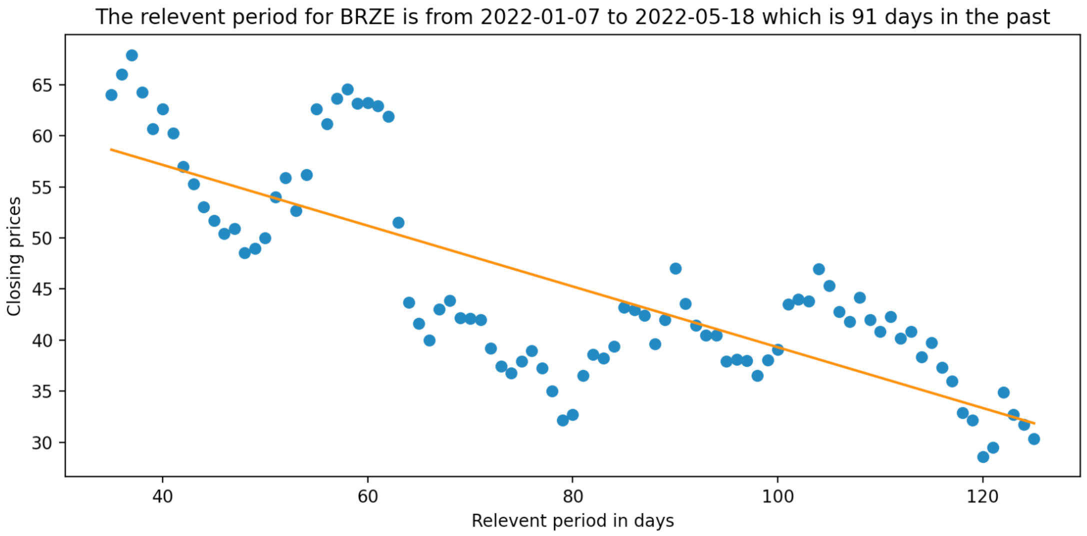
3.2 Trend

Now that we have calculated how many days in the past the stock price is relevant, we now know the relevant period. The relevant period is the period in which the autocorrelations are greater than 0, meaning that there is a correlation between today's price and the n-1 days. This is the period we deem relevant as we are going to



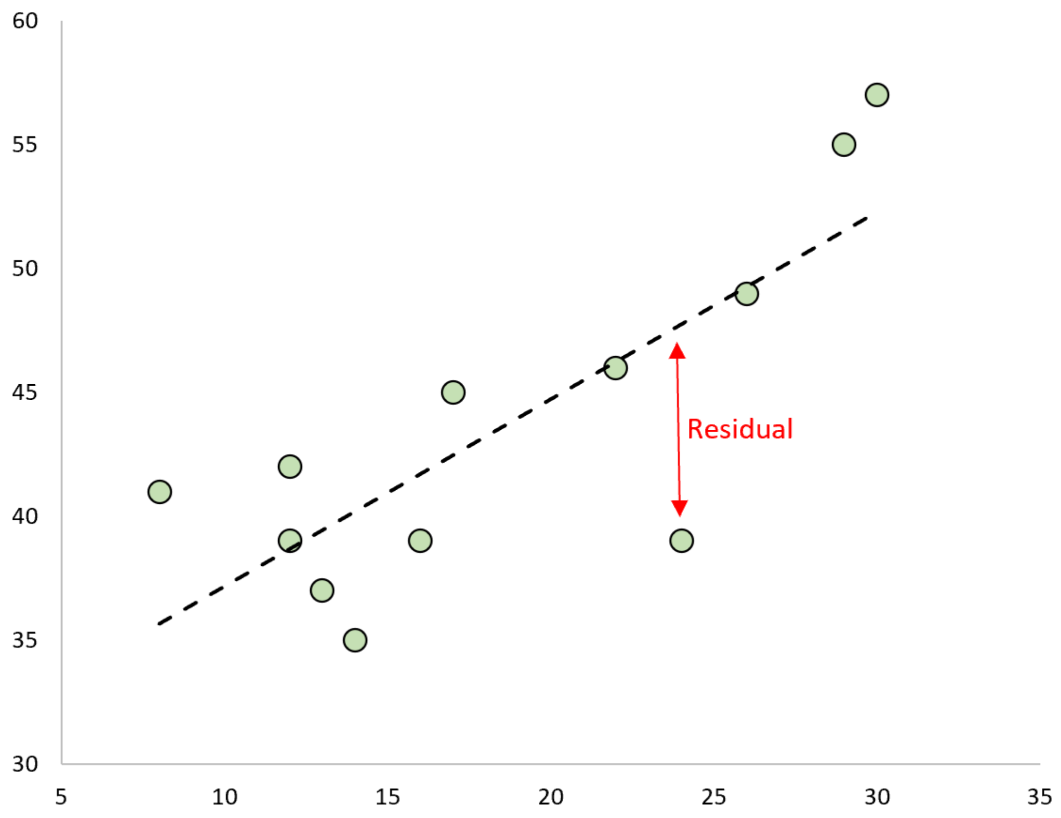
use it to predict our future stock prices. For example, the relevant period is 121 days in the past to today. We will graph that period and then draw a line of best fit, which is a line through a scatter plot of data points that best expresses the relationship between those points. The line of best fit was created using the polyfit function from the numpy library.

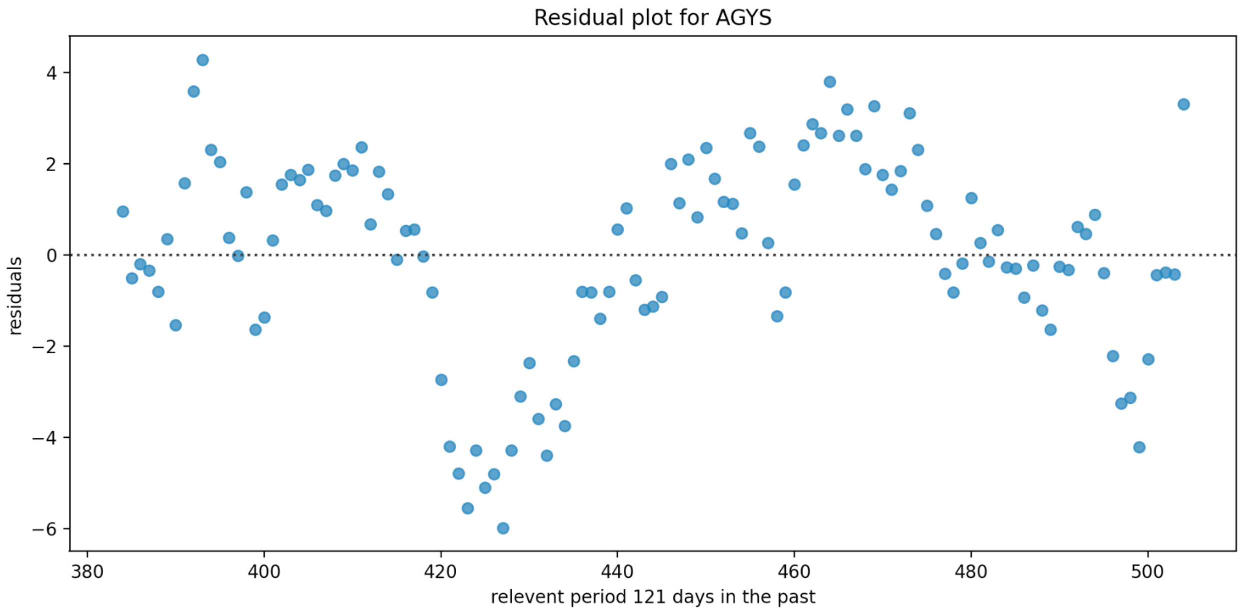

The relevent period for ALRM is from 2022-01-19 to 2022-05-18 which is 84 days in the past
Closing prices
Relevent period in days
75.0
72.5
70.0
67.5
65.0
62.5
60.0
57.5
55.0
420
440
460
480
500

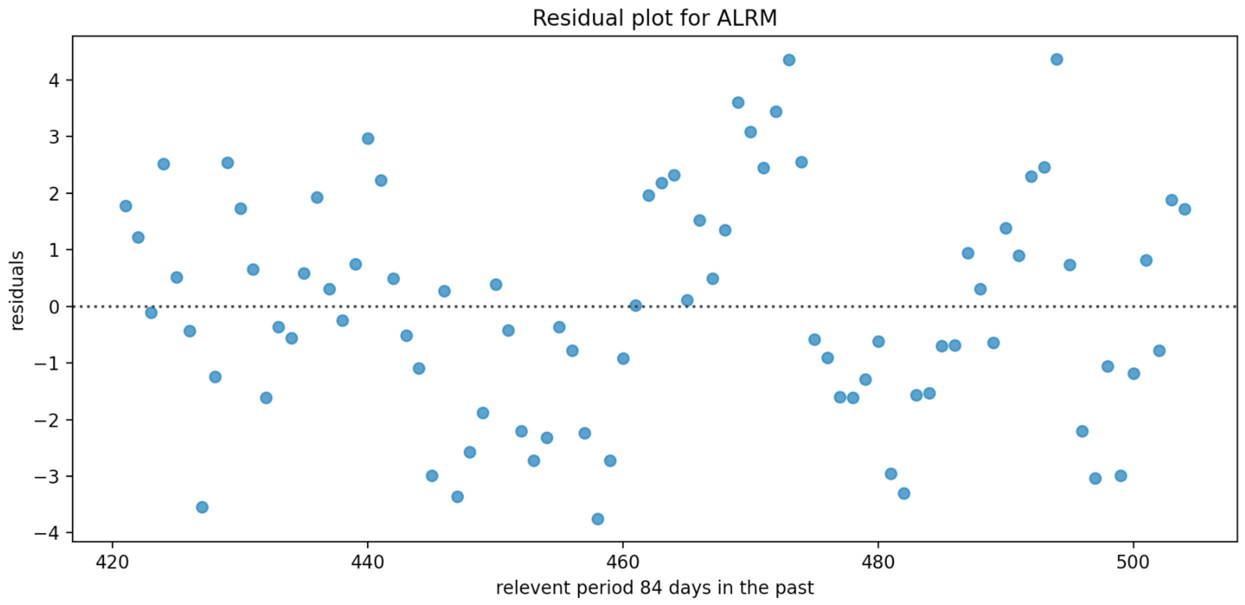

The relevent period for BRZE is from 2022-01-07 to 2022-05-18 which is 91 days in the past
Closing prices
Relevent period in days
65
60
55
50
45
40
35
30
40
60
80
100
120

3.3 Deviations from the trend

A residual is a measure of how well a line fits an individual data point. This vertical difference between point and line is known as a residual. The formula for residual is actual(actual data point)-predicted(point on line of best fit). For data points above the line, the residual is positive. For data points below the line, the residual is negative. Because there are many points above and below the line of best fit, the residuals will be negative and positive. If we think about this, the graph of the residuals will be somewhat resembling a periodic function, as seen below. This was done with a simple loop. I looped through the semi periodic relevant closing prices(from the past x days, in AGYSs' case it was 121) and subtracted the point on the line from the closing price.




Residual plot for AGYS
residuals
relevent period 121 days in the past
4
2
0
−2
−4
−6
380
400
420
440
460
480
500


Residual plot for ALRM
residuals
relevent period 84 days in the past
4
3
2
1
0
−1
−2
−3
−4
420
440
460
480
500

As we can see here the residual plot for ALARM looks a lot less periodic than the graph for AGYS, however we can still see that it looks like a wave function.

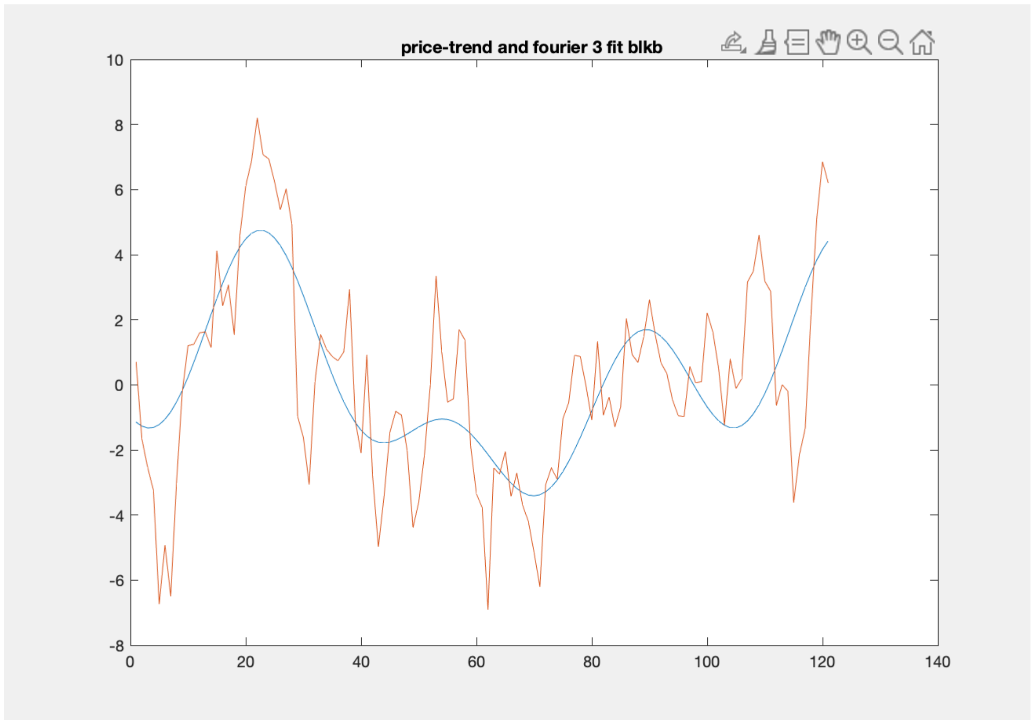
3.4 Fourier Series

Almost any kind of a wave can be written as a sum of sines and cosines. So for example, if I were to record my voice for one second saying anything, I can find its Fourier series which may look something along the lines of this.

Voice=$\sin(x)+\frac{1}{2}\sin(x)+\frac{3}{4}\cos(2x)$.....

A summation of sines and cosines can be used to approximate the wave created from the sounds waves my voice creates. In the example, there are three terms which will most likely result in a rough approximation of the original graph of my voice. The more sines and/or cosines added to the equation, the closer the equation mimics the original graph. A Fourier series is a sum that represents a periodic function as a sum of sine and cosine waves. Now, looking at the graphs of the differences between the linear fit of the closing prices and actual closing prices from the previous section, we can see that a slight wave/periodic looking graph arises. With this in mind, would we be able to approximate this wave looking function with sines and cosines? And why would we want to find this Fourier series in the first place? By Finding the fourier series and incorporating them with the trend line, we are able to improve the predictions provided by the trend line.

