Predicting the Price of a Stock

An Interactive Qualifying Project Report

submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Andrew P. Murdza

Date: August 17, 2018

Project Advisor:

Mayer Humi, PhD

Abstract

Many have tried to master the inner workings of the American stock market to reap great profits. In this project, we modeled stock prices to make short-term forecasts. Another model was developed to measure stock volatility. This model can be used to estimate the risk of model predictions for different stocks. We hope that this model for stock prices and volatility index will help investors make profitable business decisions.

Contents

Executive Summary

This Interactive Qualifying Project (IQP) focused on modeling stock prices using signal analysis and measuring stock volatility with Lyapunov exponents. The goals of this project were to create a model to predict short term stock prices and a way to determine which stocks the model would predict most accurately. A more accurate stock price model would help investors choose which stocks to invest in and when to sell their stocks to achieve the greatest profit. A way to accurately measure stock volatility would help investors avoid high-risk stocks that could lead to significant losses in their portfolios.

During the first week of the project, 10 software stocks were selected for model testing. During the next few weeks, a prototype model for stock prices was implemented in MATLAB. Moving averages were later introduced to decrease the noise in the model input data. The use of moving averages decreased the width of the prototype model prediction band by 40.4%, on average, for the 20 stocks tested. To account for factors which affected the entire stock subsector, the prototype model was replaced with a weighted average of the models for the stock prices and the S&P 500 index. On average, the index model accurately predicted the stock price for 37.5% more days than the prototype model. The last few weeks were devoted to measuring stock volatility with Lyapunov exponents. The TISEAN package was used to compute the largest Lyapunov exponent of each stock over various time intervals and embedding dimensions. It was hypothesized that stocks with larger Lyapunov exponents would be more volatile. However, we

did not find a high correlation between a stock's maximal Lyapunov exponent and its volatility.

Introduction

In recent times, it has been difficult for casual investors to compete with professionals. While big investors have access to expensive analysis packages, advanced stock data, and insider information, smaller investors rely on only general recommendations and their gut feeling. The goal of this Interactive Qualifying Project is to develop tools that casual investors can use to select profitable, low-risk stocks. The two tools that this project focused on were a model to short term predict stock prices, and new a measure of stock volatility. Accurate predictions of future stock prices can enable inexperienced investors to make more profitable investment decisions than choices based on intuition and expert recommendations alone. Small investors lack the capital to suffer major losses. For this reason, an effective indicator of stock volatility which could identify high-risk stocks would be of great help to amateur investors.

The model used to predict short term stock prices was a further development of a formulated by a previous IQP team. The TISEAN package was then applied to quantify the risk level of a stock. Small investors could use the TISEAN package to identify low-risk stocks and then apply the model to decide which stocks to invest in and when to buy and sell them.

Research: Modeling Stock Prices

We will model stock prices using time series analysis. Time series, which include stock prices, are sets of data ordered from least recent to most recent. In the first part of this section, we present several tools from time series analysis and

describe how they can be applied to predict stock prices. In the second part of this section, we will use chaos theory to model the volatility of a stock. We start by introducing some notation.

Notation

Throughout this paper, we will represent a time series of N data points by the sequence $x_1, \dots, x_N$. We call each term in the sequence an observation. We denote the times at which the observations $x_1,\dots,x_N$ occur by $t_1, \dots, t_N$. Using this notation, we may write a time series in the form $(t_1, x_1), \dots, (t_N, x_N)$. We define the mean of the observations as

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i \,. \tag{1}$$

Similarly, we define the mean of the observation times by

$$\bar{t} = \frac{1}{N}\sum_{i=1}^{N} t_i \,. \tag{2}$$

As previously mentioned, we will plan to model stock prices to make future predictions. To construct these models, we will fit a curve to the closing prices of the stock for some number of days in the past. In the following section we discuss how we will use the autocorrelation coefficient to determine the ideal number of days to use to generate our model.

The Autocorrelation Coefficient

The autocorrelation coefficient is a statistic that measures the similarity of the current value of a time series to previous observations in the series. The autocorrelation coefficient is computed at a specified number of lags. The

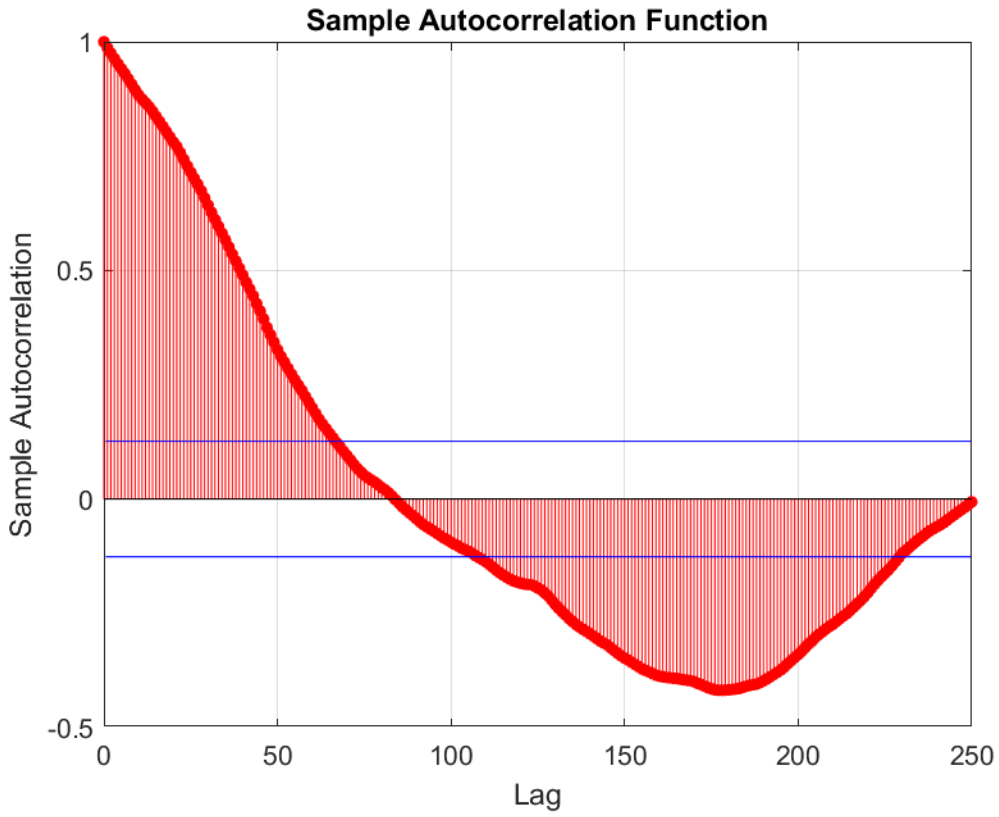
autocorrelation at lag k quantifies the strength and direction of the relationship between the first k observations and the current observation. It is the correlation between the time series and the time series after it is lagged k times. It is calculated with the equation

$$r_k = \frac{\sum_{i=1}^{N-k}(X_i - \bar{X})(X_{i+k} - \bar{X})}{\sum_{i=1}^{N}(X_i - \bar{X})^2},$$

where $\bar{X}$ is given by (1).

If the autocorrelation at lags $1, 2, \ldots, k$ are all positive, then there is some relationship between the first k observations and the current observation. From this fact we obtain a way to determine the number of points we should use to model our time series of stock prices. To find the optimal number of stock prices to be used in our models, we first compute the autocorrelation of the stock prices at lags 1 to the number of business days in the past year, 251. We then find the largest number k such that the autocorrelation at lags 1 to k are all positive. The value of k represents the maximum number of business days from the present during which all the stock price is relevant to the current stock price. Therefore, we use the k most recent stock prices for our models.

Graph of Autocorrelation for the Paycom (PAYC) Stock



The above figure displays the autocorrelation of the Paycom stock prices for lags 1-251. Note that lag 1 corresponds to 6/1/18 and lag 251 corresponds to the date 251 business days before 6/1/18, which is 6/2/17. The lag where the autocorrelation becomes negative for the first time, lag 84, corresponds to the date 84 business days before 6/1/18, 1/31/18. Stock data between 1/31/18 and 6/1/18 was used to generate the models for the Paycom stock price.

We will next present our first approach to modeling stock prices: the trend line.

The Trend Line

The trend line fit to a set of data is the line whose slope and y-intercept are chosen to minimize the average squared distance between the line and the data. It has the general form

$$X_L(t) = d_0 + d_1 t.$$

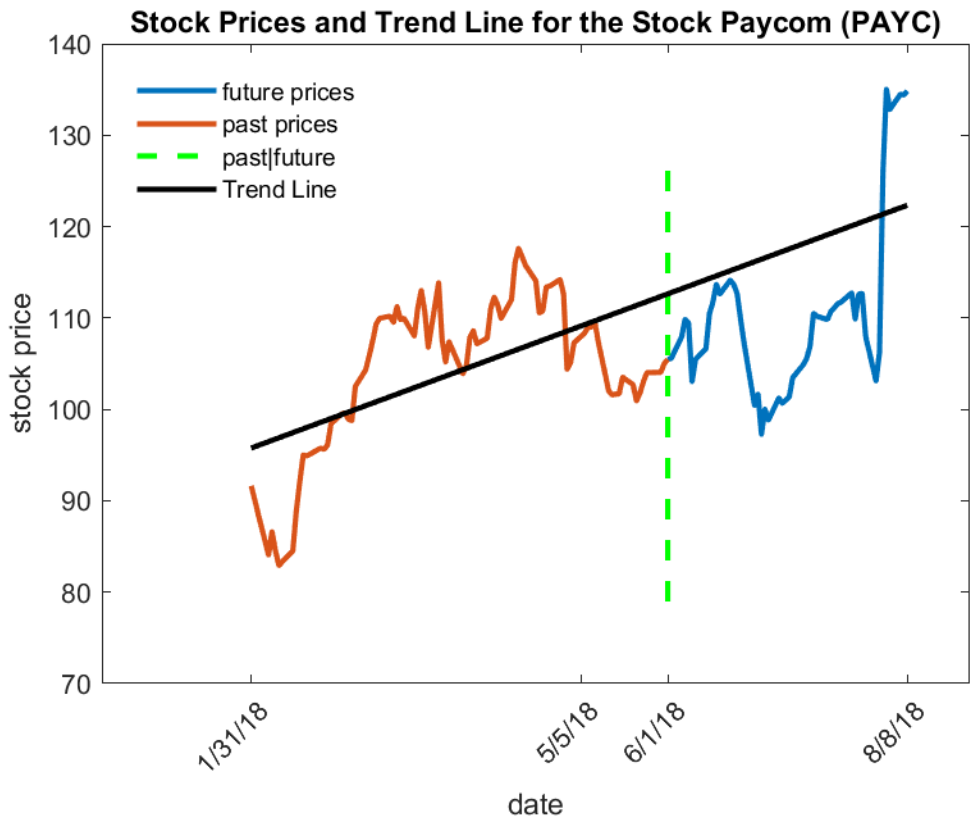
The slope of the trend line, d_1, is calculated as

$$d_1 = \frac{\sum_{i=1}^{N}[(X_i - \bar{X})(t_i - \bar{t})]}{\sum_{i=1}^{N}(t_i - \bar{t})^2},$$

where $\bar{X}$ and $\bar{t}$ are given by (1)-(2).

The y-intercept of the line, d_0, is determined by

$$d_0 = \bar{X} - c_1\bar{t}.$$



The above figure displays the (red) past and (blue) future stock prices of Paycom and the (black) trend line fit to the past prices. A green dashed line separates the past from the future.

A trend line generated from pairs of dates and stock prices gives a rough estimate of the future direction of a stock. We want more accurate predictions than

offered by the trend line alone. To decrease the model error, we introduce Fourier series.

Fourier Series

A Fourier series is a sum of a sine and cosine functions. They are useful in modeling period and oscillatory data. An n^{th} order Fourier series has the form

$$X_f(t) = \frac{1}{2}a_0 + \sum_{k=1}^{n} a_k \cos\left(\frac{2\pi nt}{T}\right) + \sum_{k=1}^{n} b_k \sin\left(\frac{2\pi nt}{T}\right).$$

When an n^{th} order Fourier series is fit to a time series $(t_1, e_1), \dots, (t_N, e_N)$, the coefficients are computed with the equations

$$a_0 = \frac{1}{N}\sum_{j=1}^{N} e_i,$$

$$a_k = \sum_{i=1}^{N} X_i \cos\left(\frac{2\pi kt_i}{T}\right) e_i,$$

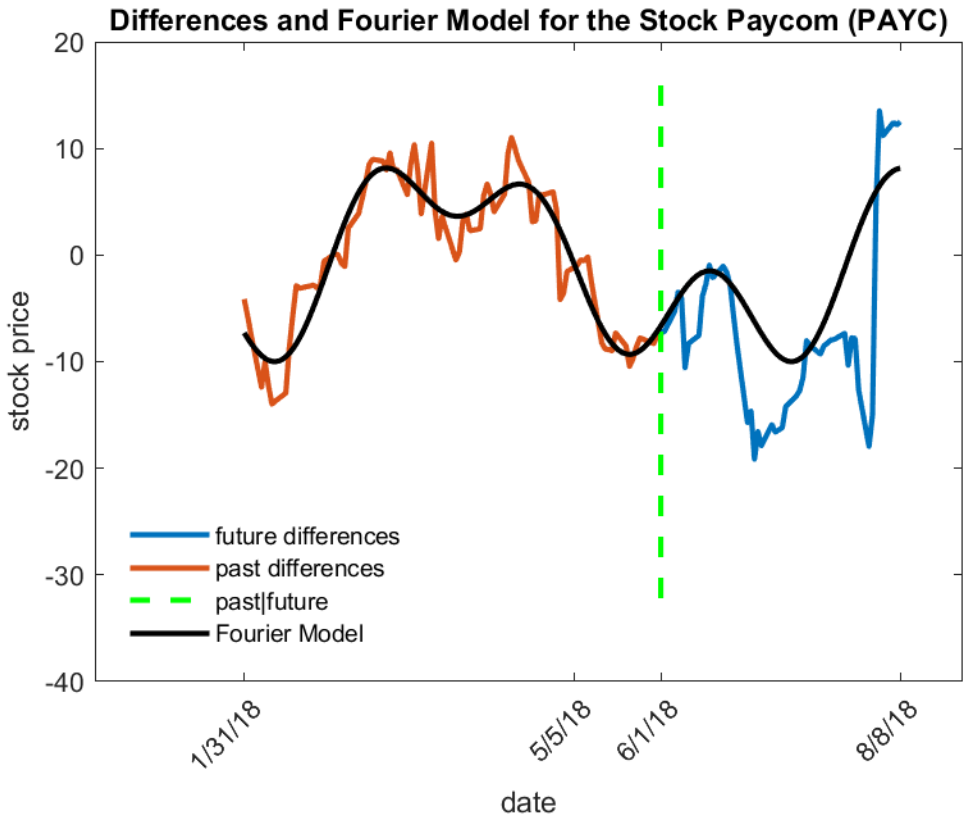
$$b_k = \sum_{i=1}^{N} X_i \sin\left(\frac{2\pi kt_i}{T}\right) e_i$$

The variable T is the period of the Fourier Series. We set it equal to the time spanned by all the stock prices, $t_N - t_1$. The ideal number of terms of the Fourier series we use depends on amount of data we generate it from. We are now ready to describe how we can improve our trend line model with Fourier series.

Using Fourier Series to Improve the Trend Line Model

In this section we will incorporate Fourier series into the trend line to increase the accuracy of the model predictions. We compute the difference between the

actual value of various stocks and the trend line, which we call the trend line residuals. We see that the trend line residuals oscillate with time. It follows that we should model these residuals with Fourier series. If there at least 60 data points, we use a third order Fourier series to model the differences. If 40 or fewer points are used, then we apply a second order Fourier series. If we have between 40 and 60 data points, then we fit both a second order and third order Fourier series to the differences and choose the series with the higher R^2 value for our model.



The above figure displays the past (red) and future (blue) trend line residuals, and the (black) Fourier model fit to the past differences. A green dashed line separates the past from the future.

Combining the trend line with the Fourier Series produces an overall model of the form

$$X_p(t) = \frac{1}{2}a_0 + d_0 + d_1 t \ + \sum_{k=1}^{n} a_k \cos\left(\frac{2\pi nt}{T}\right) + \sum_{k=1}^{n} b_k \sin\left(\frac{2\pi nt}{T}\right)$$

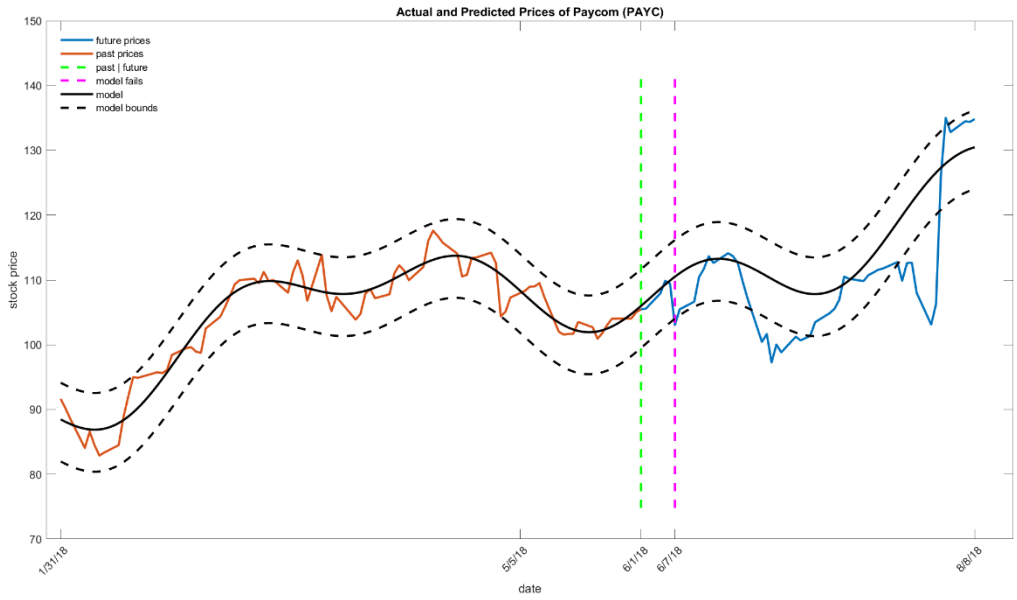
We refer to this overall model as the prototype model. We call the errors in the predictions of the prototype data noise, which represent fluctuations in prices due to factors we haven't accounted for in our model. The function $X_p(t)$ gives a single value for the price of the stock at a given time t, which we call a point estimate. Although a point estimate for the price of a stock at a given time is useful, we also want a range of values in which the stock price is mostly likely to be found. In the next section we will develop lower and upper bounds for stock price at a given time. We will also describe how to measure the precision of our models with the margin of error.

The Margin of Error

After we use our stock price data to construct our prototype model, $X_p(t)$, we will add upper and lower bounds to the model. We set the model upper bound equal to the sum of the prototype model and the maximum noise. Similarly, we add the minimum noise to the prototype model to obtain the model lower bound. This ensures that the entire time series used to generate the model is between the upper and lower bounds. We call the region between the upper and lower bounds the prediction band of the model.

The precision of the model predictions is measured by the model's margin of error. A low margin corresponds to a narrow prediction band and a precise

model. The margin of error can be computed as half of the difference of the maximum and minimum noise or as the maximum of the absolute value of the noise. We use the former method when we compute the margin of error, although we would obtain similar results if we used the latter.



The above graph includes the (red) past and (blue) future stock prices of Paycom, and the (black) prototype model fit to the past prices. A green dashed line separates the past from the future and magenta dashed line indicates where the model fails.

We next summarize how we determine the prototype model from a set of data.

A Summary of the Algorithm Used to Construct the Prototype Model

1. Compute the autocorrelation of the stock prices from the past year for lags 1 to 251.
2. Find the largest number k such that the autocorrelations corresponding to lags 1 to k are all positive.

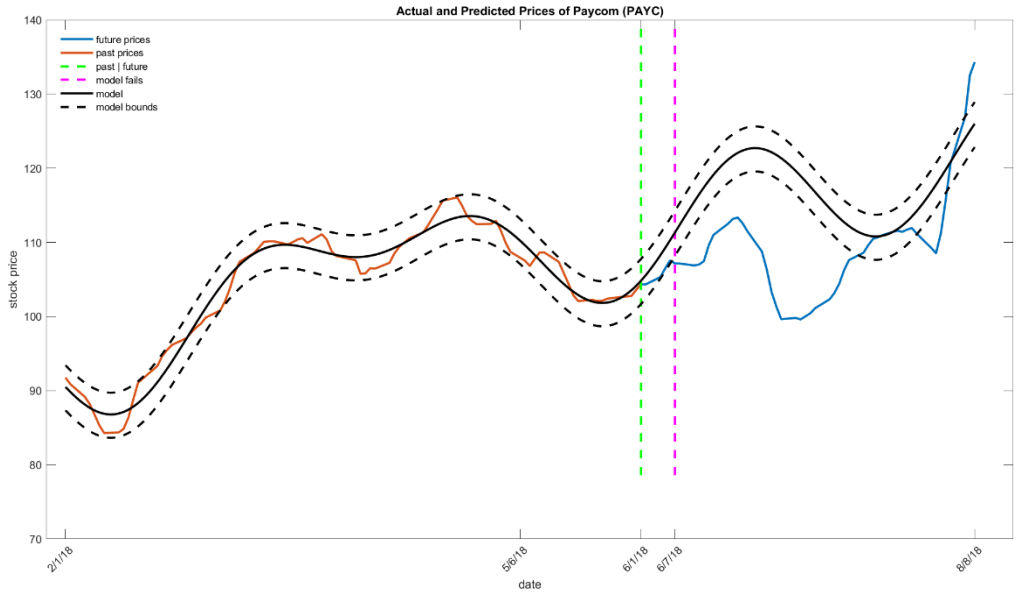
3. Fit a trend line to the k most recent stock prices.
4. Compute the difference between the k most recent stock prices and the trend line predictions.
5. Fit a Fourier Series to the differences.
6. Compute the predictions of the prototype model by summing the predictions of the trend line and the Fourier series
7. Calculate the data noise by subtracting the model predictions from the k most recent stock prices.
8. Compute the model upper bound by adding the maximum noise to the model predictions.
9. Calculate the model lower bound by adding the minimum noise to the model predictions.

Improvements on the Prototype Model

Two measures of the effectiveness of a model are its accuracy and its precision. A model is accurate if its prediction band contains the stock price for many consecutive days in the future. A precise model has a narrow prediction band. We want a model with correctly predicts the stock price for large time span and has a low margin of error.

Increasing Model Precision with Moving Averages

One way to reduce the margin of error of a model is to reduce the data noise. This can be accomplished with moving averages. A k term moving average replaces each data point with the average of the data point with the previous $k - 1$ data points. Because the variation between sample averages is less than the variation between individual data points, we can use moving averages to reduce the size of the data noise.



The above graph includes the (red) past and (blue) future stock price moving averages of Paycom, and the (black) index model fit to the past moving averages. A green dashed line separates the past from the future and magenta dashed line indicates where the model fails. We see from the plot that using moving stock averages in place of raw stock prices significantly decreases the size of the prediction band.

Increasing Model Accuracy Using Correlation with Market Indices

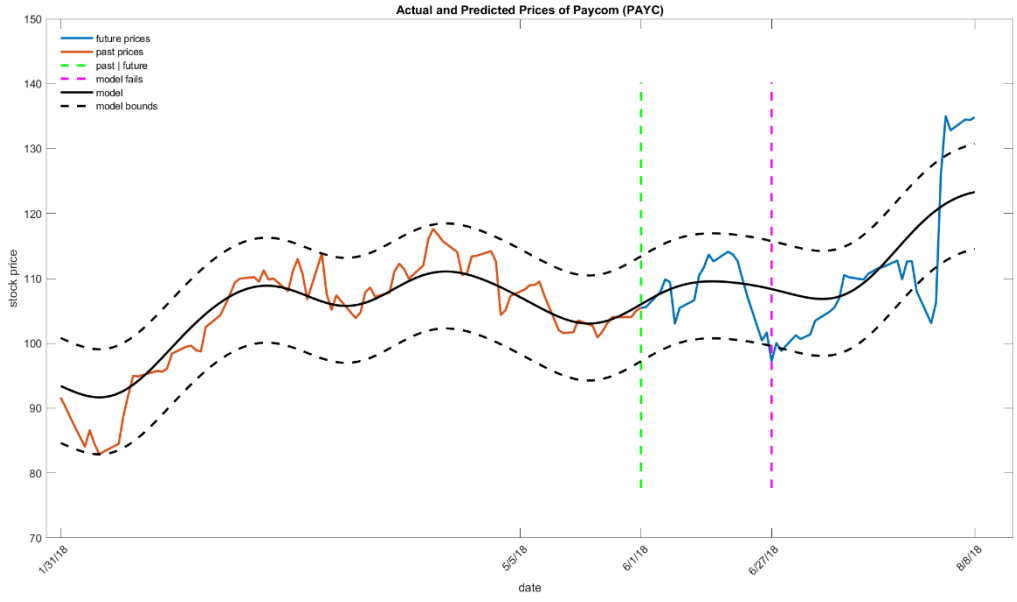
To increase the accuracy of our model, we could account for factors which affect the stock price. Stock market indices, such as the S&P 500, give rough representations of the trend of the overall market. By incorporating the S&P 500 index into the prototype model, we can account for general factors, such as unemployment and interest rates, which affect the stock prices of the whole market. Our new model will be a weighted average of the prototype model for the stock price and a prototype model for the S&P 500 index.

To prevent the S&P 500 value from overwhelming the stock value, both are normalized by dividing them by their value at the present. The improved model calculates the normalized stock price as a weighted average of the prototype model predictions of the normalized stock price, $\tilde{X}_{stock}$, and the normalized index model, $\tilde{X}_{index}$. This weighted average is given by

$$\tilde{X} = \alpha\tilde{X}_{index} + (1-\alpha)\tilde{X}_{stock},$$

where α is the correlation between the market index and the stock prices. This formula gives the index model a higher weight if it more closely follows the stock price and a negative weight if its trend is opposite to the stock trend. We obtain the index model X_{index} by multiplying the normalized index model by the current stock price

$$X_{index} = \tilde{X}(t)X_{pres}$$



The above graph includes the (red) past and (blue) future stock prices of Paycom, and the (black) index model fit to the past prices. A green dashed line separates

the past from the future and magenta dashed line indicates where the model fails. We see from the plot that the index model correctly predicts the stock price for a substantially greater number of days than the prototype model.

A Summary of the Algorithm Used to Construct the Index Model

1. Normalize the stock prices
2. Determine the prototype model for the normalized stock prices
3. Normalize the S&P 500 index
4. Fit the prototype model to the S&P 500 index
5. Compute the correlation between the stock prices and the S&P 500 index
6. Construct the normalized index model with a weighted average of the stock price and S&P 500 prototype models.
7. Multiply the normalized index model by the current stock price to obtain the index model.
8. Calculate the noise by subtracting the index model predictions from the actual stock prices
9. Compute the model upper bound by adding the maximum noise to the model predictions.
10. Calculate the model lower bound by adding the minimum noise to the model predictions.

Research: Measuring Stock Volatility

Dynamical Systems and State Space

Chaos theory based on dynamical systems. A dynamical system is a model which predicts how a set of quantities evolve over time. We call each possible

combination of values of the quantities a state. We refer to the set of all possible states as the state space of the dynamical system. The dimension of the state space is defined as the number of quantities described by the space. We assume that the system is deterministic, which means that if all the quantities of an initial state are known exactly, it is possible to determine the state of the system at any future time. We will now explain why it is difficult to predict the future state of some deterministic systems.

Chaos and Lyapunov Exponents

The evolution of some dynamical systems is very sensitive to the initial state of those systems. Chaotic dynamical systems have the property that very close starting states will diverge exponentially for some short time. The rate of exponential divergence is measured with Lyapunov exponents.

If two initial states are separated by a small distance s_0, then for a small time interval the separation will evolve according to the equation

$$s(t) \approx s_0 e^{\lambda_1 t}. \tag{3}$$

The value λ_1 is defined as the maximal Lyapunov exponent of the system. We see that chaotic systems have a positive Lyapunov exponent and the sensitivity of a system to initial conditions increases with λ_1. Systems with large maximal

Lyapunov exponents are difficult to predict because small errors in the current state of the system are amplified over time. This fact suggests that Lyapunov exponents can be used to measure the volatility of stock prices.

The Lyapunov Spectrum

As we will see later, a dynamical system with an n dimensional phase space has n Lyapunov exponents. The set of the n Lyapunov exponents Each Lyapunov exponent corresponds to "stretching" or "compressing" the distance between two initial states in a particular direction that changes over time. We only concern ourselves with the maximal Lyapunov exponent, since the "stretching" in its direction dominates over the changes in the distance in other directions.

Embedding a Time Series

We define the attractor of a dynamical as set of states which neighboring states converge to as time passes. It is important when studying the long-term behavior of the system. We want to reconstruct the n-dimensional attractor from our one-dimensional time series data. This is accomplished by embedding the time series into a higher dimensional space, called the embedded space. The dimension of the embedded space is called the embedding dimension. We construct a set of embedded vectors $\boldsymbol{x}_1, \dots, \boldsymbol{x}_M$ from time delayed values of the time series $x_1, \dots, x_N$. We compute $\boldsymbol{x}_i$ as

$$\boldsymbol{x}_i = \left(x_{i-(m-1)\tau}, x_{i-(m-2)\tau}, \dots, x_i\right),$$

where τ is the time delay and m is the embedding dimension. We choose τ so that there is no relationship between $x_{n-\tau}$ and x_n. We set τ equal to the number of lags before the autocorrelation of the time series decreases below $\frac{1}{e}$. If the embedding dimension m is too low, then the attractor will not be accurately reconstructed by the embedding series. If we use an embedding which is too high, then there will be more Lyapunov exponents in the embedded system than the original dynamical system. If one of the extra "spurious" Lyapunov exponents is greater than the maximal Lyapunov exponent of the original system, then the maximal Lyapunov exponent of the embedded space will be higher than it should be. We will describe one method of determining the ideal value of m in the next section.

The Method of False Nearest Neighbors

If we use an embedding dimension which is too low, the embedded space is not topologically equivalent to the attractor. Some points in such an embedded space, called false neighbors, will appear to be closer than they are on the attractor. False neighbors should be closer in the low embedding dimension than they are in a higher embedding dimension which more closely resembles the attractor. We denote the i^{th} m-dimensional state vector as Y_i and the i^{th} $m+1$-

dimensional state vector as Y_i. We define the nearest neighbor of X_i, X_{j_i}, as the state vector which is the closest distance to X_i. We introduce Y_{j_i} similarly. For convenience, we set $R_i(m)$ and $R_i(m+1)$ equal to the distances between X_i and X_{j_i} and Y_i and Y_{j_i} respectively.

We say that the i^{th} nearest neighbor is false if

$$\sqrt{\frac{[R_i(m+1)]^2 - [R_i(m)]^2}{[R_i(m)]^2}}$$

rises above some tolerance D_{tol}, i.e.

$$\sqrt{\frac{[R_i(m+1)]^2 - [R_i(m)]^2}{[R_i(m)]^2}} < D_{tol}. \tag{4}$$

We see that (4) holds if $R_i(m+1)$ and $R_i(m)$ differ by an amount which is large relative to $R_i(m)$. This is consistent with our intuition of false nearest neighbors.

Noting that

$$X_i = \left(x_{i-(m-1)\tau}, x_{i-(m-2)\tau}, \dots, x_i\right)$$

and

$$X_{i_j} = \left(x_{i_j-(m-1)\tau}, x_{i_j-(m-2)\tau}, \dots, x_{i_j}\right)$$

we see that

$$[R_i(m)]^2 = \left\| X_i - X_{i_j} \right\|^2 = \sum_{k=0}^{m-1} \left(x_{i-k\tau} - x_{i_j-k\tau} \right)^2 \quad (5)$$

We similarly note that, because

$$Y_i = \left(x_{i-(m+1-1)\tau}, x_{i-(m-1)\tau}, x_{i-(m-2)\tau}, \dots, x_i \right)$$

and

$$Y_{i_j} = \left(x_{i-(m+1-1)\tau}, x_{i_j-(m-1)\tau}, x_{i_j-(m-2)\tau}, \dots, x_{i_j} \right),$$

$$[R_i(m+1)]^2 = \left\| Y_i - Y_{i_j} \right\|^2 = \sum_{k=0}^{m} \left(x_{i-k\tau} - x_{i_j-k\tau} \right)^2 \quad (6)$$

After substituting (5) and (6) into (4), we find that the i^{th} nearest neighbor is false if

$$D_{tol} < \sqrt{\frac{\sum_{k=0}^{m} \left(x_{i-k\tau} - x_{i_j-k\tau} \right)^2 - \sum_{k=0}^{m-1} \left(x_{i-k\tau} - x_{i_j-k\tau} \right)^2}{[R_i(m)]^2}}$$

$$= \sqrt{\frac{\left(x_{i-m\tau} - x_{i_j-m\tau} \right)^2}{[R_i(m)]^2}}$$

Simplifying, we see that the i^{th} nearest neighbor is false if

$$\frac{\left|x_{i-m\tau} - x_{i_j-m\tau}\right|}{\left\|X_i - X_{i_j}\right\|} > D_{tol}. \tag{7}$$

We also conclude that a nearest neighbor is false if they become too distant when the embedding dimension is increased. We estimate that the radius of the attractor, R_A, is approximately the standard deviation of the time series. If the nearest neighbors are separated by a distance greater than twice the radius of the attractor in a higher embedding dimension, then they are false, even if (7). We express this second criterion as

$$\frac{R_i(m+1)}{R_A} > 2 \tag{8}$$

If either (7) or (8) hold, we say that the nearest neighbor is false.

To determine the ideal embedding dimension m, we compute the percentage of nearest neighbors which are false for increasing m until it falls below some threshold. The lowest value of m for which the percentage of false neighbors is below the threshold is our estimate for the ideal embedding dimension.

Computing the Maximal Lyapunov Exponent from a Time Series

We now discuss how to compute maximal Lyapunov exponents using the algorithm developed by Rosenstein et al. To compute the maximal Lyapunov exponent, we first embed the time series into a set of state vectors $X_1, \ldots, X_M$.

Noting that (3) is valid when the two initial states are close together, we find the nearest viable neighbor to each state vector X_i. Two states are viable neighbors if they occur at times separated by more than one time delay. We add this viability requirement to avoid false nearest neighbors. We denote the nearest viable neighbor of X_i by X_{j_i}. To determine how rapidly X_i and its nearest neighbor, X_{j_i}, separate, we compute the distance between them after k time steps

$$d_i(k) = \left\| X_{i+k} - X_{j_i+k} \right\|.$$

To compute $y(t)$ after k time steps, we require the logarithm of $d_i(k)$:

$$y_i(k\Delta t) = \ln\left\| X_{i+k} - X_{j_i+K} \right\|$$

We take the average of the distances between each pair of nearest neighbors by summing the distance $d_i(k)$ and dividing by the total number of pairs:

$$y(k\Delta t) = \frac{1}{M-k} \sum_{i=1}^{M-k} y_i(k\Delta t)$$

Only the pairs of the first $M - k$ nearest neighbors are used because the there is no data for the remaining neighbors after k time steps. We compute $y(k\Delta t)$ for $k = 1, \ldots, K$, then fit a trend line to $\big(\Delta t, y(\Delta t)\big), \ldots, \big(K\Delta t, y(K\Delta t)\big)$. The slope of the trend line is our estimation to the maximal Lyapunov exponent of the

dynamical system. This process is often repeated for increasing values of K with the hope that the Lyapunov exponent estimate converges to some value.