As one can see, this does not exactly look like a typical sine/cosine graph. If you look closely, this graph looks very similar to the residual scatter plot for the AGYS stock from section 3.3, except that the dots are connected. Because 121 fourier components were used, which is the length of the relevant period(121 days), the fourier series graph looks very similar if not identical to the original scatter plot from section 3.3. By finding the fourier series with 121 terms, we have found an equation containing many sines and cosines that calculates the difference between the predicted and actual closing prices from our linear model. If we were to take the end of this graph and extend it, we could theoretically be able to predict how different the actual closing price, say 2 days from today, differs from the predicted closing price from our linear model. Of course we would also have to extend our linear model as well. Here are a couple more fourier series graphs.



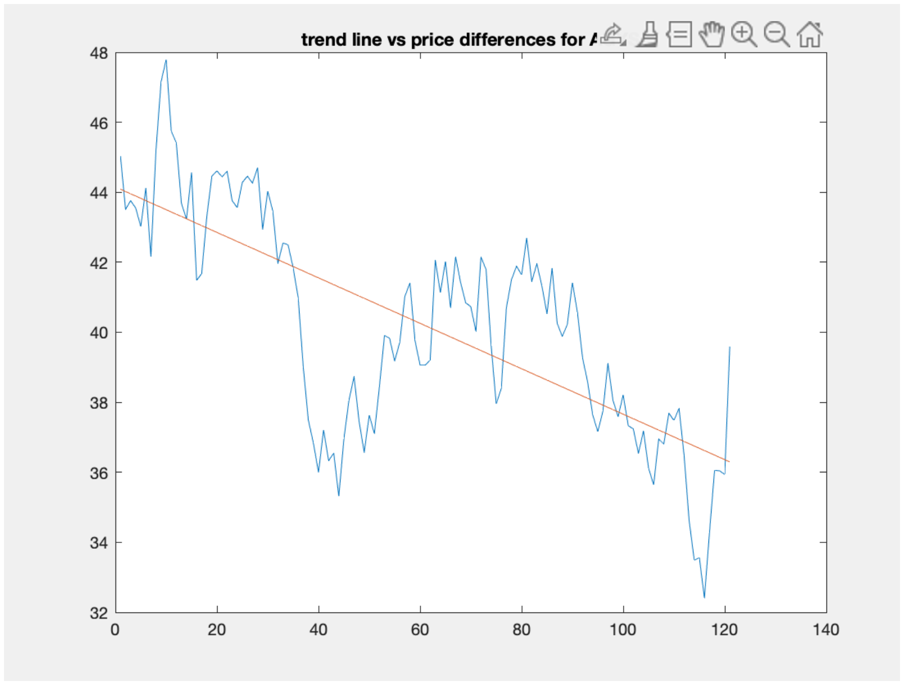
Again, because we are using 84 components for our fourier series(meaning 84 terms containing either a sine or cosine summed together), we can find an equation that directly represents the residuals. This is great because we can predict what the next residual will be with the fourier series. If we graph the fourier series of the other 13 stocks, the graphs would come out looking something like these previous 2 graphs.

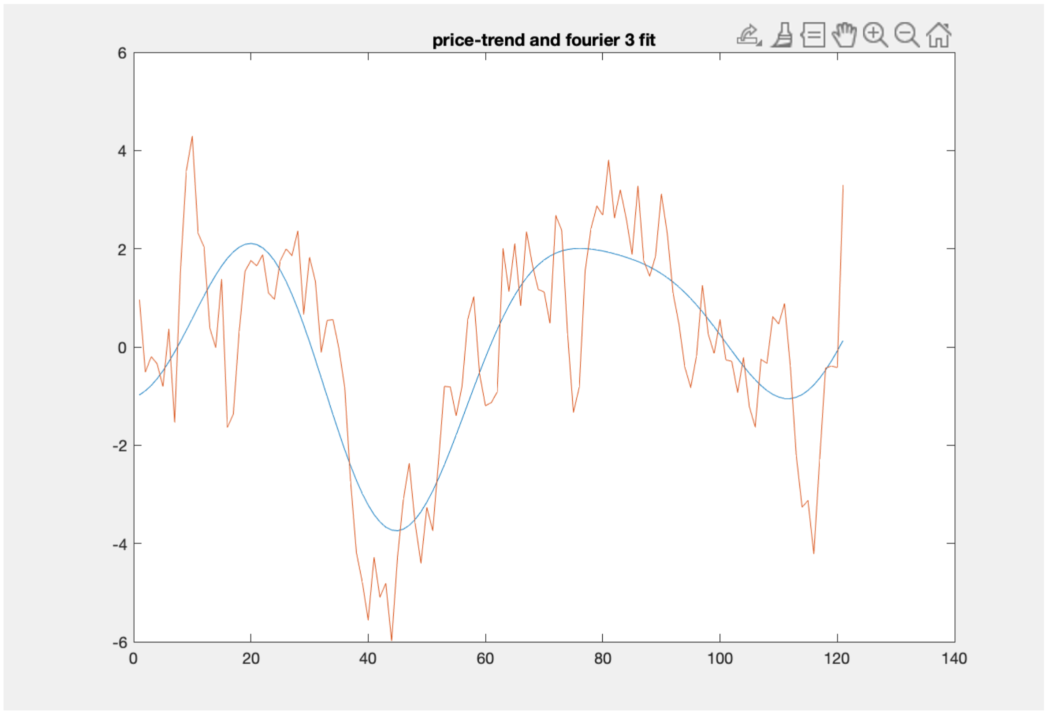
3.5 Combining our Fourier Series and Linear Model

In section 3.3, we built a linear model to fit our daily closing prices. If we were to extend the line of best fit, we would have a somewhat decent idea of what the stock price will look like in the near future. In section 3.4, we came up with a defined Fourier Series to fit the residuals, which are the difference between the actual closing price of a given day and the predicted closing price calculated from our linear model. By extending this line we could predict how far our predicted closing price will differ from the actual closing price of a stock. Now, what happens if we take these two equations and add them together? We will have an equation resembling the following,

y=mx+b+sin(x)+a sin(omega x) + b cos(omega x)......
Where omega is some variable.

This equation is combining the trend of the stock prices(our linear model) and the trend of the residuals(our fourier series). If we combine our linear model with our fourier series, we should get an equation that predicts our stock price more accurately than the linear model by itself because we are taking into account how our original linear model deviates from the actual price measured. Let's graph this equation as well as test out our predictions.





If we take a look at this graph, we can see that it resembles the Fourier Series graph from section 3.4, but shifted up by some value. This value is the y=mx+b formula of our linear model. Now, let’s put our equation to the test. Here are the first 10 stock prices in list format from our relevant period for AGYS, which starts 121 days before 5-19-2022.

```
[45.040001, 43.509998, 43.759998, 43.549999, 43.02, 44.119999, 42.16, 45.209999,
47.16, 47.790001]
```

Here are the first 10 stock prices in list format that our equation generates. How did we come up with this? I looped over the first 10 days in the relative period and plugged them into our equation(linear model + fourier series).

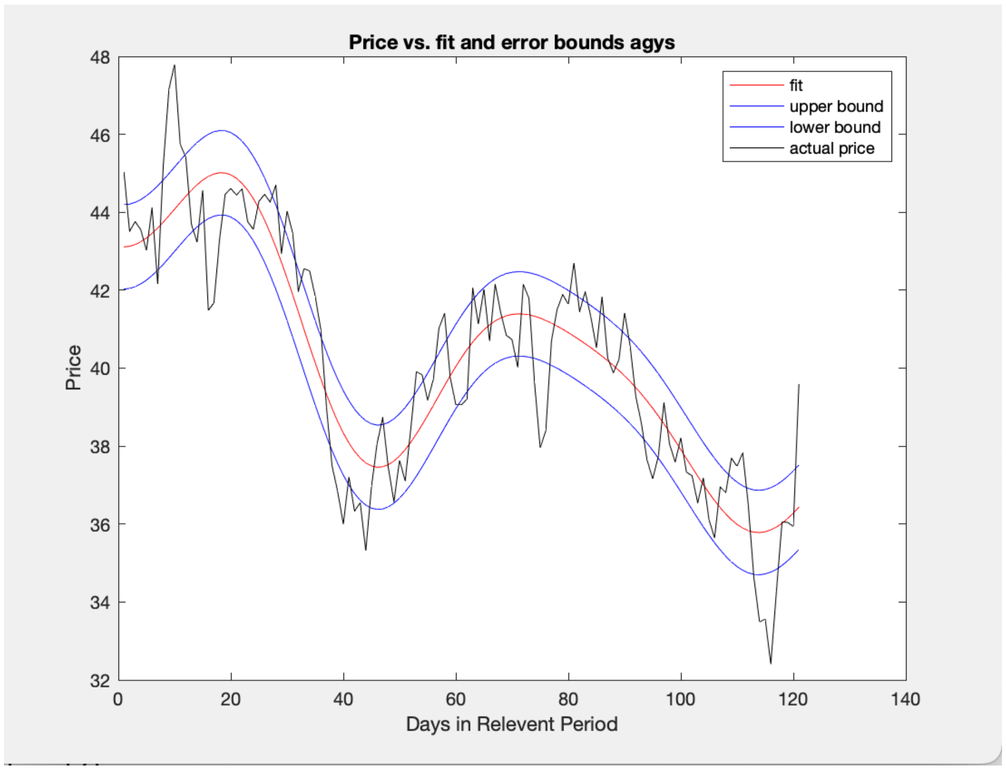
```
[45.040001+3.81696511e-16j, 43.509998+2.05528891e-16j,
43.759998-1.46806350e-16j, 43.549999-3.52335241e-16j, 43.02-2.12869208e-16j,
44.119999+6.45947942e-16j, 42.16+6.45947942e-16j, 45.209999-8.80838102e-17j,
47.16+0.00000000e+00j, 47.790001+4.69780321e-16j]
```

The first thing to notice is that the numbers generated have a real and imaginary part. This is where I noticed my current imitations with python. The python program I created in order to calculate the fourier series included the imaginary part. And because of the time limit on this project, since I knew that matlab would not include the imaginary part, I started to think about switching for this part.

[45.040001, 43.509998, 43.759998, 43.549999, 43.02, 44.119999, 42.16, 45.209999, 47.16, 47.790001]

The stock prices generated from the equation match up perfectly with the real stock prices taken from Yahoo Finance. Why is it that the prices generated from our equation match up perfectly with the real stock data? Let’s go back to our linear model. The purpose of our linear model was to graph the trend of where the stock data is heading. A line of best fit is the term used earlier to describe this trend line.. We were then able to generate an equation of line of best fit in the form y=mx+b. When using this line to fit the data, there will be differences between our trend line and the actual data. We then graphed these stock price differences and calculated a fourier series to generate formulas to approximate the price differences. Because we used the maximum fourier coefficients we could, we didn’t just approximate the price differences. We were able to perfectly model the price differences. When adding the linear model(models trend) and the fourier series(models price difference), the stock prices generated using this equation match up perfectly with the real stock data. This sounds like it would work, but we are making a bad assumption. We are assuming that the future period will mimic the relevant period. So the stock price on the first day after the current day will equal the first stock price in the relevant period. In order to combat this assumption, we will lower the terms so that our fourier series does not fit the actual data perfectly. Because I was not able to We will calculate the errors and find a margin of error. With a margin of error, we can implement error bounds to give us a rough estimate of how stock price might look in the future. Now, what will happen when we use this formula on a day in the future? How close will our prediction be?

3.6 Prediction



with a margin of error 1.3267578024156983

Prediction 1 day after current day is 44.07258088354782

Prediction 2 day after current day is 44.090837216982614

Prediction 3 day after current day is 44.1093071414788

Prediction 4 day after current day is 44.126885331200285

Prediction 5 day after current day is 44.142474033891055

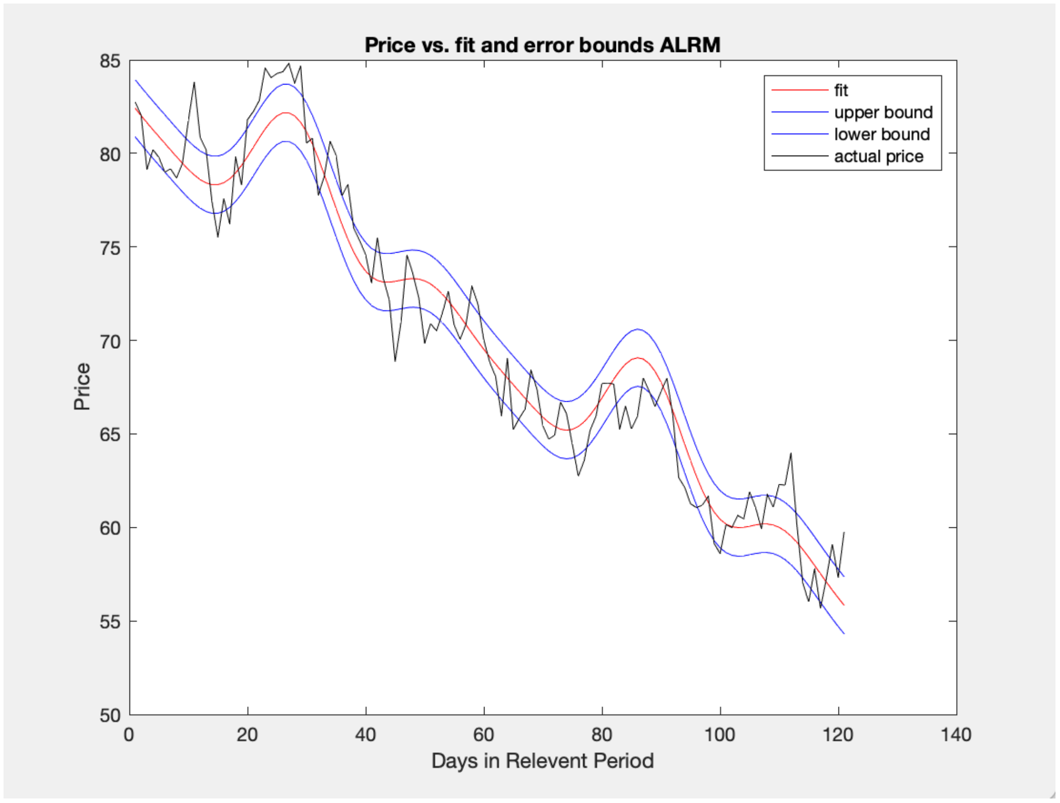
Prediction 6 day after current day is 44.15499546317748