Results: Stock Models

Ten stocks were chosen to compare the accuracy and precision of the models. For each stock, a prototype model, a prototype model using moving averages, an index model, and an index model using moving averages were generated. The models were tested with moving averages of between 5 and 15 terms, and the five-term average performed best overall. The end results presented here were generated with 5 term moving averages. The data set used for the stock models ranged from 6/02/17 to 6/01/17. The models were compared to "future values" between 6/02/18 and 8/9/18. The margin of error of each model and the number of business days the model correctly predicted were computed for each stock and organized in the below table:

Summary Table for Model Performance

Stock		Prototype Model		Prototype Mov. Avgs		Index Model		Index Mov. Avgs	
Name	Symbol	Days	% ME	Days	% ME	Days	% ME	Days	% ME
Interxion Holdings	INXN	5	3.5	7	2.67	8	6.38	48	6.64
58.com	WUBA	6	5.96	7	3.97	17	9.53	19	8.24
Progress Software	PRGS	14	8.58	12	3.57	13	11.07	16	10.42
Black Baud	BLKB	17	4.75	10	2.75	17	4.68	10	3.51
Commvalut	CVLT	7	2.27	9	1.49	8	2.19	9	1.64
Changyou	CYOU	6	4.14	6	3.64	6	4.97	7	4.67
Imperva	IMPV	8	3.76	10	2.72	8	3.96	8	3.47
Guidewire Software	GWRE	4	2.86	5	1.44	5	3.85	7	2.33
Paycom	PAYC	5	5.83	5	2.92	19	7.76	19	7.57
Talend	TLND	4	3.91	7	2.00	9	8.3	7	5.06
Mean		7.6	4.56	7.8	2.72	11	6.27	15	5.36
Standard Deviation		4.4	0.92	2.3	0.44	5.1	1.41	12.6	1.4

The table contains the margin of error and the number of business predicted corresponding to each stock and each model. The last two rows list the mean and standard deviation of the margin of error and the number of days predicted for each model

Introducing moving averages significantly decreased the average margin of error of the respective prototype models from 4.56% to 2.72% (a relative decrease of 40.4%). It did not appreciably change the accuracy of the prototype model. Moving averages increased the number of days predicted by the model from 11 to 15. However, this increase is primarily due to the outlier Interxion Holdings. The average number of days predicted for the index models and is 11.3 with moving averages and 11.7 without moving averages, which is not a meaningful change. However, adding moving averages to the index model decreases the mean margin of error from 6.27 to 5.36 (a 14.5% relative decrease).

Although incorporating correlation with the stock index to into the prototype model increased the average margin of error from 4.56% to 6.27% (a relative increase of 37.5%), it also increased the number of days predicted without moving averages from 7.6 to 11 (a relative increase of 44.7%). It increased the margin of error of the prototype model with moving averages from 2.72% to 5.36% (a 97% relative increase), but increased the number of days predicted from 7.8 to 11.7 (without the outlier), which corresponds to a relative increase of 50.0%.

We find that introducing moving averages substantially increases the model precision and does not affect its accuracy. Incorporating correlation with a market index increases the accuracy of the model but decreases its precision. According to these results, one should apply moving averages to the model to increase its precision. One should add market indices to the model if higher accuracy is worth lower precision.

Results: Measuring Volatility

To model stock volatility, we applied Lyapunov exponents in four different ways.

1. Lyapunov exponents of the raw stock prices
2. Lyapunov exponents of the stock prices after moving averages were applied

3. A weighted average of the Lyapunov exponent of each stock's prices and the exchange index, given by
$$\lambda_{index} = (1-\alpha)\lambda_{stock} + \alpha\lambda_{SP500}, \quad (6)$$
where α is the correlation between the stock prices and the S&P 500 index.
4. The Lyapunov exponents computed with (6), where λ_{stock} and λ_{SP500} were calculated with moving averages.

We also attempted to model stock volatility with the normalized standard deviation, given by

$$s = \frac{\sigma}{\bar{X}},$$

where σ and $\bar{X}$ are the standard deviation and mean of the stock prices used to generate the model. We used the normalized standard deviation in four ways:

1. The normalized standard deviation of the raw stock prices
2. The normalized standard deviation of the stock prices after moving averages were applied
3. A weighted average of the normalized standard deviation of each stock's prices and the exchange index, given by
$$s_{index} = (1-\alpha)s_{stock} + \alpha s_{SP500}, \quad (7)$$
where α is the correlation between the stock prices and the S&P 500 index.
4. The normalized standard deviation computed with (7), where s_{stock} and s_{SP500} were calculated with moving averages.

Correlation Between One-Year Lyapunov Exponents and Days Predicted

		One Year Lyapunov Exponent											
		Prototype			Moving Averages			Index			Index Averages		
		m=1	m=2	m=3	m=1	m=2	m=3	m=1	m=2	m=3	m=1	m=2	m=3
Stock Model	Prototype	-0.293	-0.391	-0.514	-0.269	-0.291	-0.431	-0.112	-0.186	-0.451	-0.065	-0.123	-0.397
	Moving Avgs	0.107	-0.16	-0.17	0.261	0.045	-0.102	0.114	0.133	-0.166	0.367	0.294	-0.034
	Index	-0.342	-0.304	-0.544	-0.353	-0.284	-0.475	-0.208	-0.133	-0.526	-0.187	-0.105	-0.463
	Index Avgs	-0.316	-0.178	-0.241	-0.411	-0.272	-0.272	-0.097	-0.201	-0.196	-0.241	-0.239	-0.24

The below table displays the correlation between each type of maximal Lyapunov exponent for embedding dimensions 1,2,3 and the days predicted by each stock model. The Lyapunov exponents were computed with one year of stock prices. The correlation was calculated with data from 20 stocks.

We see that none of the correlations are large enough to be meaningful. From this we realize that the Lyapunov exponents we used in this section do not accurately model stock volatility. We now check if the results are better when the Lyapunov exponents are computed from stock prices spanning two years with the below table.

Correlation Between Two-Year Lyapunov Exponents and Days Predicted

		Two Year Lyapunov Exponent											
		Prototype			Moving Averages			Index			Index Averages		
		m=1	m=2	m=3	m=1	m=2	m=3	m=1	m=2	m=3	m=1	m=2	m=3
Stock Model	Prototype	-0.407	-0.224	-0.012	-0.417	-0.157	-0.016	-0.302	-0.15	-0.046	-0.276	-0.098	-0.042
	Moving Avgs	-0.216	-0.032	0.102	0.003	0.148	0.216	-0.082	0.113	0.153	0.097	0.256	0.231
	Index	-0.346	-0.525	-0.396	-0.408	-0.454	-0.37	-0.212	-0.268	-0.215	-0.245	-0.187	-0.181
	Index Avgs	-0.35	-0.395	-0.274	-0.441	-0.413	-0.302	-0.301	-0.271	-0.207	-0.337	-0.262	-0.205

The below table displays the correlation between each type of maximal Lyapunov exponent for embedding dimensions 1,2,3 and the days predicted by each stock model. The Lyapunov exponents were computed with two years of stock prices. The correlation was calculated with data from 20 stocks.

We again find that there is not a meaningful relationship between Lyapunov exponents and stock volatility.

We finally compare the stock volatility to the normalized standard deviation of the stock prices throughout the past:

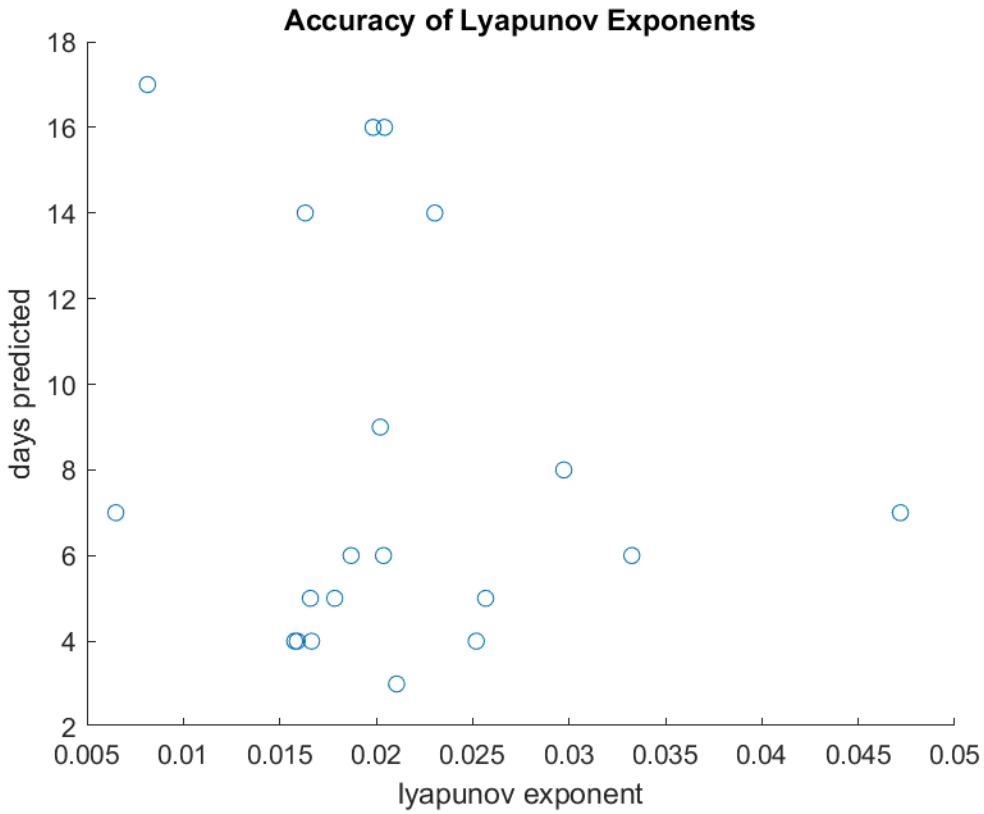
Correlation Between Normalized Standard Deviation

		Normalized standard Deviation			
		Prototype	Moving Avgs	Index	Index Avgs
Stock Model	Prototype	-0.109	-0.106	0.182	0.173
	Moving Avgs	-0.177	-0.065	0.29	0.368
	Index	0.074	0.134	0.459	0.503
	Index Avgs	-0.133	-0.12	-0.129	-0.119

This table presents the correlation between the number of days predicted by each model and each form of normalized standard deviation.

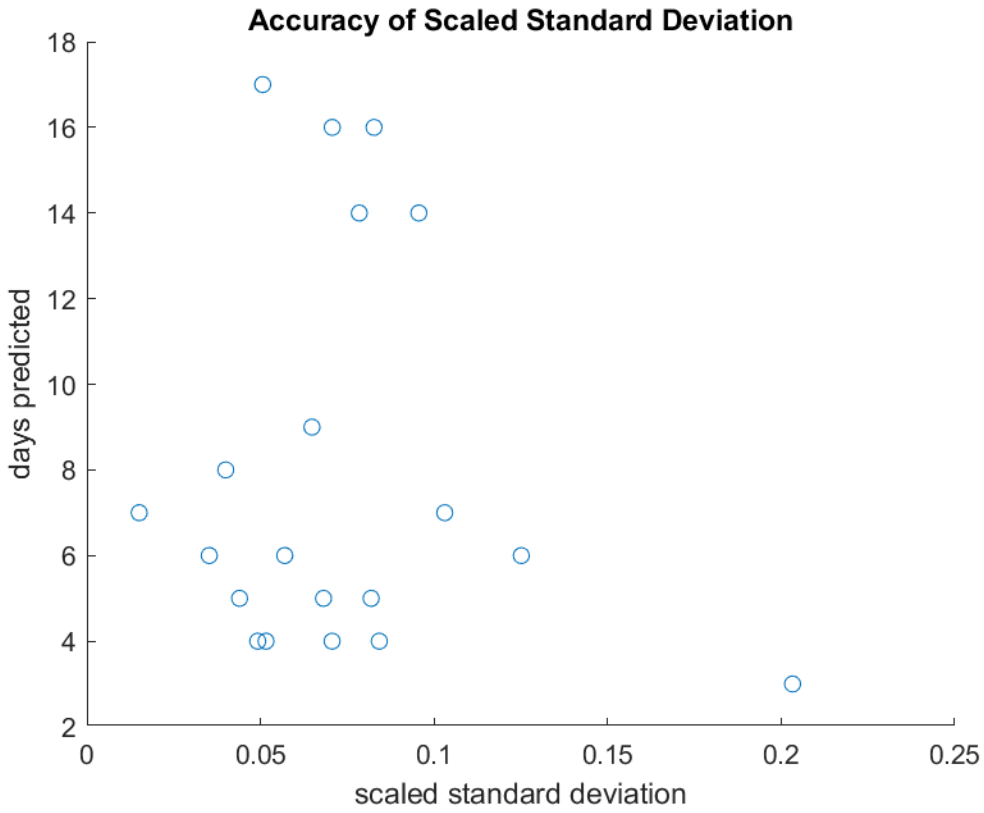
We conclude that these correlations are not large enough to suggest that there is a strong relationship between the normalized standard deviation of a stock's prices and the its volatility.

To visually represent the lack of a relationship between the Lyapunov exponents and the number of days predicted by model, we plot them in a scatterplot. The Lyapunov exponents of the raw stock data with embedding dimension 3 were graphed vs. the number of days the prototype model predicted correctly.



The scatterplot seems to show no association between the maximal Lyapunov exponents and the number of days the prototype model correctly predicts.

We also plot the number of days predicted by the prototype model vs. the normalized standard deviation of the raw stock prices:



We again see no pattern in the scatter plot. There seems to be an outlier with a scaled volatility of over 0.2. It corresponds to the stock Nutanix (NTNX).

Although our analysis suggests that there is not a strong relationship between Lyapunov exponents and stock volatility, there are two other variables which, by themselves or addition to Lyapunov exponents, may be used to accurate model stock volatility. The CBOE Volatility Index (VIX) measures the expected volatility of the stock market based on data from S&P 500 index options. Although it calculated for options rather than stocks, it could be combined with an estimate

of a single's stock volatility using the correlation between the S&P 500 index and the prices of the stock being modeled. The volume of a stock may also be used to estimate its volatility. Stocks with high volume, which are traded more often than lower volume stocks, tend to have a lower volatility than stocks with a lower volatility. Future researchers should consider a stock's volume and the VIX index when modeling stock volatility.

Conclusion

During the first part of this project, the prototype stock price model was improved by incorporating moving averages and market indices. The prototype model and the three improved models were implemented in MATLAB and tested for 20 software stocks. The models were able to predict stock prices within a small margin for a week or greater on average. We hope that this will help investors estimate the future trend of a stock. The accuracy of the models varied based on the volatility of the stocks, which motivated us to find a way to measure a stock's volatility. We applied Lyapunov exponents to estimate the volatility and predictability of the 20 software stocks. We found that the Lyapunov exponents were not effective in measuring stock volatility.

Below we list several some topics that we recommend for future researchers:

1. There are several ways to quantify a stock's volatility other than Lyapunov exponents and normalized volatility, such as the VIX index. Some of these may be more effective measures than Lyapunov exponents.
2. One could also research other factors which explain the difference in predictability of stocks, such as their volume.
3. One could vary the method used to compute the maximal Lyapunov exponents. The predicted Lyapunov exponents differ based on how they were calculated. Considering a different time spans and different ways to determine the mean period and time delay may improve the accuracy of the Lyapunov exponents.
4. There are many components of the stock market that affect a stock's price which have not been considered in this project. One could study the effect external factors, such as interest rates, inflation, and tax levels. It is also worth considering stock specific factors such as volume, dividend, and yield.
5. Replacing the S&P 500 index with an indicator specific to the subsector of the stock being studied may increase the accuracy of the index model.

6. The effect of social media on a stock's price is another interesting research topic.
7. One could test if investment decisions based on these models would actually make a profit. If this topic is explored, commission fees for the transactions should be taken into account.
8. It may be advantageous to find a different way to measure the accuracy of the models tested. Although the index models generally predicted the stock price for a greater number of days, this was, in many cases, due to the size of the margin of error. One could try measuring the accuracy as the standard deviation of the future noise or the mean of the absolute value of the noise.
9. Although the Fourier series tend to closely follow the trendline residuals, they often have difficulty modeling the differences in the future. This happens because the differences are not periodic, at least not with a period equal to the number of lags where the autocorrelation is positive. One could try varying the period to a value that reflects seasonal changes in stock prices, such as three months or six months. It may be possible to develop some way to determine an ideal period using the stock prices in addition to the time they span.

References:

1. Fillman, J. J. (2006, June 1). 1.3.5.12. Autocorrelation. Retrieved August 10, 2018, from https://itl.nist.gov/div898/handbook/eda/section3/eda35c.htm
2. Lynch-Stieglitz, J. (2005, Spring). Autocorrelation. Retrieved August 10, 2018, from http://shadow.eas.gatech.edu/~jean/paleo/Meko_Autocorrelation.pdf
3. Korthauer, K. (n.d.). The Least-Squares Line. Retrieved August 10, 2018, from http://bcb.dfci.harvard.edu/~keegan/stat324/STAT324_0414_SLR.pdf
4. Weisstein, E. W. (n.d.). Dynamical System. Retrieved August 10, 2018, from http://mathworld.wolfram.com/DynamicalSystem.html
5. Cvitanovic, P. (2017, January 28). Lyapunov exponents. Retrieved August 10, 2018, from http://chaosbook.org/chapters/Lyapunov.pdf
6. M. T. Rosenstein, J. J. Collins, C. J. De Luca, *A practical method for calculating largest Lyapunov exponents from small data sets*, Physica D **65**, 117 (1993).
7. M. B. Kennel, R. Brown, and H. D. I. Abarbanel, *Determining embedding dimension for phase-space reconstruction using a geometrical construction*, Phys. Rev. A **45**, 3403 (1992).

8. Scheiber, T. (1999, January 6). Delay coordinates. Retrieved August 10, 2018, from https://www.pks.mpg.de/~tisean/TISEAN_2.1/docs/chaospaper/node6.html

9. Cross, M. (n.d.). Lyapunov Exponents. Retrieved August 17, 2018, from http://www.cmp.caltech.edu/~mcc/Chaos_Course/Lesson7/Lyapunov.pdf

10. INXN Historical Prices | InterXion Holding N.V. Ordinary Stock. (2018, August 17). Retrieved August 17, 2018, from https://finance.yahoo.com/quote/INXN/history?p=INXN

11. Kantz, H., & Scheiber, T. (2004). Phase Space Methods. In *Nonlinear Time Series Analysis* (Second ed.). Cambridge, Massachusetts: Cambridge University Press.

12. Kantz, H., & Scheiber, T. (2004). Instability: Lyapunov Exponents. In *Nonlinear Time Series Analysis* (Second ed.). Cambridge, Massachusetts: Cambridge University Press.

13. Kantz, H., & Scheiber, T. (2004). Lyapunov Exponents II. In *Nonlinear Time Series Analysis* (Second ed.). Cambridge, Massachusetts: Cambridge University Press.

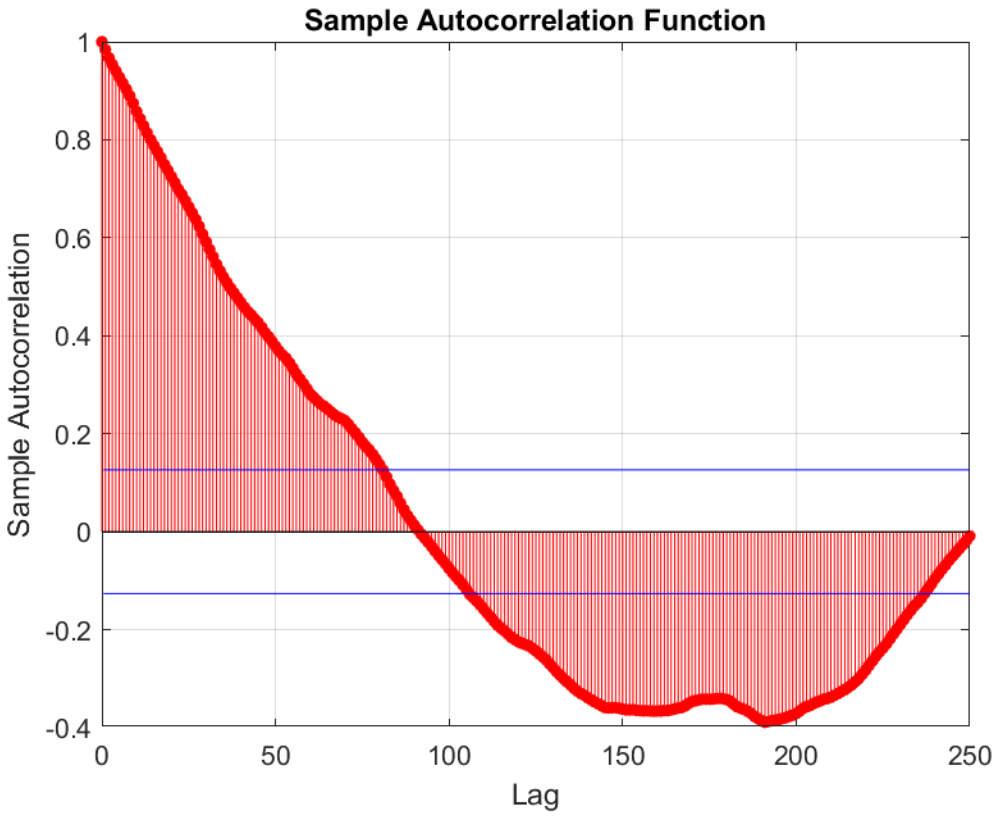
14. Mitra, A. K. (n.d.). Curve Fitting and Fourier Series. Retrieved August 17, 2018, from http://www.public.iastate.edu/~akmitra/aero361/design_web/crvft.html

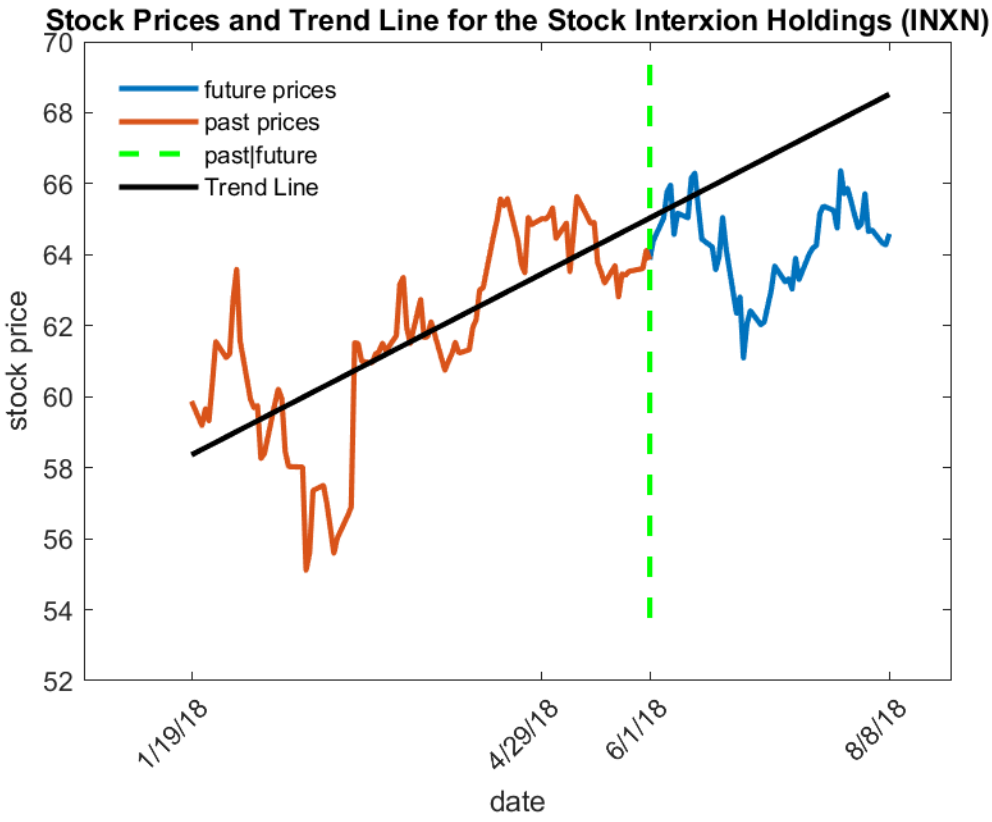
15. Strang, G. (n.d.). Fourier Series and Integrals. Retrieved August 17, 2018, from http://math.mit.edu/~gs/cse/websections/cse41.pdf

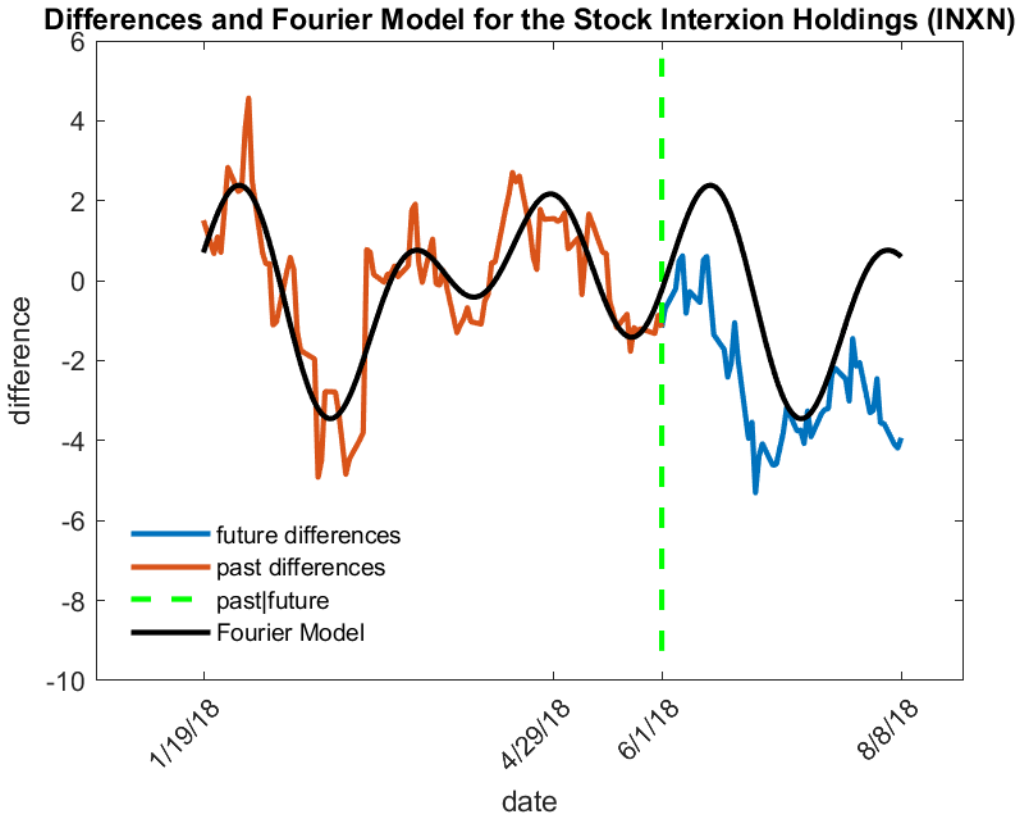
16. Weidman, M. (n.d.). Download Daily Data from Google and Yahoo! Finance - File Exchange - MATLAB Central. Retrieved August 17, 2018, from https://www.mathworks.com/matlabcentral/fileexchange/43627-download-daily-data-from-google-and-yahoo-finance

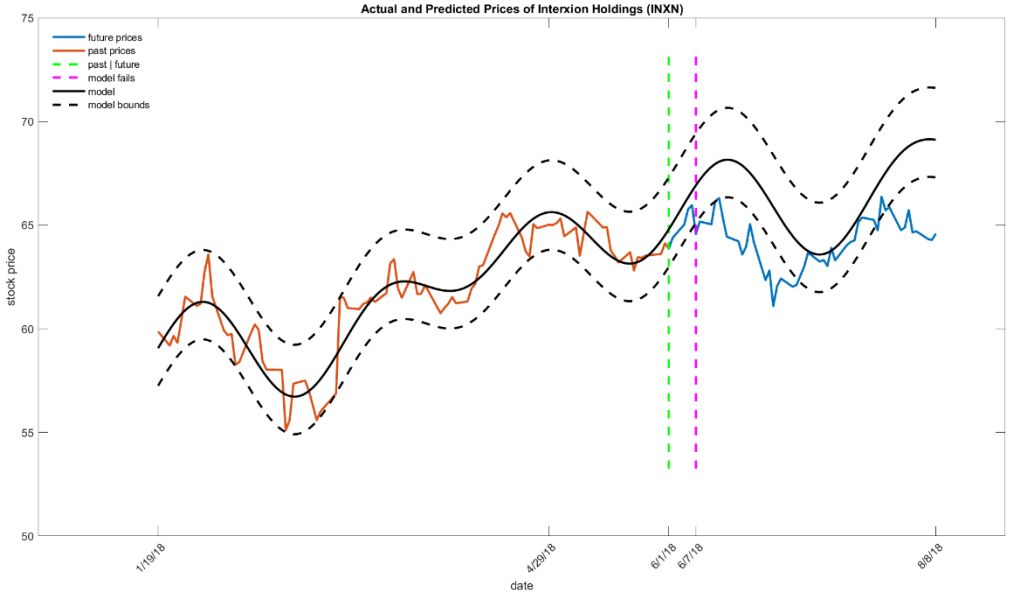
Appendix: Individual Stock Graphs

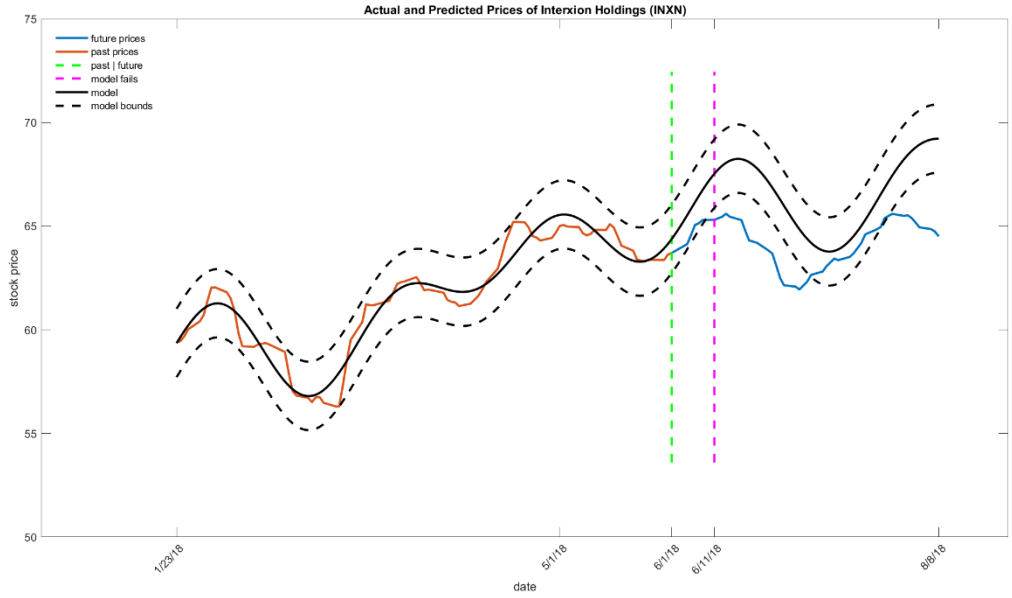
Interxion Holdings (INXN)

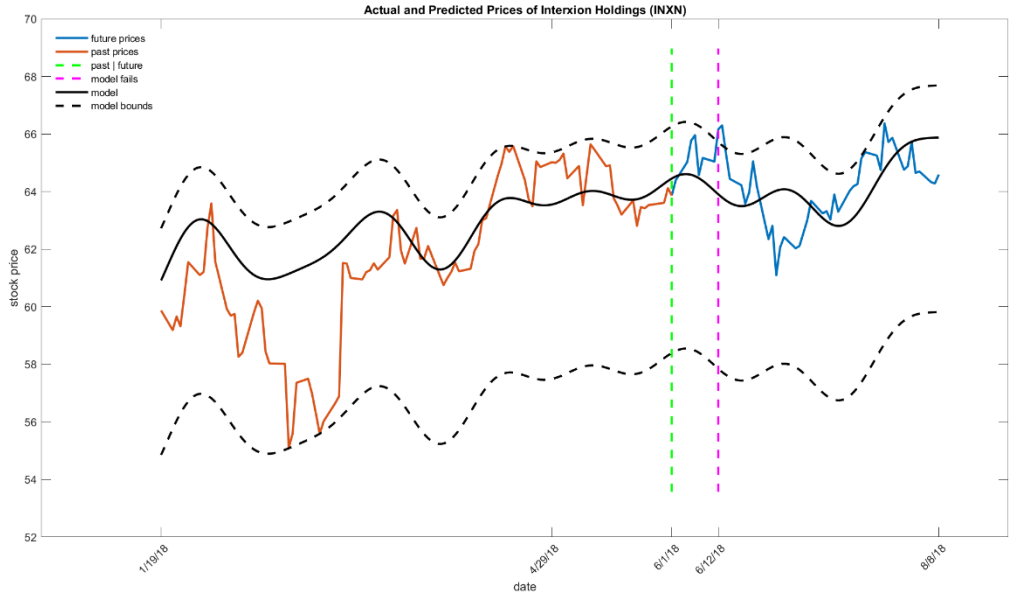


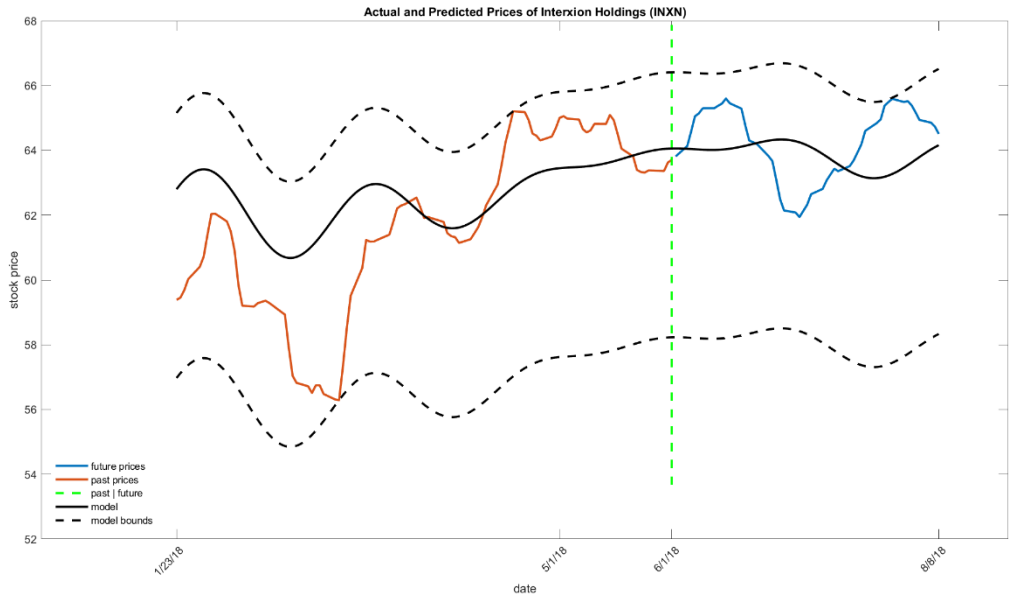

Stock Prices and Trend Line for the Stock Interxion Holdings (INXN)
future prices
past prices
past|future
Trend Line
stock price
70
68
66
64
62
60
58
56
54
52
1/19/18
4/29/18
6/1/18
8/8/18
date


Differences and Fourier Model for the Stock Interxion Holdings (INXN)
difference
date
6
4
2
0
-2
-4
-6
-8
-10
1/19/18
4/29/18
6/1/18
8/8/18
future differences
past differences
past|future
Fourier Model


Actual and Predicted Prices of Interxion Holdings (INXN)
stock price
date
75
70
65
60
55
50
1/19/18
4/29/18
6/1/18
6/7/18
8/8/18
future prices
past prices
past | future
model fails
model
model bounds

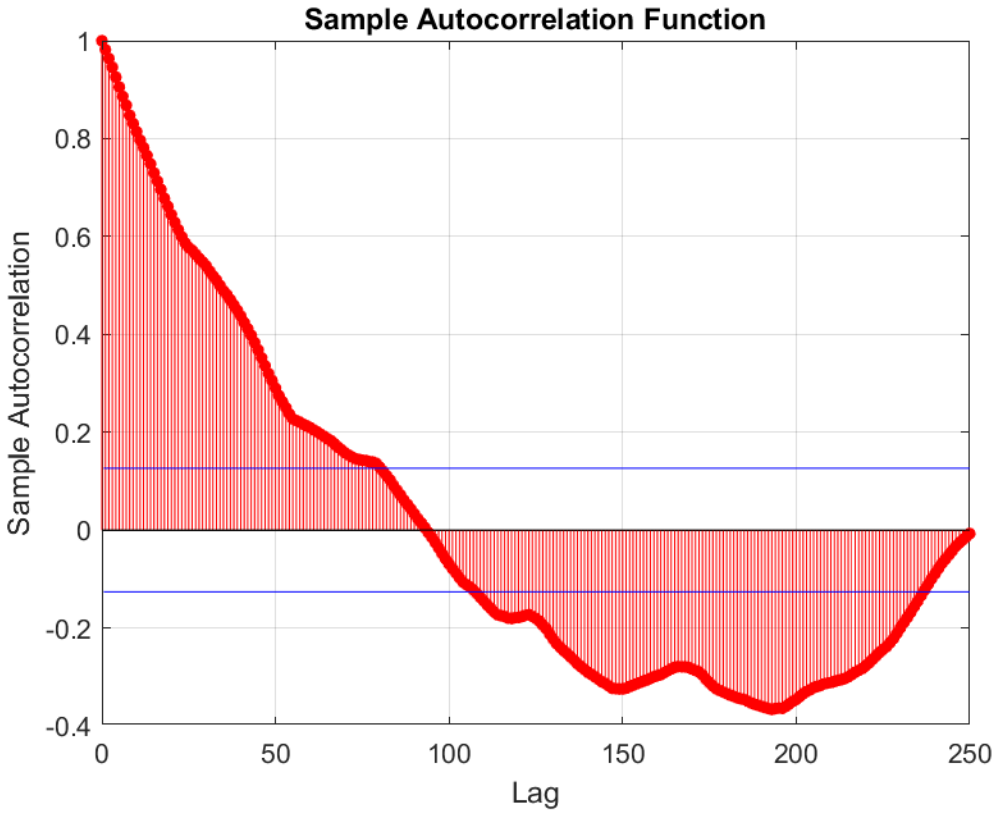

Actual and Predicted Prices of Interxion Holdings (INXN)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
75
70
65
60
55
50
1/23/18
5/1/18
6/1/18
6/11/18
8/8/18

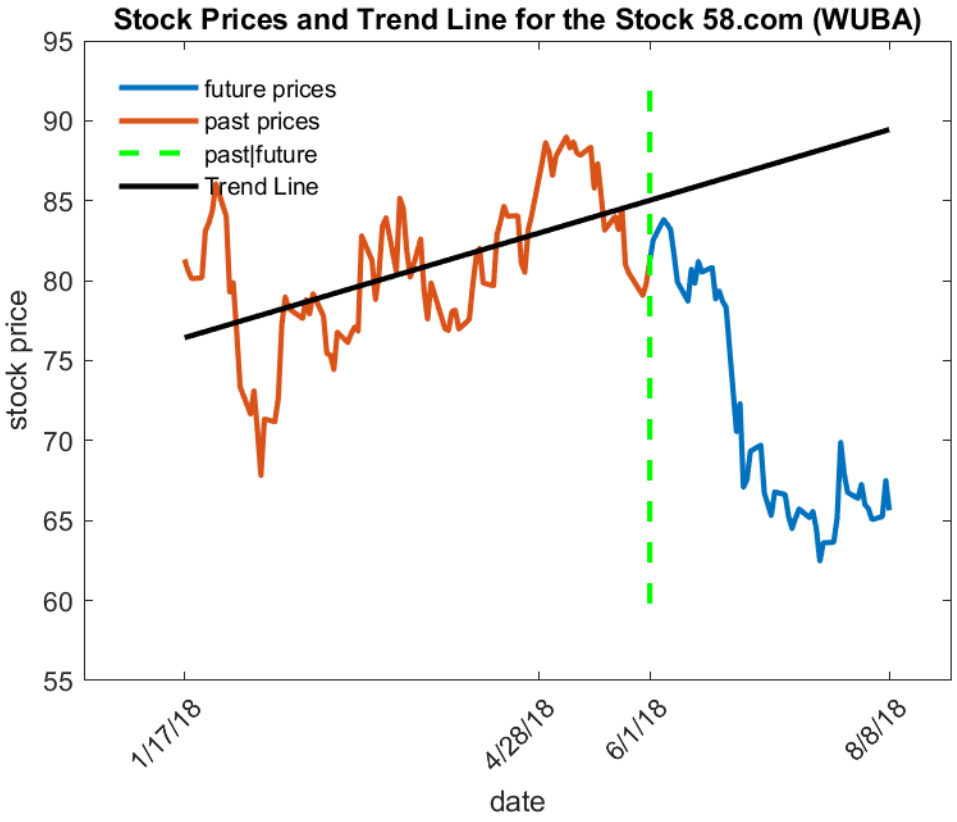

Actual and Predicted Prices of Interxion Holdings (INXN)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
70
68
66
64
62
60
58
56
54
52
1/19/18
4/29/18
6/1/18
6/12/18
8/8/18

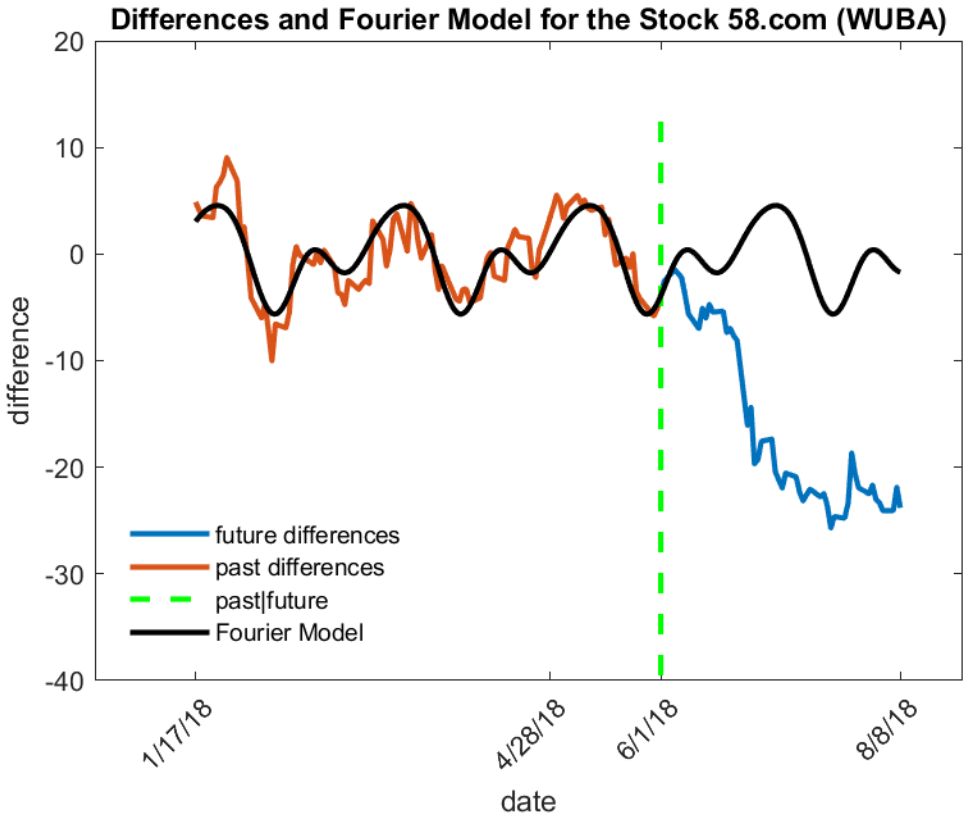


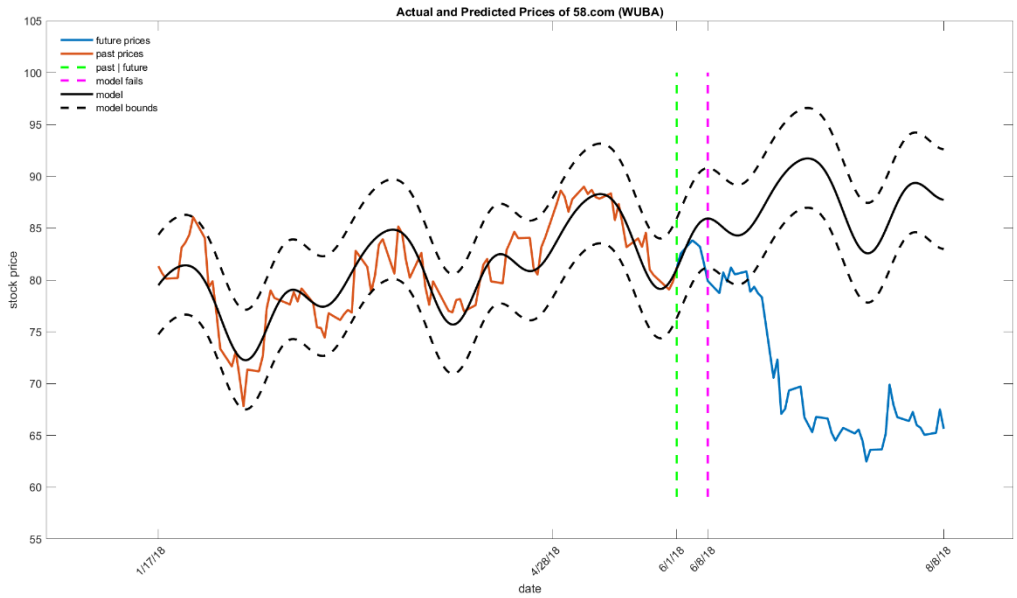
The prototype model fails to predict the stock price after 5 business days. This occurs because the Fourier series predicts an increase in the trend line residuals after 6/1/18 (to mirror the increase after 1/19/18). In reality, the stock briefly increases, then drops. The prototype model with moving averages fails for the same reason. The index model fails later because it has a wider prediction margin. It fails at the second peak in the stock price after 6/1/18. However, it correctly predicts the stock price for almost all of the 48 business days after 6/1/18. Introducing moving averages decreases the height of the peak, causing the index model to predict all 48 days correctly.

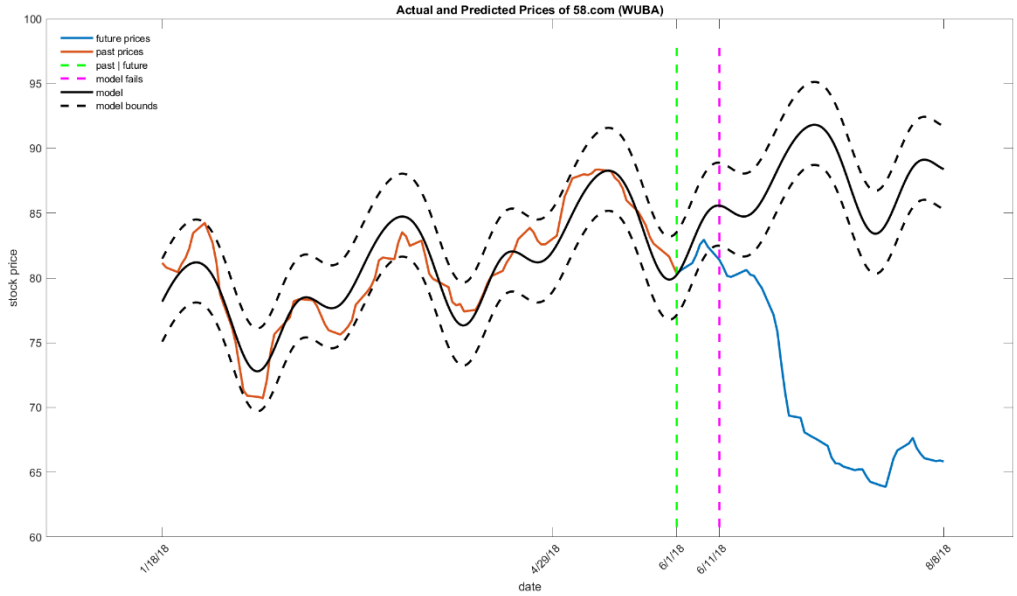
58.com (WUBA)

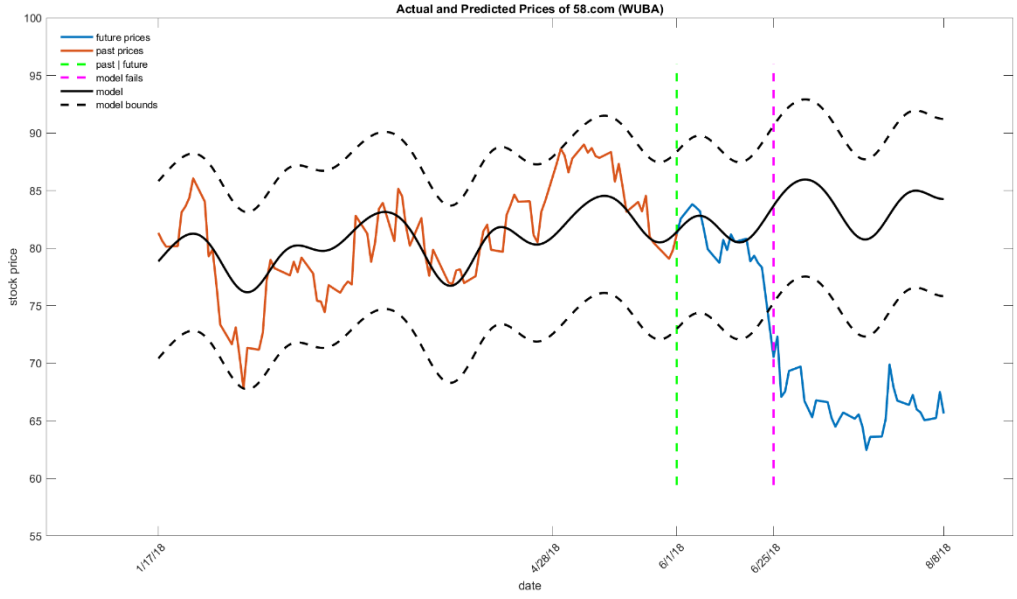

Sample Autocorrelation Function
Sample Autocorrelation
Lag
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
50
100
150
200
250

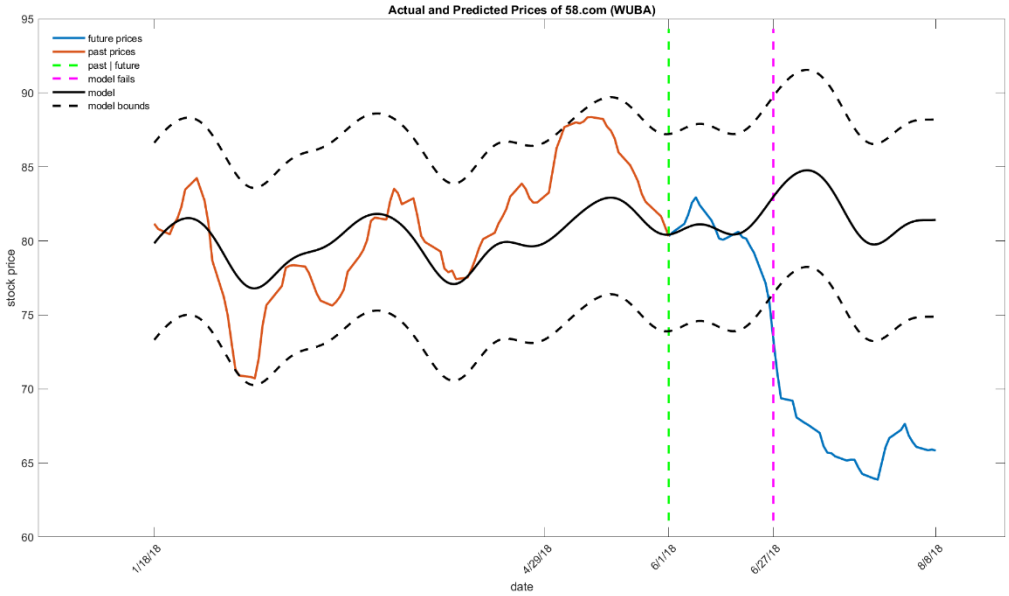

Stock Prices and Trend Line for the Stock 58.com (WUBA)
future prices
past prices
past|future
Trend Line
stock price
95
90
85
80
75
70
65
60
55
1/17/18
4/28/18
6/1/18
8/8/18
date


Differences and Fourier Model for the Stock 58.com (WUBA)
20
10
0
-10
-20
-30
-40
difference
future differences
past differences
past|future
Fourier Model
1/17/18
4/28/18
6/1/18
8/8/18
date


Actual and Predicted Prices of 58.com (WUBA)
future prices
past prices
past | future
model fails
model
model bounds
105
100
95
90
85
80
75
70
65
60
55
stock price
1/17/18
4/28/18
6/1/18
6/8/18
8/8/18
date

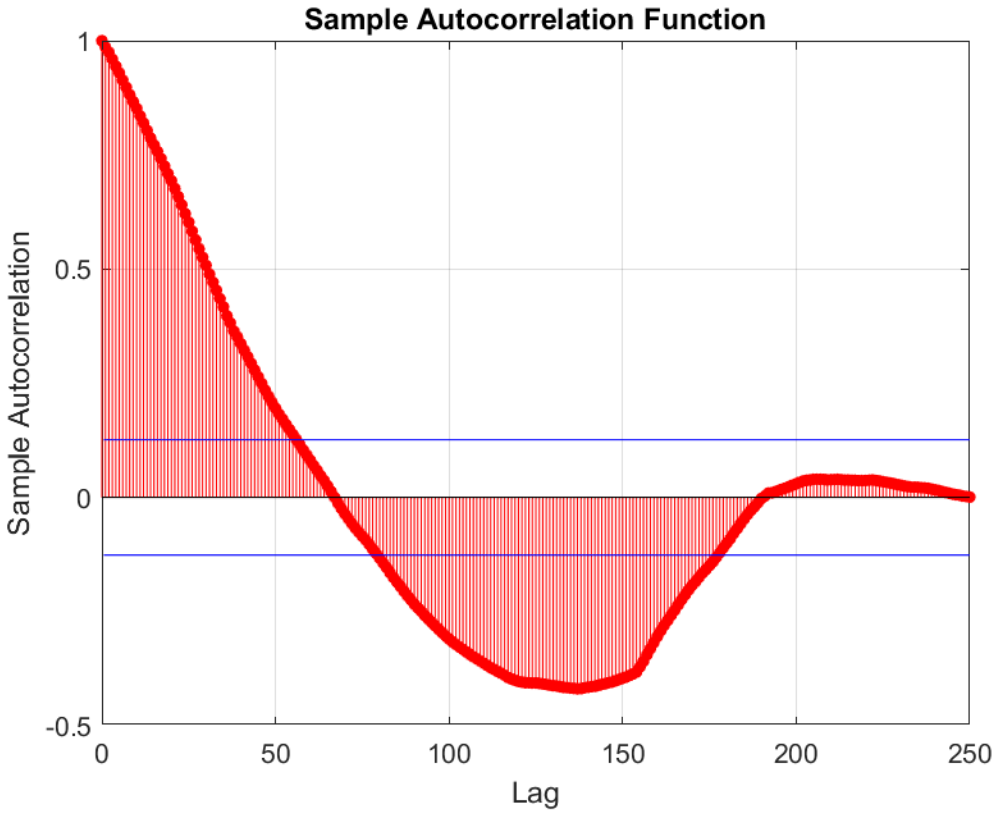

Actual and Predicted Prices of 58.com (WUBA)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
100
95
90
85
80
75
70
65
60
1/18/18
4/29/18
6/1/18
6/11/18
8/8/18

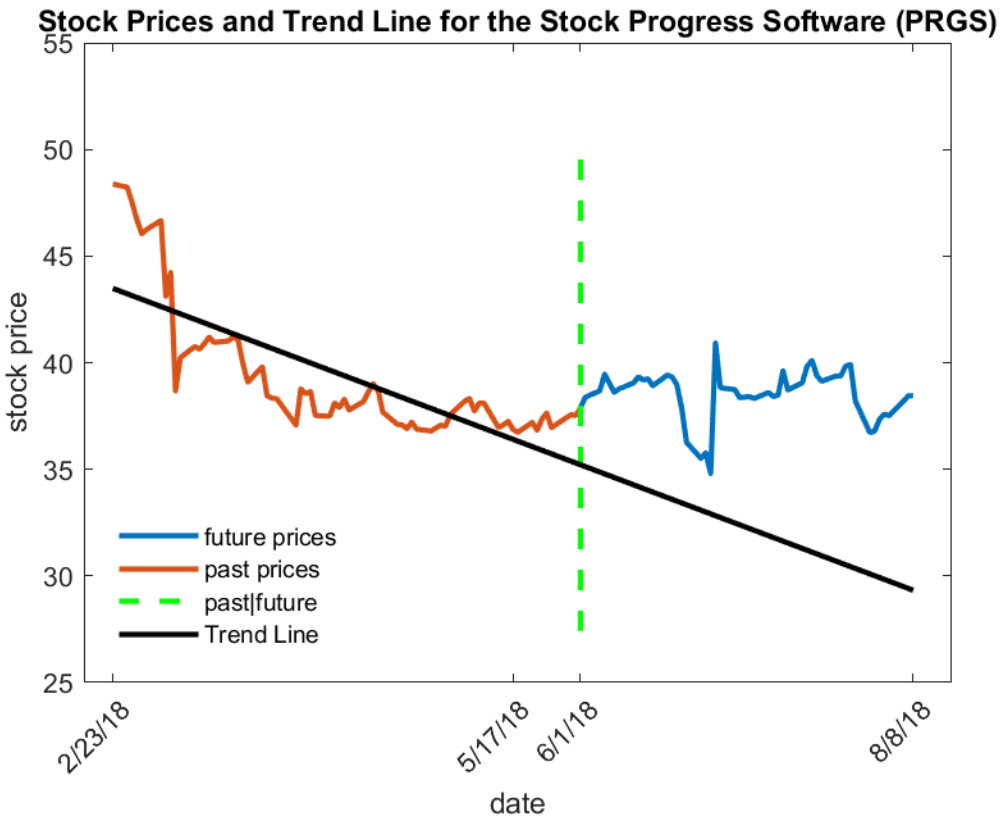

Actual and Predicted Prices of 58.com (WUBA)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
100
95
90
85
80
75
70
65
60
55
1/17/18
4/29/18
6/1/18
6/25/18
8/8/18

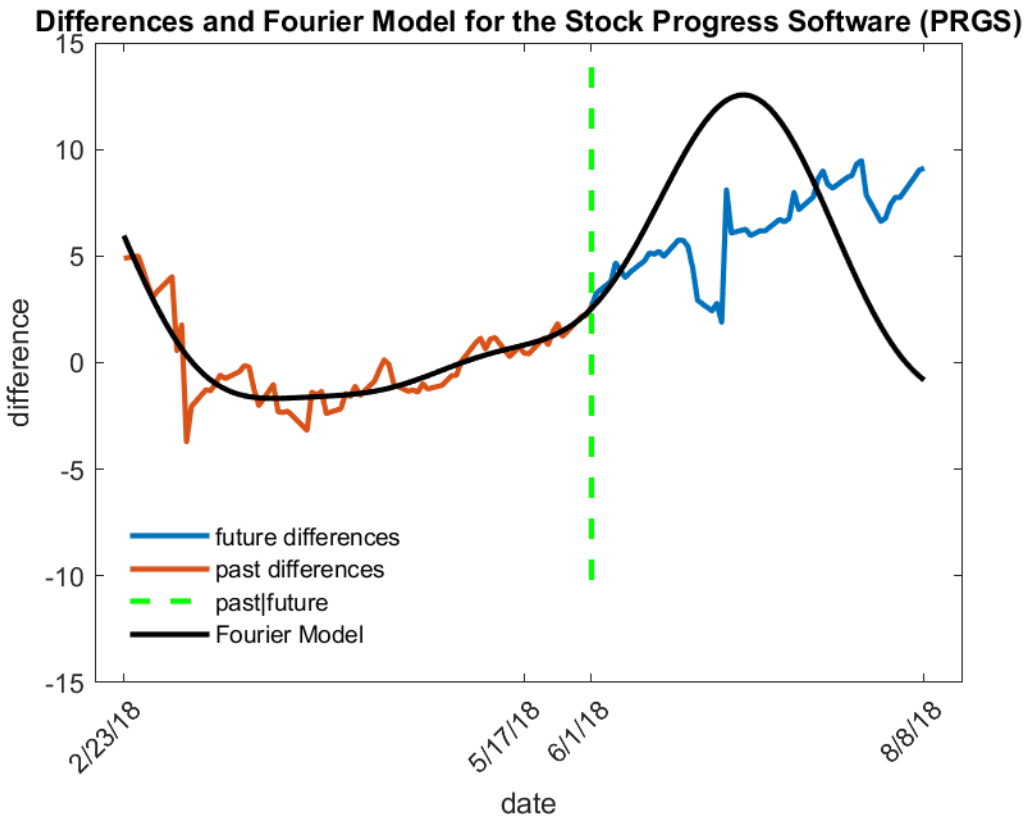


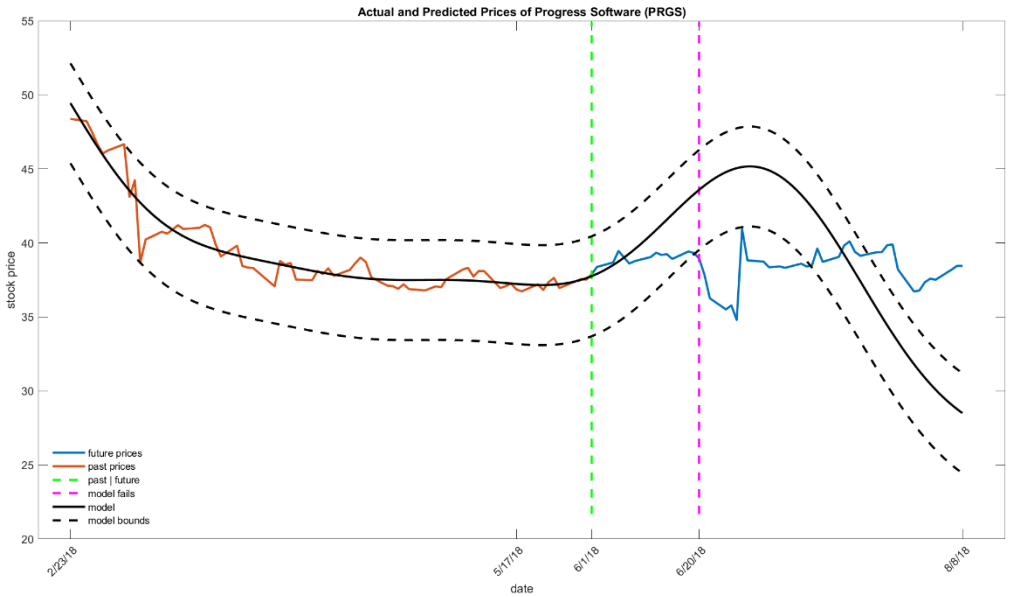
Although the price of 58.com sharply drops, the prototype model fails before the drop. This failure occurs because the Fourier Series predicts that the trend line residuals will increase but they actually decrease. Because the index models have a higher margin of error than the prototype models, they correctly predict the stock's price until the stock price drops sharply.

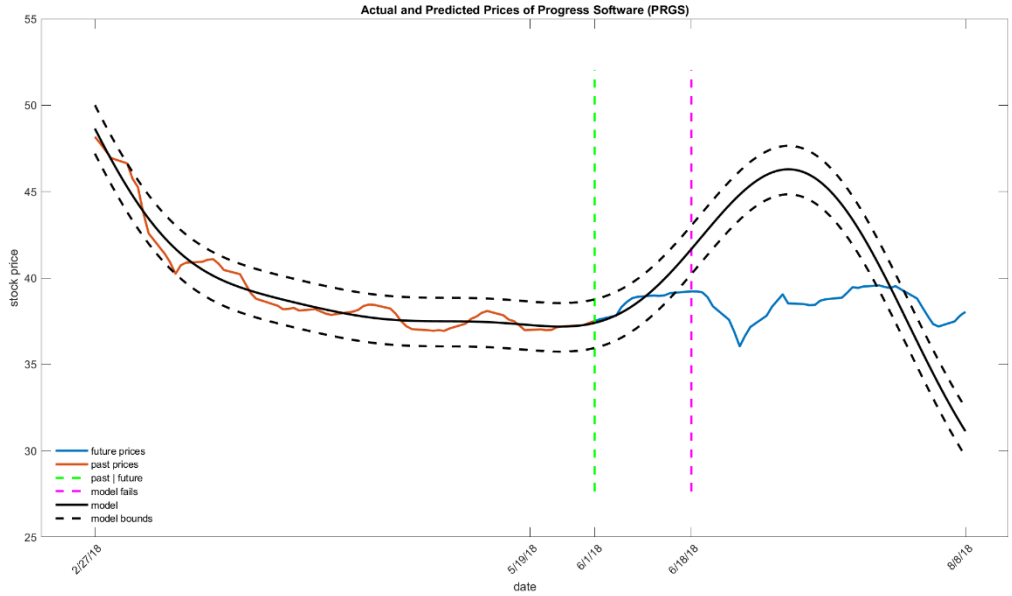
Progress Software (PRGS)

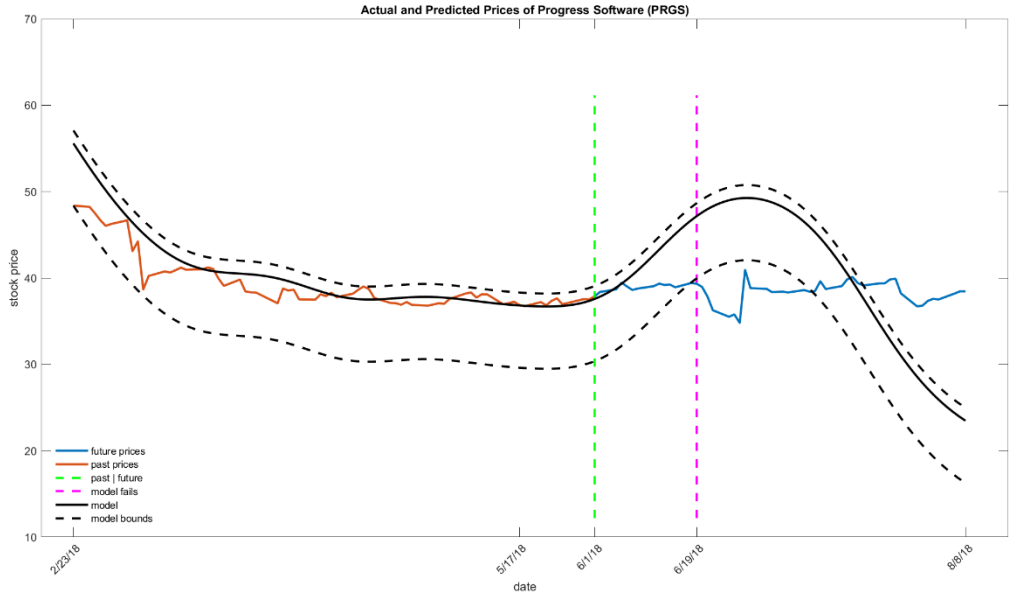


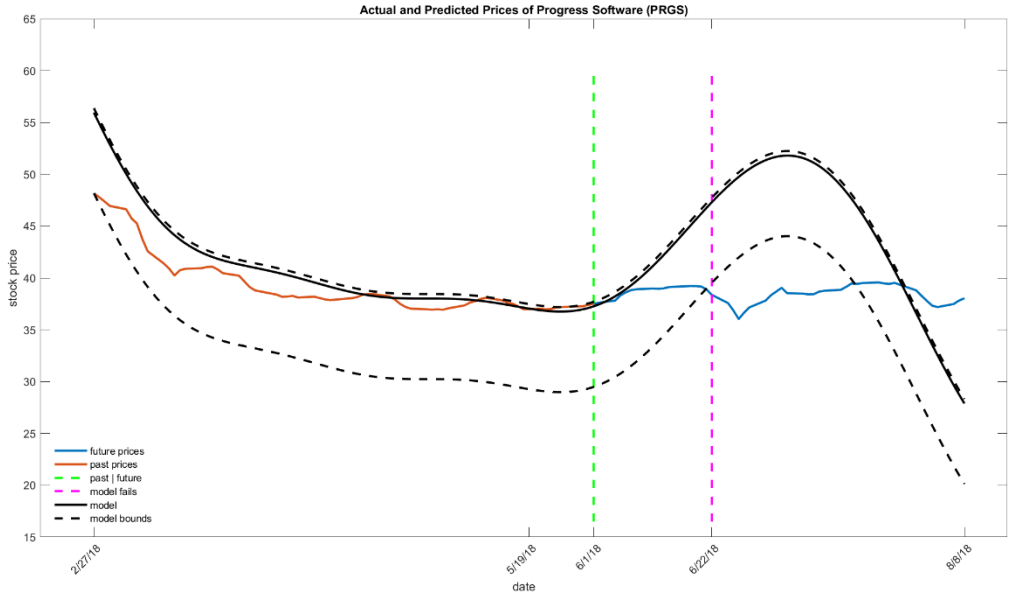

Stock Prices and Trend Line for the Stock Progress Software (PRGS)
55
50
45
40
35
30
25
stock price
future prices
past prices
past|future
Trend Line
2/23/18
5/17/18
6/1/18
8/8/18
date


Differences and Fourier Model for the Stock Progress Software (PRGS)
15
10
5
0
-5
-10
-15
difference
future differences
past differences
past|future
Fourier Model
2/23/18
5/17/18
6/1/18
8/8/18
date


Actual and Predicted Prices of Progress Software (PRGS)
55
50
45
40
35
30
25
20
stock price
future prices
past prices
past | future
model fails
model
model bounds
2/23/18
5/17/18
6/1/18
6/20/18
8/8/18
date

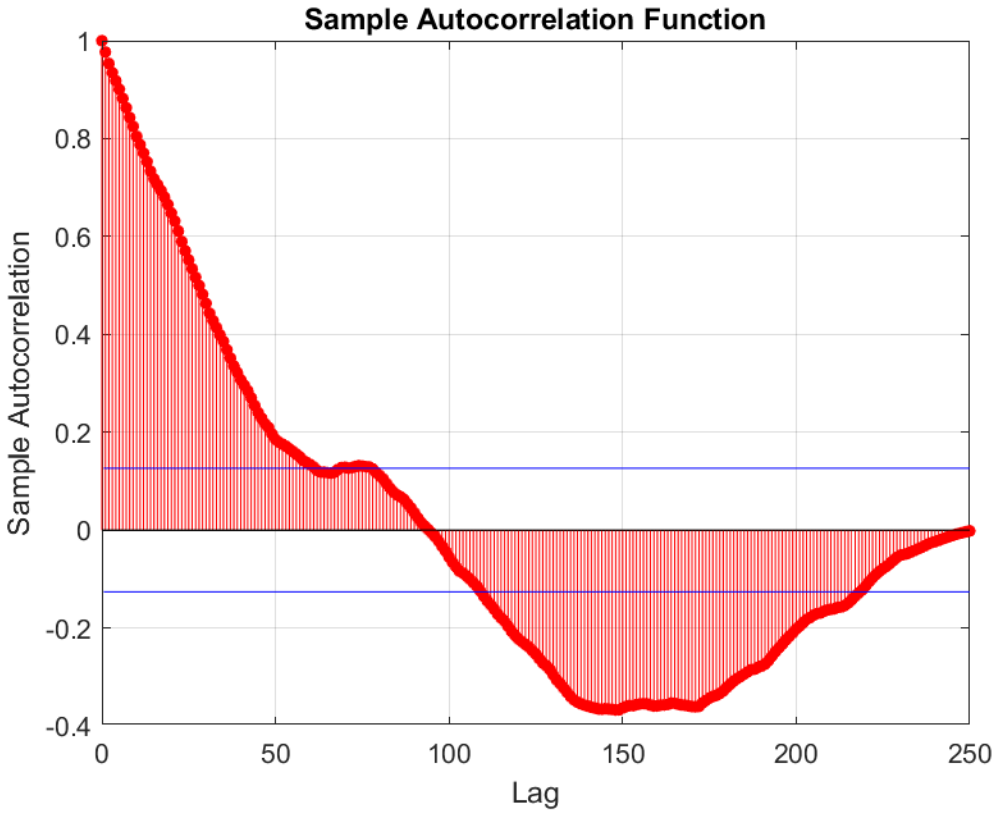

Actual and Predicted Prices of Progress Software (PRGS)
stock price
date
55
50
45
40
35
30
25
2/27/18
5/19/18
6/1/18
6/18/18
8/8/18
future prices
past prices
past | future
model fails
model
model bounds