Prediction 7 day after current day is 44.163403979688354

Prediction 8 day after current day is 44.166697928200485

Prediction 9 day after current day is 44.16393100171769

Prediction 10 day after current day is 44.15422300848638



With a margin of error 1.541797154812492

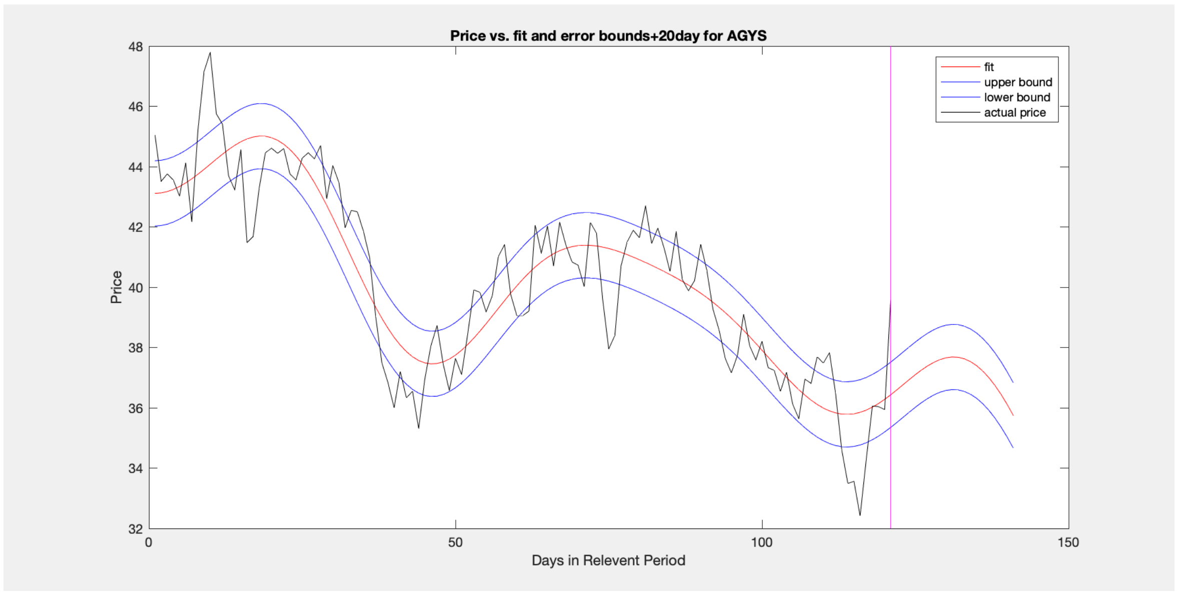
Prediction 1 day after current day is 73.87279572424549

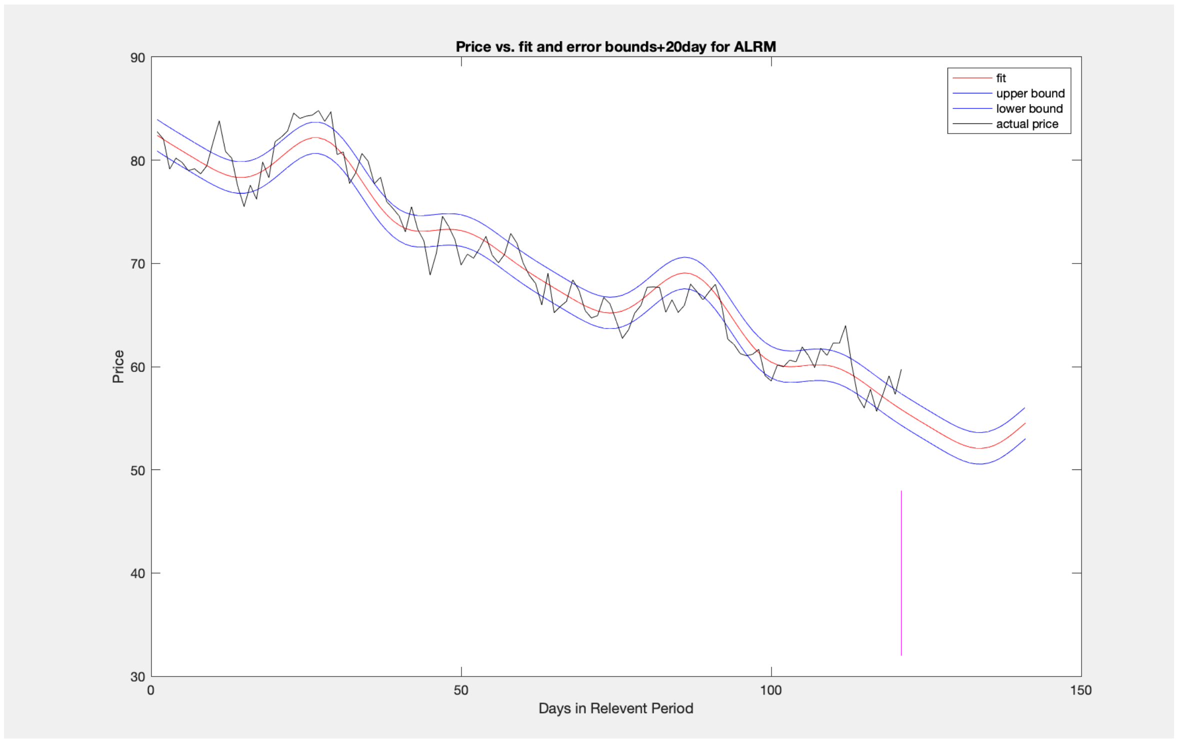
```
Prediction 2 day after current day is 73.72759419587425

Prediction 3 day after current day is 73.57519454468155

Prediction 4 day after current day is 73.41452841297858

Prediction 5 day after current day is 73.24470801855512

Prediction 6 day after current day is 73.06504716663835

Prediction 7 day after current day is 72.87507777496009

Prediction 8 day after current day is 72.6745615360973

Prediction 9 day after current day is 72.46349644982395

Prediction 10 day after current day is 72.24211807279032
```

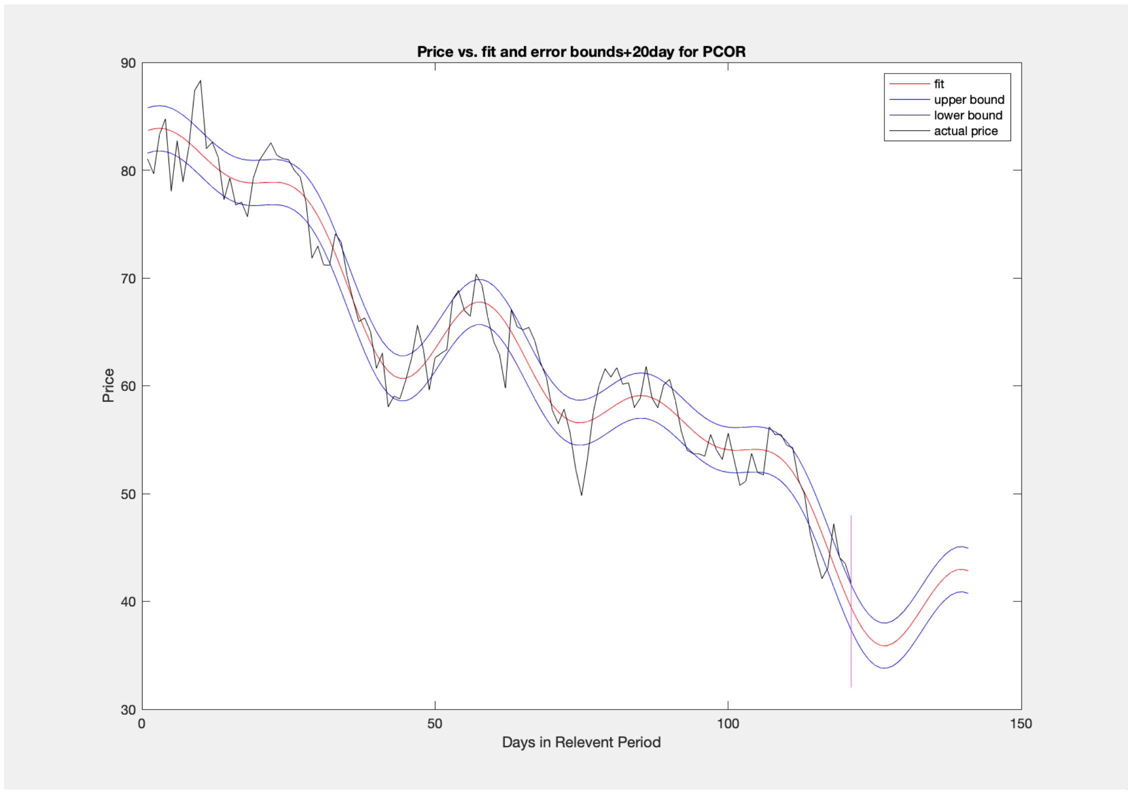
The above graphs are what our first iteration of what our models look like.We need to add error bounds as well as lower the amount of fourier terms so that our graph does not model the data perfectly.

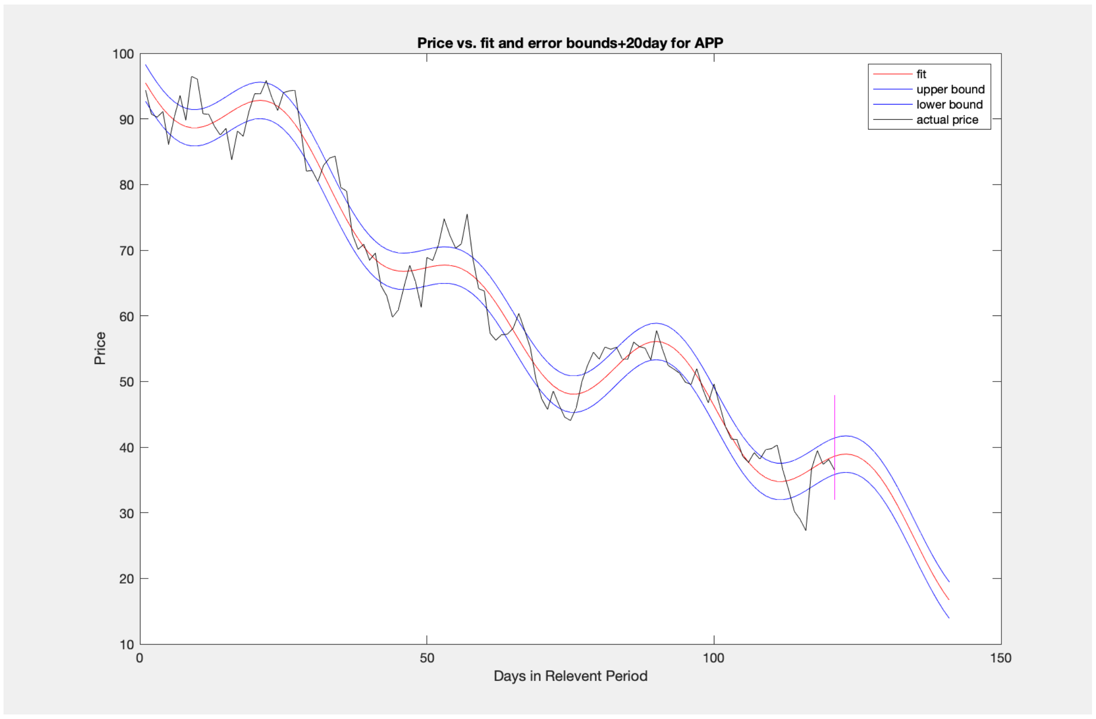
Switch to matlab

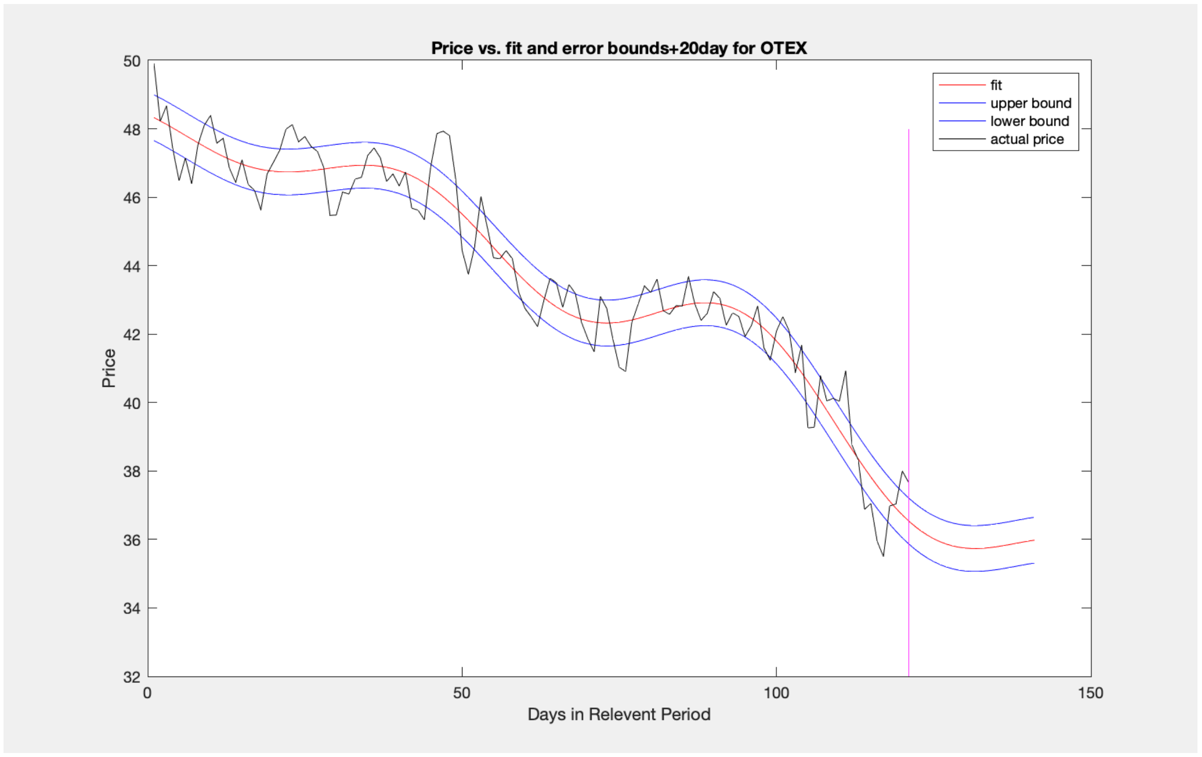
When creating a fourier series function in python, an issue arose. I was unable to extend the line to fit the residuals. So I decided to switch to matlab and test in the built in fourier series function. The built in fourier series worked perfectly and I decided to use matlab for this section. The following graphs are used using matlab.

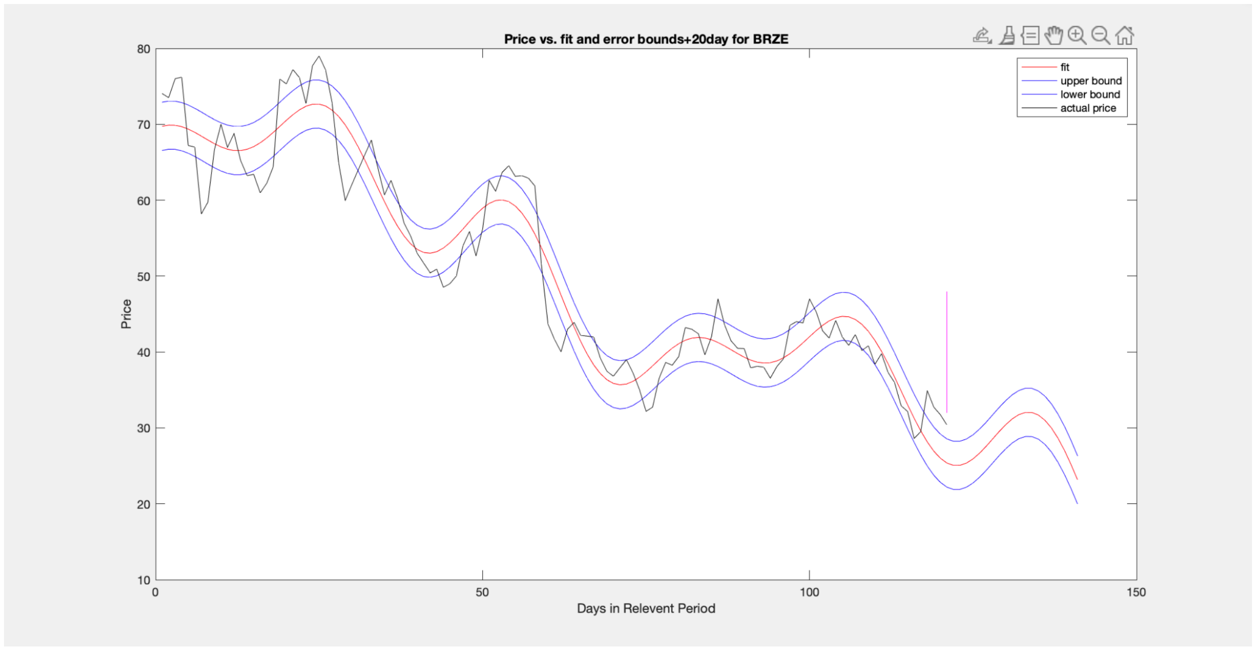


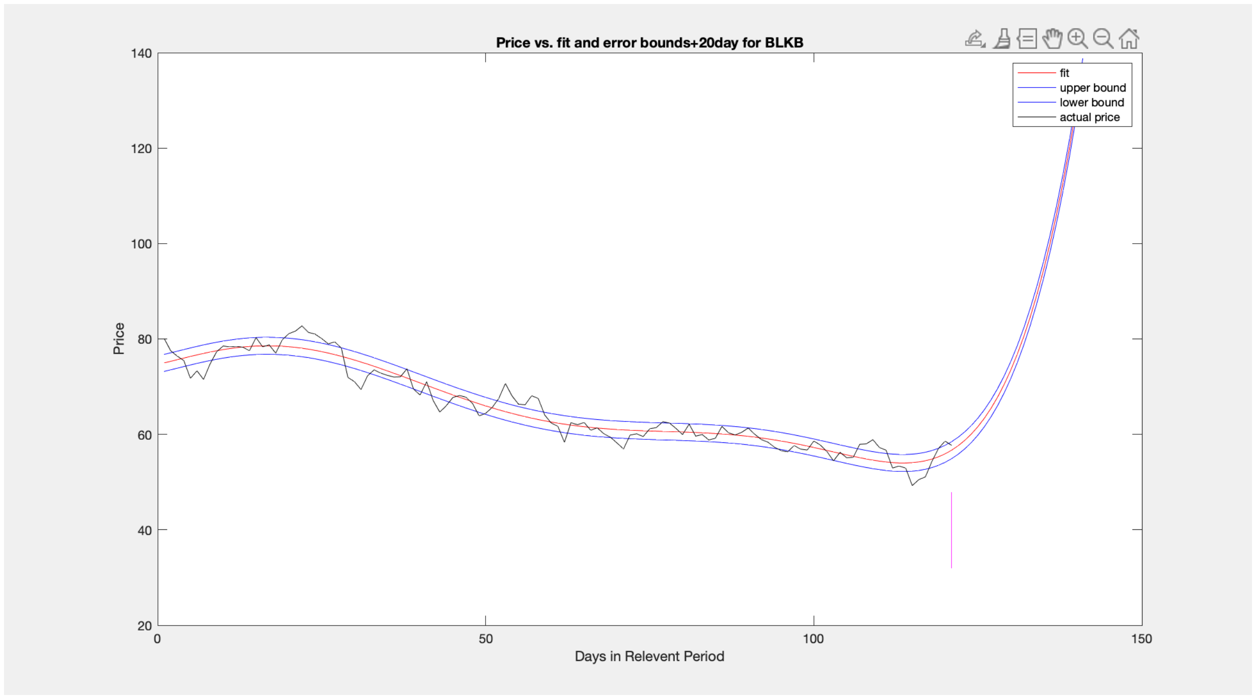

Price vs. fit and error bounds+20day for ALRM
fit
upper bound
lower bound
actual price
Price
Days in Relevent Period
90
80
70
60
50
40
30
0
50
100
150

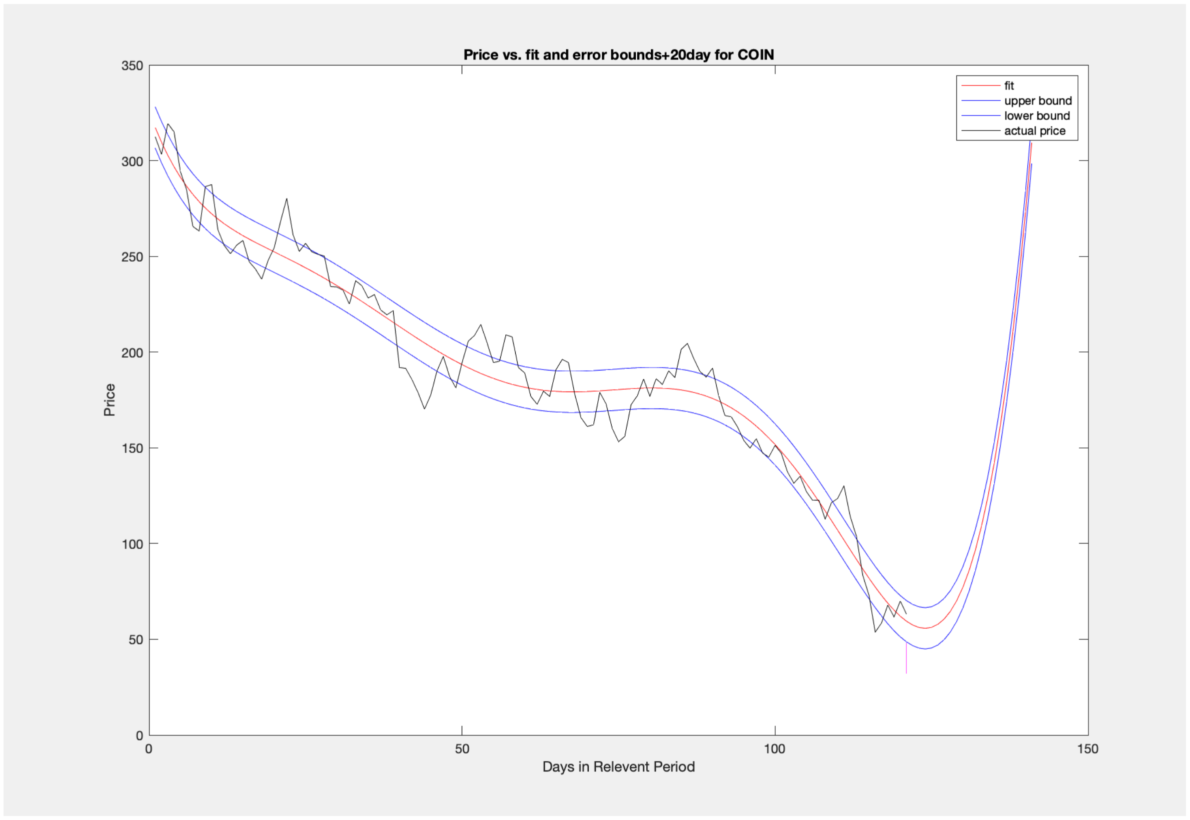

Price vs. fit and error bounds+20day for PCOR
fit
upper bound
lower bound
actual price
Price
Days in Relevent Period
90
80
70
60
50
40
30
0
50
100
150


Price vs. fit and error bounds+20day for APP
fit
upper bound
lower bound
actual price
Price
Days in Relevent Period
100
90
80
70
60
50
40
30
20
10
0
50
100
150


Price vs. fit and error bounds+20day for OTEX
fit
upper bound
lower bound
actual price
Price
Days in Relevent Period
50
48
46
44
42
40
38
36
34
32
0
50
100
150

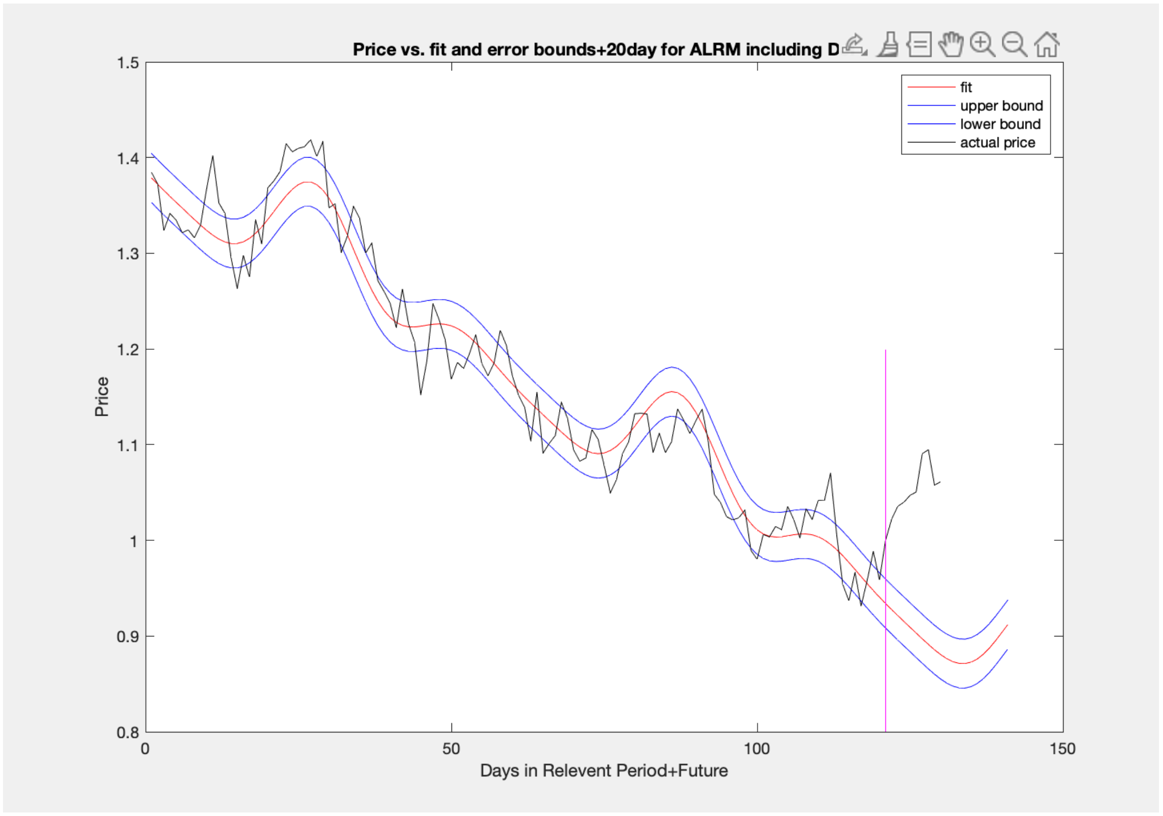

Price vs. fit and error bounds+20day for BRZE
fit
upper bound
lower bound
actual price
Price
Days in Relevent Period
80
70
60
50
40
30
20
10
0
50
100
150

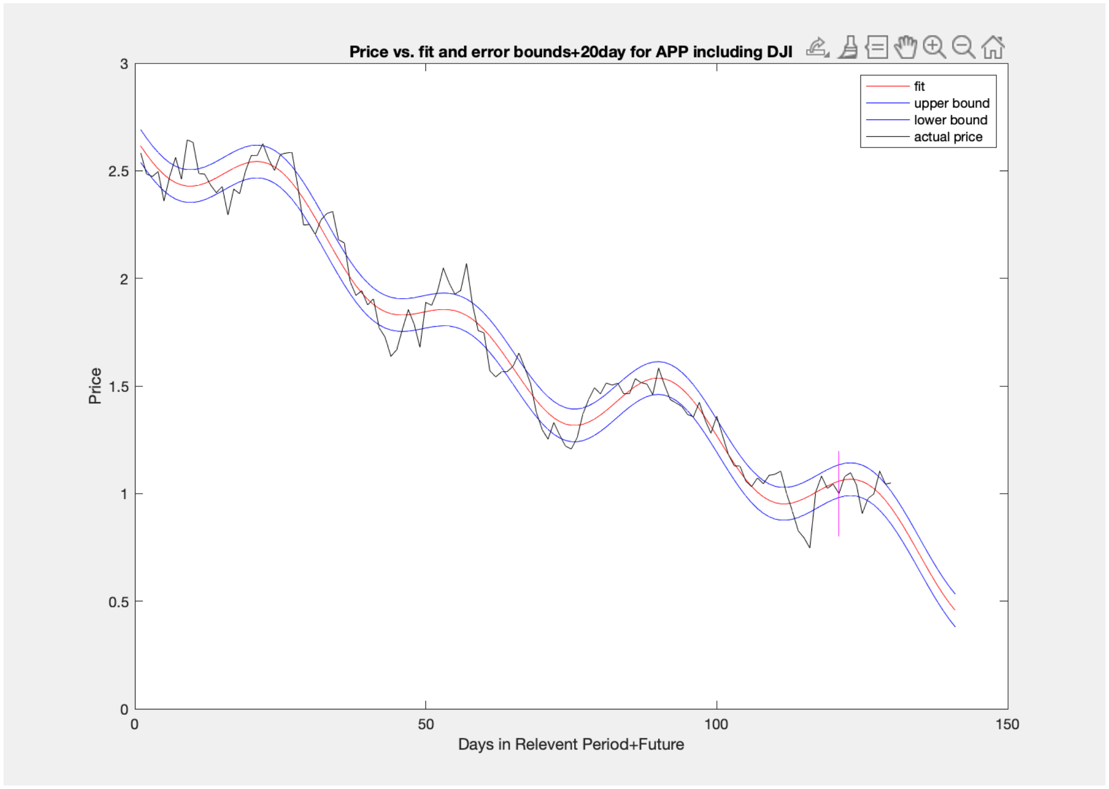

Price vs. fit and error bounds+20day for BLKB
fit
upper bound
lower bound
actual price
Price
Days in Relevent Period
140
120
100
80
60
40
20
0
50
100
150