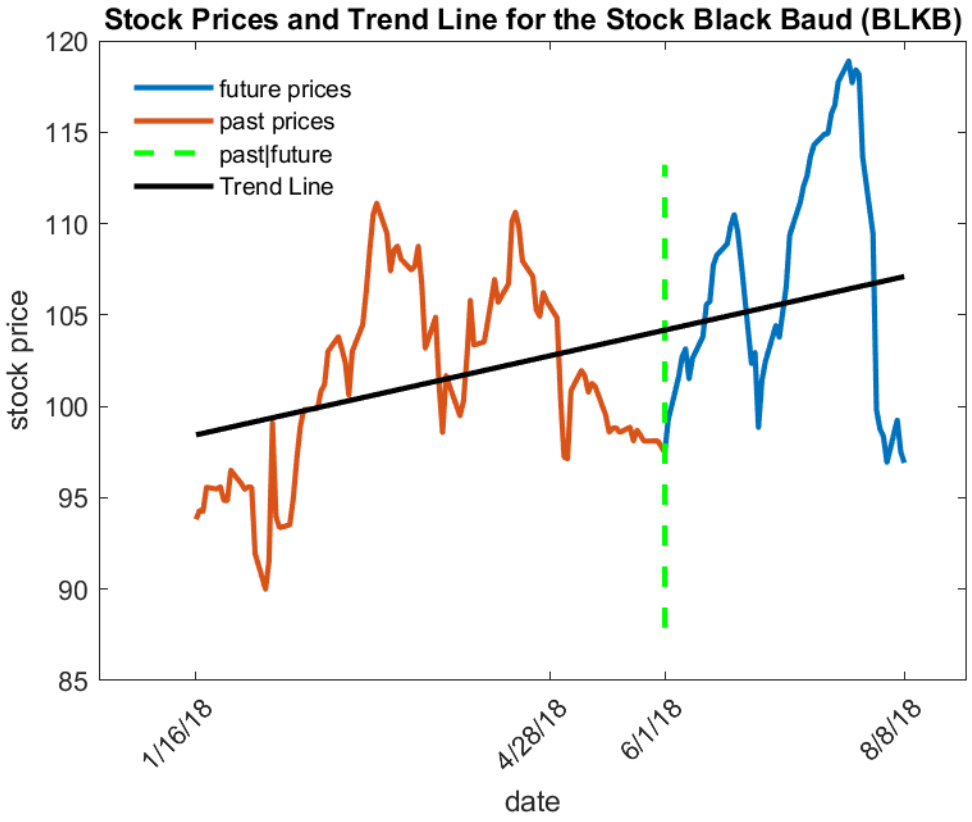

Actual and Predicted Prices of Progress Software (PRGS)
stock price
date
70
60
50
40
30
20
10
2/23/18
5/17/18
6/1/18
6/19/18
8/6/18
future prices
past prices
past | future
model fails
model
model bounds

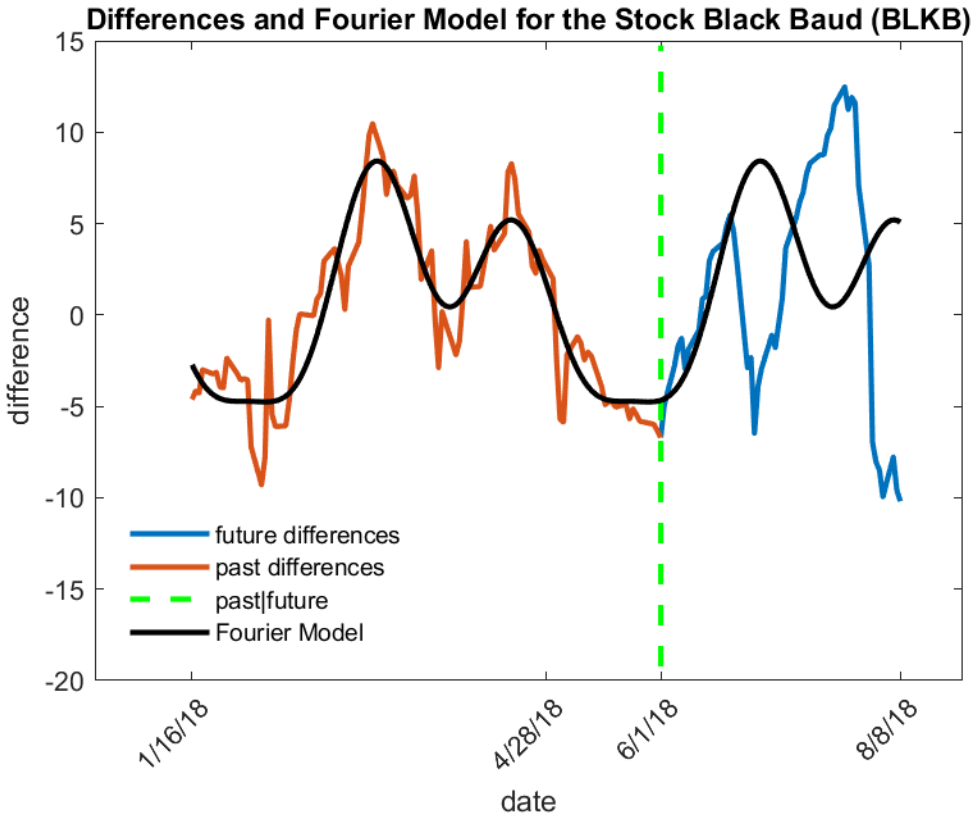


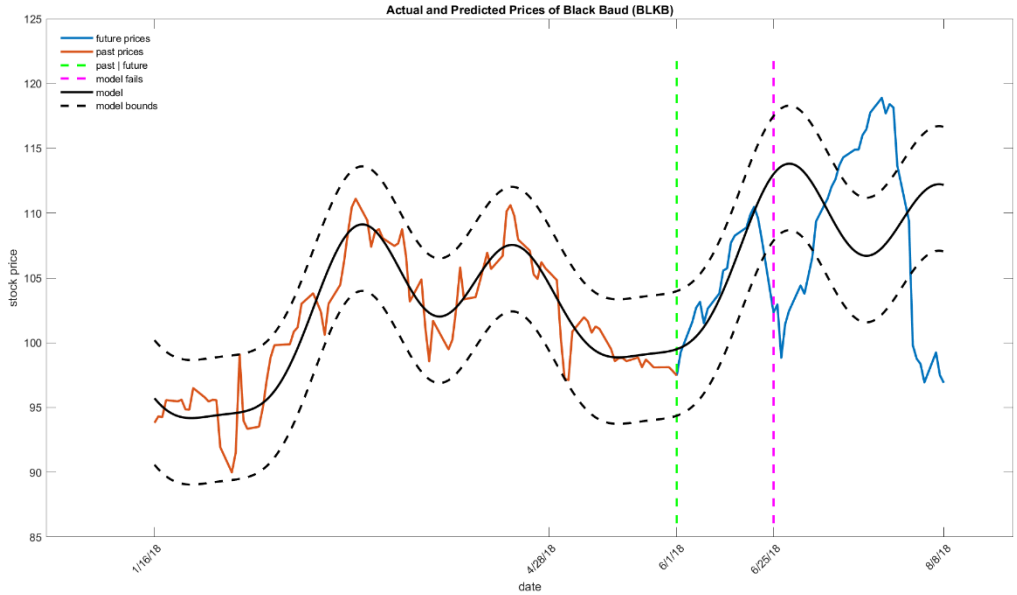
The Fourier model accurately estimates the past trend line residuals. However, the oscillation of the trend line residuals is dominated by a larger change in the trend line residuals due to the rapid drop in the stock prices shortly after 2/23/18. This results in a substantial increase in the Fourier model until early July. Because the actual stock price does not correspondingly increase, the model fails.

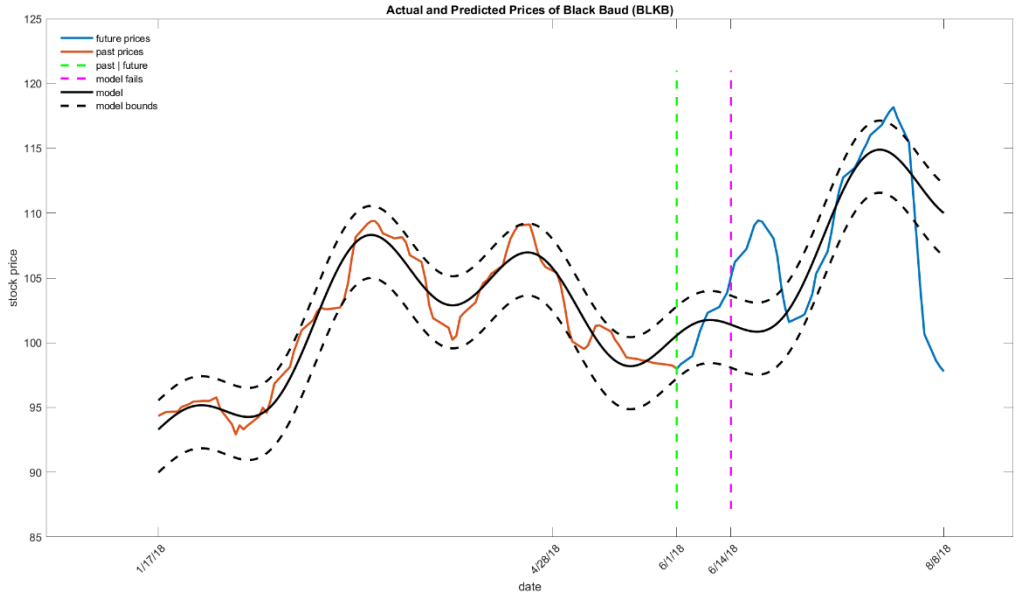
Black Baud (BLKB)

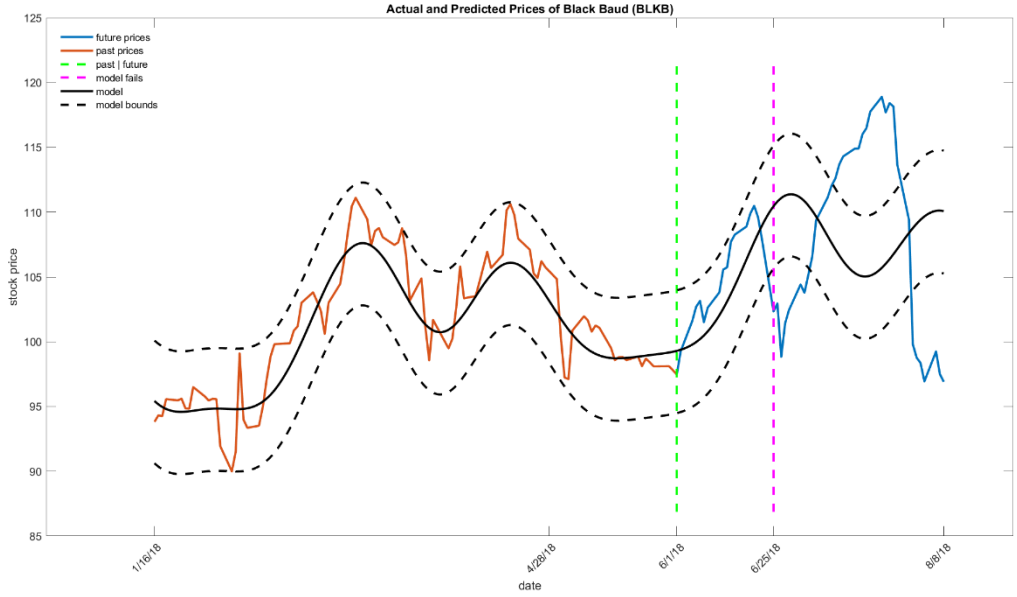


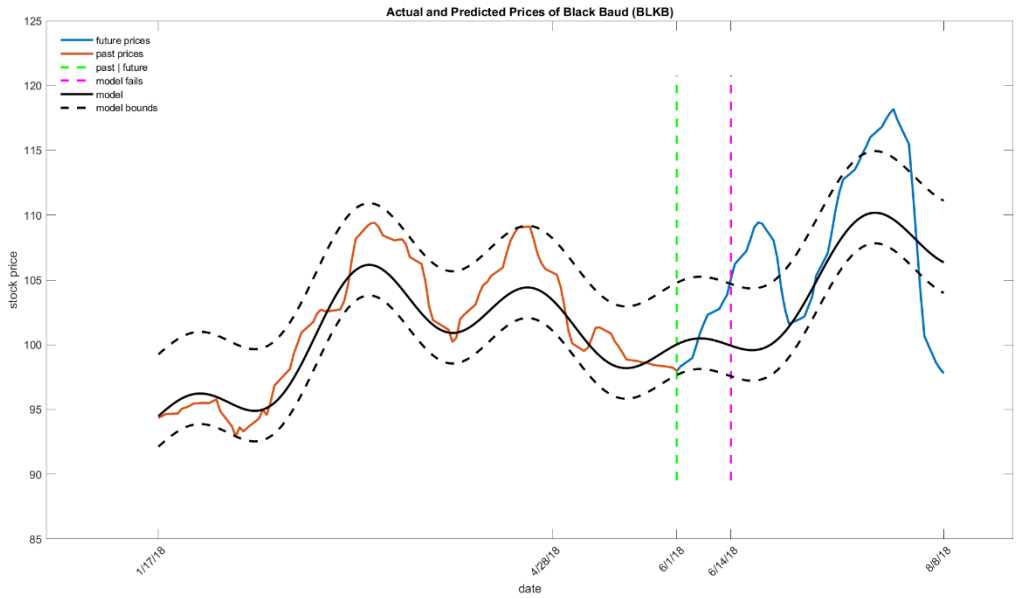

Stock Prices and Trend Line for the Stock Black Baud (BLKB)
future prices
past prices
past|future
Trend Line
stock price
120
115
110
105
100
95
90
85
1/16/18
4/28/18
6/1/18
8/8/18
date


Differences and Fourier Model for the Stock Black Baud (BLKB)
difference
date
15
10
5
0
-5
-10
-15
-20
1/16/18
4/28/18
6/1/18
8/8/18
future differences
past differences
past|future
Fourier Model


Actual and Predicted Prices of Black Baud (BLKB)
stock price
date
125
120
115
110
105
100
95
90
85
1/16/18
4/28/18
6/1/18
6/25/18
8/8/18
future prices
past prices
past | future
model fails
model
model bounds

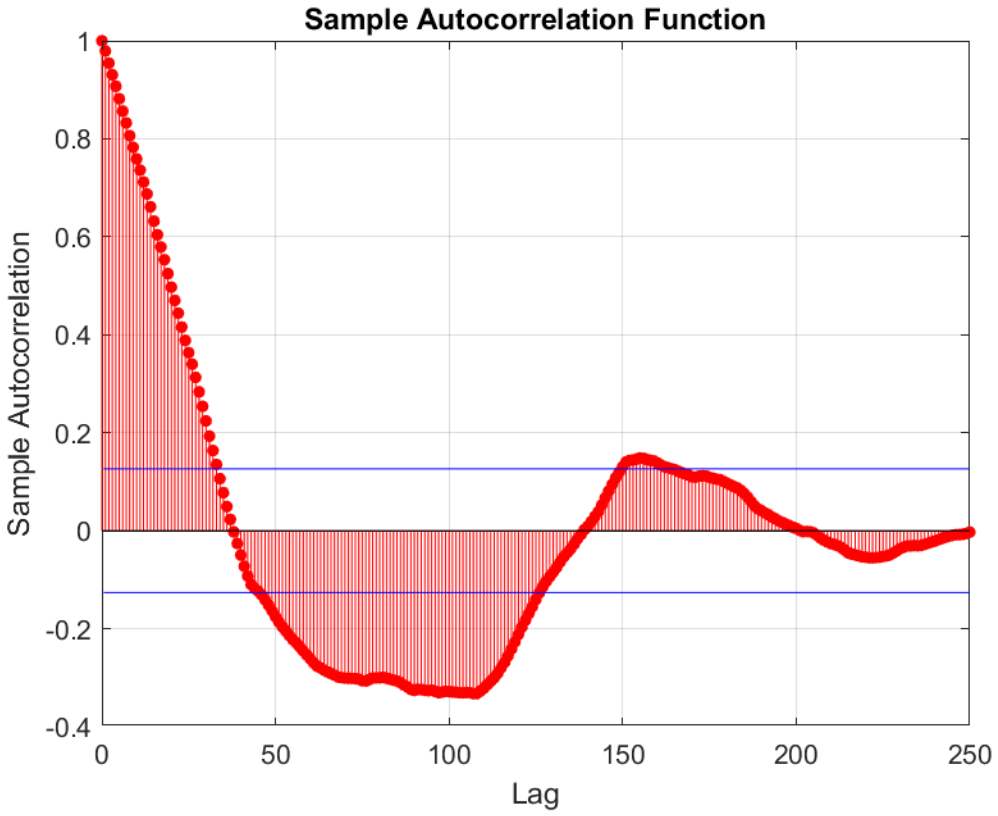

Actual and Predicted Prices of Black Baud (BLKB)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
125
120
115
110
105
100
95
90
85
1/17/18
4/28/18
6/1/18
6/14/18
8/8/18

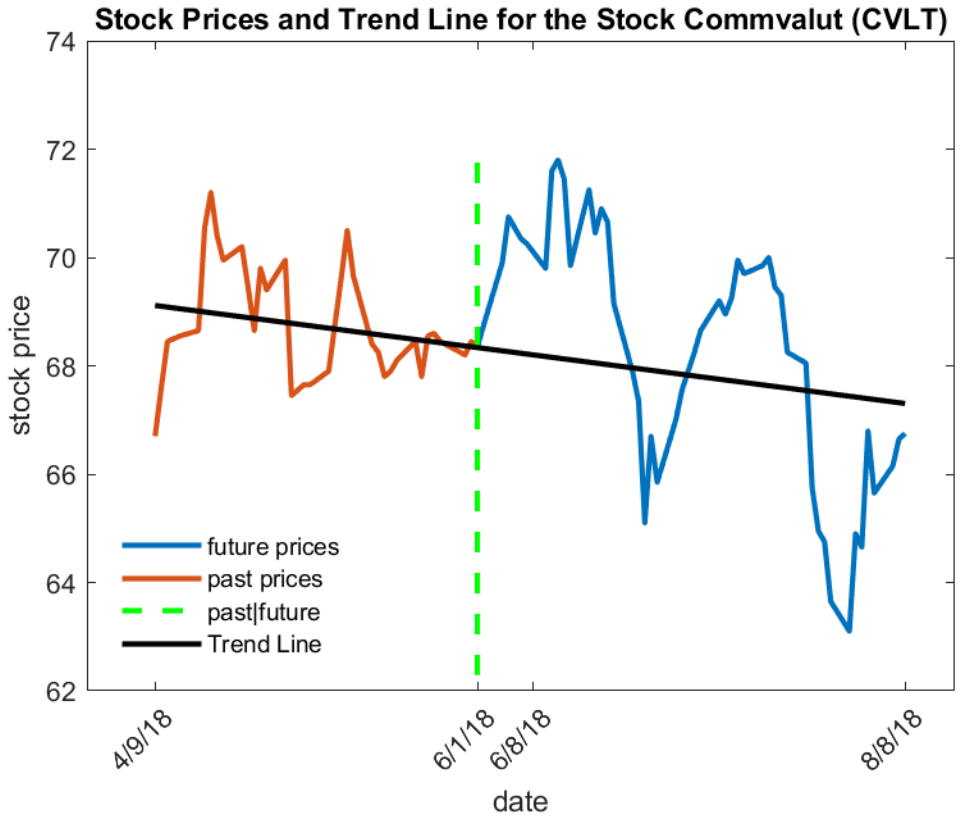

Actual and Predicted Prices of Black Baud (BLKB)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
125
120
115
110
105
100
95
90
85
1/16/18
4/28/18
6/1/18
6/25/18
8/8/18

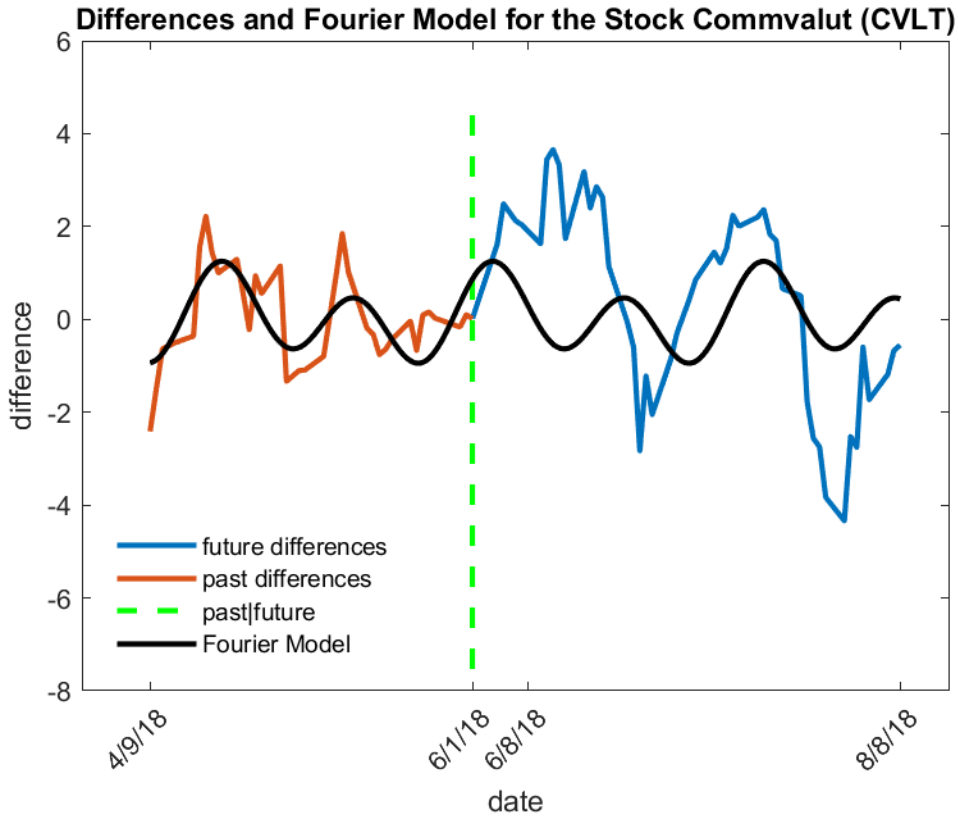


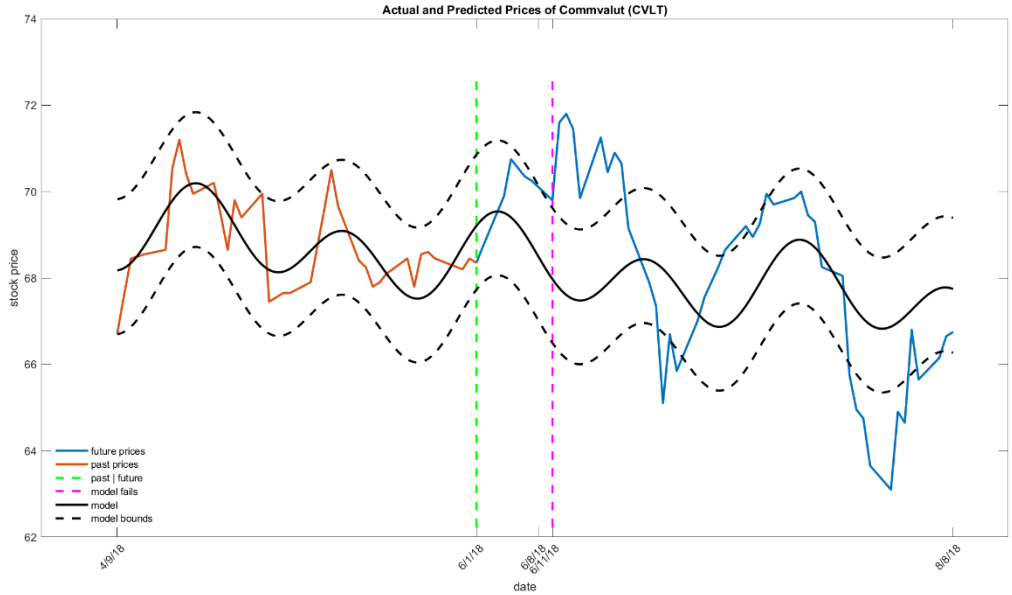
Both the prototype and index models perform well without averages, predicting 17 days correctly. They eventually fail when the stock price drops sharply. When moving averages are introduced, the first future peak is amplified in size, causing the models to fail before the peak. The prototype and index models were similar because the correlation between the stock price and the S&P 500 was near zero.

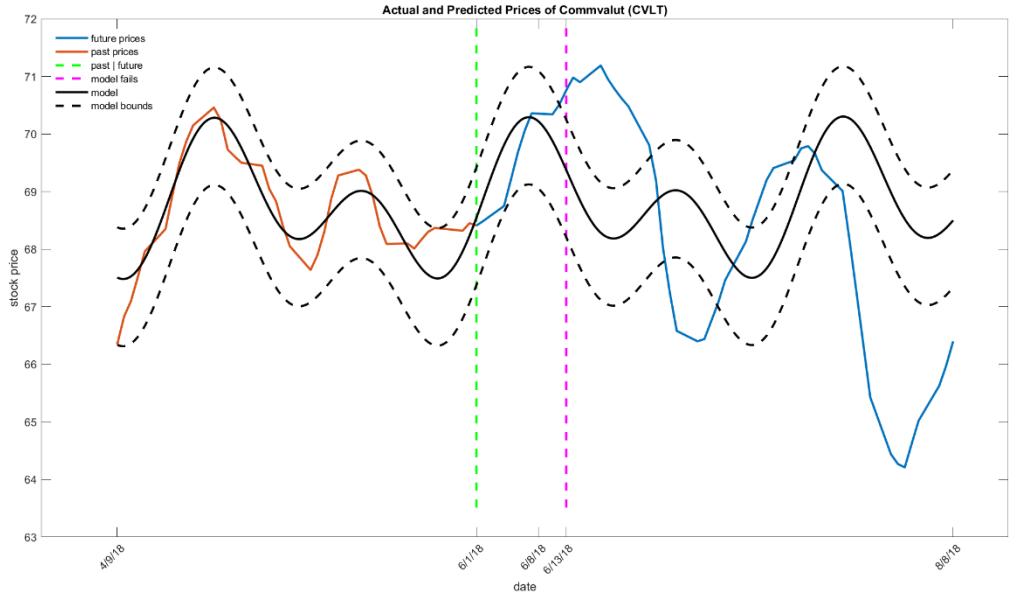
Commvault (CVLT)

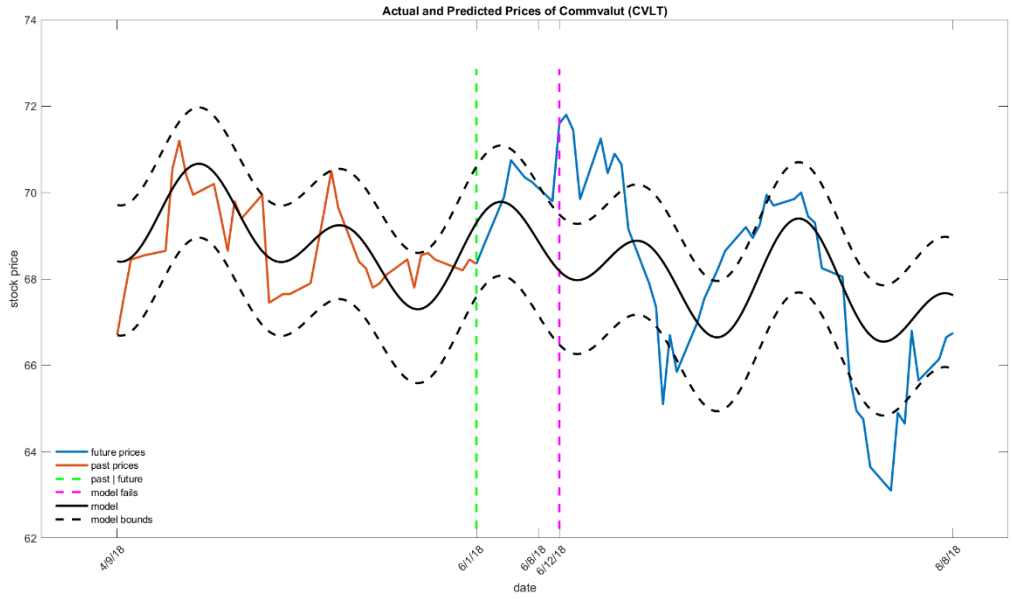


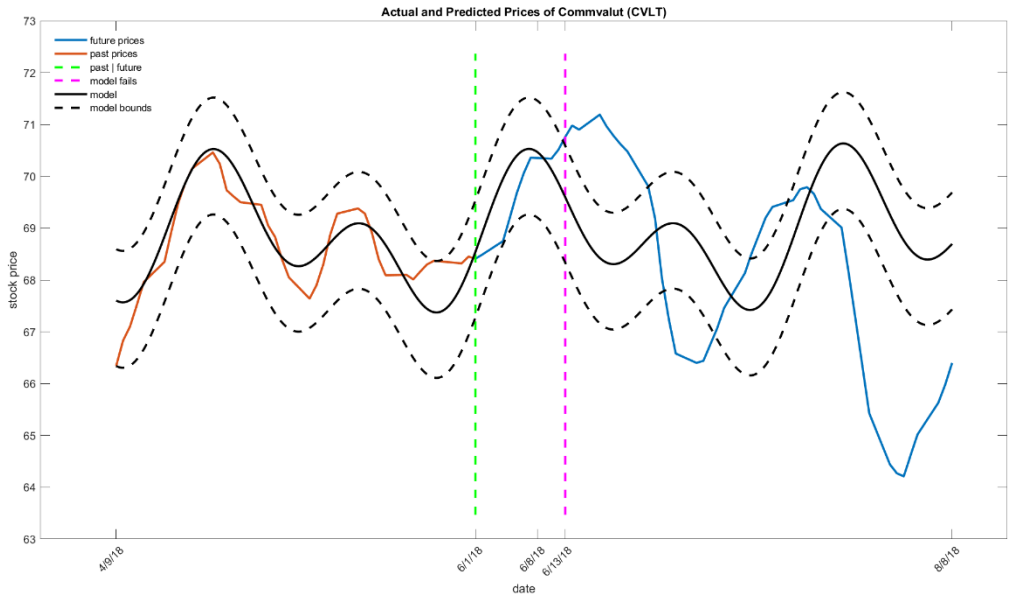

Stock Prices and Trend Line for the Stock Commvalut (CVLT)
74
72
70
68
66
64
62
stock price
future prices
past prices
past|future
Trend Line
4/9/18
6/1/18
6/8/18
8/8/18
date


Differences and Fourier Model for the Stock Commvalut (CVLT)
difference
date
4/9/18
6/1/18
6/8/18
8/8/18
future differences
past differences
past|future
Fourier Model


Actual and Predicted Prices of Commvalut (CVLT)
stock price
date
4/9/18
6/1/18
6/8/18
6/11/18
8/8/18
future prices
past prices
past | future
model fails
model
model bounds

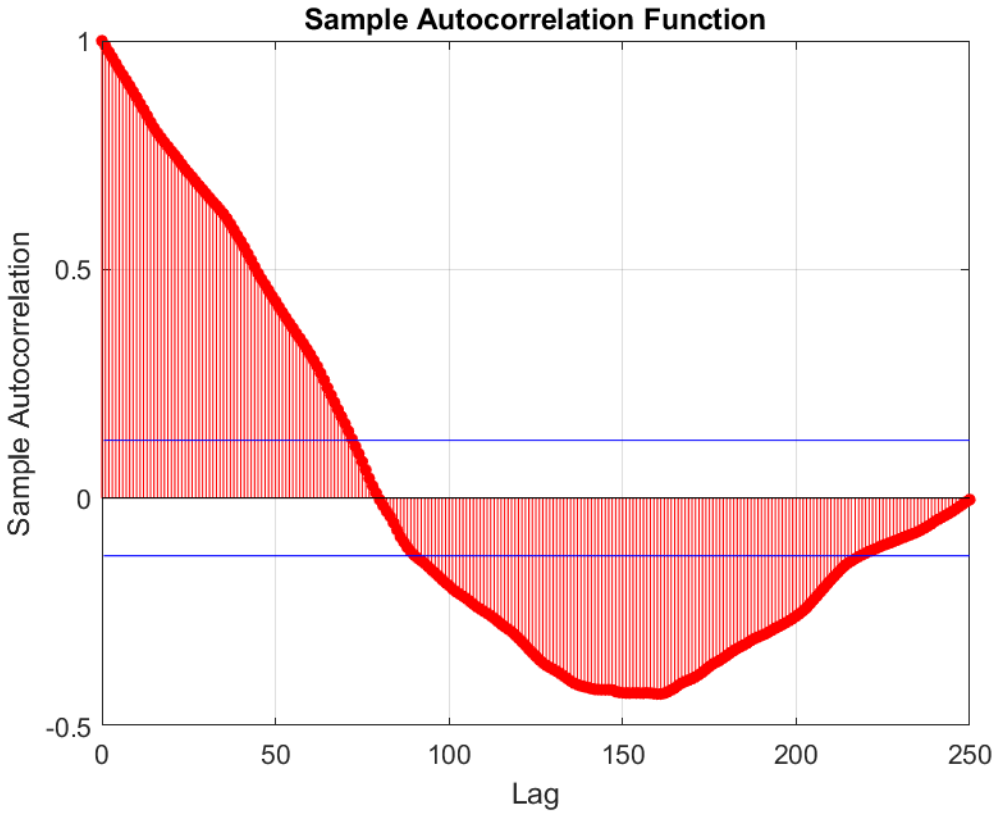

Actual and Predicted Prices of Commvalut (CVLT)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
4/9/18
6/1/18
6/8/18
6/13/18
8/8/18

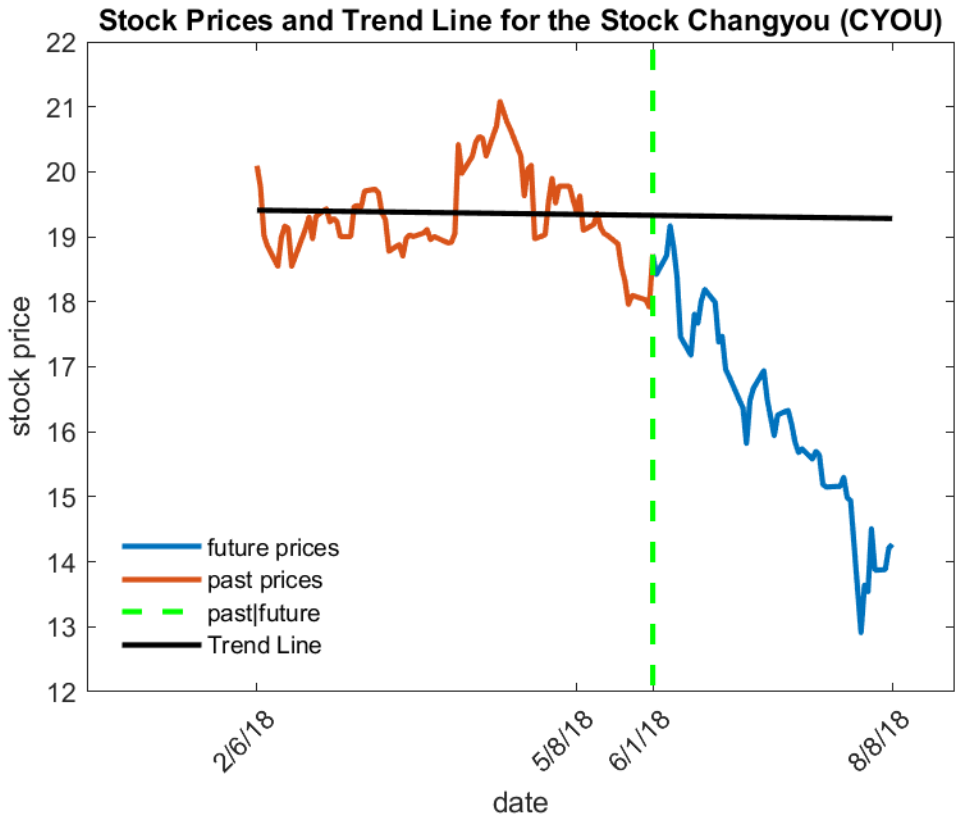

Actual and Predicted Prices of Commvalut (CVLT)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
4/9/18
6/1/18
6/8/18
6/12/18
8/8/18

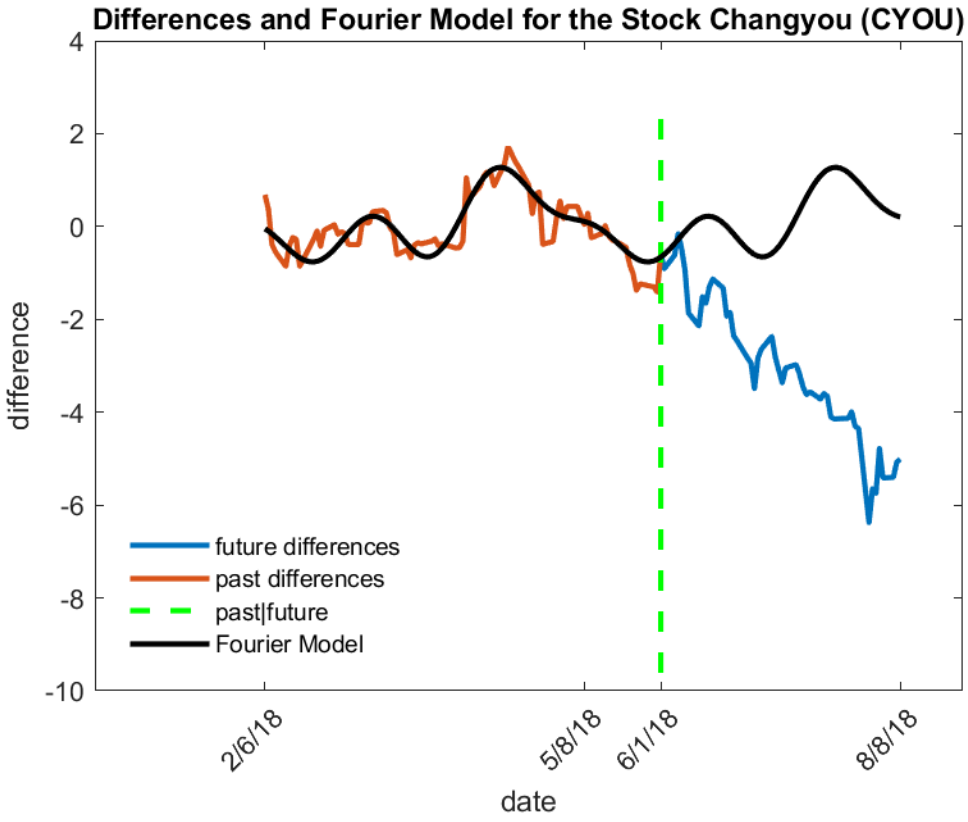


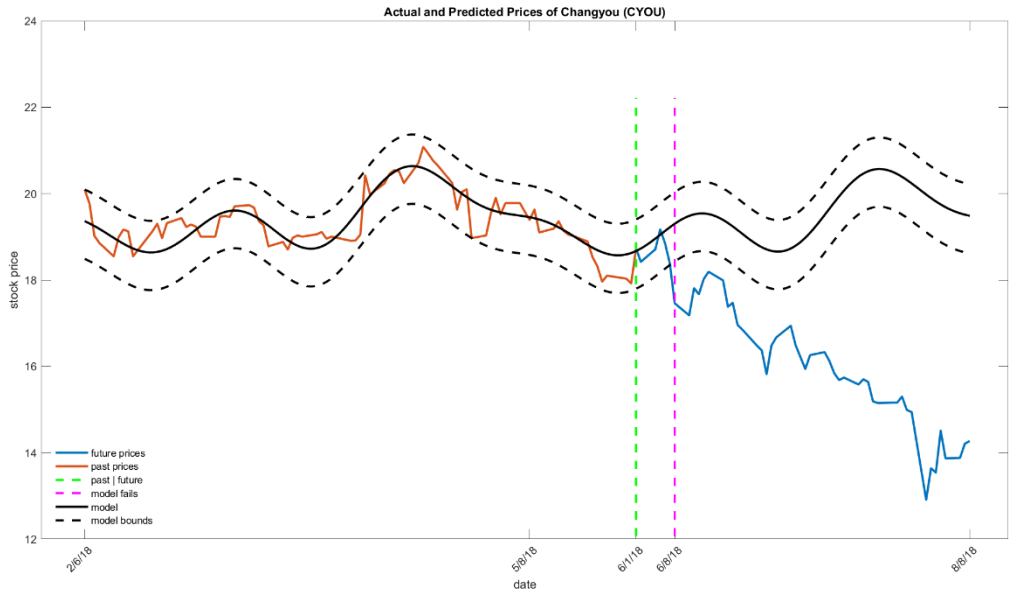
All the models fail when the Commvault stock price spikes. The Fourier Series was not as accurate as for the other stocks because only 38 stock prices were used by the model. This is likely the cause of the model's inaccuracy.