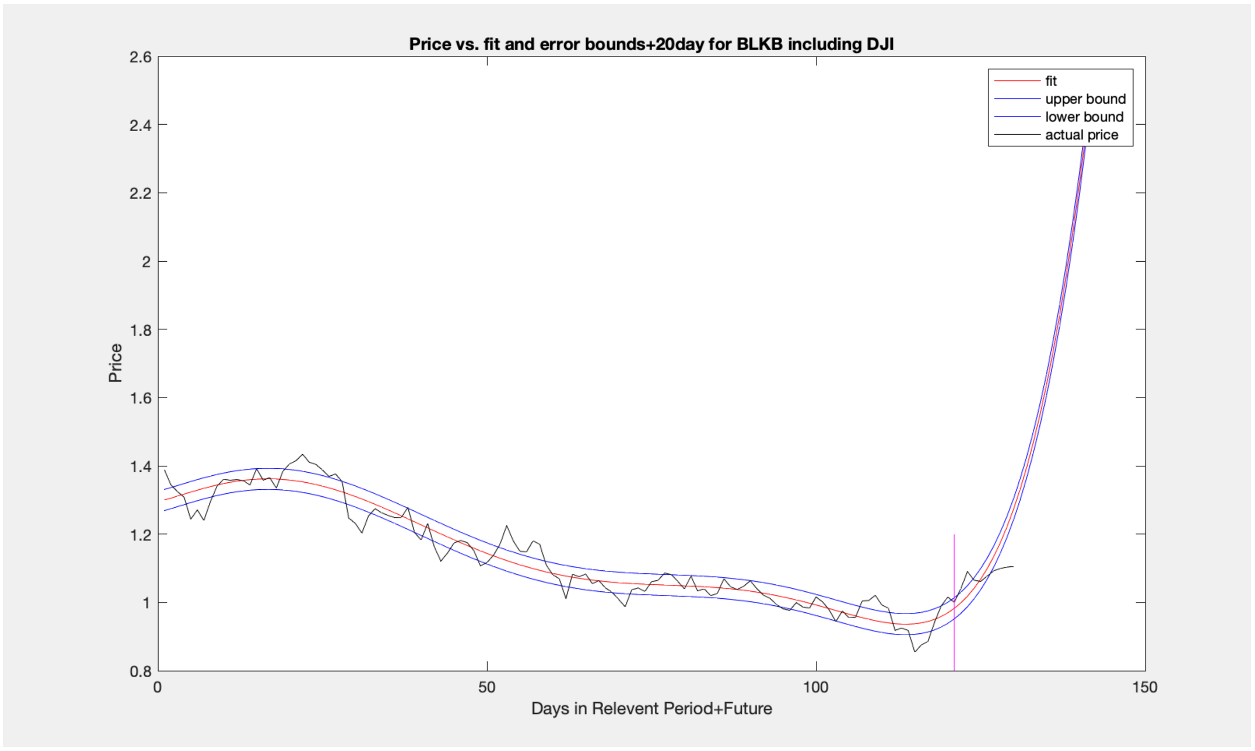


3.7 Extending the Line

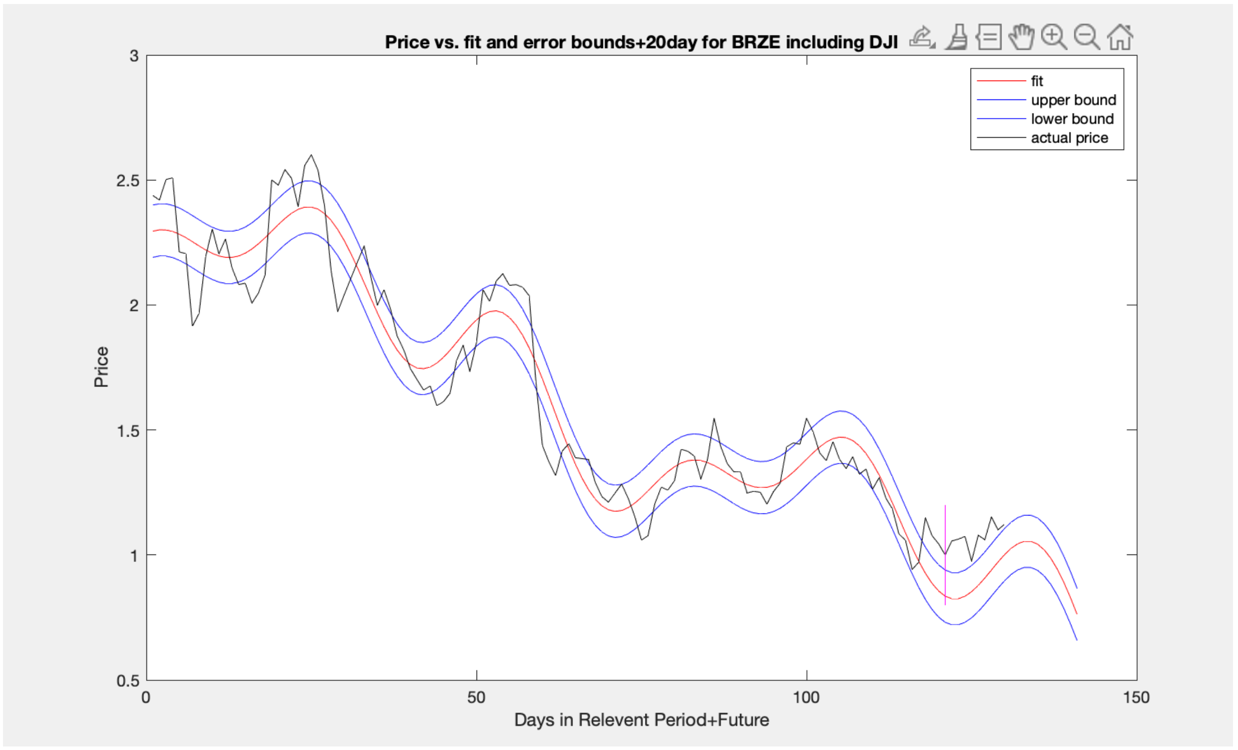
In order to finally make our predictions, we need to extend the line past the relevant period. Using the linear and fourier model we will extend the lint to try to predict the price. Again, this was done in matlab. Because of the variable nature of stocks, we will include an error bound. We will count how many days into the future our graph stays between the error bounds. This is a very common technique in statistics to make predictions. We can’t come up with a definite number so we create error bounds to create a range of values our prediction will lie in.


Price vs. fit and error bounds+20day for ALRM including D
fit
upper bound
lower bound
actual price
Price
Days in Relevent Period+Future
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0
50
100
150

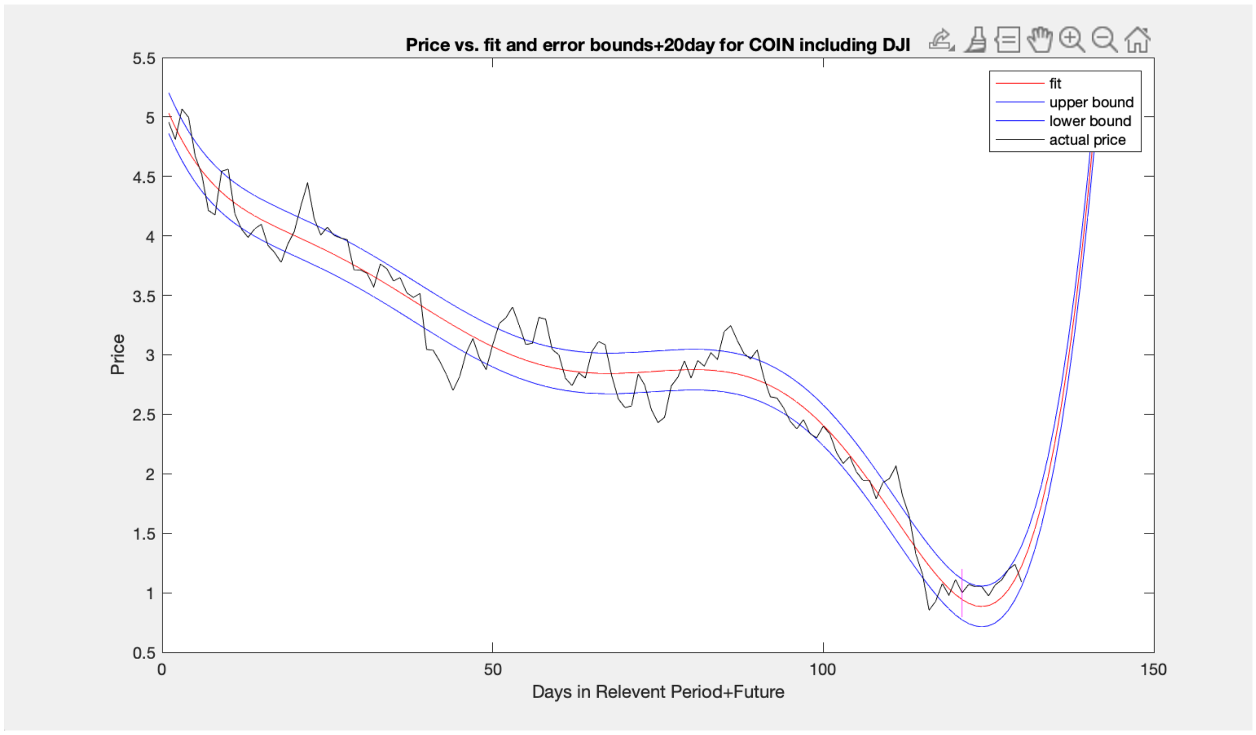

Price vs. fit and error bounds+20day for APP including DJI
fit
upper bound
lower bound
actual price
Price
Days in Relevent Period+Future
3
2.5
2
1.5
1
0.5
0
0
50
100
150

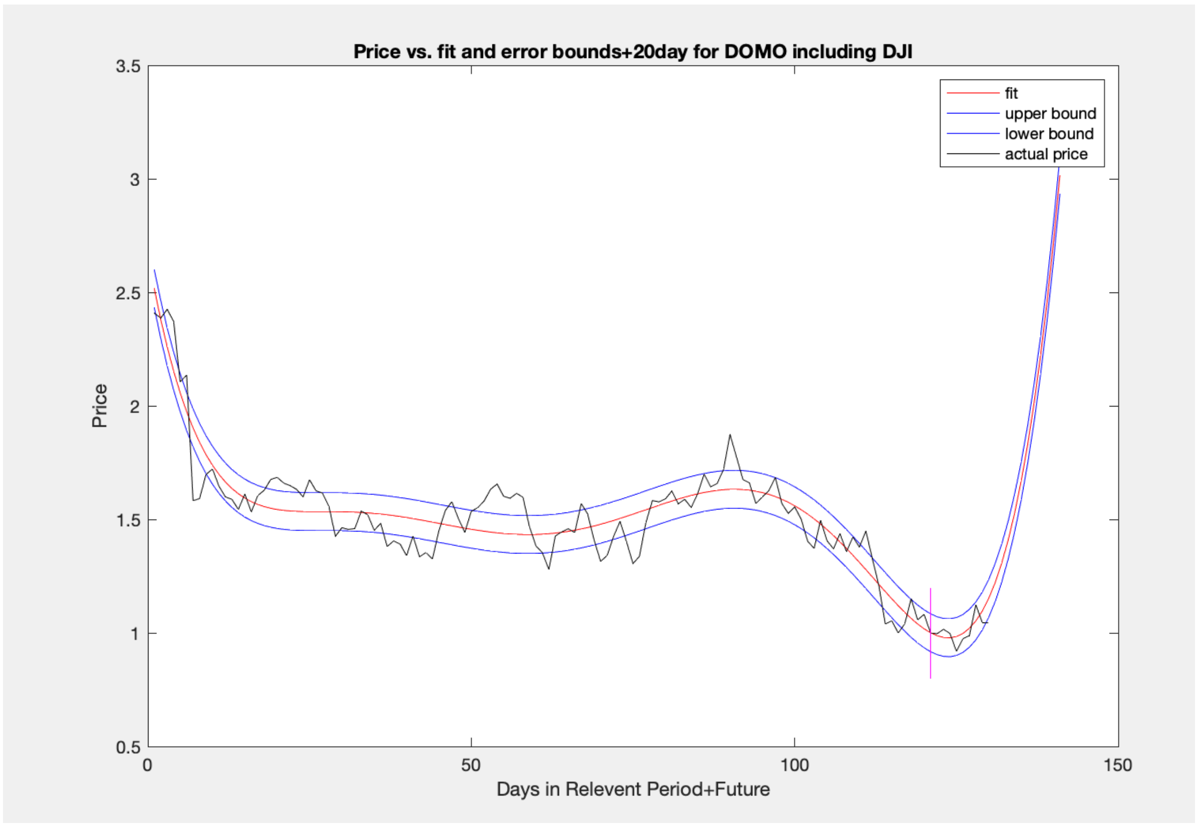


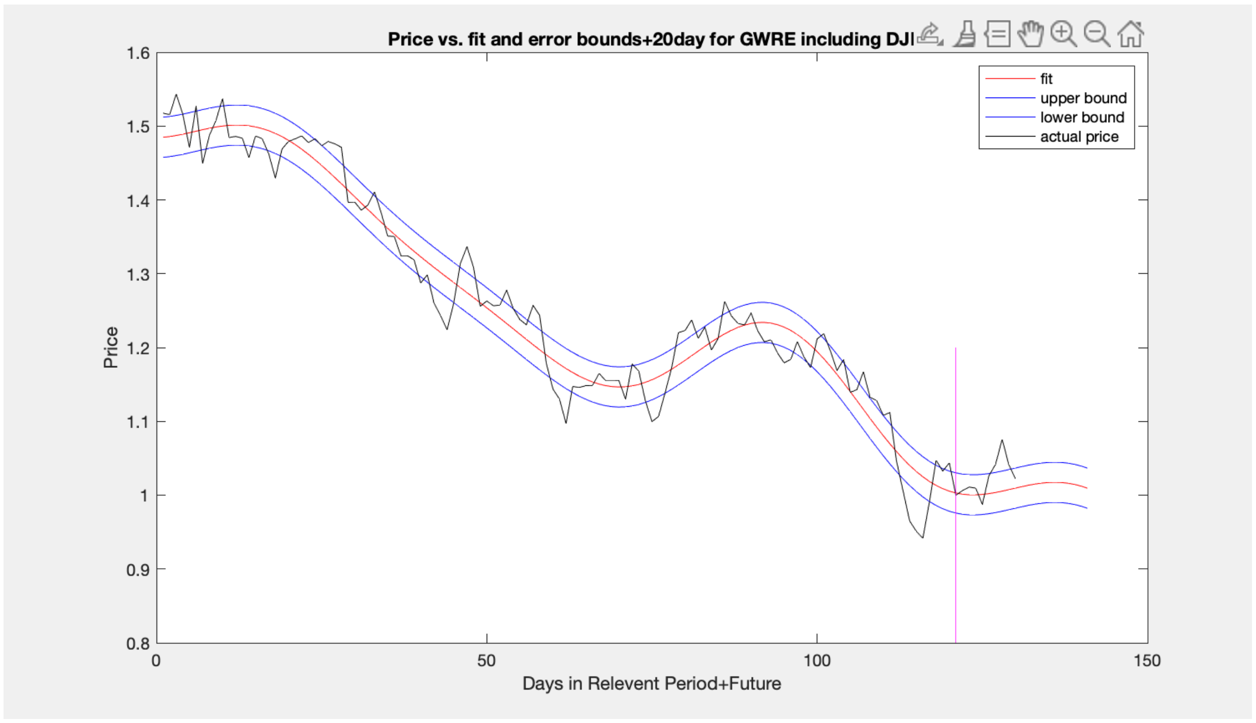
Price vs. fit and error bounds+20day for BLKB including DJI
fit
upper bound
lower bound
actual price
Price
Days in Relevent Period+Future
0
50
100
150
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6