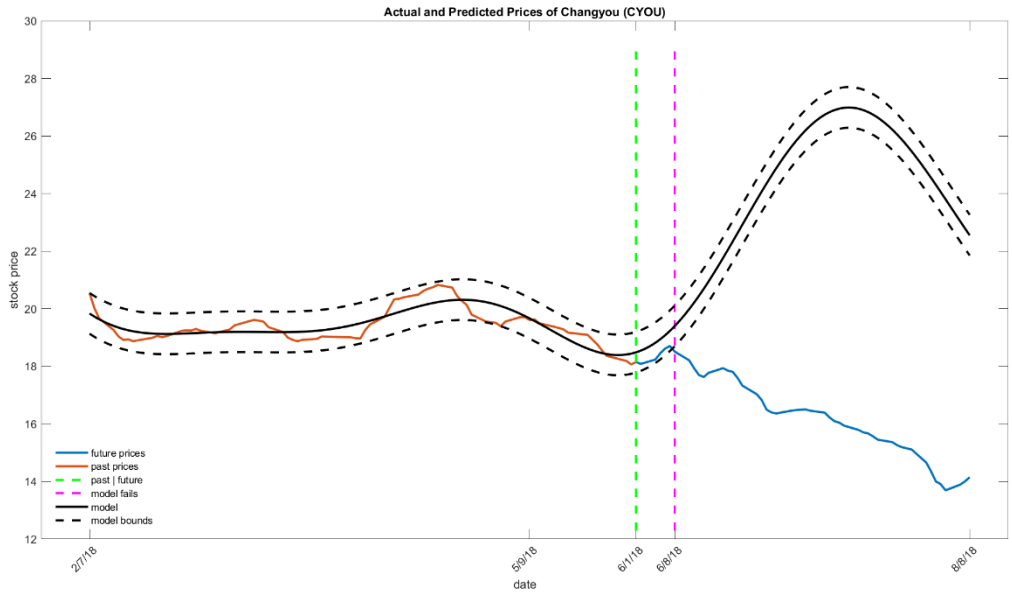
Changyou (CYOU)

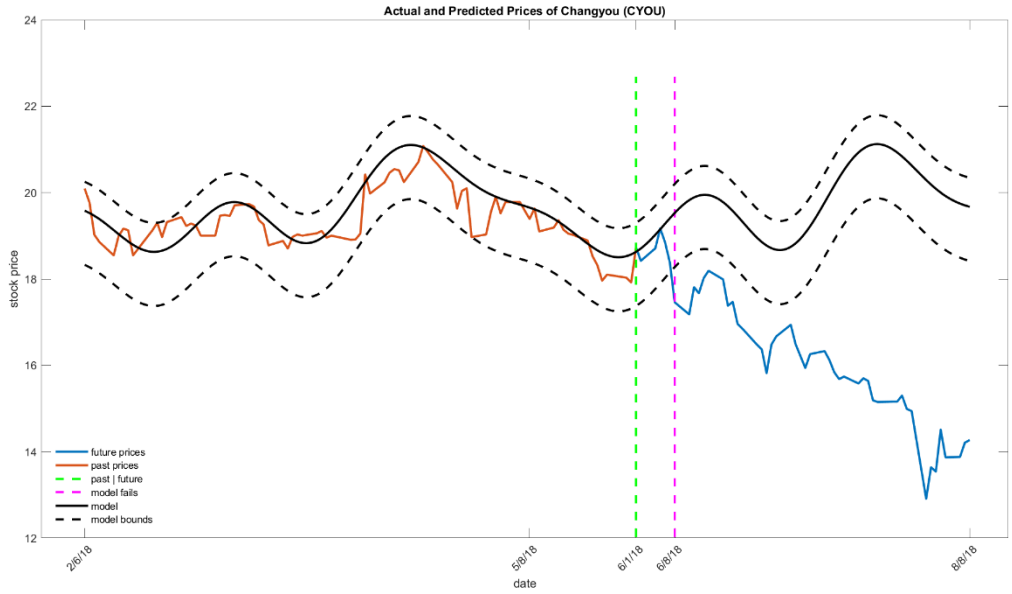

Sample Autocorrelation Function
Sample Autocorrelation
Lag
1
0.5
0
-0.5
0
50
100
150
200
250

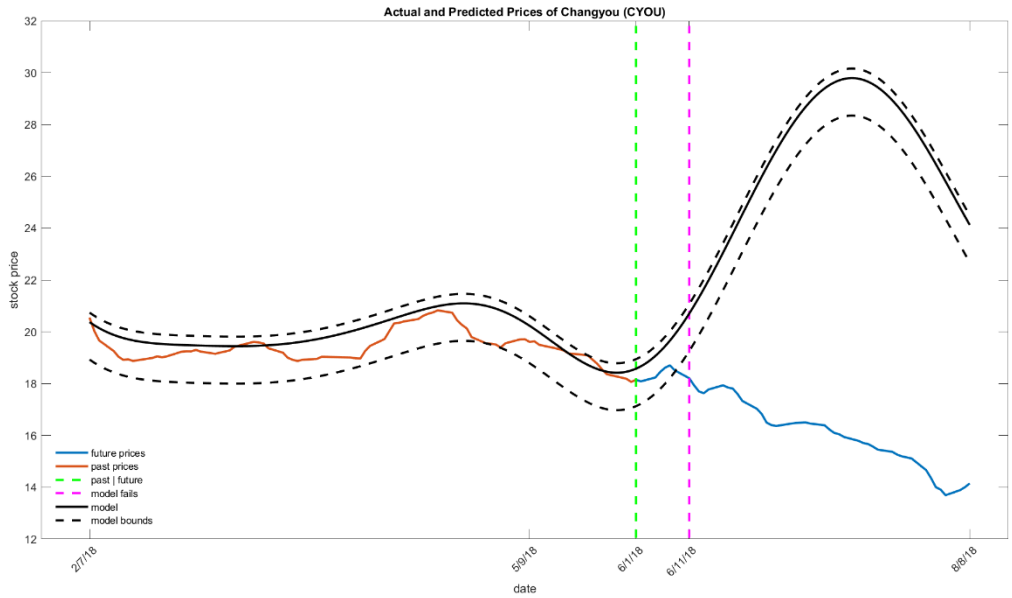

Stock Prices and Trend Line for the Stock Changyou (CYOU)
stock price
date
future prices
past prices
past|future
Trend Line
22
21
20
19
18
17
16
15
14
13
12
2/6/18
5/8/18
6/1/18
8/8/18


Differences and Fourier Model for the Stock Changyou (CYOU)
difference
date
4
2
0
-2
-4
-6
-8
-10
2/6/18
5/8/18
6/1/18
8/8/18
future differences
past differences
past|future
Fourier Model


Actual and Predicted Prices of Changyou (CYOU)
stock price
date
24
22
20
18
16
14
12
2/6/18
5/8/18
6/1/18
6/8/18
8/8/18
future prices
past prices
past | future
model fails
model
model bounds

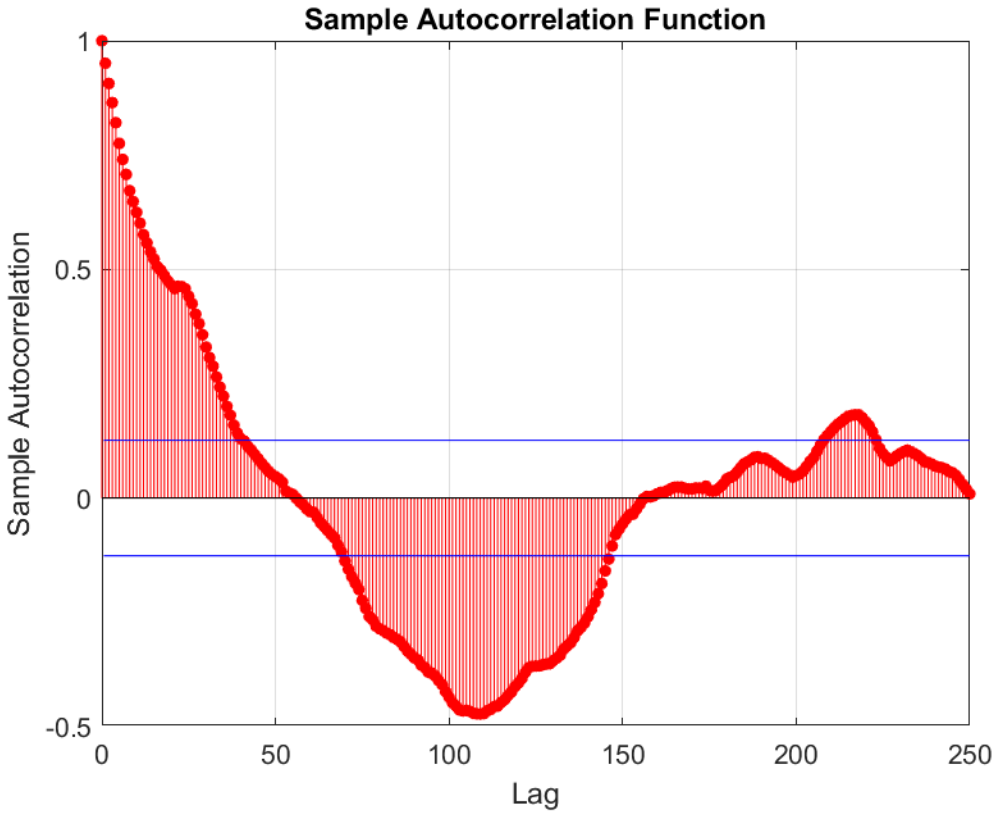

Actual and Predicted Prices of Changyou (CYOU)
stock price
date
30
28
26
24
22
20
18
16
14
12
2/7/18
5/9/18
6/1/18
6/8/18
8/8/18
future prices
past prices
past | future
model fails
model
model bounds

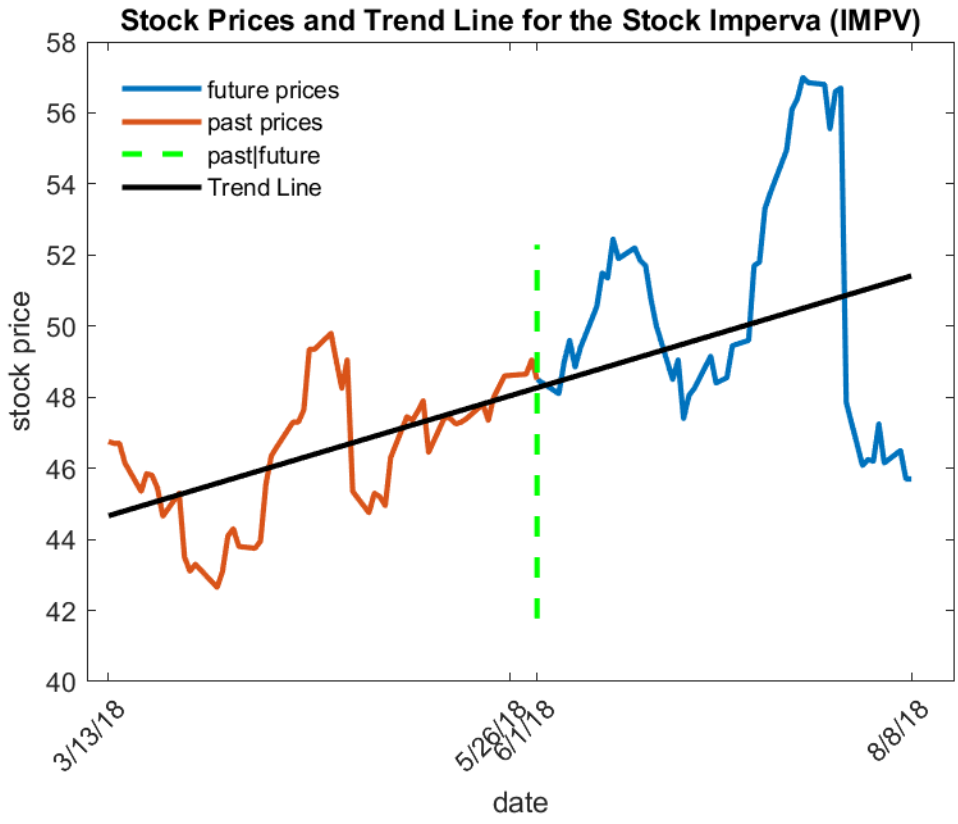

Actual and Predicted Prices of Changyou (CYOU)
stock price
date
24
22
20
18
16
14
12
2/6/18
5/8/18
6/1/18
6/8/18
8/8/18
future prices
past prices
past | future
model fails
model
model bounds

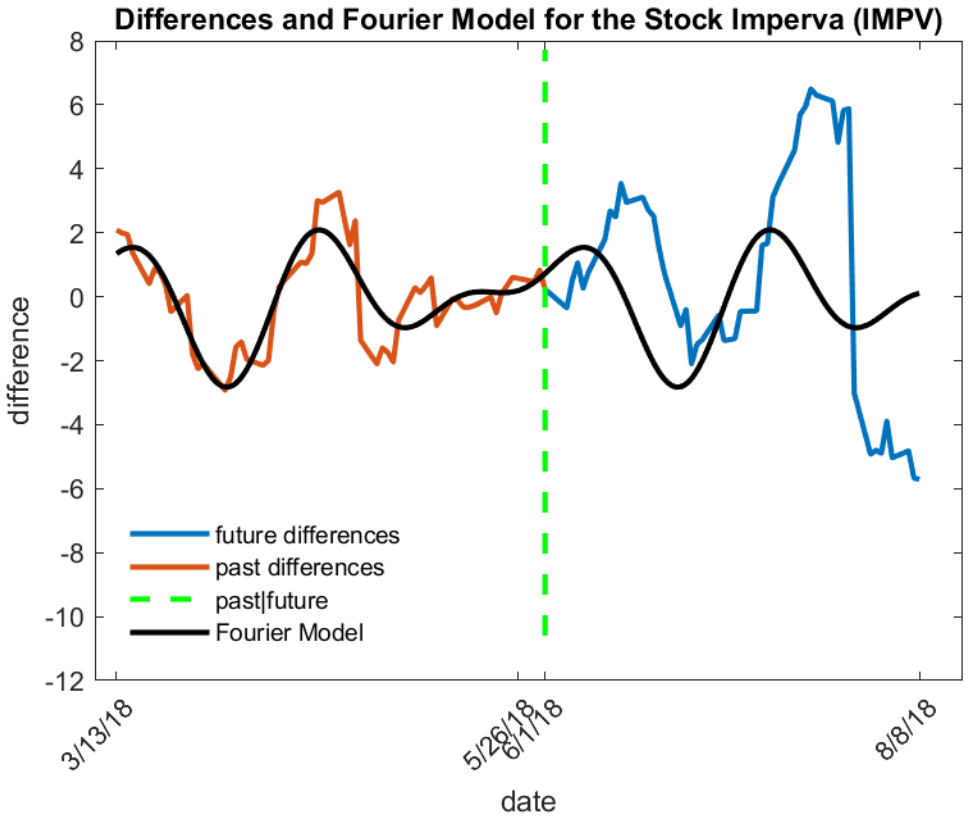


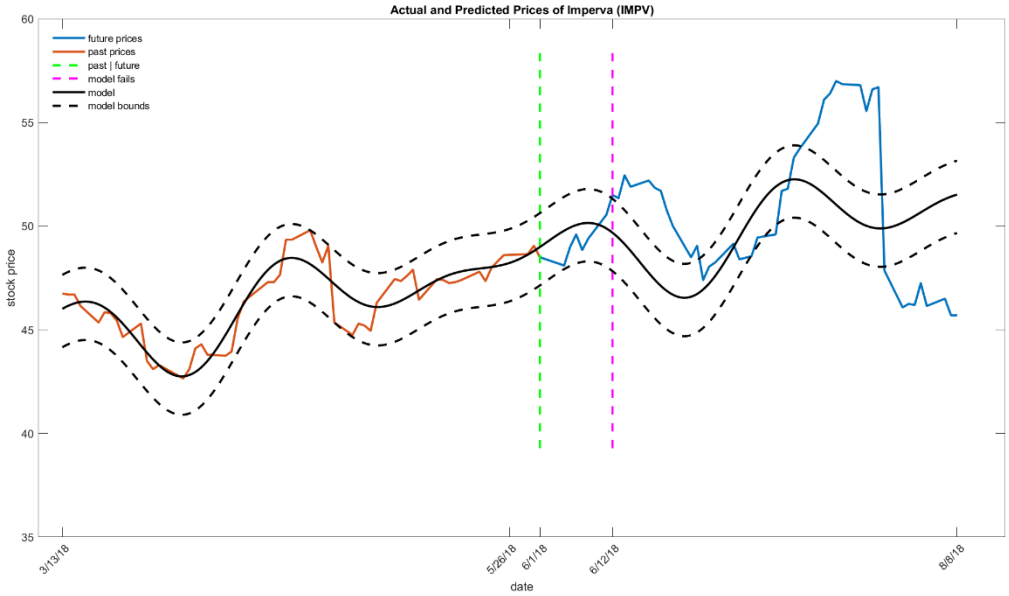
All the models have difficulty predicting the sudden downward trend of the Changyou stock. Before 6/1/18, the stock did not have a substantial upward or downward trend.

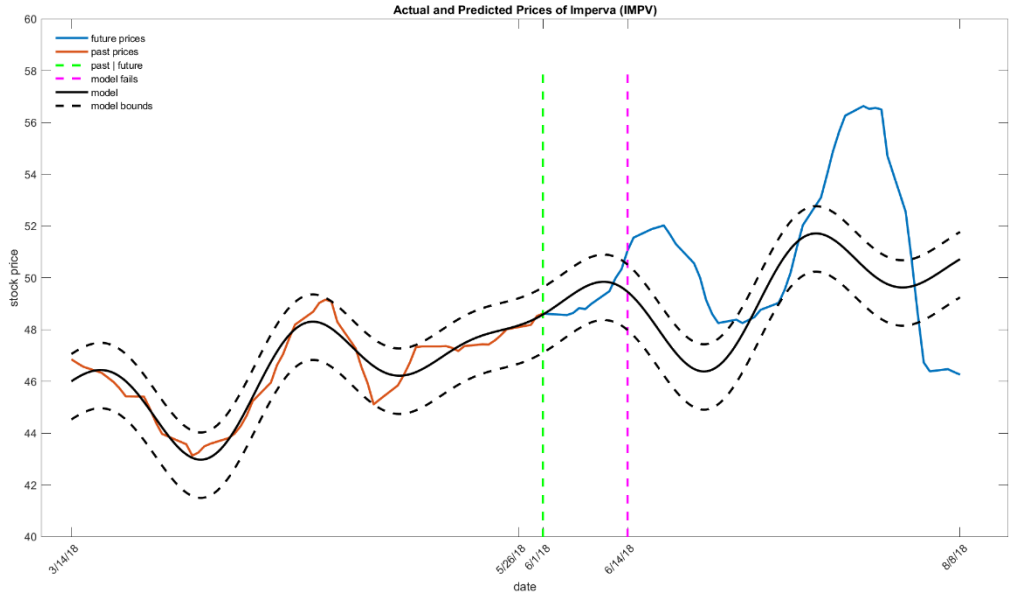
Imperva (IMPV)

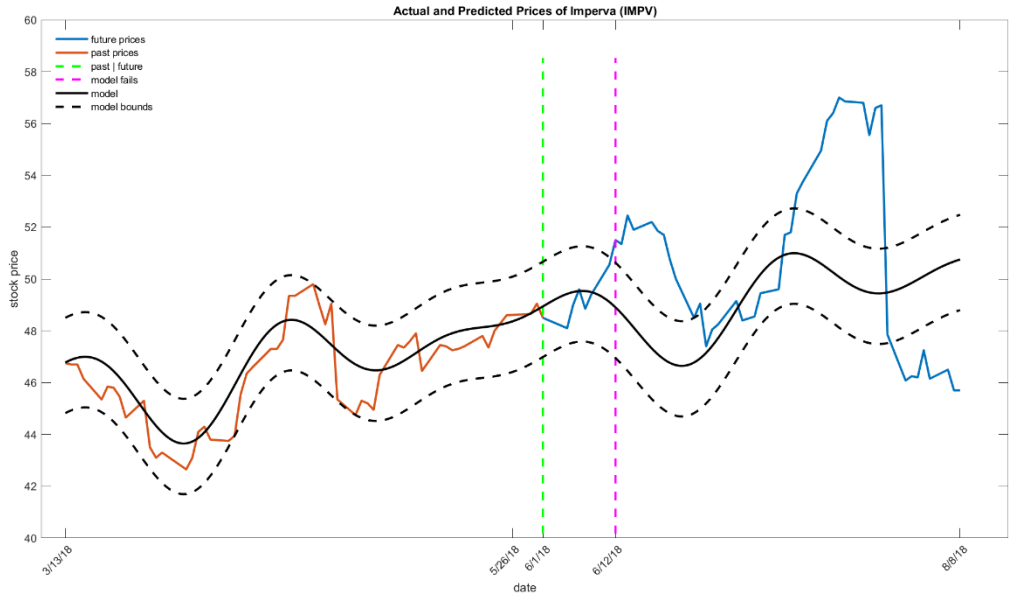

Sample Autocorrelation Function
Sample Autocorrelation
Lag
1
0.5
0
-0.5
0
50
100
150
200
250

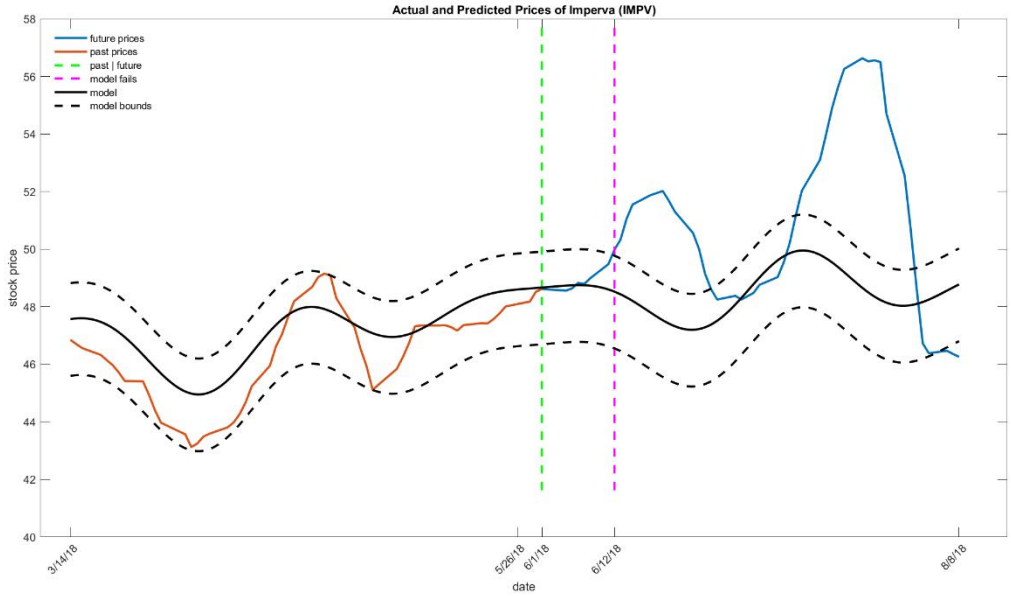

Stock Prices and Trend Line for the Stock Imperva (IMPV)
future prices
past prices
past|future
Trend Line
stock price
date
58
56
54
52
50
48
46
44
42
40
3/13/18
5/26/18
6/1/18
8/8/18


Differences and Fourier Model for the Stock Imperva (IMPV)
difference
date
future differences
past differences
past|future
Fourier Model
3/13/18
5/26/18
6/1/18
8/8/18


Actual and Predicted Prices of Imperva (IMPV)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
3/13/18
5/26/18
6/1/18
6/12/18
8/8/18

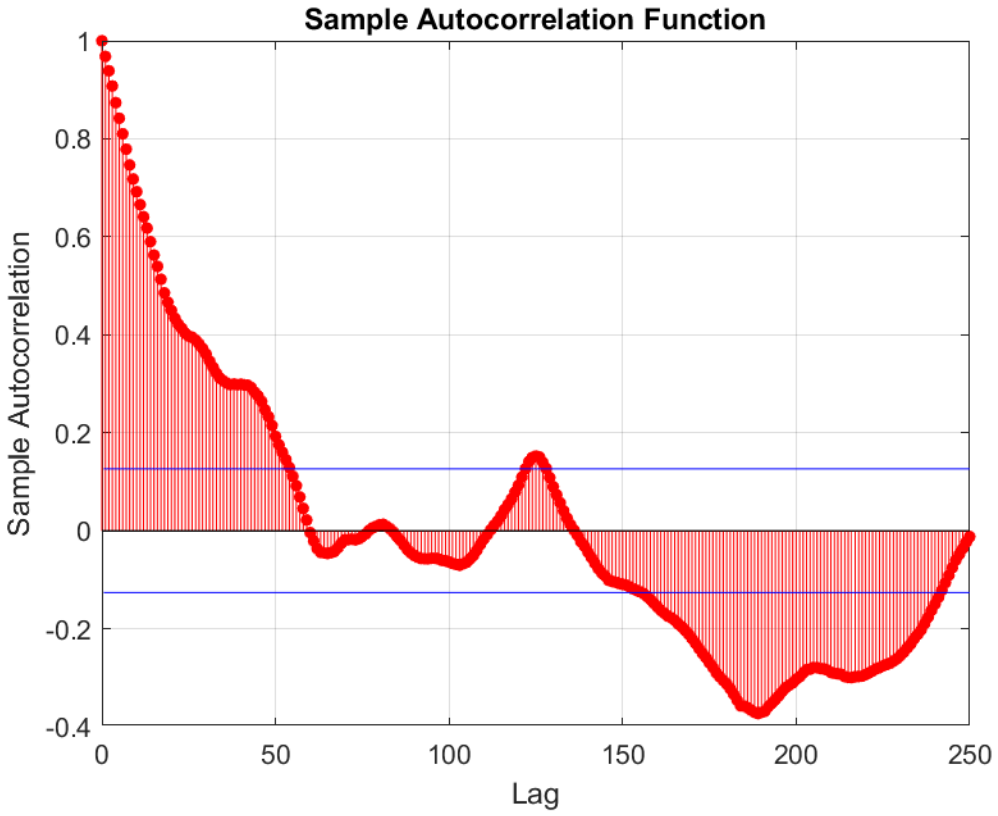

Actual and Predicted Prices of Imperva (IMPV)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
3/14/18
5/26/18
6/1/18
6/14/18
8/8/18

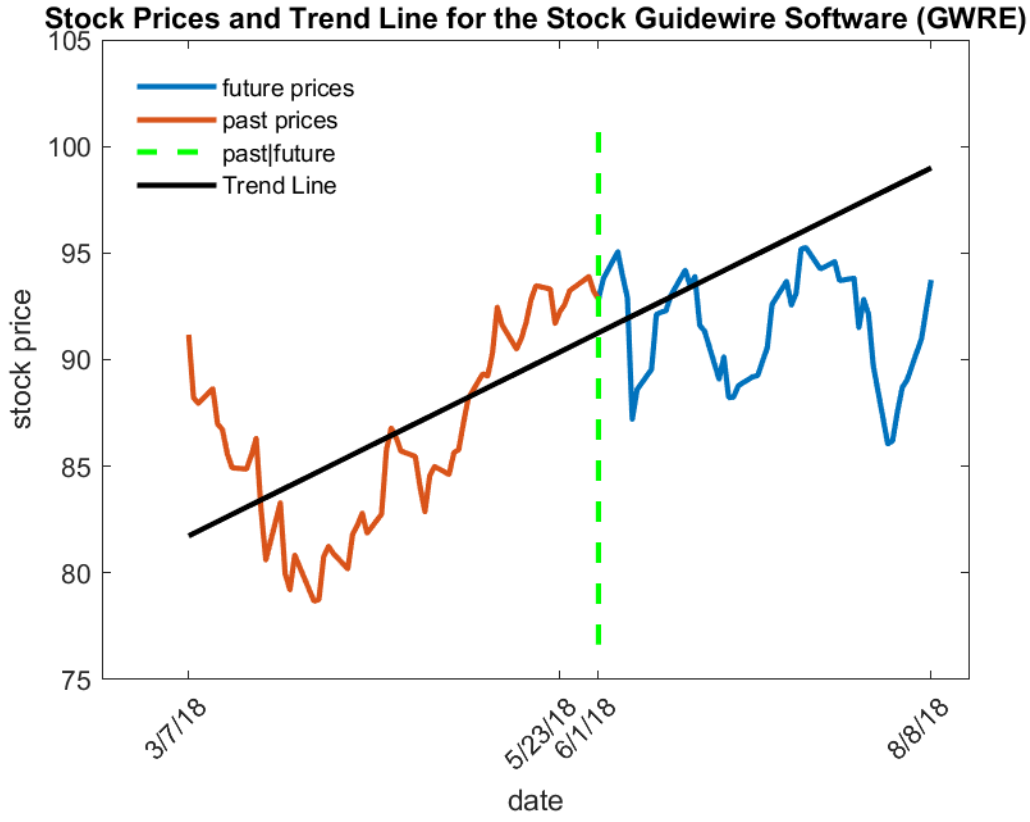

Actual and Predicted Prices of Imperva (IMPV)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
3/13/18
5/26/18
6/1/18
6/12/18
8/8/18

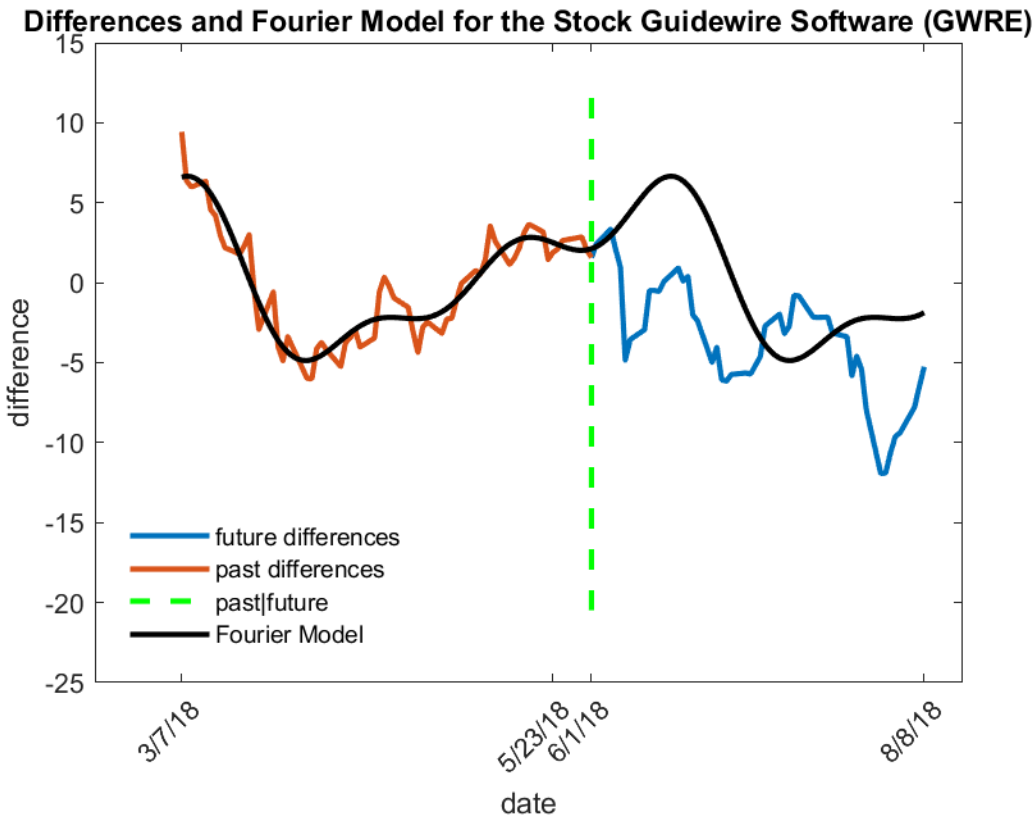


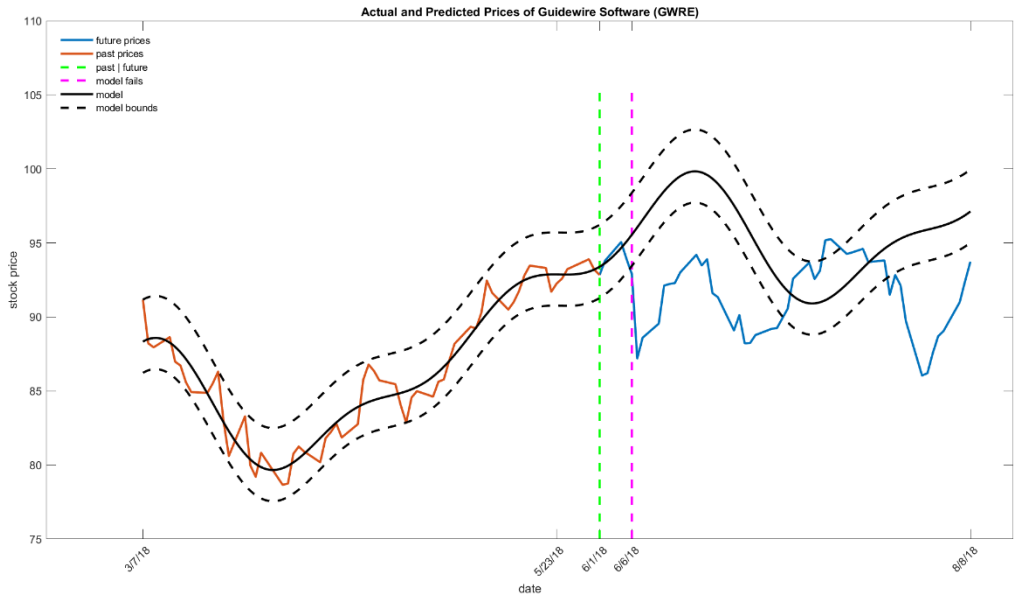
The models all fail at the first peak of the Imperva stock price. The Fourier Series predicts that the trend line residuals will decrease rather than peak. This caused the models to fail.

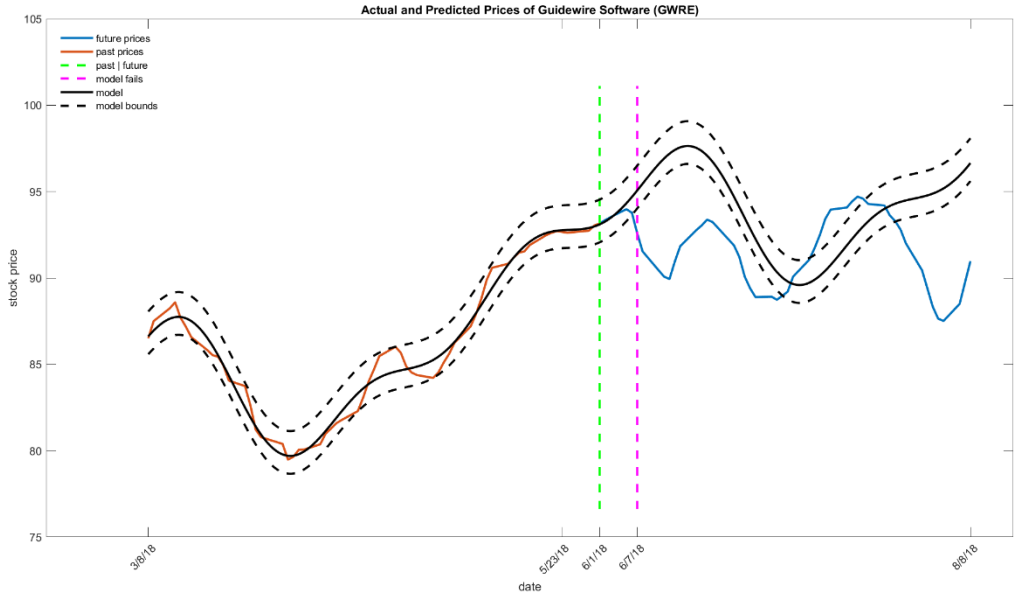
Guidewire Software (GWRE)

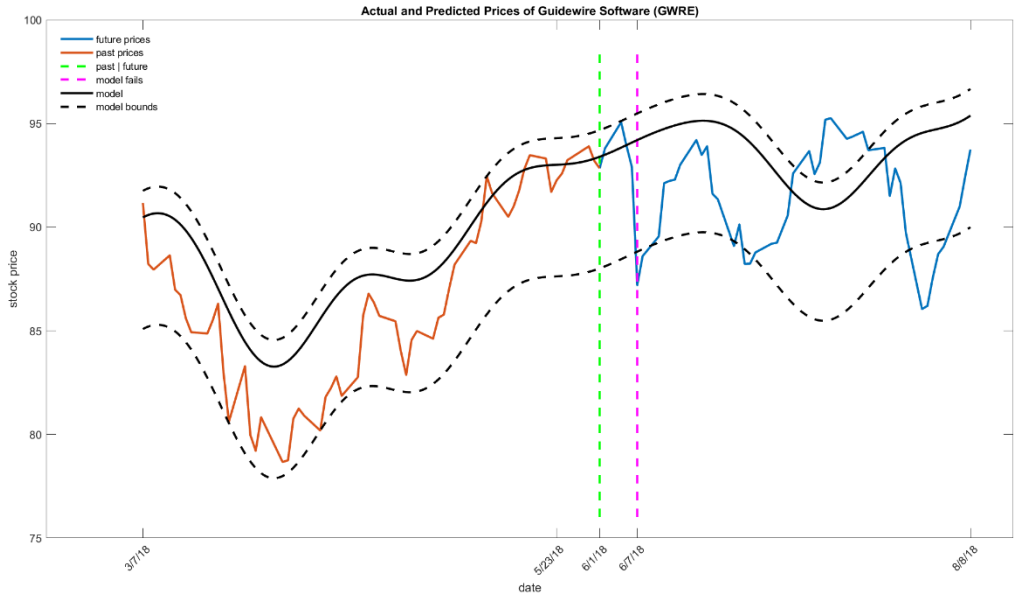

Sample Autocorrelation Function
Sample Autocorrelation
Lag
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
50
100
150
200
250

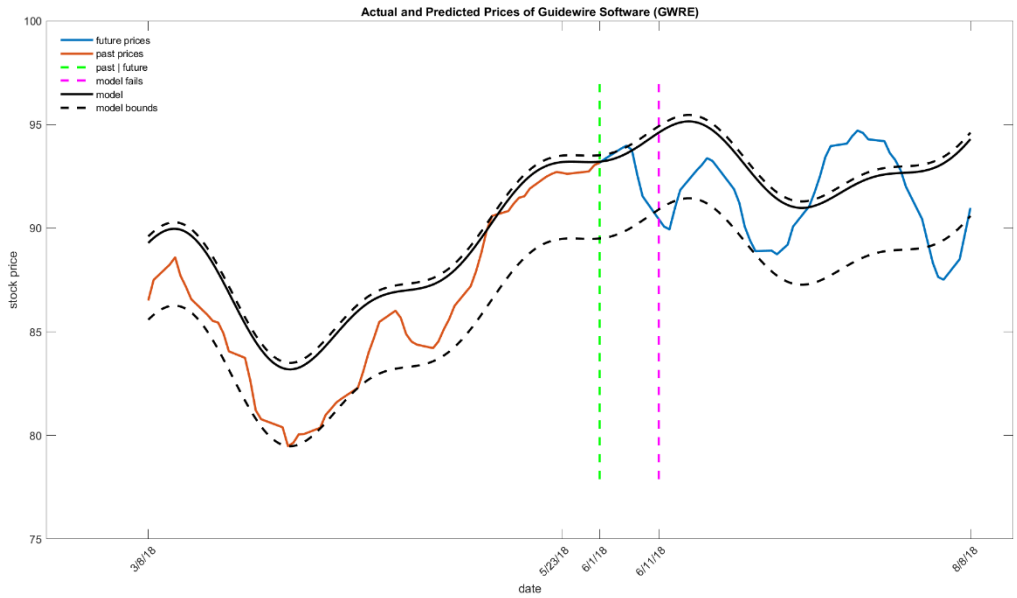

Stock Prices and Trend Line for the Stock Guidewire Software (GWRE)
future prices
past prices
past|future
Trend Line
stock price
date
105
100
95
90
85
80
75
3/7/18
5/23/18
6/1/18
8/8/18


Differences and Fourier Model for the Stock Guidewire Software (GWRE)
difference
date
future differences
past differences
past|future
Fourier Model
3/7/18
5/23/18
6/1/18
8/8/18


Actual and Predicted Prices of Guidewire Software (GWRE)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
3/7/18
5/23/18
6/1/18
6/6/18
8/6/18

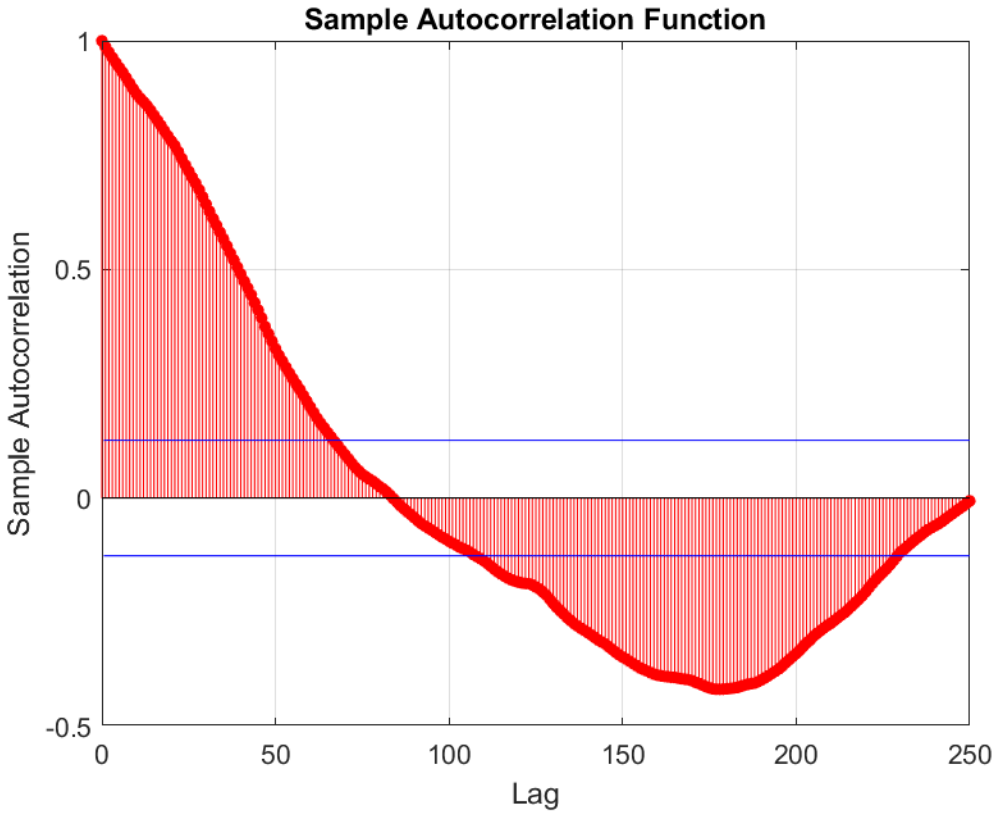

Actual and Predicted Prices of Guidewire Software (GWRE)
future prices
past prices
past | future
model fails
model
model bounds
stock price
105
100
95
90
85
80
75
3/8/18
5/23/18
6/1/18
6/7/18
8/8/18
date

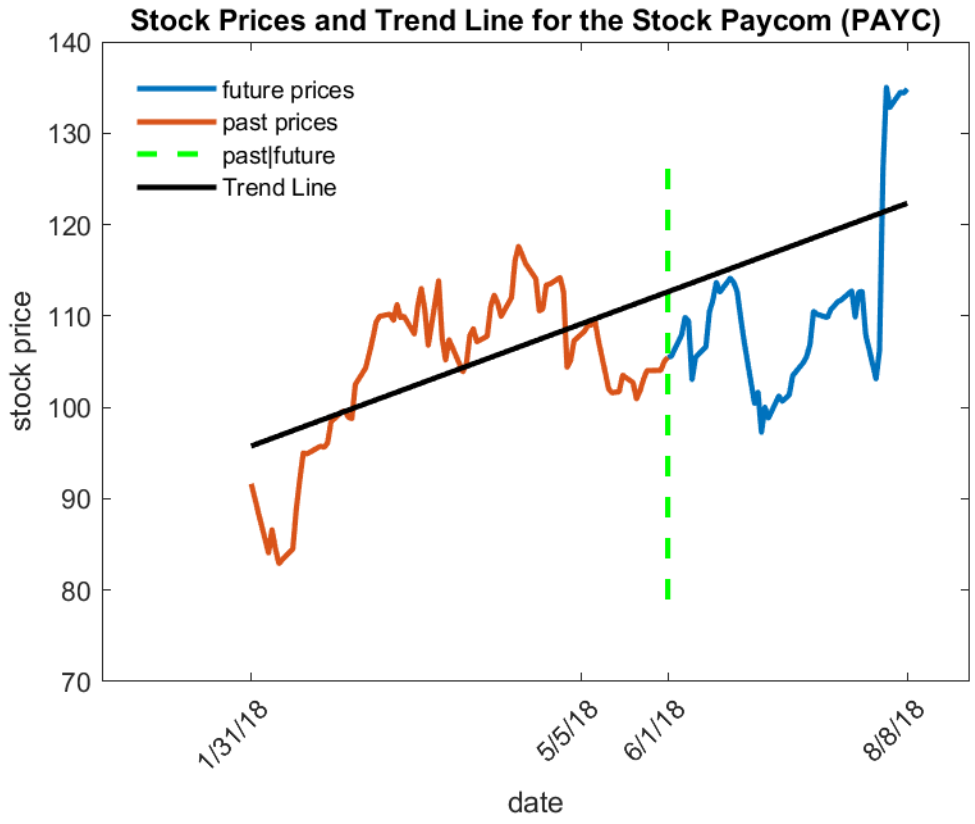

Actual and Predicted Prices of Guidewire Software (GWRE)
future prices
past prices
past | future
model fails
model
model bounds
stock price
100
95
90
85
80
75
3/7/18
5/23/18
6/1/18
6/7/18
8/8/18
date

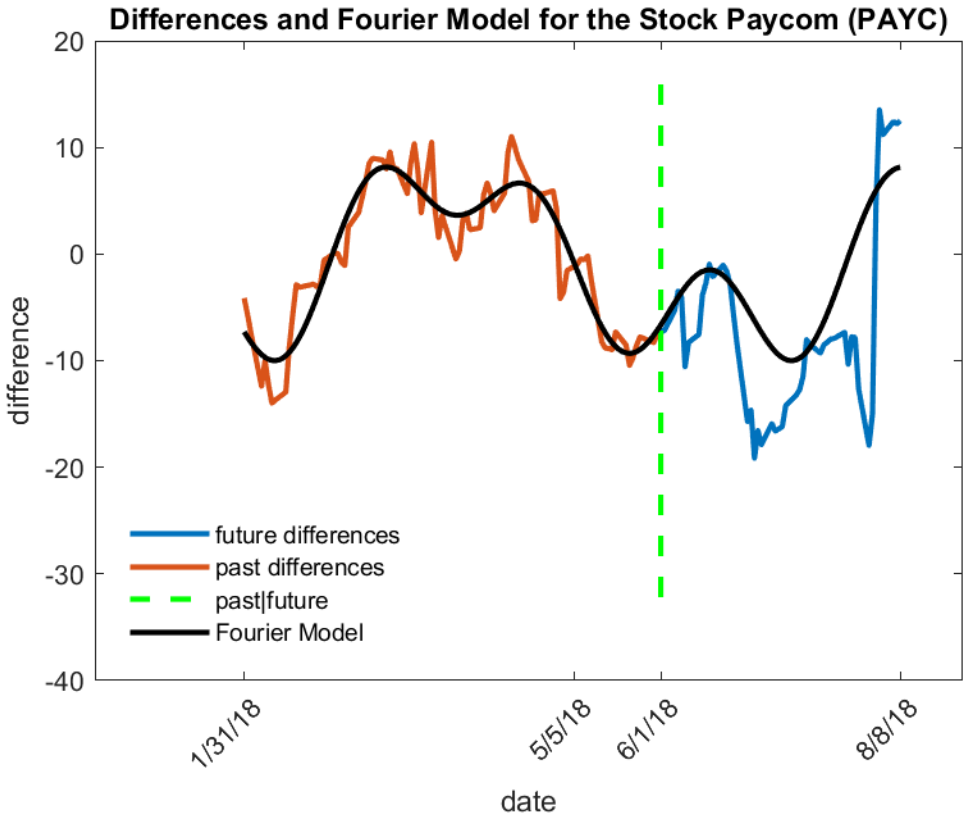


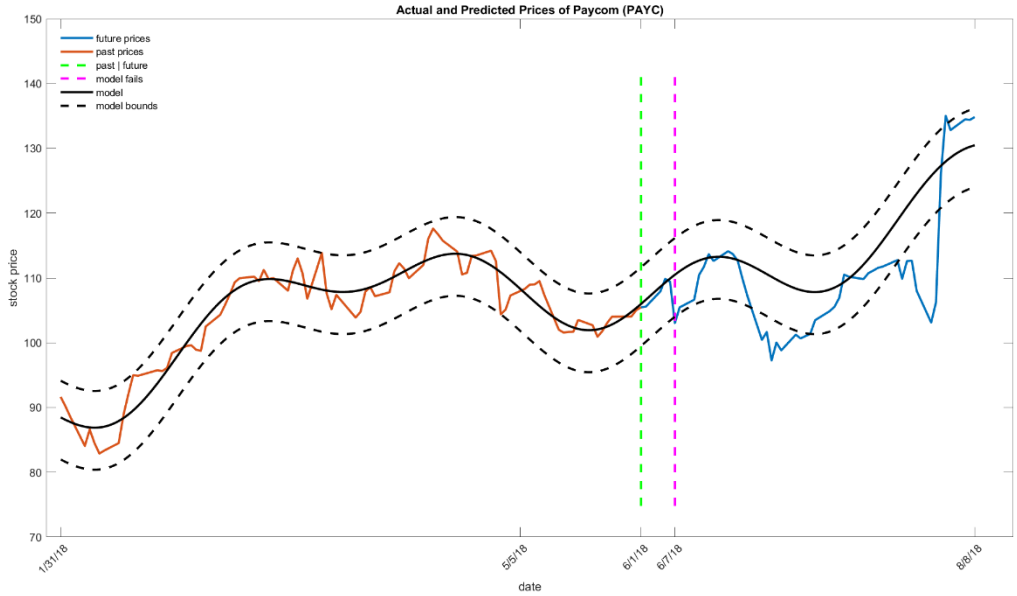
The models fail when Guidewire Software stock price sharply drops.

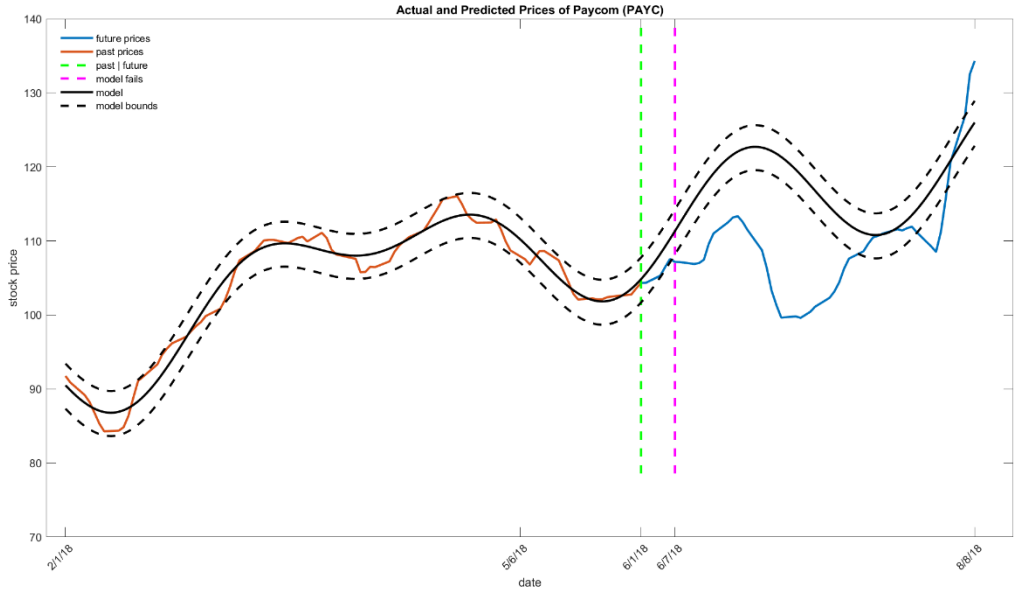
Paycom (PAYC)

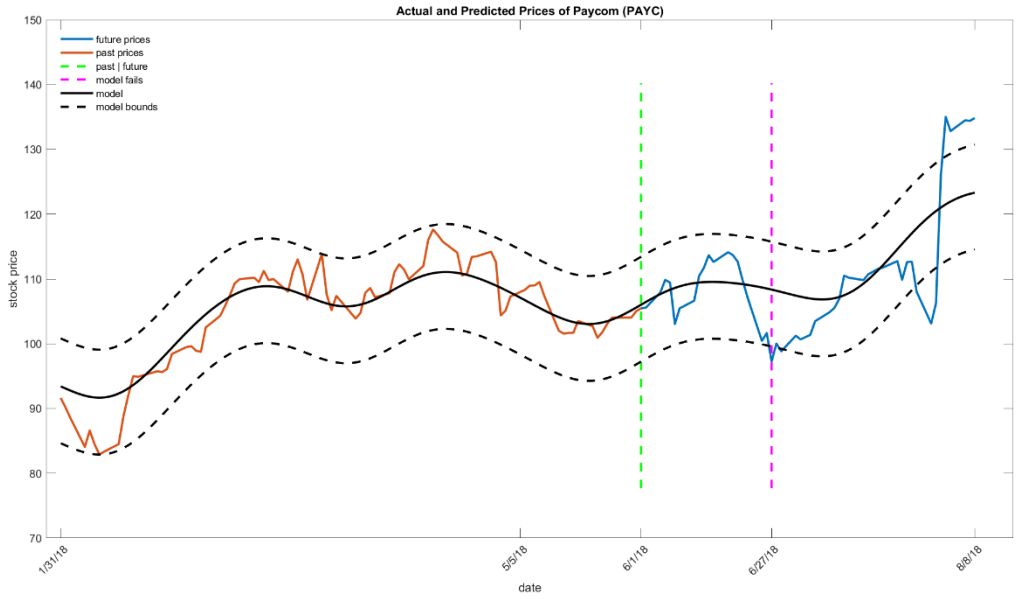

Sample Autocorrelation Function
Sample Autocorrelation
Lag
1
0.5
0
-0.5
0
50
100
150
200
250

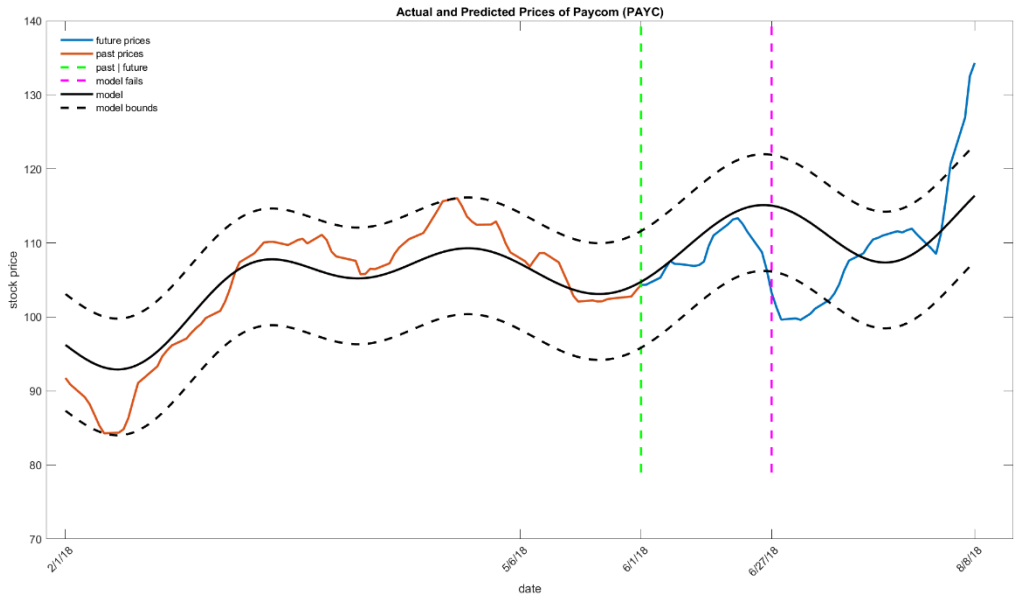

Stock Prices and Trend Line for the Stock Paycom (PAYC)
future prices
past prices
past|future
Trend Line
stock price
date
140
130
120
110
100
90
80
70
1/31/18
5/5/18
6/1/18
8/8/18


Differences and Fourier Model for the Stock Paycom (PAYC)
20
10
0
-10
-20
-30
-40
difference
future differences
past differences
past|future
Fourier Model
1/31/18
5/5/18
6/1/18
8/8/18
date


Actual and Predicted Prices of Paycom (PAYC)
future prices
past prices
past | future
model fails
model
model bounds
150
140
130
120
110
100
90
80
70
stock price
1/31/18
5/5/18
6/1/18
6/7/18
8/8/18
date

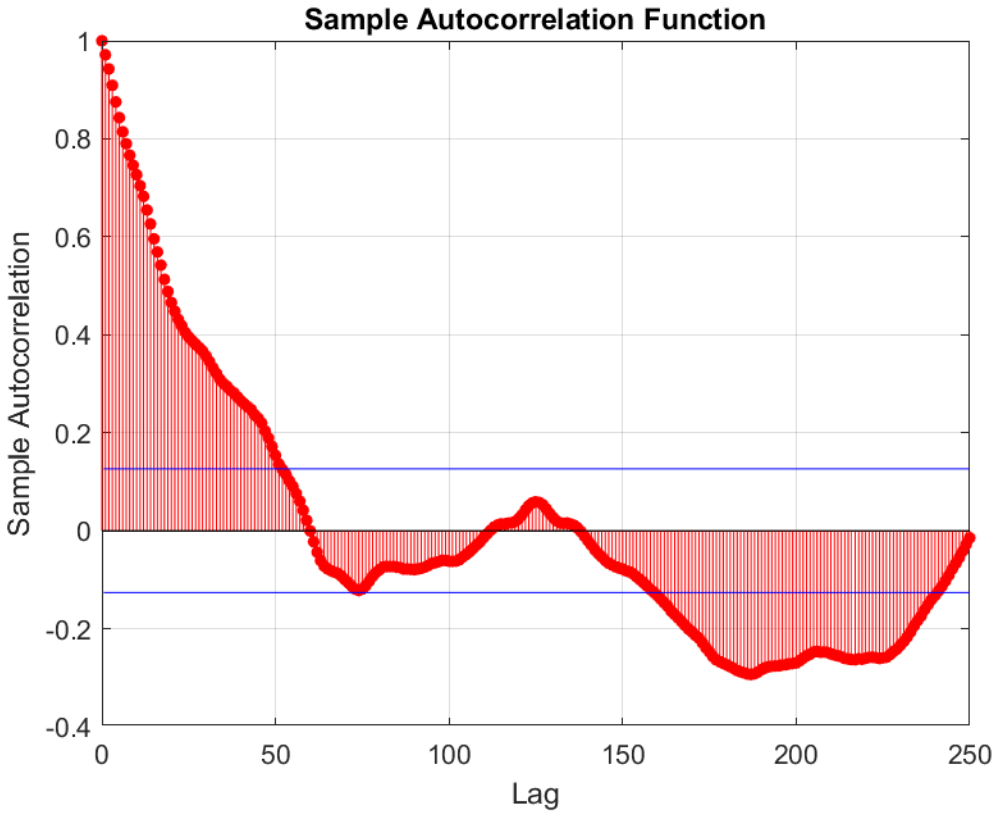

Actual and Predicted Prices of Paycom (PAYC)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
2/1/18
5/6/18
6/1/18
6/7/18
8/8/18

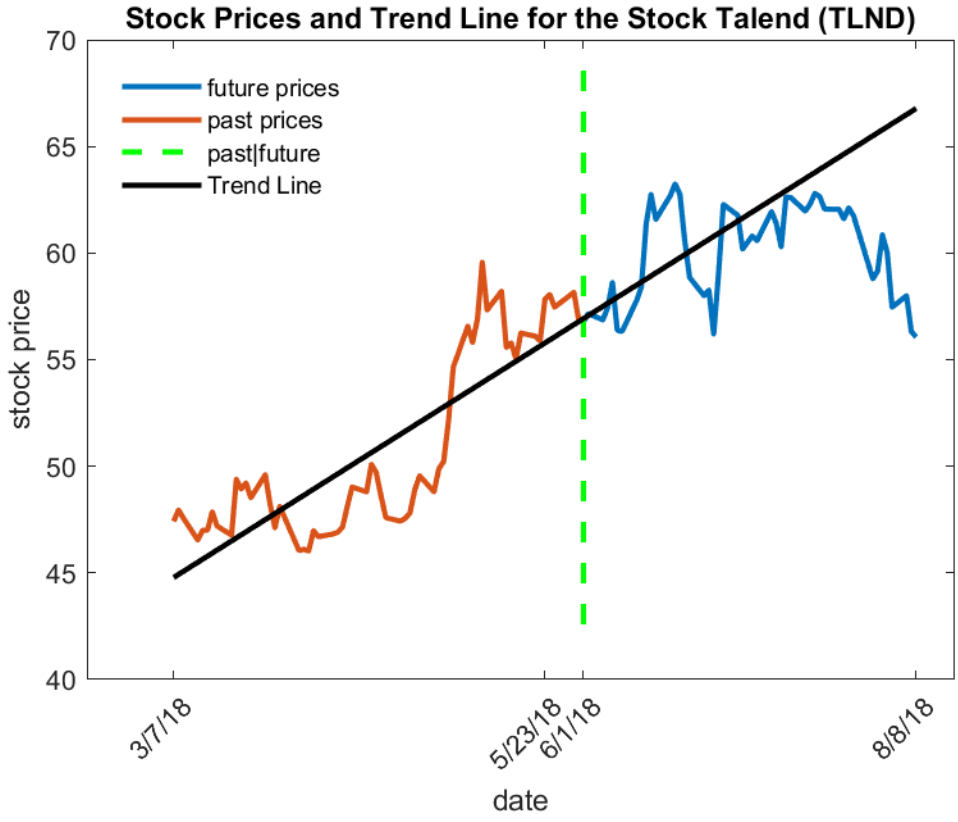

Actual and Predicted Prices of Paycom (PAYC)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
1/31/18
5/5/18
6/1/18
6/27/18
8/8/18

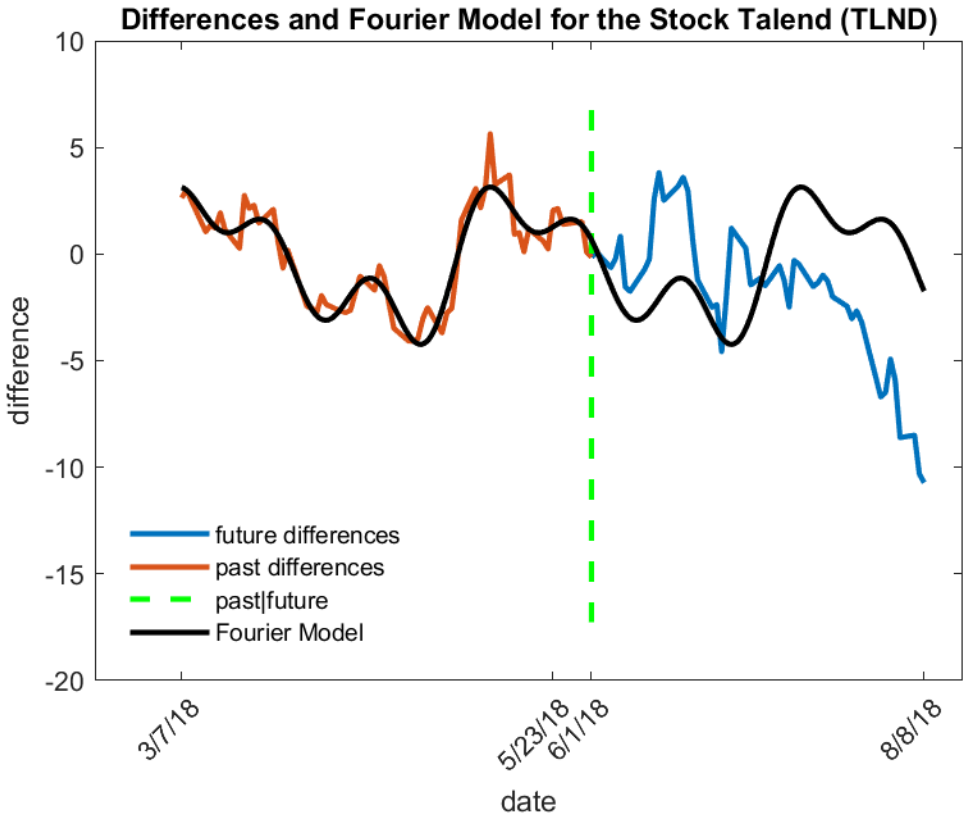


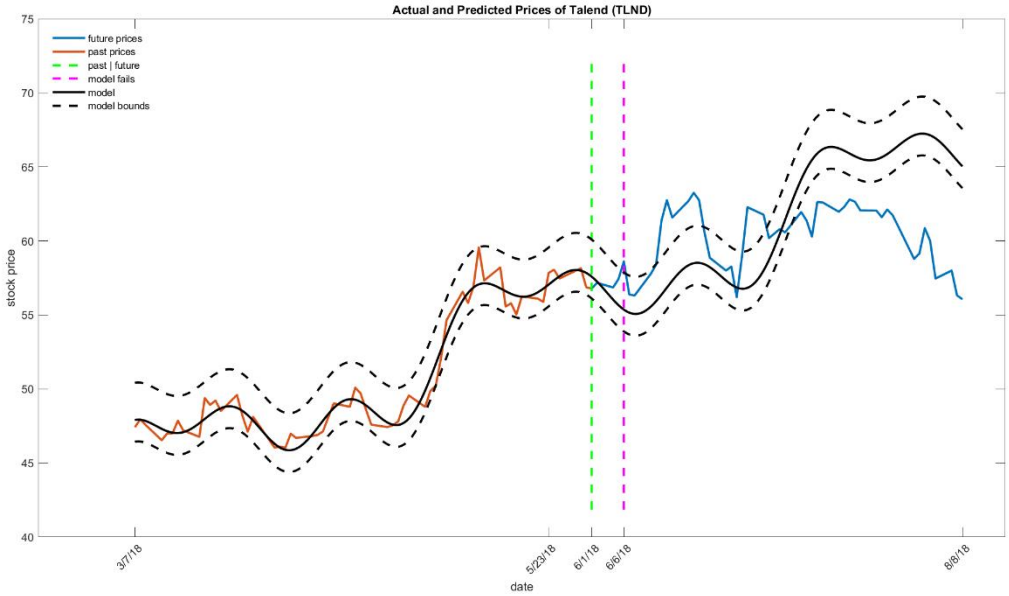
The prototype model without moving averages barely exceeds a distance 105% of the margin of error from the stock price when it fails. With a slightly larger margin of error, it would have failed when the stock sharply dropped. The prototype model with moving averages failed early because the Fourier model predicted a large peak in the price which did not occur in reality. The index models lasted until the sharp drop in price because they had a wider margin of error than the prototype model.