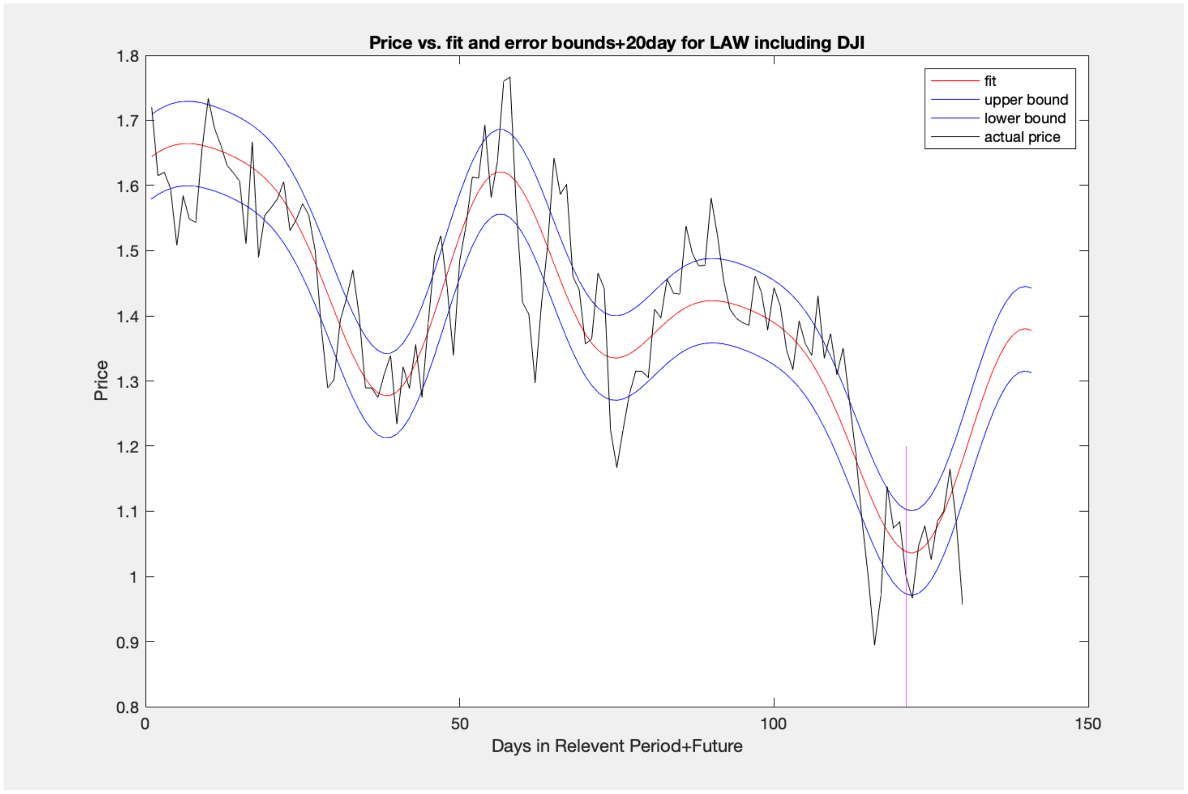


Price vs. fit and error bounds+20day for BRZE including DJI
fit
upper bound
lower bound
actual price
Price
Days in Relevent Period+Future
0
50
100
150
0.5
1
1.5
2
2.5
3


Price vs. fit and error bounds+20day for COIN including DJI
fit
upper bound
lower bound
actual price
Price
Days in Relevent Period+Future
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
50
100
150


Price vs. fit and error bounds+20day for DOMO including DJI
fit
upper bound
lower bound
actual price
Price
Days in Relevent Period+Future
3.5
3
2.5
2
1.5
1
0.5
0
50
100
150


Price vs. fit and error bounds+20day for GWRE including DJI
fit
upper bound
lower bound
actual price
Price
Days in Relevent Period+Future
1.6
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0
50
100
150


Price vs. fit and error bounds+20day for LAW including DJI
fit
upper bound
lower bound
actual price
Price
Days in Relevent Period+Future
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0
50
100
150

How many days into the future does the stock price stay within the error bounds? On the left side we have the stock. On the right side of the table, we have how many days the stock remained in the error bounds.

Stock	How many days
AGYS	0
ALRM	0
APP	3
BLKB	0
BRZE	0
COIN	1
DOMO	1
GWRE	1
LAW	1
LPSN	0
NCNO	0
OTEX	0
PCOR	0
PEGA	0
SMAR	2
	0.6 days on avg

On average, the stock stayed within the error bounds for 0.6 days, with a standard deviation of 0.88. Clearly, this is not a significant amount of time. Less than a day. And because of the volatility of the market, this is not too surprising.

Looking through all the graphs, our predictions using the fourier+linear model are inconsistent. However, there is some promise. For a few of the stocks, the predicted stock price stays in the bonds for a few days. For other stocks, the predictions exit the error bounds within the first two days. Sometimes, after exiting the error bounds, the prediction enters back in the bounds after a few days. This is the nature of the predictions. While the predictions may not stay in the error bounds, our predictions still give an idea of the trend and overall direction in which the actual prices are going. The main limitation in this model is that it does not take into account the state of the market or economy. Our model only uses past stock prices as variables to predict future prices. There are many variables in the market that can affect the stock market. For example, GDP, the gross domestic product. GDP is mainly used to determine how much a country produces and the price of individual stocks. So if the GDP is low,production is low, so companies will be making less revenue which results in lower stock prices. We won't be using this variable to add on to our model for now, because of the complicated nature of it. One variable that we can use however that might give us a better picture of the state of the market is the DJI, or the Dow Jones Industrial Average. The Dow Jones Industrial Average, Dow Jones, or simply the Dow, is a price-weighted measurement stock market index of 30 large companies listed on stock exchanges in the United States. The idea is that this Average gives us a rough idea of what the trend of the entire market looks like.

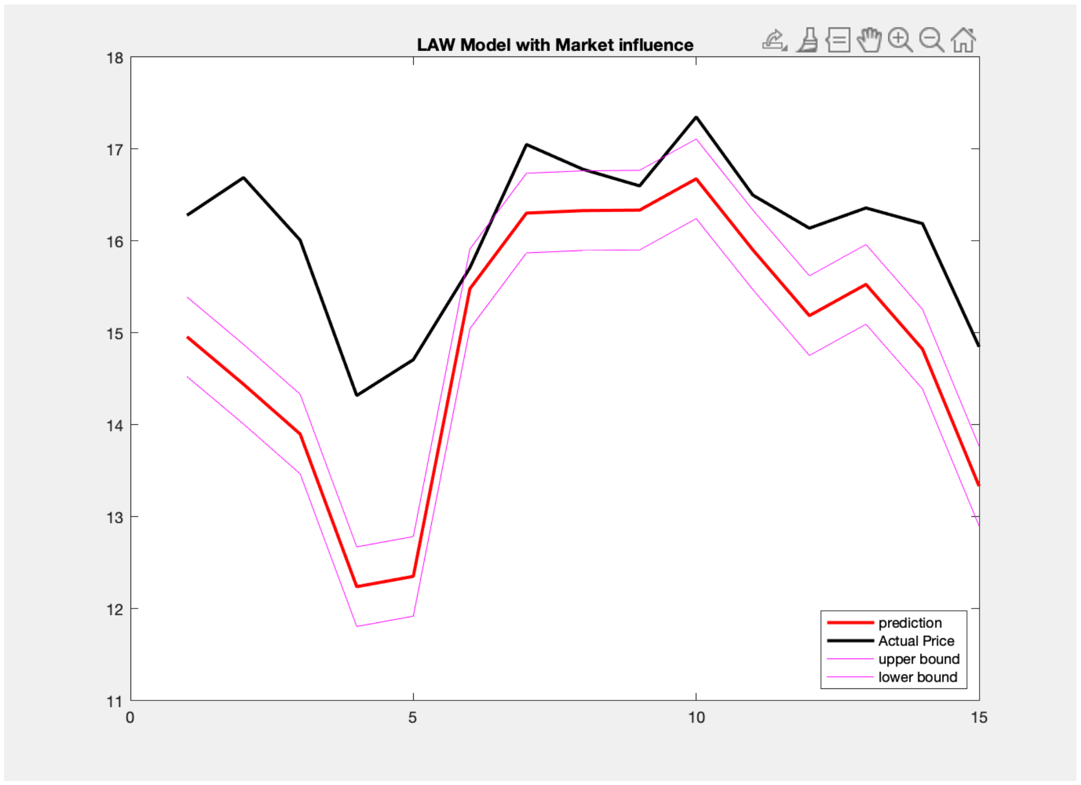
4. Market Effect

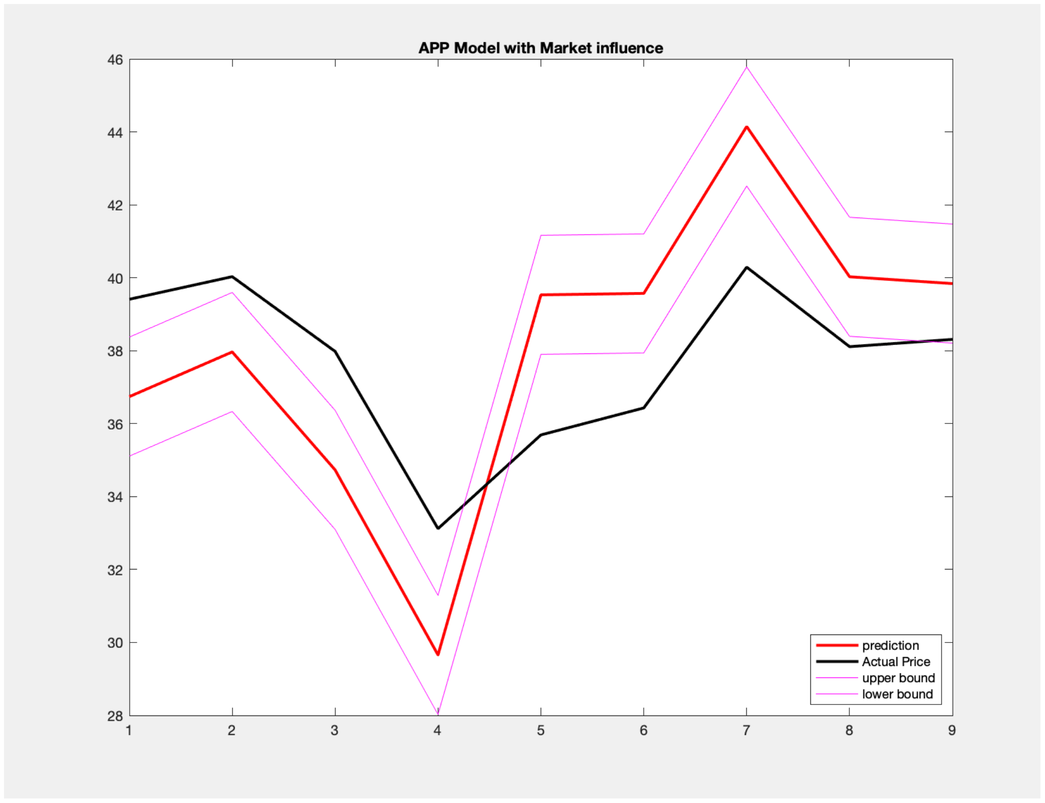
As mentioned previously, we want to somehow incorporate the state of the market and economy into our model. We are going to do this with the Dow Jones Industrial Average.

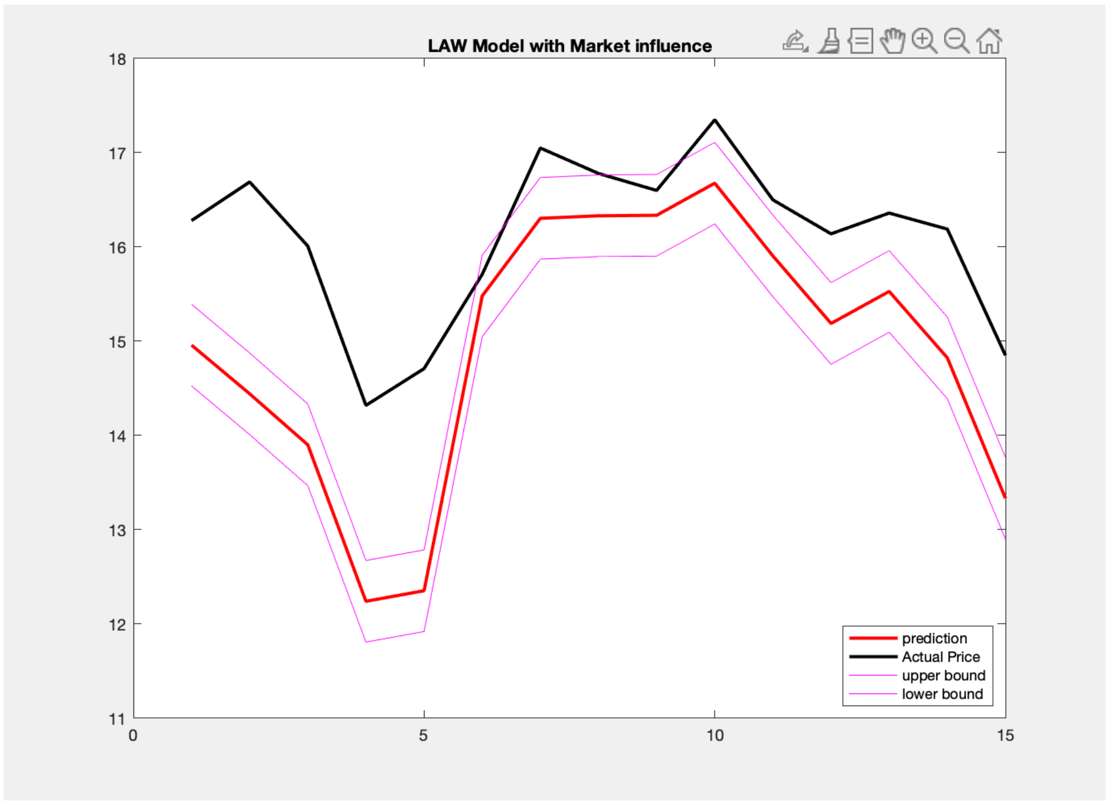
So how did we do this? We treat the Dow Jones Industrial Average(DJI) as another “stock”. So we find the relevant period and the trend just like we did with our other 15 stocks. Next, we need to normalize our DJI values to one by dividing by the value of DJI on day 1. Then we find the correlation coefficient between the stock, say AGYS, and DJI during the relevant period of the stock. This is the exact same process we did for our 15 previous stocks, but as mentioned, we are treating the Dow Jones Industrial Average as another stock. Let’s say the correlation coefficient is A. To predict the future price of the stock including DJI we use

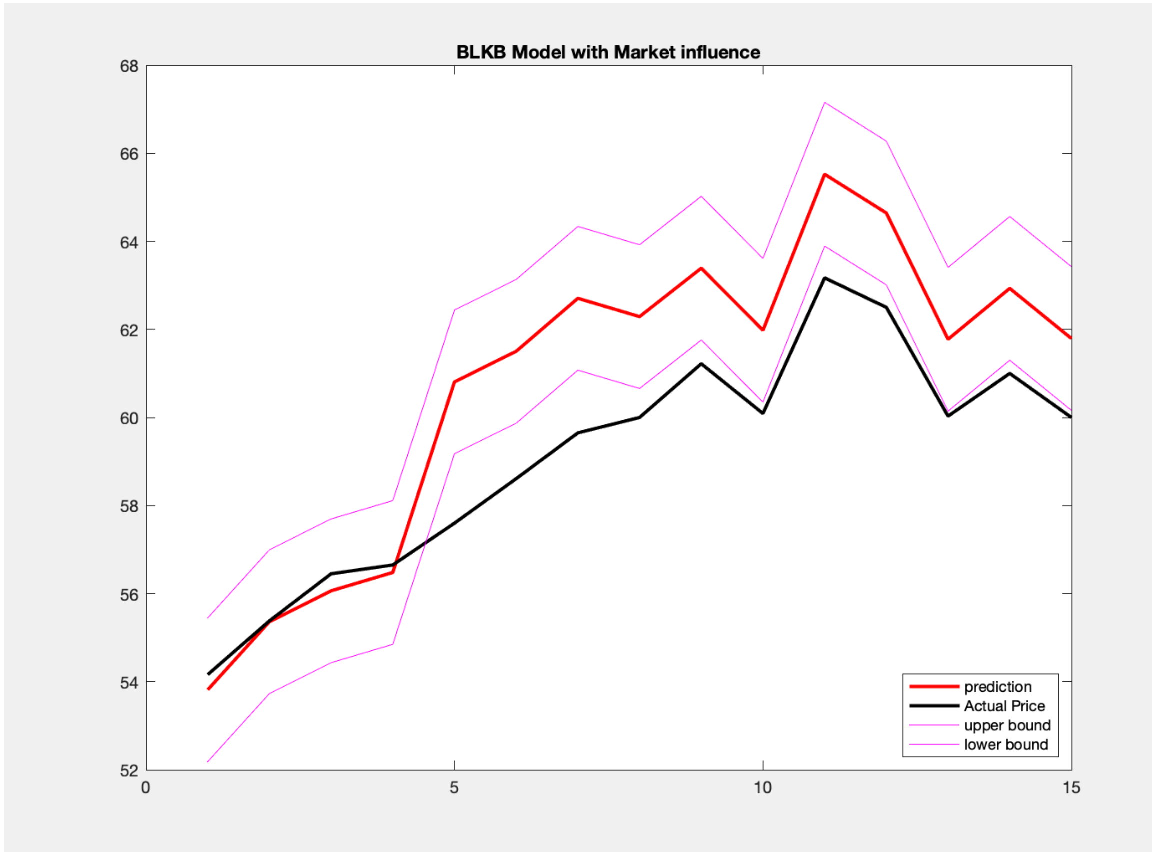
stock(i day in future)=A * DJI(i) + (1-A) * stock_price(i)

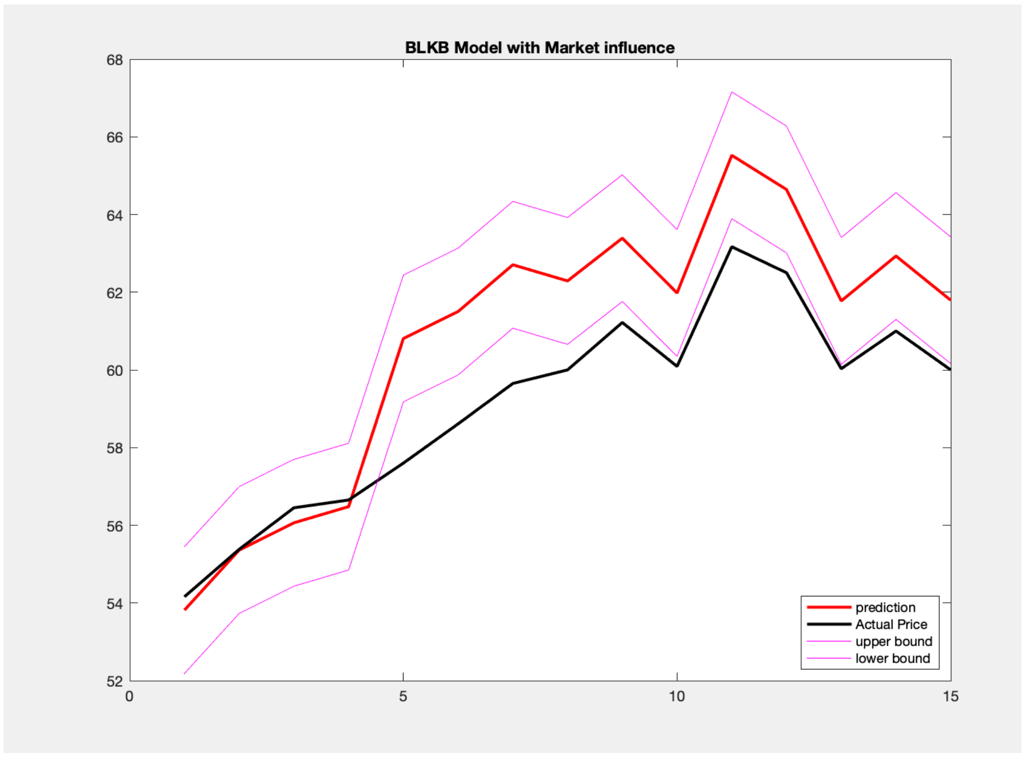
We always give a higher weight to the stock itself. If A>5, the coefficients in the formula are exchanged. After we find the price, we need to convert the normalized prices back into their original prices by multiplying by the price of the stock on the first day.


LAW Model with Market influence
prediction
Actual Price
upper bound
lower bound


APP Model with Market influence
prediction
Actual Price
upper bound
lower bound


LAW Model with Market influence
18
17
16
15
14
13
12
11
0
5
10
15
prediction
Actual Price
upper bound
lower bound


BLKB Model with Market influence
68
66
64
62
60
58
56
54
52
0
5
10
15
prediction
Actual Price
upper bound
lower bound



Here is a table detailing the number of days that the stock stayed within the error bounds with DJI and without DJI.

Stock	Days within error bounds	Days within error bounds+DJI
AGYS	0	0
ALRM	0	0
APP	3	3.5
BLKB	0	1
BRZE	0	1
COIN	1	7
DOMO	1	9
GWRE	1	5
LAW	1	8

LPSN	0	0
NCNO	0	9
OTEX	0	0
PCOR	0	0
PEGA	0	0
SMAR	2	2
	0.6 days on avg	3 days on avg

From our previous calculations, the stock stayed on average within the error bounds for 0.6 days, with a standard deviation of 0.88. We found this to not be a significant amount of time. Now, when we include the Dow Jones Industrial Average, our values are a little different. The stocks stayed on average within the error bounds for 3 days with a standard deviation of 3.47. The stock stayed in the error bounds for a much longer time than when not taking into account the market state. However, the standard deviation is also a lot higher, indicating the current volatility of the market.

Firstly, because of covid and the economy, the stock market is extremely volatile. But based on our findings, including the market effect of the Dow Jones Industrial average seems to serve as a better predictor than without it. Without the Dow Jones Industrial average, the average amount of days that our prediction stayed within the error bounds was less than 1 day. With the Dow Jones Industrial Average, the average amount of days that our predictions stayed within the range was around 3 days. However, the average number of days with the DJI is quite volatile, which can be attributed to the current volatility of the stock market.

5.

Savitzky Golay

A Savitzky–Golay is a method that can be applied to a set of data points for the purpose of smoothing the data. That is, to increase the precision of the data without messing with the signal tendency. This is achieved by a process known as convolution. This is done by fitting successive sub-sets of adjacent data points with a low-degree polynomial by the method of linear least squares. By changing the degree of a polynomial, the hope is that the curve will fit the data better. We used a trend line to create a linear prediction for our stocks. What will happen if we change the degree of that formula. So instead of having a degree of 1, we will change the degree to be in between 0.9 and 1. Will our new trend line fit the data better and lead to better predictions?

The plan is to use the linear model but change the power. Our linear model was y=mx+b. First, we change the power of the x. We will use all power from 0.9 to 1, incrementing by 0.01. So, 0.91, 0.92 etc.

Similar to our previous linear model, we will calculate the residuals. By taking the predicted price from the model with a power different than one, and the actual stock price. Because some of the residuals will be negative, we will square them and then take the square root. We will add up these residuals and take the square root. We will do this for all powers.

4.1 Savitzky Golay for AGYS

Power	Avg Residual
0.90	2.1168
0.91	2.0581
0.92	2.0027
0.93	1.9538
0.94	1.9091
0.95	1.8650
0.96	1.8262
0.97	1.7899
0.98	1.7575
0.99	1.7418
1.00	1.7364
1.001	1.7358
1.01	1.7409
1.02	1.8020
1.03	1.8738

As shown in this table, our Savitzky Golay model where the power is 0.99 has the lowest average residual. This makes sense as it is the closest to one, which was used in the linear model. How did we do this? First we took the y=mx+b formula and changed it so we raise the x to a different power. So instead of our trend line having a power of 1, y=mx+b, we are changing the power so our formula is y=mx^(y)+b. Then we calculate the residuals by subtracting the predictions generated by the new formula and the actual stock prices. We square the residuals because some of them will be negative. Then, we take the square root. We find the average of these residuals by adding them up and dividing by how many there are.

Stock	Days within error bounds
AGYS	0

Like with our previous models, we measured how many days the predictions stayed within the error bounds. If we compare this to our linear model with the fourier series for the first five stocks, the Savitsky Golay method produces very similar predictions.So by changing the power, our predictions do not get more accurate. However, we changed our powers by an increment of 0.01 from 0.9 to 0.99. If we change the increments so that we change the power to 0.998, then it is possible that the predictions may become more accurate.

4.2 Savitzky Golay for ALRM

Power	Avg Residual
0.90	5.4750
0.91	5.1330

0.92	4.4032
0.93	4.0159
0.94	4.0159
0.95	3.6140
0.96	3.2188
0.97	2.8477
0.98	2.5369
0.99	2.3198
1.00	2.2012
1.001	2.1925
1.002	2.1847
1.003	2.1776
1.004	2.1735
1.005	2.1756
1.006	2.1800
1.01	2.2169
1.02	2.4531
1.03	2.8549

Similar to AGYS, the power that results in the lowest residual is close to one. It is in fact 1.004, and not 1. We will now use this to see if our predictions can get any better than using the original power of 1 in our linear model using our new power calculated from our Savitzky Golay.

Stock	Days within error bounds
ALRM	0

Even though the average residuals for our coefficient of 1.003 is less than the linear model coefficient of 1, the prediction has not changed. It appears as the difference is not large enough to result in a better prediction. While 1.003 might be the optimal coefficient, its results are not visible.

4.3 Savitzky Golay for APP

Power	Avg Residual
0.90	12.3926
0.91	11.5489
0.92	10.6749
0.93	9.7635
0.94	8.8295
0.95	7.8560
0.96	6.8723
0.97	5.9105
0.98	5.1452
0.99	4.7878
1.00	4.8067
1.001	4.8233
1.002	4.8454

1.01	5.2025

The coefficient with the lowest average residual is 1, which corresponds with our original linear model. And if we run the linear model. We will just get the same as what we had before.

4.4 Savitzky Golay for BLKB

Power	Avg Residual
0.95	3.9124
0.96	3.5041
0.97	3.1166
0.98	3.1166
0.99	2.5612
1.00	2.4409
1.001	2.4366
1.002	2.4356
1.003	2.4388
1.01	2.5130

The power with the lower average residual is 1.002 by a small margin. We will use this power to make our predictions and see if the result is any different.

Stock	Days within error bounds
BLKB	0

By replacing our linear equation, y=mx+b with y=mx^(1.002)+b, we ran the matlab program and found that the predictions were not any better than using a power of 1.

4.5 Savitzky Golay for BRZE

Power	Avg Residual
0.95	6.9458
0.96	6.4294
0.97	6.0671
0.98	5.7963
0.99	5.6098
1.00	5.5140
1.001	5.5160
1.002	5.5194
1.003	5.5257
1.01	5.6321

The power with the lowest residuals is 1, which is the same power used in the linear model. y=mx+b. And since we already ran this model, this is the best possible power for this specific stock.