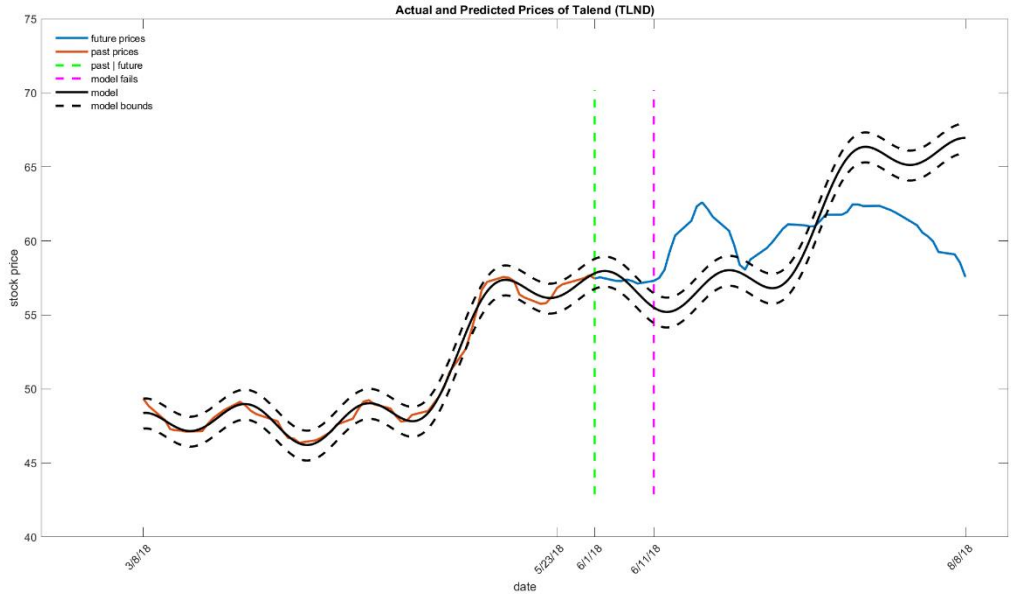
Talend (TLND)

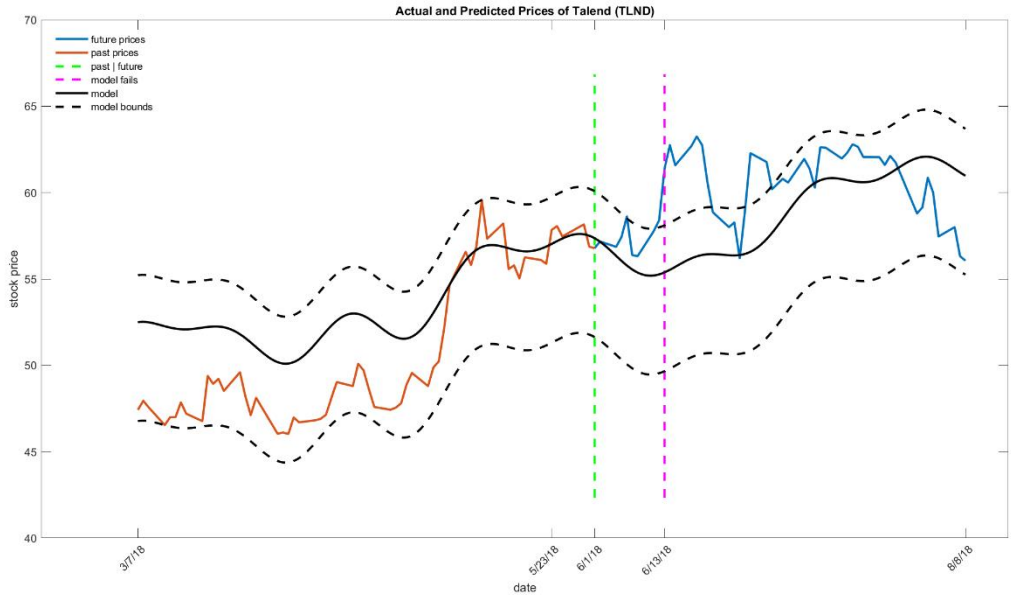

Sample Autocorrelation Function
Sample Autocorrelation
Lag
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
50
100
150
200
250

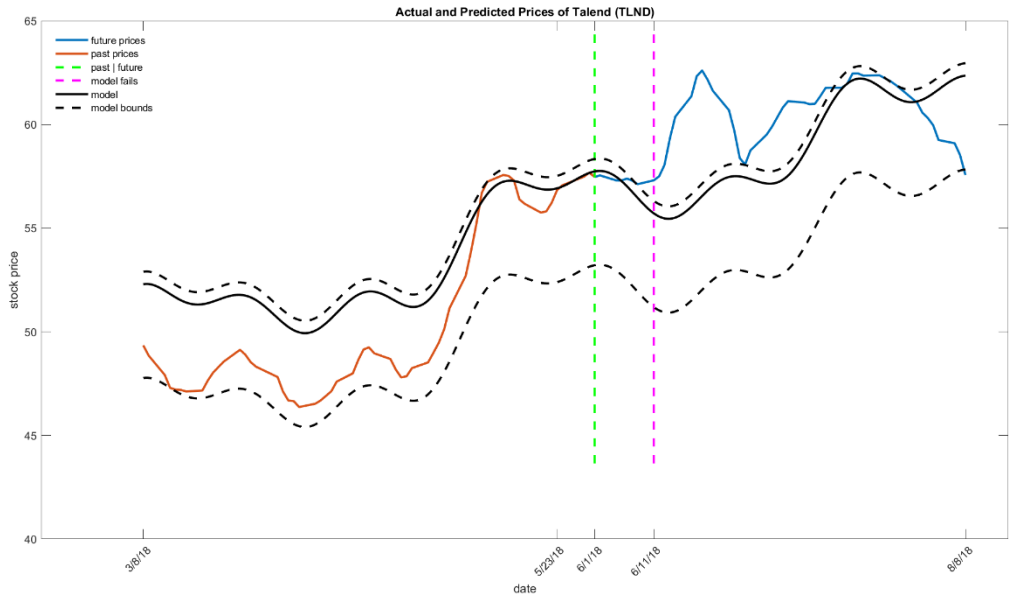

Stock Prices and Trend Line for the Stock Talend (TLND)
future prices
past prices
past|future
Trend Line
stock price
date
70
65
60
55
50
45
40
3/7/18
5/23/18
6/1/18
8/8/18


Differences and Fourier Model for the Stock Talend (TLND)
10
5
0
-5
-10
-15
-20
difference
future differences
past differences
past|future
Fourier Model
3/7/18
5/23/18
6/1/18
8/8/18
date


Actual and Predicted Prices of Talend (TLND)
future prices
past prices
past | future
model fails
model
model bounds
75
70
65
60
55
50
45
40
stock price
3/7/18
5/23/18
6/1/18
6/6/18
8/6/18
date


Actual and Predicted Prices of Talend (TLND)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
3/8/18
5/23/18
6/1/18
6/11/18
8/8/18


Actual and Predicted Prices of Talend (TLND)
future prices
past prices
past | future
model fails
model
model bounds
stock price
date
3/7/18
5/23/18
6/1/18
6/13/18
8/8/18



The prototype model barely past a distance of 105% of the margin of error from the stock price. It should have failed when the model peaked. The prototype model with moving averages failed early because it had a margin of error that was very small compared to the change in the stock within the time analyzed. The index models failed when the stock price peaked.

Appendix: Matlab Code

DataGeneratorNew

```
%Generates Performance Data for the Various Models

close all;
clear all;

%stock symbols and names
stocks={'INXN','WUBA','PRGS','BLKB','CVLT','CYOU','IMPV','G
WRE','PAYC','TLND'};
stocknames={'Interxion Holdings','58.com','Progress
Software','Black Baud',...
     'Commvalut','Changyou','Imperva','Guidewire
Software','Paycom','Talend'};
fouriercutoff1=40;
fouriercutoff2=60;
%Defines number of terms of moving average
termsinavg=5;
%maximum embedding dimension
mmax=3;
%maximum number of iterations for Lyapunov
maxiter=10;
presentdate='09-August-2018';
tic
%Additional Data from Yahoo Finance
[paststockvalues,pastdates,futurestockvalues,futuredates,..
.
    paststockaverages,futurestockaverages,pastdatesavgs,...
    SP500values,SP500dates,SP500avgs,SP500datesavgs]...
    =DataCollector(stocks,termsinavg,'01-June-2018','01-
June-2018','02-June-2017',presentdate);
toc
tic
%Gets Data For Prototype Model
[percentmargins(1,:),meanmargins(1),stdmargins(1),dayspredi
cted(1,:),meandays(1),stddays(1),vol(1,:)]...
```

```
=PrototypeModelDataNew(paststockvalues,pastdates,futurestoc
kvalues,futuredates,fouriercutoff1,fouriercutoff2,stocks,st
ocknames,1,presentdate);
%Gets Data For Prototype Model with Moving Averages
[percentmargins(2,:),meanmargins(2),stdmargins(2),dayspredi
cted(2,:),meandays(2),stddays(2),volavg]...

=PrototypeModelDataNew(paststockaverages,pastdatesavgs,futu
restockaverages,futuredates,fouriercutoff1,fouriercutoff2,s
tocks,stocknames,2,presentdate);
%Gets Data For Index Model
[percentmargins(3,:),meanmargins(3),stdmargins(3),dayspredi
cted(3,:),meandays(3),stddays(3),correlation,volSP500]...

=IndexModelDataNew(paststockvalues,pastdates,futurestockval
ues,futuredates,SP500dates,SP500values,fouriercutoff1,fouri
ercutoff2,stocks,stocknames,1,presentdate);
%Gets Data For Index Model with Moving Averages
[percentmargins(4,:),meanmargins(4),stdmargins(4),dayspredi
cted(4,:),meandays(4),stddays(4),correlationavgs,volSP500av
gs]...

=IndexModelDataNew(paststockaverages,pastdatesavgs,futurest
ockaverages,futuredates,SP500datesavgs,SP500avgs,fouriercut
off1,fouriercutoff2,stocks,stocknames,2,presentdate);
percentmarginsdisplay=round(percentmargins*100,2);
numericaldata=[];
for cnt1=1:4

numericaldata=[numericaldata,dayspredicted(cnt1,:)',percent
marginsdisplay(cnt1,:)'];
end
save('numericaldata');


means=[];
 for cnt=1:4

means=[means,round(meandays(cnt),1),round(meanmargins(cnt)*
100,2)];
 end
stds=[];
for cnt=1:4
```

```
stds=[stds,round(stddays(cnt),1)',round(stdmargins(cnt)'*10
0/2,2)];
end

numericaldata=[numericaldata;means;stds];
save('alldata2')
xlswrite('ModelPredictionData2',numericaldata,'D4:K15');
xlswrite('ModelPredictionData2',stocknames','B4:B13');
xlswrite('ModelPredictionData2',stocks','C4:C13');
toc
```

DataGeneratorAvgs

```
close all;
clear all;

%stock symbols and names
stocks={'INXN','WUBA','PRGS','BLKB','CVLT','CYOU','IMPV','G
WRE','PAYC','TLND'};
stocknames={'Interxion Holdings','58.com','Progress
Software','Black Baud',...
     'Commvalut','Changyou','Imperva','Guidewire
Software','Paycom','Talend'};

fouriercutoff1=40;
fouriercutoff2=60;
%Defines number of terms of moving average
termsinavgs=5:15;
%maximum embedding dimension
mmax=3;
%maximum number of iterations for Lyapunov
maxiter=10;
presentdate='09-August-2018';
%Collects Additional Data from Yahoo Finance
tic
[paststockvalues,pastdates,futurestockvalues,futuredates,..
.

paststockaveragesall,futurestockaveragesall,pastdatesavgsal
l,...

SP500values,SP500dates,SP500avgsall,SP500datesavgsall]...
```

```
    =DataCollectorAvgs(stocks,termsinavgs,'01-June-
2018','01-June-2018','02-June-2017');
toc
tic
for cnt=termsinavgs
%Selects Information Corresponding to the Number of Terms
in the Moving
%Average
termsinavg=cnt;
index=cnt-termsinavgs(1)+1;
paststockaverages=paststockaveragesall{index};
futurestockaverages=futurestockaveragesall{index};
pastdatesavgs=pastdatesavgsall{index};
SP500datesavgs=SP500datesavgsall{index};
SP500avgs=SP500avgsall{index};
%Gets Data For Prototype Model
[percentmargins(1,:),meanmargins(1),stdmargins(1),dayspredi
cted(1,:),meandays(1),stddays(1),vol(1,:)]...

=PrototypeModelDataNew(paststockvalues,pastdates,futurestoc
kvalues,futuredates,fouriercutoff1,fouriercutoff2,stocks,st
ocknames);
%Gets Data For Prototype Model with Moving Averages
[percentmargins(2,:),meanmargins(2),stdmargins(2),dayspredi
cted(2,:),meandays(2),stddays(2),volavg]...

=PrototypeModelDataNew(paststockaverages,pastdatesavgs,futu
restockaverages,futuredates,fouriercutoff1,fouriercutoff2,s
tocks,stocknames);
%Gets Data For Index Model
[percentmargins(3,:),meanmargins(3),stdmargins(3),dayspredi
cted(3,:),meandays(3),stddays(3),correlation,volSP500]...

=IndexModelDataNew(paststockvalues,pastdates,futurestockval
ues,futuredates,SP500dates,SP500values,fouriercutoff1,fouri
ercutoff2,stocks);
%Gets Data For Index Model with Moving Averages
[percentmargins(4,:),meanmargins(4),stdmargins(4),dayspredi
cted(4,:),meandays(4),stddays(4),correlationavgs,volSP500av
gs]...

=IndexModelDataNew(paststockaverages,pastdatesavgs,futurest
ockaverages,futuredates,SP500datesavgs,SP500avgs,fouriercut
off1,fouriercutoff2,stocks);
```

```
numericaldata=[];
for cnt1=1:4

numericaldata=[numericaldata,dayspredicted(cnt1,:)',percent
margins(cnt1,:)'];
end
numericaldatas(:,:,cnt-termsinavgs(1)+1)=numericaldata;
end
toc
```

get_yahoo_stockdata3

```
function stock = get_yahoo_stockdata3(ticker,d1,d2,freq)
% Updated from v3 when in May 2017, yahoo went and changed
how stock data
% was shown on web pages. From Michael Weidman
%
% INPUTS:
%
% ticker  <-- Yahoo ticker symbol for desired security.
This can be a char
%             string for a single stock, or data can be
retrieved for
%             multiple stocks by using a cellstr array.
%
% d1      <-- start date for data. Can be a matlab
datenumber or a date string.
%             Default = 100 days ago
%
% d2      <-- end date for data. Can be a matlab datenumber
or a date string.
%             Default = today
%
% freq    <-- data frequency 'd' (daily), 'w' (weekly), or
'm' (monthly).
%             Default = 'd'
%
% OUTPUT:
%
% stock  <-- matlab data structure with output data.
%
% Examples:
```

```
%
%  Get data for the past 100 days.
%  stock = get_yahoo_stockdata3('goog');
%  stock = get_yahoo_stockdata3({'goog', 'aapl', 'fb'});
%
%  Get data from 01-Mar-2008 to now.
%  stock = get_yahoo_stockdata3('goog','01-Mar-2008');
%
%  Get data for the past 5 years.
%  stock = get_yahoo_stockdata3('goog', now-5*365, now);
%
%  Get data for specific date range, but weekly instead of
daily
%  stock = get_yahoo_stockdata3({'goog', 'aapl', 'fb'},'01-
Jan-2009','01-Apr-2010','w');
%
% Captain Awesome, November 2017

if nargin<4
  freq = 'd';
end
if nargin<3
  d2 = now;
end
if nargin<1
  d1 = d2-100;
end

d1 = floor(datenum(d1));
d2 = floor(datenum(d2));
ticker = upper(ticker);

if d1>d2
  error('bad date order');
end

if isempty(ticker)
  error('No ticker given.');
end

if sum(strcmpi(freq,{'daily','day','d'}))
  freq = 'd';
elseif sum(strcmpi(freq,{'weekly','week','w','wk'}))
  freq = 'wk';
```

```
elseif sum(strcmpi(freq,{'monthly','month','mmowk'}))
  freq = 'mo';
else
  error('data frequency not recognized');
end

% If given a cellstr array of tickers, then this will
recursively call this
% function for each ticker, output will be a cell array of
stock data
% structures.
if iscell(ticker)
  stock = cellfun(@(x)
get_yahoo_stockdata3(x,d1,d2,freq),...
    ticker, 'uniformoutput', false);
  return
end

clear stockData
stock.ticker     = ticker;
stock.dataSource = 'Yahoo Finance';
stock.dataUpdate = datestr(now,0);
stock.errorMsg   = '';
stock.dataFreq   = freq;

% Yahoo finance uses a unix serial date number, so will
have to convert to
% that.  That's a UNIX timestamp -- the number of seconds
since January 1, 1970.
unix_epoch = datenum(1970,1,1,0,0,0);
d1u = floor(num2str((datenum(d1) - unix_epoch)*86400));
d2u = floor(num2str((datenum(d2) - unix_epoch)*86400));

site=strcat('https://finance.yahoo.com/quote/',ticker,'/his
tory?',...

'period1=',d1u,'&period2=',d2u,'&interval=1',freq,'&filter=
history&',...
  'frequency=1',freq);

[temp, status] = urlread(site);

if ~status
  warning(['stock data download failed: ',ticker]);
```

```
  stock.errorMsg=['stock data download failed (',...
    datestr(now,0),'): ',ticker];
  return
end

%% Check that this is the right ticker data
C = strsplit(temp,'Ta(start) ')';

for k = 1:length(C)

  s = C{k};

  if sum(strfind(lower(s),lower('ticker=')))
    t = strsplit(s,'ticker=');
    t = t{2};
    t = textscan(t,'%s');
    t = t{1}{1};
    if ~strcmp(t, ticker)
      error('ticker mismatch');
    end
    break
  end
  clear s

end

clear k C

%% Get data from 'Historical Prices' section

C = strsplit(temp,'"HistoricalPriceStore":{"prices":[');

if length(C)==1
  % In this case all the data was displayed to the screen
and there was no
  % extra data in "HistoricalPriceStore"

  data = [];
  ddata = [];
  sdata = [];

else
  % In this case only some data was initially displayed and
the rest was
```

```
  % put in this "HistoricalPriceStore" section and a script
would display
  % it as the user scrolled down.

  C = strsplit(C{2},'},');

  n = length(C);
  data = NaN(n,7);  % stock data
  ddata = NaN(n,3); % dividend data
  sdata = NaN(n,3); % split events

  for k=1:n

    s = C{k};

    if length(s)<13
      continue
    end

    if strcmp(s(1:13),'"eventsData":')
      break
    end

    if sum(strfind(lower(s),lower('"splitRatio"')))

      x = textscan(s,['{"date":%f "numerator":%f
"denominator":%f%*s'],...
        'delimiter',',','TreatAsEmpty','null');

      if sum(cellfun('isempty',x))
        error('badness');
      end

      sdata(k,:) = cell2mat(x);
      clear x


    elseif strcmp(s(1:8),'{"date":')
      x = textscan(s,['{"date":%f "open":%f "high":%f
"low":%f "close":%f "volume":%f "adjclose":%f}'],...
        'delimiter',',','TreatAsEmpty','null');

      if sum(cellfun('isempty',x))
        error('badness');
      end
```

```
      data(k,:) = cell2mat(x);
      clear x

    elseif strcmp(s(1:10),'{"amount":')
      x =
textscan(s,['{"amount":%f,"date":%f,"type":"DIVIDEND","data
":%f'],...
        'delimiter','','TreatAsEmpty','null');

      if sum(cellfun('isempty',x))
        error('badness');
      end

      ddata(k,:) = cell2mat(x);
      clear x

    end

    clear s

  end
  clear n k

  % data columns: date, open, high, low, close, volume,
adjclose
  data(isnan(data(:,1)),:) = [];
  data(:,1) =
datenum(datetime(data(:,1),'ConvertFrom','posixtime'));
  data = sortrows(data,1);

  % ddate columns: Amount, date, data
  ddata(isnan(ddata(:,1)),:) = [];
  ddata(:,2) =
datenum(datetime(ddata(:,2),'ConvertFrom','posixtime'));
  ddata = sortrows(ddata,2);

  %sdata columns: Date numerator demonitor
  sdata(isnan(sdata(:,1)),:) = [];
  sdata(:,1) =
datenum(datetime(sdata(:,1),'ConvertFrom','posixtime'));
  sdata = sortrows(sdata,2);

end
```

```
clear C

%% Assign data to output structure

if isempty(ddata)
  stock.dividends = [];
else
  stock.dividends.DateTime = ddata(:,2);
  stock.dividends.Amount = ddata(:,1);
end

if isempty(sdata)
  stock.splits = [];
else
  stock.splits.DateTime = sdata(:,1);
  stock.splits.numerator = sdata(:,2);
  stock.splits.denominator = sdata(:,3);
end


if isempty(data)
  stock.errorMsg=['No data found in stock data download
(',datestr(now,0),'): ',ticker];
  warning(['No data found in stock data download:
',ticker]);
  return
end

stock.range = [datestr(data(1,1),1),...
    ' to ',datestr(data(end,1),1)];

stock.varnotes={...
% Variable          Units   Description
Format
  'DateTime',       '[EST]', 'Date of stock quote',
'yyyy-mm-dd';...
  'openPrice',      '[$]',   'Opening price of stock',
'%.2f';...
  'highPrice',      '[$]',   'High price of stock',
'%.2f';...
  'lowPrice',       '[$]',   'Low price of stock',
'%.2f';...
```

```
  'closePrice',    '[$]',   'Closing price of stock',
'%.2f';...
  'adjClosePrice', '[$]',   'Adjusted close price of
stock',   '%.2f';...
  'volume',        '[-]',   'Trading volume',
'%.0f'};

stock.DateTime      = data(:,1);
stock.openPrice     = data(:,2);
stock.highPrice     = data(:,3);
stock.lowPrice      = data(:,4);
stock.closePrice    = data(:,5);
stock.volume        = data(:,6);
stock.adjClosePrice = data(:,7);


end % function get_yahoo_stockdata3
```

DataCollector

```
function
[paststockvalues,pastdates,futurestockvalues,futuredates,..
.
    paststockaverages,futurestockaverages,pastdatesavgs,...
    SP500values,SP500dates,SP500avgs,SP500datesavgs]...

=DataCollector(stocks,termsinavg,present,present1,endpast,e
ndfuture)
%Collects Data From Yahoo Finance

%Preallocates Data for Arrays
paststockvalues=cell(1,length(stocks));
pastdates=cell(1,length(stocks));
futurestockvalues=cell(1,length(stocks));
futuredates=cell(1,length(stocks));
paststockaverages=cell(1,length(stocks));
futurestockaverages=cell(1,length(stocks));
pastdatesavgs=cell(1,length(stocks));

%Retrieves Data from Yahoo Finance
futureinfo=get_yahoo_stockdata3(stocks,present1,endfuture);
```

```
pastinfo=get_yahoo_stockdata3([stocks,{'^SP500TR'}],endpast
,present);
save('datafile');
%Determines past and future dates and stock values for each
stock
for cnt=1:length(stocks)
    paststockvalues{1,cnt}=pastinfo{1,cnt}.adjClosePrice;
    pastisnan=find(isnan(paststockvalues{1,cnt}));
    pastdatenumbers=round(pastinfo{1,cnt}.DateTime);
    pastdates{1,cnt}=(pastdatenumbers-pastdatenumbers(1));
    paststockvalues{1,cnt}(pastisnan)=[];
    pastdates{1,cnt}(pastisnan)=[];

futurestockvalues{1,cnt}=futureinfo{1,cnt}.adjClosePrice;
    futureisnan=find(isnan(futurestockvalues{1,cnt}));
    futuredatenumbers=round(futureinfo{1,cnt}.DateTime);
    futuredates{1,cnt}=(futuredatenumbers-
pastdatenumbers(1));
    futurestockvalues{1,cnt}(futureisnan)=[];
    futuredates{1,cnt}(futureisnan)=[];
    %Computes moving averages

paststockaveragesall=movmean(paststockvalues{1,cnt},[termsi
navg-1 0]);

futurestockaveragesall=movmean([paststockvalues{1,cnt}(end-
termsinavg+1:end-1);futurestockvalues{1,cnt}],[termsinavg-1
0]);
    %Removes the first few terms of the moving past moving
averages
    %because they do not correspond to actual moving
averages

paststockaverages{1,cnt}=paststockaveragesall(termsinavg:en
d);

futurestockaverages{1,cnt}=futurestockaveragesall(termsinav
g:end);
    pastdatesavgs{1,cnt}=pastdates{1,cnt}(termsinavg:end);
end

%Obtains S&P 500 values and dates
SP500values=pastinfo{1,end}.adjClosePrice;
SP500times=pastinfo{1,end}.DateTime;
```

```
SP500dates=(SP500times-SP500times(1));

%Calculates S&P 500 moving average
SP500avgsall=movmean(SP500values,[termsinavg-1 0]);
SP500avgs=SP500avgsall(termsinavg:end);
SP500datesavgs=SP500dates(termsinavg:end);
save('datafile1');
end
```

DataCollectorAvgs

```
function
[paststockvalues,pastdates,futurestockvalues,futuredates,...
.
    paststockaverages,futurestockaverages,pastdatesavgs,...
    SP500values,SP500dates,SP500avgs,SP500datesavgs]...

=DataCollectorAvgs(stocks,termsinavgs,present,present1,endp
ast)
%Collects Data From Yahoo Finance For Moving Averages with
a range of
%different terms

%Preallocates Data for Arrays
paststockvalues=cell(1,length(stocks));
pastdates=cell(1,length(stocks));
futurestockvalues=cell(1,length(stocks));
futuredates=cell(1,length(stocks));
paststockaverages=cell(1,length(stocks));
futurestockaverages=cell(1,length(stocks));
pastdatesavgs=cell(1,length(stocks));
SP500avgs=cell(1,length(termsinavgs));
SP500datesavgs=cell(1,length(termsinavgs));
%Retrieves Data from Yahoo Finance
futureinfo=get_yahoo_stockdata3(stocks,present1);
pastinfo=get_yahoo_stockdata3([stocks,{'^SP500TR'}],endpast
,present);

%Determines past and future dates and stock values for each
stock
for cnt=1:length(stocks)
    paststockvalues{1,cnt}=pastinfo{1,cnt}.adjClosePrice;
```

```
    pastisnan=find(isnan(paststockvalues{1,cnt}));
    pastdatenumbers=round(pastinfo{1,cnt}.DateTime);
    pastdates{1,cnt}=(pastdatenumbers-pastdatenumbers(1));
    paststockvalues{1,cnt}(pastisnan)=[];
    pastdates{1,cnt}(pastisnan)=[];

futurestockvalues{1,cnt}=futureinfo{1,cnt}.adjClosePrice;
    futureisnan=find(isnan(futurestockvalues{1,cnt}));
    futuredatenumbers=round(futureinfo{1,cnt}.DateTime);
    futuredates{1,cnt}=(futuredatenumbers-
pastdatenumbers(1));
    futurestockvalues{1,cnt}(futureisnan)=[];
    futuredates{1,cnt}(futureisnan)=[];
    for termsinavg=termsinavgs
        index=termsinavg-termsinavgs(1)+1;
        %Computes moving averages

paststockaveragesall=movmean(paststockvalues{1,cnt},[termsi
navg-1 0]);

futurestockaveragesall=movmean([paststockvalues{1,cnt}(end-
termsinavg+1:end-1);futurestockvalues{1,cnt}],[termsinavg-1
0]);
        %Removes the first few terms of the moving past
moving averages
        %because they do not correspond to actual moving
averages

paststockaverages{1,index}{1,cnt}=paststockaveragesall(term
sinavg:end);

futurestockaverages{1,index}{1,cnt}=futurestockaveragesall(
termsinavg:end);

pastdatesavgs{1,index}{1,cnt}=pastdates{1,cnt}(termsinavg:e
nd);
    end
end

%Obtains S&P 500 values and dates
SP500values=pastinfo{1,end}.adjClosePrice;
SP500times=pastinfo{1,end}.DateTime;
SP500dates=(SP500times-SP500times(1));
for termsinavg=termsinavgs
```

```
    index=termsinavg-termsinavgs(1)+1;
    %Calculates S&P 500 moving average
    SP500avgsall=movmean(SP500values,[termsinavg-1 0]);
    SP500avgs{1,index}=SP500avgsall(termsinavg:end);
    SP500datesavgs{1,index}=SP500dates(termsinavg:end);
end
end
```