It appears that while 1 might not always have the smallest average residual, its predictions are not necessarily any worse.Of the 5 stocks we implemented the savitzky golay method, 4 of the stocks had a power with a smaller average residual than 1. However, a lot of these powers were extremely close to 1 so we can assume that the predictions would not change by much if at all because of the close proximity to 1. And this seems to hold true as when using these new powers, none of the predictions seemed to change. Using proper terminology, this power is called a gamma. So we can say that based on these 5 stocks, there is no optimal gamma as the predictions remained the same. In addition to there being no optimal gamma, it can be concluded that the Savitsky Golay method did not improve the span of the predictions.

What could have gone better?

The biggest obstacle in this project was programming a reliable method to calculate the fourier coefficients in python. After searching up countless websites and answers on stack overflow, I could not find a way to calculate the fourier series for this project. While I have a decent understanding of the Fourier series, it was definitely not too easy to learn the math on the fly during this project. Same with the Savitzky Golay method, but that was a lot easier to learn and research.

Advice to future students

In my opinion, it is important to try to really understand the math that is going on. It is easy to use the given functions in python and matlab, but understanding how the fourier series or savitsky golay works is a good idea.

Conclusion

The first method to make short term predictions combined a trend line with the fourier series. This method did not result in very accurate predictions. The predicted stock price stayed within the error bounds with an average of 0.6 days. When including the state of the market, or market influence, the predicted stock price stayed within the error bounds with an average of 3 days. By including the current state of the market with the Dow Jones Industrial Average, the predictions stayed within the error bounds 2.4 days longer on average than without taking into account market influence. The next model, the savitzky golay, was not found to have any significant impact on predictions. Overall I learned a lot about the ways we can use math to make predictions. In the case of this project, these models were used to predict the stock market. However, these methods can be used to predict a lot of different phenomena.

Bibliography

Wikimedia Foundation. (2022, June 23). *Savitzky–Golay Filter*. Wikipedia. Retrieved August 1, 2022, from https://en.wikipedia.org/wiki/Savitzky%E2%80%93Golay_filter

Wikipedia contributors. (2022, July 22). *Fourier series*. Wikipedia. https://en.wikipedia.org/wiki/Fourier_series#:%7E:text=A%20Fourier%20series%20(%2F%CB%88f,be%20determined%20using%20harmonic%20analysis.

Fourier Analysis. (2022, May 19). Investopedia. https://www.investopedia.com/terms/f/fourieranalysis.asp#:%7E:text=Numerous%20studies%20have%20explored%20Fourier,as%20most%20similar%20strategies%20are.

A Simple Overview of Quantitative Analysis. (n.d.). Investopedia. Retrieved August 8, 2022, from https://www.investopedia.com/articles/investing/041114/simple-overview-quantitative-analysis.asp

All Python Functions

#All Libraries and import statements

```
import re

import json

import csv

from io import StringIO

from bs4 import BeautifulSoup

import requests

import pandas

import matplotlib.pyplot as plt

from pylab import *

import numpy as np

from yahoo_fin import stock_info as si

import numpy

import os

import time

import datetime

import math

import statsmodels.api as sm

from statsmodels.graphics import tsaplots

from shapely.geometry import LineString

import seaborn

import scipy

import pyttsx3

from symfit import parameters, variables, sin, cos, Fit
```

```
#Extract yahoo stock data. This is how we get all our data.
Create a table with the price of each stock from sp500, nasdaq,
and dow jones
dfsp500 = pandas.DataFrame(si.tickers_sp500())
dfnasdaq = pandas.DataFrame(si.tickers_nasdaq())
dfdow = pandas.DataFrame(si.tickers_dow())
dfother = pandas.DataFrame(si.tickers_other())

setsp500 = set(x for x in dfsp500[0].values.tolist())
setspnasdaq = set(x for x in dfnasdaq[0].values.tolist())
setdow = set(x for x in dfdow[0].values.tolist())
setother = set(x for x in dfother[0].values.tolist())

setAllStocks = setsp500.union(setspnasdaq, setdow, setother)
```

```
#determine if the ticker stock symbol entered is a valid stock.
Is the stock present in the 4 markets?
def valid_Stock():
   global stock

   stock = input("What stock do you want to analyze?")

   for val in setAllStocks:
       validStock = False
       if (stock == val):
           validStock = True
           break

   if validStock == False:
       print("Not a valid stock try again")
       clearConsole()
       valid_Stock()
   else:
       print("Is a valid stock")
       clearConsole()
```

```
#This function takes in a period, for example, from 5-12-2020 to
5-12-2022, as well as a ticker symbol. It then returns all
prices for that specific stock that you entered in the specific
time frame you entered
def stockAnalyze():
   #asks use to input period to view stock history
   global period1
   global period2
   print("Let's enter in period1")
   time.sleep(1)
   year1 = int(input("Enter in Year(Example: 2021):"))
   month1 = int(input("Enter in month(Example: 12):"))
   day1 = int(input("Enter in day(Example: 1):"))
   period1 = int(time.mktime(datetime.datetime(year1, month1,
day1).timetuple()))
   clearConsole()
   print("Let's enter in period2")
   time.sleep(1)
   year2 = int(input("Enter in Year(Example: 2021):"))
   month2 = int(input("Enter in month(Example: 12):"))
   day2 = int(input("Enter in day(Example: 1):"))
   period2 = int(int(time.mktime(datetime.datetime(year2, month2,
day2).timetuple())))
   clearConsole()
   interval = input("Let's enter the interval(Example: 1wk for 1 week, 10d for
10 days, 4m for 4 months ")
   clearConsole()
```

```
    url_history =
f'https://query1.finance.yahoo.com/v7/finance/download/{stock}?period1={period
1}&period2={period2}&interval={interval}&events=history&includeAdjustedClose=t
rue'
    df = pandas.read_csv(url_history)
    period1 = datetime.datetime(year1, month1, day1).timetuple()
    period2 = datetime.datetime(year2, month2, day2).timetuple()
    return df
```

```
#When looking at the stock prices, we want the closing prices.
All this function does is find the closing prices of the stock
you want. The input to this function is a table of the stock
prices in that specific time frame you entered in the function
before
def get_closing_prices(dataframe):
   closing_prices = np.array([])

   for index, row in dataframe.iterrows():
       closing_prices = np.append(closing_prices, row['Close'])

   return closing_prices

def get_dates(dataframe):
   dates = np.array([])

   for index, row in dataframe.iterrows():
       dates = np.append(dates, row['Date'])

   return dates
```

```
#This function just finds the autocorrelations. Look back to
page 15.
def find_auto_corr(dataframe):
   closing_prices = get_closing_prices(dataframe)

   #finds autocorrelations by finding correlations between stock
and stock-1 etvc
   autocorrelation = []
   for shift in range(1, len(closing_prices)):
       if np.std(closing_prices[:-shift]) == 0 or
np.std(closing_prices[shift:]) == 0:
           #autocorrelation.append(nan)
           continue
       else:
           correlation = np.corrcoef(closing_prices[:-shift],
closing_prices[shift:])[0, 1]
           autocorrelation.append(correlation)


   autocorrelation.insert(0, autocorrelation.pop())
   return autocorrelation
#graphs autocorrelations
def plot_auto_corr(array):
   # Graphs auto correlations
   plt.plot(array)
   plt.axhline(y=0)
   plt.xlabel("Days in the past")
   plt.ylabel("Correlation coefficient")
```

```
    plt.title(stock + " autocorrelation graph")
    plt.show()
#This function finds the relative period. This is the period when the
autocorrelation becomes 0
def find_rel_period(rel_day, closing_prices, dates):
    rel_days_in_past = []
    #rel_days_in_past = np.array([])

    for x in (range(len(dates) - rel_day, len(dates))):
        rel_days_in_past.append(int(x))
        #np.insert(rel_days_in_past, int(x))

    rel_dates = []
    rel_closing_price = []

    for index, elem in enumerate(dates):
        if index >= len(dates) - rel_day:
            rel_dates.append(elem)



    for index, elem in enumerate(closing_prices):
        if index >= len(closing_prices) - rel_day:
            rel_closing_price.append(elem)
```

```
#This Function finds the difference between the trendline and
the actual stock prices for a given stock
def find_residuals(rel_day, rel_days_in_past, rel_dates, rel_closing_price):
   m, b = np.polyfit(rel_days_in_past, rel_closing_price, 1)

   predicted = []
   residuals =[]
   #for price, x in zip( rel_closing_price, range(len(rel_closing_price))):
   predicted = m*np.array(rel_days_in_past)+ b


   for predicted, actual  in zip(predicted, rel_closing_price):
       residual = actual - predicted
       residuals.append(residual)

   return residuals
```

```
#Graphs the fourier series
def plot_fourier(residuals, rel_days_in_past, rel_closing_price):
   m, b = np.polyfit(rel_days_in_past, rel_closing_price, 1)
   n = len(residuals)
   COMPONENTS = [10]

   for c in COMPONENTS:
       colors = np.linspace(start=100, stop=255, num=c)

       for i in range(c):
           Y = np.fft.fft(residuals)
           numpy.put(Y, range(0, i), 0.0)
           np.put(Y, range(i + 1, n), 0.0)
           ifft = np.fft.ifft(Y)

       #plt.plot(rel_days_in_past, residuals, label="Original dataset")

       print(Y)
       plt.plot(rel_days_in_past, ifft,  alpha=.70)
       plt.plot(rel_days_in_past, residuals)

       plt.title("First {c} fourier components".format(c=c) + " for " + stock)
       plt.xlabel("Relevent period")
       plt.ylabel("Residuals")
       plt.grid(linestyle='dashed')
       plt.legend()
       plt.show()
```

```
#Combines the fourier series with the linear model. All this
does is combines the y=mx+b that graphs the trend line with the
equation describing the semi periodic function defined by the
fourier series
def plot_fourier_plus_linear(residuals, rel_days_in_past, rel_closing_price):
   m, b = np.polyfit(rel_days_in_past, rel_closing_price, 1)



   n = len(residuals)
   COMPONENTS = [6]


   for c in COMPONENTS:
       colors = np.linspace(start=100, stop=255, num=c)


       for i in range(c):
           Y = np.fft.fft(residuals)


           np.put(Y, range(i + 1, n), 0.0)
           ifft = np.fft.ifft(Y)



       fourier_plus_linear = ifft + (m * np.array(rel_days_in_past) + b)
       return np.real(fourier_plus_linear)
```

First version provided by Professor Mayer Humi
Matlab code

```
load('DJI.csv');%load the data file
prclose=DJI(:,9);%closing price of the stock in 5th col
prclose=prclose/253.13
[m,n]=size(prclose)% find out the number of data points
xx=linspace(1,m,m);%creat an array 1,2,...m
plot(xx,prclose);%plot the closing price for the period
pause
autocorr(prclose,m-1)% find the autocorrelation plot and find
its first 0.
pause
nrelp=121; % number of days in the relevant period
%the relevant data period is at the bottom of the data file
for i=1:nrelp
RV(i)=prclose(m-nrelp+i);%stock price during the relevant period
end
[m1,n1]=size(RV)
pause
yy=linspace(1,n1,n1);
p=polyfit(transpose(yy),RV,1);%find the equation for the trend
line
for i=1:n1+20
trend(i)=p(1)*i+p(2);%create the trend line
end
plot(yy,RV)
hold
plot(yy,trend(1:n1))
hold
pause
prdiff=RV-trend(1:n1);%difference between the actual price and
the trend
f=fit(yy',prdiff','fourier3')%fit fuorier exapnsion to the
difference
%These are the coefficients
for i=1:n1+20
ff(i)=f(i);%store the fourier value in ff
end
plot(yy,ff(1:n1))
hold
```

```
plot(yy,prdiff)
hold
title('price-trend and fourier 3 fit')
pause
%now find what is the residual between price and the sum of
% trend+fourier 3 fit
err=prdiff-ff(1:n1);
plot(yy,err)
pause
%find the mean of the abslute error
aerr=0;
for i=1:n1
aerr=aerr+abs(err(i));
end
aerr=aerr/n1;
%plot the trend +fourier with error bounds
for i=1:n1+20
model(i)=trend(i)+ff(i);%the model values
upr(i)=trend(i)+ff(i)+aerr;%upper bound
lpr(i)=trend(i)+ff(i)-aerr;%lower bound
end
plot(yy,model(1:n1),'r')
hold
plot(yy,upr(1:n1),'b')
plot(yy,lpr(1:n1),'b')
plot(yy,RV,'k')
title('Price vs. fit and error bounds')
legend('fit','upper bound', 'lower bound','actual price')
xlabel('Days in Relevent Period')
ylabel('Price')
hold
pause
%No continue the model 20 days into the future
zz=linspace(1,n1+20,n1+20);
yy2=linspace(1,n1+9,n1+9);%curreny future consis of 9days
z(1)=121;
z(2)=121;
y5(1)=32;
y5(2)=48;

plot(zz,model,'r')
```

```
hold
plot(zz,upr,'b')
plot(zz,lpr,'b')
plot(yy,RV,'k')
plot(z,y5,'m')
title('Price vs. fit and error bounds+20day')
legend('fit','upper bound', 'lower bound','actual price')
xlabel('Days in Relevent Period')
ylabel('Price')
hold
% include the future data in the RV
RV(n1+1)=1;
RV(n1+2)=0.94025615761;
RV(n1+3)=0.9869151126;
RV(n1+4)=0.9869151126;
RV(n1+5)=0.99223538781;
RV(n1+6)=0.9989376894;
RV(n1+7)=0.98162040261;
RV(n1+8)=0.96607343297;
RV(n1+9)=0.96030892425;
plot(zz,model,'r')
hold
plot(zz,upr,'b')
plot(zz,lpr,'b')
plot(yy2,RV,'k')
plot(z,y5,'m')
title('Price vs. fit and error bounds+20day for AGYS')
legend('fit','upper bound', 'lower bound','actual price')
xlabel('Days in Relevent Period+Future')
ylabel('Price')
hold
rv2= RV             %for stock i did rv=RV
rv2 = rv2(1:166)
A= corrcoef(rv1,rv2)
A= A(1,2)
p1 = A* DJI(size(rv2)+1)+(1-A)*stock_price(size(rv1)+1)
p2 = A* DJI(size(rv2)+2)+(1-A)*stock_price(size(rv1)+2)
p3 = A* DJI(size(rv2)+3)+(1-A)*stock_price(size(rv1)+3)
p4 = A* DJI(size(rv2)+4)+(1-A)*stock_price(size(rv1)+4)
p5 = A* DJI(size(rv2)+5)+(1-A)*stock_price(size(rv1)+5)
p6 = A* DJI(size(rv2)+6)+(1-A)*stock_price(size(rv1)+6)
```

```
p7 = A* DJI(size(rv2)+7)+(1-A)*stock_price(size(rv1)+7)
```