PrototypeModelDataNew

```
function
[percentmargin,meanmargin,stdmargin,dayspredicted,meandays,
...

stddays,scaledvolatility,pastnoise,futurenoise,meanstockpas
t,...

meanstockfuture,stdpast,stdfuture,stdall]=PrototypeModelDat
aNew...

(paststockvaluesall,pastdatesall,futurestockvaluesall,futur
edatesall,...

fouriercutoff1,fouriercutoff2,stocks,stocknames,graph,prese
ntdate)
%Fits Prototype Model to Stock Data
%Generates Several Graphs if graph>0
percentmargin=zeros(1,length(stocks));
dayspredicted=zeros(1,length(stocks));
scaledvolatility=zeros(1,length(stocks));
meanstockpast=zeros(1,length(stocks));
meanstockfuture=zeros(1,length(stocks));
stdpast=zeros(1,length(stocks));
stdfuture=zeros(1,length(stocks));
stdall=zeros(1,length(stocks));
pastnoise=cell(1,length(stocks));
futurenoise=cell(1,length(stocks));
for stocknum=1:length(stocks)

    %Extracts the data for the particular stock being
studied
    paststockvalues=paststockvaluesall{1,stocknum};
```

```
    pastdates=pastdatesall{1,stocknum};
    futuredates=futuredatesall{1,stocknum};
    futurestockvalues=futurestockvaluesall{1,stocknum};
    %calculates autocorrelation and uses it to determine
how much data to use

[acf]=autocorr(flip(paststockvalues),length(paststockvalues
)-1);
    [~,datasize]=max(acf<0);
    datasize=datasize-1;
    if(graph>0)
        %plots autocorrelation
        figure('NumberTitle','off','Name',['Autocorrelation
from present (',stocknames{stocknum},')']);
        autocorr(paststockvalues,size(paststockvalues,1)-
1);
%        xlabel('lags');
%        ylabel('autocorrelation');
%        title(['Autocorrelation of
',stocknames{stocknum},' (',stocks{stocknum},')']);
   %saves the graph as a .png file
        if(graph>1)

%print(['C:\Users\apmur\OneDrive\Documents\IQPGraphs\',stoc
ks{stocknum},'\autocorrelationavgs'],'-dpng');
        else

print(['C:\Users\apmur\OneDrive\Documents\IQPGraphs\',stock
s{stocknum},'\A'],'-dpng');
        end
    end
    %selects most recent "past" data for use in building
model
    selectedstockvalues=paststockvalues(end-
datasize+1:end);
    selecteddates=pastdates(end-datasize+1:end);

    %Determines linear model

linearmodel=fit(selecteddates,selectedstockvalues,'poly1');

    %Calculates differences between linear model and actual
stock prices
```

```
    differences=selectedstockvalues-
linearmodel(selecteddates);

    %Fits a Fourier Model to the Differences
    if(datasize<=fouriercutoff1)

fouriermodel=fit(selecteddates,differences,'fourier2');
    elseif(datasize>=fouriercutoff2)

fouriermodel=fit(selecteddates,differences,'fourier3');
    else

[fouriermodel1,gof1]=fit(selecteddates,differences,'fourier
2');

[fouriermodel2,gof2]=fit(selecteddates,differences,'fourier
3');
        if(gof1.rsquare>gof2.rsquare)
            fouriermodel=fouriermodel1;
        else
            fouriermodel=fouriermodel2;
        end
    end

    if(graph==1)
        %Evaluates linear model and fourier model at date
values for plotting
        numberofpoints=1000;

datevalues=linspace(selecteddates(1),futuredates(end),numbe
rofpoints);
        linearmodelpredictions=linearmodel(datevalues);
        fouriermodelpredictions=fouriermodel(datevalues);
        futuredifferences=futurestockvalues-
linearmodel(futuredates);
        plotdifferences=[differences;futuredifferences];
        %Generates a vertical line separating past and
future for trend
        %line plot
        numlinepts=100;

maxstock=max([max(differences),max(selectedstockvalues),max
(linearmodelpredictions)]);
```

```
minstock=min([min(futurestockvalues),min(selectedstockvalue
s),min(linearmodelpredictions)]);
        maxheight=maxstock+0.1*(maxstock-minstock);
        minheight=minstock-0.1*(maxstock-minstock);
        liney=linspace(minheight,maxheight,numlinepts)';
        linex=ones(numlinepts,1)*selecteddates(end);

        %Generates a vertical line separating past and
future for fourier
        %series plot

maxdiff=max([max(plotdifferences),max(fouriermodelpredictio
ns)]);

mindiff=min([min(plotdifferences),min(fouriermodelpredictio
ns)]);
        maxheightdiff=maxdiff+0.1*(maxdiff-mindiff);
        minheightdiff=mindiff-0.4*(maxdiff-mindiff);

lineydiff=linspace(minheightdiff,maxheightdiff,numlinepts)'
;

        %finds important dates
        presentnum=datenum(presentdate,'dd-mmm-yyyy');
        dates=[selecteddates;futuredates];
        middledatevalue=round((dates(1)+dates(end))/2);
        beginningofpast=presentnum-1-dates(end)+dates(1);
        present=presentnum-1-
futuredates(end)+futuredates(1);
        middledate=presentnum-1-round((dates(end)-
dates(1))/2);
        futureend=presentnum-1;

        datevectors1(1,:)=datevec(beginningofpast);
        datevectors1(2,:)=datevec(present);
        datevectors1(3,:)=datevec(middledate);
        datevectors1(4,:)=datevec(futureend);

        %orders the dates from oldest to newest

M=[selecteddates(1);selecteddates(end);middledatevalue;futu
redates(end)];
        [M,ia]=unique(M);
```

```
        datenums=datevectors1(ia,:);

        %turns the date into a string
        datelist1=cell(1,4);
        for i=1:size(M,1)
            yearstring=sprintf('%.0f',datenums(i,1));
            a=char(yearstring);
            yearshortened=string(a(end-1:end));

datelist1{i}=char(strcat(sprintf('%.0f',datenums(i,2)),'/',
sprintf('%.0f',datenums(i,3)),'/',yearshortened));
        end

        %plots linear model and stock prices
        figure('NumberTitle','off','Name',['Line Plot
(',stocknames{stocknum},')']);

plot([selecteddates(end);futuredates],[selectedstockvalues(
end);futurestockvalues],selecteddates,selectedstockvalues,l
inex,liney,'--
g',datevalues,linearmodelpredictions,'k','LineWidth',2);
        title(['Stock Prices and Trend Line for the Stock
',stocknames{stocknum},' (',stocks{stocknum},')']);

        %legend placed on whether the slope of the line is
positive or negative
        linearmodelcoefficients=coeffvalues(linearmodel);
        linearmodelslope=linearmodelcoefficients(1);
        %Puts in Legend. Positions them to avoid
overlapping with the graph
        %based on the line slope
        if(linearmodelslope>0)
            legend({'future prices','past
prices','past|future','Trend
Line'},'Location','NorthWest');
        else
            legend({'future prices','past
prices','past|future','Trend
Line'},'Location','SouthWest');
        end

        %labels important dates on x-axis
        xticks(M)
        xticklabels(datelist1)
```

```
        xtickangle(45)

        %labels axes
        legend('boxoff');
        xlabel('date');
        ylabel('stock price');


print(['C:\Users\apmur\OneDrive\Documents\IQPGraphs\',stock
s{stocknum},'\B',],'-dpng');

        figure('NumberTitle','off','Name',['Fourier Plot
(',stocknames{stocknum},')']);

plot([selecteddates(end);futuredates],plotdifferences(datas
ize:end),selecteddates,plotdifferences(1:datasize),linex,li
neydiff,'--
g',datevalues,fouriermodelpredictions,'k','LineWidth',2);
        title(['Differences and Fourier Model for the Stock
',stocknames{stocknum},' (',stocks{stocknum},')']);

        %labels important dates on x-axis
        xticks(M)
        xticklabels(datelist1)
        xtickangle(45)

        %labels axes
        legend('boxoff');
        xlabel('date');
        ylabel('difference');
        legend({'future differences','past
differences','past|future','Fourier
Model'},'Location','SouthWest');

print(['C:\Users\apmur\OneDrive\Documents\IQPGraphs\',stock
s{stocknum},'\C',],'-dpng');
    end


    %computes noise, maxnoise, minnoise
    noise=differences-fouriermodel(selecteddates);
    pastnoise{stocknum}=noise;
    maxnoise=max(noise);
    minnoise=min(noise);
```

```
    %computes margin of error and percent margin
    margin=(maxnoise-minnoise)/2;
    averagestock=mean(selectedstockvalues);
    percentmargin(stocknum)=margin/averagestock;

    %Computes model upper and lower bounds

upperbound=linearmodel(futuredates)+fouriermodel(futuredate
s)+maxnoise;

lowerbound=linearmodel(futuredates)+fouriermodel(futuredate
s)+minnoise;

    %Adds a little padding on each side of the prediction
band
    upperbound=upperbound+0.05*(maxnoise-minnoise);
    lowerbound=lowerbound-0.05*(maxnoise-minnoise);

    %Determines if the stock exceeds the upper bound
    exceeded=max(futurestockvalues>upperbound);
    %
    %Determines if the stock drops below the lower bound
    dropped=max(futurestockvalues<upperbound);

    %Determines when the stock leaves the prediction band
    if(exceeded>0&&dropped>0)
        [~,exceedtimeup]=max(futurestockvalues>upperbound);

[~,exceedtimelow]=max(futurestockvalues<lowerbound);

dayspredicted(stocknum)=min(exceedtimeup,exceedtimelow);
    elseif(exceeded>0)

[~,dayspredicted(stocknum)]=max(futurestockvalues>upperboun
d);
    elseif(dropped>0)

[~,dayspredicted(stocknum)]=max(futurestockvalues<lowerboun
d);
    else
        dayspredicted(stocknum)=length(futuredates);
    end
    %date where model fails
```

```
    faildate=futuredates(dayspredicted(stocknum));
    %computes the scaled volatility

scaledvolatility(stocknum)=std(selectedstockvalues)/mean(se
lectedstockvalues);

    %computes mean stock value
    meanstockpast(stocknum)=mean(paststockvalues);
    meanstockfuture(stocknum)=mean(futurestockvalues);

    %computes mean stock value
    stdpast(stocknum)=std(paststockvalues);
    stdfuture(stocknum)=std(futurestockvalues);

stdall(stocknum)=std([paststockvalues;futurestockvalues]);

    %computes noise for future data
    futurenoise{stocknum}=futurestockvalues-
(linearmodel(futuredates)+fouriermodel(futuredates));

    if(graph>0)
        %computes model predictions and model bounds for
graphing
        numberofpoints=1000;

datevalues=linspace(selecteddates(1),futuredates(end),numbe
rofpoints);

modelpredictions=linearmodel(datevalues)+fouriermodel(datev
alues);
        dashedtop=modelpredictions+maxnoise;
        dashedbottom=modelpredictions+minnoise;

        %forms a vertical line at the date dividing the
"past" and the
        %"future"
        numlinepts=100;

maxstock=max([max(futurestockvalues),max(selectedstockvalue
s),max(dashedtop)]);

minstock=min([min(futurestockvalues),min(selectedstockvalue
s),min(dashedbottom)]);
        maxheight=maxstock+0.1*(maxstock-minstock);
```

```
        minheight=minstock-0.1*(maxstock-minstock);
        liney=linspace(minheight,maxheight,numlinepts)';
        linex=ones(numlinepts,1)*selecteddates(end);

        %makes a vertical line where the prediction fails
        line1y=linspace(minheight,maxheight,numlinepts)';
        line1x=ones(numlinepts,1)*faildate;

        %assigns string dates to the oldest stock value
used in the model,
        %the time halfway between the oldest and stockvalue
furthest into
        %the future, the newest stock value used in the
model (the present
        %day), and the date of the stock value farthest in
the future
        presentnum=datenum(presentdate,'dd-mmm-yyyy');
        dates=[selecteddates;futuredates];
        middledatevalue=round((dates(1)+dates(end))/2);
        beginningofpast=presentnum-1-dates(end)+dates(1);
        present=presentnum-1-
futuredates(end)+futuredates(1);
        modelfails=presentnum-1-futuredates(end)+faildate;
        middledate=presentnum-1-round((dates(end)-
dates(1))/2);
        futureend=presentnum-1;
        datevectors(1,:)=datevec(beginningofpast);
        datevectors(2,:)=datevec(present);
        datevectors(3,:)=datevec(modelfails);
        datevectors(4,:)=datevec(middledate);
        datevectors(5,:)=datevec(futureend);


        %orders the dates from oldest to newest

M=[selecteddates(1);selecteddates(end);faildate;middledatev
alue;futuredates(end)];
        [M,ia]=unique(M);
        datenums=datevectors(ia,:);

        %turns the date into a string
        datelist=cell(1,5);
        for i=1:size(M,1)
            yearstring=sprintf('%.0f',datenums(i,1));
```

```
            a=char(yearstring);
            yearshortened=string(a(end-1:end));

datelist{i}=char(strcat(sprintf('%.0f',datenums(i,2)),'/',s
printf('%.0f',datenums(i,3)),'/',yearshortened));
        end

        %generates the figure and makes it fullscreen

figure('Name',stocknames{stocknum},'units','normalized','ou
terposition',[0 0 1 1])
        %plots the past stock values used in the model,
future stock values,
        %model predictions, model bounds, and vertical
lines
        if(dayspredicted(stocknum)>0&&(exceeded||dropped))

plot([selecteddates(end);futuredates],[selectedstockvalues(
end);futurestockvalues],selecteddates,selectedstockvalues,l
inex,liney,'g--',line1x,line1y,'m--
',datevalues,modelpredictions,'k',datevalues,dashedtop,'k--
',datevalues,dashedbottom,'k--','LineWidth',2);
        else

plot(futuredates,futurestockvalues,selecteddates,selectedst
ockvalues,linex,liney,'g--
',datevalues,modelpredictions,'k',datevalues,dashedtop,'k--
',datevalues,dashedbottom,'k--','LineWidth',2);
        end

        %labels important dates on x-axis
        xticks(M)
        xticklabels(datelist)
        xtickangle(45)

        %legend placed on whether the slope of the line is
positive or negative
        linearmodelcoefficients=coeffvalues(linearmodel);
        linearmodelslope=linearmodelcoefficients(1);
        %Puts in Legend. Positions them to avoid
overlapping with the graph
        %based on the line slope
        %Adjusts Legend based on whether the model
predicted the stock for
```

```
        %the whole time period
        if(dayspredicted(stocknum)>0&&(exceeded||dropped))
            if(linearmodelslope>0&&(exceeded||dropped))
                legend('future prices','past prices','past
| future','model fails','model','model
bounds','Location','NorthWest');
            else
                legend('future prices','past prices','past
| future','model fails','model','model
bounds','Location','SouthWest');
            end
        else
            if(linearmodelslope>0)
                legend('future prices','past prices','past
| future','model','model bounds','Location','SouthWest');
            else
                legend('future prices','past prices','past
| future','model','model bounds','Location','NorthWest');
            end
        end

        %labels axes
        legend('boxoff');
        xlabel('date');
        ylabel('stock price');

        %titles graph
        title(['Actual and Predicted Prices of
',stocknames{stocknum},' (',stocks{stocknum},')']);

        %saves the graph as a .png file
        if(graph>1)

print(['C:\Users\apmur\OneDrive\Documents\IQPGraphs\',stock
s{stocknum},'\E'],'-dpng');
        else

print(['C:\Users\apmur\OneDrive\Documents\IQPGraphs\',stock
s{stocknum},'\D'],'-dpng');
        end
    end
end
meanmargin=mean(percentmargin);
stdmargin=std(percentmargin);
```

```
meandays=mean(dayspredicted);
stddays=std(dayspredicted);
end
```

IndexModelDataNew

```
function
[percentmargin,meanmargin,stdmargin,dayspredicted,meandays,
...
    stddays,alphas,volSP500,pastnoise,futurenoise]=...

IndexModelDataNew(paststockvaluesall,pastdatesall,futuresto
ckvaluesall,...

futuredatesall,SP500dates,SP500values,fouriercutoff1,...
    fouriercutoff2,stocks,stocknames,graph,presentdate)
%Fits Prototype Model to Stock Data
%Generates A Graph if graph>0
percentmargin=zeros(1,length(stocks));
dayspredicted=zeros(1,length(stocks));
alphas=zeros(1,length(stocks));
pastnoise=cell(1,length(stocks));
futurenoise=cell(1,length(stocks));
%calculates autocorrelation and uses it to determine how
much data to use
[acf]=autocorr(flip(SP500values),length(SP500values)-1);
[~,datasizeSP500]=max(acf<0);
datasizeSP500=datasizeSP500-1;

%selects most recent "past" data for use in building model
selectedSP500values=SP500values(end-datasizeSP500+1:end);
selectedSP500dates=SP500dates(end-datasizeSP500+1:end);

%Determines linear model for SP500
linearmodelSP500=fit(selectedSP500dates,selectedSP500values
,'poly1');

%Calculates differences between linear model and actual SP
500 values
differencesSP500=selectedSP500values-
linearmodelSP500(selectedSP500dates);

%Fits a Fourier Model to the Differences
if(datasizeSP500<=fouriercutoff1)
```

```
fouriermodelSP500=fit(selectedSP500dates,differencesSP500,'
fourier2');
elseif(datasizeSP500>=fouriercutoff2)

fouriermodelSP500=fit(selectedSP500dates,differencesSP500,'
fourier3');
else

[fouriermodel1,gof1]=fit(selectedSP500dates,differencesSP50
0,'fourier3');

[fouriermodel2,gof2]=fit(selectedSP500dates,differencesSP50
0,'fourier3');
    if(gof1.rsquare>gof2.rsquare)
        fouriermodelSP500=fouriermodel1;
    else
        fouriermodelSP500=fouriermodel2;
    end
end

for stocknum=1:1:size(stocks,2)

    %Extracts the data for the particular stock being
studied
    paststockvalues=paststockvaluesall{1,stocknum};
    pastdates=pastdatesall{1,stocknum};
    futuredates=futuredatesall{1,stocknum};
    futurestockvalues=futurestockvaluesall{1,stocknum};

    %calculates autocorrelation and uses it to determine
how much data to use

[acf]=autocorr(flip(paststockvalues),size(paststockvalues,1
)-1);
    [~,datasize]=max(acf<0);
    datasize=datasize-1;

    %selects most recent "past" data for use in building
model
    selectedstockvalues=paststockvalues(end-
datasize+1:end);
    selecteddates=pastdates(end-datasize+1:end);
```

```
    %Determines linear model

linearmodel=fit(selecteddates,selectedstockvalues,'poly1');

    %Calculates differences between linear model and actual
stock prices
    differences=selectedstockvalues-
linearmodel(selecteddates);

    %Fits a Fourier Model to the Differences
    if(datasize<=fouriercutoff1)

fouriermodel=fit(selecteddates,differences,'fourier2');
    elseif(datasize>=fouriercutoff2)

fouriermodel=fit(selecteddates,differences,'fourier3');
    else

[fouriermodel1,gof1]=fit(selecteddates,differences,'fourier
3');

[fouriermodel2,gof2]=fit(selecteddates,differences,'fourier
3');
        if(gof1.rsquare>gof2.rsquare)
            fouriermodel=fouriermodel1;
        else
            fouriermodel=fouriermodel2;
        end
    end

    %computes correlation between the stock and the S&P 500
index

alphas(stocknum)=corr(selectedstockvalues,SP500values(1:dat
asize));

    %Uses the S&P 500 model to approximate the S&P 500 at
the points where
    %we are using data for the model.

SP500model=linearmodelSP500(selecteddates)+fouriermodelSP50
0(selecteddates);

    %Normalized S&P 500 Model
```

```
    SP500modelnormalized=SP500model/SP500values(end);

    %Prototype Model

prototypemodel=linearmodel(selecteddates)+fouriermodel(sele
cteddates);

    %Normalized Prototype Model

prototypemodelnormalized=prototypemodel/paststockvalues(end
);

    %Computes Normalized Index Model

IndexModelNormalized=SP500modelnormalized*alphas(stocknum)+
(1-alphas(stocknum))*prototypemodelnormalized;

    %Computes Index Model
    IndexModel=IndexModelNormalized*paststockvalues(end);

    %computes noise, maxnoise, minnoise
    noise=selectedstockvalues-IndexModel;
    pastnoise{stocknum}=noise;
    maxnoise=max(noise);
    minnoise=min(noise);

    %computes margin of error and percent margin
    margin=(maxnoise-minnoise)/2;
    averagestock=mean(selectedstockvalues);
    percentmargin(stocknum)=margin/averagestock;

    %Uses the S&P 500 model to make predictions

SP500modelpred=linearmodelSP500(futuredates)+fouriermodelSP
500(futuredates);

    %Normalized S&P 500 Model

SP500modelnormalizedpred=SP500modelpred/SP500values(end);

    %Prototype Model Predictions

prototypemodelpred=linearmodel(futuredates)+fouriermodel(fu
turedates);
```

```
    %Normalized Prototype Model

prototypemodelnormalizedpred=prototypemodelpred/paststockva
lues(end);

    %Computes Normalized Index Model

IndexModelNormalizedpred=SP500modelnormalizedpred*alphas(st
ocknum)+(1-alphas(stocknum))*prototypemodelnormalizedpred;

    %Computes Index Model

IndexModelpred=IndexModelNormalizedpred*paststockvalues(end
);

    %Computes model upper and lower bounds
    upperbound=IndexModelpred+maxnoise;
    lowerbound=IndexModelpred+minnoise;

    %Adds a little padding on each side of the prediction
band
    upperbound=upperbound+0.05*(maxnoise-minnoise);
    lowerbound=lowerbound-0.05*(maxnoise-minnoise);

    %Determines if the stock exceeds the upper bound
    exceeded=max(futurestockvalues>upperbound);

    %Determines if the stock drops below the lower bound
    dropped=max(futurestockvalues<lowerbound);

    %Determines when the stock leaves the prediction band
    if(exceeded>0&&dropped>0)
        [~,exceedtimeup]=max(futurestockvalues>upperbound);

[~,exceedtimelow]=max(futurestockvalues<lowerbound);

dayspredicted(stocknum)=min(exceedtimeup,exceedtimelow);
    elseif(exceeded>0)

[~,dayspredicted(stocknum)]=max(futurestockvalues>upperboun
d);
    elseif(dropped>0)
```

```
[~,dayspredicted(stocknum)]=max(futurestockvalues<lowerboun
d);
    else
        dayspredicted(stocknum)=length(futuredates);
    end
    faildate=futuredates(dayspredicted(stocknum));
    %computes the scaled volatility of SP500

volSP500=std(selectedSP500values)/mean(selectedSP500values)
;
    %computes noise for both past and future data
    futurenoise{stocknum}=futurestockvalues-IndexModelpred;

    if(graph>0)
        %computes model predictions and model bounds for
graphing
        numberofpoints=1000;

        %dates used for graphing the index model

datevalues=linspace(selecteddates(1),futuredates(end),numbe
rofpoints);

        %Uses the S&P 500 model to approximate the S&P 500
at the points we are
        %using in the graph

SP500predictions=linearmodelSP500(datevalues)+fouriermodelS
P500(datevalues);

        %Normalized S&P 500 Model at graph points

SP500normalizedpredictions=SP500predictions/SP500values(end
);

        %Prototype Model at graph points

prototypemodelpredictions=linearmodel(datevalues)+fouriermo
del(datevalues);

        %Normalized Prototype Model at graph points
```

```
prototypemodelnormalizedpredictions=prototypemodelpredictio
ns/paststockvalues(end);

        %Computes Normalized Index Model at graph points

IndexModelNormalizedpredictions=SP500normalizedpredictions*
alphas(stocknum)+...
            (1-
alphas(stocknum))*prototypemodelnormalizedpredictions;

        %Computes Index Model at graph points

modelpredictions=IndexModelNormalizedpredictions*paststockv
alues(end);

        dashedtop=modelpredictions+maxnoise;
        dashedbottom=modelpredictions+minnoise;

        %forms a vertical line at the date dividing the
"past" and the
        %"future"
        numlinepts=100;

maxstock=max([max(futurestockvalues),max(selectedstockvalue
s),max(dashedtop)]);

minstock=min([min(futurestockvalues),min(selectedstockvalue
s),min(dashedbottom)]);
        maxheight=maxstock+0.1*(maxstock-minstock);
        minheight=minstock-0.1*(maxstock-minstock);
        liney=linspace(minheight,maxheight,numlinepts)';
        linex=ones(numlinepts,1)*selecteddates(end);

        %makes a vertical line where the prediction fails
        line1y=linspace(minheight,maxheight,numlinepts)';
        line1x=ones(numlinepts,1)*faildate;

        %assigns string dates to the oldest stock value
used in the model,
        %the time halfway between the oldest and stockvalue
furthest into
        %the future, the newest stock value used in the
model (the present
```

```
        %day), and the date of the stock value farthest in
the future
        presentnum=datenum(presentdate,'dd-mmm-yyyy');
        dates=[selecteddates;futuredates];
        middledatevalue=round((dates(1)+dates(end))/2);
        beginningofpast=presentnum-1-dates(end)+dates(1);
        present=presentnum-1-
futuredates(end)+futuredates(1);
        modelfails=presentnum-1-futuredates(end)+faildate;
        middledate=presentnum-1-round((dates(end)-
dates(1))/2);
        futureend=presentnum-1;
datevectors(1,:)=datevec(beginningofpast);
        datevectors(2,:)=datevec(present);
        datevectors(3,:)=datevec(modelfails);
        datevectors(4,:)=datevec(middledate);
        datevectors(5,:)=datevec(futureend);


        %orders the dates from oldest to newest

M=[selecteddates(1);selecteddates(end);faildate;middledatev
alue;futuredates(end)];
        [M,ia]=unique(M);
        datenums=datevectors(ia,:);

        %turns the date into a string
        datelist=cell(1,5);
        for i=1:size(M,1)
            yearstring=sprintf('%.0f',datenums(i,1));
            a=char(yearstring);
            yearshortened=string(a(end-1:end));

datelist{i}=char(strcat(sprintf('%.0f',datenums(i,2)),'/',s
printf('%.0f',datenums(i,3)),'/',yearshortened));
        end

        %generates the figure and makes it fullscreen

figure('Name',stocknames{stocknum},'units','normalized','ou
terposition',[0 0 1 1])

        %plots the past stock values used in the model,
future stock values,
```

```
        %model predictions, model bounds, and vertical
lines
        if(dayspredicted(stocknum)>0&&(exceeded||dropped))

plot([selecteddates(end);futuredates],[selectedstockvalues(
end);futurestockvalues],selecteddates,selectedstockvalues,l
inex,liney,'g--',line1x,line1y,'m--
',datevalues,modelpredictions,'k',datevalues,dashedtop,'k--
',datevalues,dashedbottom,'k--','LineWidth',2);
        else

plot(futuredates,futurestockvalues,selecteddates,selectedst
ockvalues,linex,liney,'g--
',datevalues,modelpredictions,'k',datevalues,dashedtop,'k--
',datevalues,dashedbottom,'k--','LineWidth',2);
        end

        %labels important dates on x-axis
        xticks(M)
        xticklabels(datelist)
        xtickangle(45)

        %legend placed on whether the slope of the line is
positive or negative
        linearmodelcoefficients=coeffvalues(linearmodel);
        linearmodelslope=linearmodelcoefficients(1);
        %Puts in Legend. Positions them to avoid
overlapping with the graph
        %based on the line slope
        %Adjusts Legend based on whether the model
predicted the stock for
        %the whole time period
        if(dayspredicted(stocknum)>0&&(exceeded||dropped))
            if(linearmodelslope>0&&(exceeded||dropped))
                legend('future prices','past prices','past
| future','model fails','model','model
bounds','Location','NorthWest');
            else
                legend('future prices','past prices','past
| future','model fails','model','model
bounds','Location','SouthWest');
            end
        else
            if(linearmodelslope>0)
```

```
                legend('future prices','past prices','past
| future','model','model bounds','Location','SouthWest');
            else
                legend('future prices','past prices','past
| future','model','model bounds','Location','NorthWest');
            end
        end

        %labels axes
        legend('boxoff');
        xlabel('date');
        ylabel('stock price');
        %titles graph
        title(['Actual and Predicted Prices of
',stocknames{stocknum},' (',stocks{stocknum},')']);

        %saves the graph as a .png file        if(graph>1)

        if(graph>1)

print(['C:\Users\apmur\OneDrive\Documents\IQPGraphs\',stock
s{stocknum},'\H'],'-dpng');
        else

print(['C:\Users\apmur\OneDrive\Documents\IQPGraphs\',stock
s{stocknum},'\G'],'-dpng');
        end
    end
end
meanmargin=mean(percentmargin);
stdmargin=std(percentmargin);
meandays=mean(dayspredicted);
stddays=std(dayspredicted);
end
```

embed

```
function Y=embed(x,tau,m)
%Length of Time Series
N=length(x);
%Number of Embedding Vectors
M=N-m*tau;
%Computes Embedding Vectors
```

```
[U,T]=ndgrid(1:M,0:m-1);
X=ndgrid(x,x);
indices=U+T*tau;
Y=X(indices);
end
```

falsenearest

```
function percentfalse=falsenearest(x,mmax,tau,Dtol,Atol)
x=x';
%Intializes Variables
percentfalse=zeros(1,mmax);
%Length of Time Series
N=length(x);
%Initializes Vector Storing embedding vectors for each
embedding dimension
Ys=cell(1,mmax);
%Embedding Dimension 1 corresponds to original time series
Ys{1}=x;
%Standard deviation of time series; approximate radius of
the attractor
s=std(x);
%Minimum distances between nearest neighbors
mindistsq=cell(1,mmax);
%Locations of nearest neighbors
nearestpos=cell(1,mmax);
for m=1:mmax
    %Embedded vectors in next dimension
    Ys{m+1}=embed(x,tau,m+1);
    %Embedded vectors in current dimension
    Y=Ys{m};
    %Number of embedded vectors in current dimension
    M(m)=N-tau*m;
    %square of the distances between each pair of embedded
vectors
    distancessq=zeros(M(m));
    for cnt=1:m
        %computes pairwise distances between embedded
vectors
        [Ygrid,Ygrid2]=ndgrid(Y(:,cnt),Y(:,cnt));
        distancessq=distancessq+(Ygrid-Ygrid2).^2;
    end
```

```
    %makes it so the nearest neighbor cannot be the vector
itself
    distancessq=distancessq+10^6*eye(M(m));
    %finds nearest neighbor
    [mindistsq{m},nearestpos{m}]=min(distancessq);
end
for m=1:mmax
    %distances between nearest neighbors
    Rim=mindistsq{m}(1:M(m));
    %distances between neighbors in original time series
    dist1=abs(x(m*tau+1:end)-x(nearestpos{m}+m*tau));
    %check if the ratio of distances are below tolerance
    D=dist1./Rim;
    Dokay=D<Dtol;
    %distances in next dimension
    Rim1=Rim.^2+dist1.^2;
    R=Rim1/s;
    %checks if distances are too far
    Rokay=R<Atol;
    %When both conditions indicate neighbors are true
    okay=Dokay.*Rokay;
    %percent of false nearest neighbors
    percentfalse(m)=1-sum(okay)/M(m);
end
end
```

lyapunovnew

```
function lyap=lyapunovnew(t,x,m,tau,meanperiod,maxiter)
%used to compute Lyapunov exponents with Rosenstein's
Algorithm

%Constructs Embedded Vectors
Y=embed(x,tau,m);
%Length of time series
N=length(x);
%Number of embedded vectors
M=N-tau*m;
%distance between pairs of embedded vectors
distancessq=zeros(M);
%minimum distance between viable neighbors
radius=floor(meanperiod/2);
%computes squares of distances between points
for cnt=1:m
```

```
    [Ygrid,Ygrid2]=ndgrid(Y(:,cnt),Y(:,cnt));
    distancessq=distancessq+(Ygrid-Ygrid2).^2;
end
%prevents neighbors closer than one mean period from being
chosen as
%nearest
A=10^6*ones(M);
A=A-triu(A,radius+1);
A=A-tril(A,-radius-1);
distancessq=distancessq+A;
%find distances between nearest neighbors
[~,nearestpos]=min(distancessq);

%Initializes mean logarithm of distances
meanlogs=zeros(1,maxiter);
for k=1:maxiter
    %Number of embedding vectors used in calculation
    maxindex=M-k;
    %locations of nearest neighbors
    indicies=find(nearestpos(1:maxindex)<=maxindex);
    %distances between nearest neighbors after k time steps
    distances=vecnorm(Y(indicies+k,:)-
Y(nearestpos(indicies)+k,:));
    %computes mean of logs of distances
    logdistances=log(distances);
    meanlogs(k)=mean(logdistances);
end
%fits a least squares line
leastsquaresline=fit(t(1:maxiter),meanlogs','poly1');
%finds line slope=lyapunov exponent
linecoefficients=coeffvalues(leastsquaresline);
lyap=linecoefficients(1);
```

Lyapunov2

```
function lyaps=Lyapunov2(t,x,mmax,maxiter)
%Computes Lyapunov Exponents for a range of embedding
dimensions
%calculates autocorrelation and uses it to determine how
much data to use
[acf]=autocorr(flip(x),length(x)-1);
[~,datasize]=max(acf<exp(-1));
%determines delay time and mean period
```

```
tau=datasize-1;
meanperiod=tau;
lyaps=zeros(1,mmax);
for m=1:mmax
    lyaps(m)=lyapunovnew(t,x,m,tau,meanperiod,maxiter);
end
end
```

DataGeneratorLyapunov

```
%Generates Lyapunov Exponent Data and Writes it to an Excel
File
%Saves Scatter Plots
close all;
clear all;
tic
%Fourier Series fit to data with fewer than this number of
points is 2nd Order.
fouriercutoff1=40;
%Fourier Series fit to data with at least than this number
of points is 3rd
%Order
fouriercutoff2=60;
%Defines number of terms of moving average
termsinavg=5;
%maximum embedding dimension. If this is changed, the excel
region should
%be correspondingly changed.
mmax=3;
%maximum number of iterations for Lyapunov
maxiter=10;

%Forces the late future date to be the day before this date
presentdate='09-August-2018';

%stock symbols
stocks={'INXN','WUBA','TSG','MTCH','PTC','PRGS','CTXS','BLK
B','CVLT','MDSO','CYOU','RP','IMPV','GWRE','SPLK','PCTY','P
AYC','NTNX','TLND','MFGP'};

%stocknames. A placeholder since no graphs are generated
stocknames=stocks;
```

```
%Data From Yahoo Finance
[paststockvalues,pastdates,~,~,paststockaverages,~,pastdate
savgs,...
    SP500values,SP500dates,SP500avgs,SP500datesavgs]...
    =DataCollector(stocks,termsinavg,'01-June-2018','01-
June-2018',...
    '02-June-2017',presentdate);

%Calculates Scaled Standard Deviation of Stock Prices
[~,~,~,~,~,~,vol]...

=PrototypeModelDataNew(paststockvalues,pastdates,futurestoc
kvalues,futuredates,fouriercutoff1,fouriercutoff2,stocks,st
ocknames,1,presentdate);
%Calculates Scaled Standard Deviation of Stock Averages
[~,~,~,~,~,~,volavgs]...

=PrototypeModelDataNew(paststockaverages,pastdatesavgs,futu
restockaverages,futuredates,fouriercutoff1,fouriercutoff2,s
tocks,stocknames,2,presentdate);
%Gets Correlation Between Stock Prices and Index and Scaled
Standard
%Deviation of S&P 500
[~,~,~,~,~,~,correlationavgs,volSP500avgs]...

=IndexModelDataNew(paststockvalues,pastdates,futurestockval
ues,futuredates,SP500dates,SP500values,fouriercutoff1,fouri
ercutoff2,stocks,stocknames,1,presentdate);
%Gets Correlation Between Stock Prices and Index and Scaled
Standard
%Deviation of S&P 500 for Moving Averages
[percentmargins(4,:),meanmargins(4),stdmargins(4),dayspredi
cted(4,:),meandays(4),stddays(4),correlationavgs,volSP500av
gs]...

=IndexModelDataNew(paststockaverages,pastdatesavgs,futurest
ockaverages,futuredates,SP500datesavgs,SP500avgs,fouriercut
off1,fouriercutoff2,stocks,stocknames,2,presentdate);

%Intializes One Year Lyapunov Exponents
lyapexp1yr=zeros(length(stocks),mmax);
lyapexp1yravg=zeros(length(stocks),mmax);
```

```
%Computes One Year Lyapunov Exponents
for stocknum=1:length(stocks)

lyapexp1yr(stocknum,:)=Lyapunov2(pastdates{stocknum},paststockvalues{stocknum},mmax,maxiter);

lyapexp1yravg(stocknum,:)=Lyapunov2(pastdatesavgs{stocknum},paststockaverages{stocknum},mmax,maxiter);
end
lyapexp1yrSP5001yr=Lyapunov2(SP500dates,SP500values,mmax,maxiter);
lyapexp1yrSP5001yravg=Lyapunov2(SP500datesavgs,SP500avgs,mmax,maxiter);

%%%%% TWO YEARS

%Collects Additional Data from Yahoo Finance For 2 years
[paststockvalues,pastdates,futurestockvalues,futuredates,...
    paststockaverages,futurestockaverages,pastdatesavgs,...
    SP500values,SP500dates,SP500avgs,SP500datesavgs]...
    =DataCollector(stocks,termsinavg,'01-June-2018','01-June-2018','02-June-2016',presentdate);

%Calculates Scaled Standard Deviation of Stock Prices
[~,~,~,~,~,~,vol]...

=PrototypeModelDataNew(paststockvalues,pastdates,futurestockvalues,futuredates,fouriercutoff1,fouriercutoff2,stocks,stocknames,1,presentdate);
%Calculates Scaled Standard Deviation of Stock Averages
[~,~,~,~,~,~,volavgs]...

=PrototypeModelDataNew(paststockaverages,pastdatesavgs,futurestockaverages,futuredates,fouriercutoff1,fouriercutoff2,stocks,stocknames,2,presentdate);
%Gets Correlation Between Stock Prices and Index and Scaled Standard
%Deviation of S&P 500
[~,~,~,~,~,~,correlationavgs,volSP500avgs]...

=IndexModelDataNew(paststockvalues,pastdates,futurestockvalues,futuredates,SP500dates,SP500values,fouriercutoff1,fouriercutoff2,stocks,stocknames,1,presentdate);
```

```
%Gets Correlation Between Stock Prices and Index and Scaled
Standard
%Deviation of S&P 500 for Moving Averages
[percentmargins(4,:),meanmargins(4),stdmargins(4),dayspredi
cted(4,:),meandays(4),stddays(4),correlationavgs,volSP500av
gs]...

=IndexModelDataNew(paststockaverages,pastdatesavgs,futurest
ockaverages,futuredates,SP500datesavgs,SP500avgs,fouriercut
off1,fouriercutoff2,stocks,stocknames,2,presentdate);


%Intializes Two Year Lyapunov Exponents
lyapexp2yrs=zeros(length(stocks),mmax);
lyapexp2yrsavg=zeros(length(stocks),mmax);

%Computes Two Year Lyapunov Exponents
for stocknum=1:length(stocks)

lyapexp2yrs(stocknum,:)=Lyapunov2(pastdates{stocknum},pasts
tockvalues{stocknum},mmax,maxiter);


lyapexp2yrsavg(stocknum,:)=Lyapunov2(pastdatesavgs{stocknum
},paststockaverages{stocknum},mmax,maxiter);
end
%Computes Lyapunov Exponent of S&P 500
lyapexp1yrSP5002yrs=Lyapunov2(SP500dates,SP500values,mmax,m
axiter);
lyapexp1yrSP5002yrsavg=Lyapunov2(SP500datesavgs,SP500avgs,m
max,maxiter);

%Computes One-Year Index Lyapunov Exponents
lyapexpindex1yr=lyapexp1yrSP5001yr.*correlation'+lyapexp1yr
.*(1-correlation)';
lyapexpindexavg1yr=lyapexp1yrSP5001yravg.*correlation'+lyap
exp1yravg.*(1-correlation)';

%Computes Two-Year Index Lyapunov Exponents
lyapexpindex2yrs=lyapexp1yrSP5002yrs.*correlation'+lyapexp2
yrs.*(1-correlation)';
lyapexpindexavg2yrs=lyapexp1yrSP5002yrsavg.*correlation'+ly
apexp2yrsavg.*(1-correlation)';
```

```
%Computes Index Scaled Standard Deviation
volindex=volSP500.*correlation+vol.*(1-correlation);
volindexavgs=volSP500avgs.*correlationavgs+volavg.*(1-
correlation);

volmatrix=[vol',volavg',volindex',volindexavgs'];

%Reorganizes Lyapunov Exponents for Correlation Calculation
lyapunovexps1yr=[lyapexp1yr,lyapexp1yravg,...
    lyapexpindex1yr,lyapexpindexavg1yr];
lyapunovexps2yrs=[lyapexp2yrs,lyapexp2yrsavg,...
   lyapexpindex2yrs,lyapexpindexavg2yrs];

%Computes correlation between One Year Lyapunov Exponents
and Days Predicted
%by each model
corrmatrix=zeros(4,size(lyapunovexps1yr,2));
for i=1:4
    for j=1:size(lyapunovexps1yr,2)

corrmatrix(i,j)=corr(lyapunovexps1yr(:,j),dayspredicted(i,:
)');
    end
end

%Computes correlation between Two Year Lyapunov Exponents
and Days Predicted
%by each model
corrmatrix2=zeros(4,size(lyapunovexps2yrs,2));
for i=1:4
    for j=1:size(lyapunovexps2yrs,2)

corrmatrix2(i,j)=corr(lyapunovexps2yrs(:,j),dayspredicted(i
,:)');
    end
end

%Computes correlation between scaled standard deviation and
days predicted
corrvol=zeros(4,4);
for i=1:4
    for j=1:4

corrvol(i,j)=corr(volmatrix(:,j),dayspredicted(i,:)');
```

```
    end
end

%Calculates Row and Column Averages
corrcolavgs=mean(corrmatrix,1);
corrrowavgs=mean(corrmatrix,2);
corrcolavgs2=mean(corrmatrix2,1);
corrrowavgs2=mean(corrmatrix2,2);
corrvolcolavgs=mean(corrvol,1);
corrvolrowavgs=mean(corrvol,2);

%Scatterplot of Days Predicted vs. Lyapunov Exponent
figure(1)
scatter(lyapexpindex2yrs(:,1),dayspredicted(1,:))
xlabel('lyapunov exponent');
ylabel('days predicted');
title('Accuracy of Lyapunov Exponents');
%saves the graph as a .pdf file
print('C:\Users\apmur\OneDrive\Documents\MATLAB\IQP\Volatil
ity\lyapunovplot','-dpng');

%Scatterplot of Days Predicted vs. Scaled Standard
Deviation
figure(2)
scatter(vol,dayspredicted(1,:))
xlabel('scaled standard deviation');
ylabel('days predicted');
title('Accuracy of Scaled Standard Deviation');
print('C:\Users\apmur\OneDrive\Documents\MATLAB\IQP\Volatil
ity\stdovermeanplot','-dpng');

%Writes Results to Excel File
xlswrite('ModelPredictionData4',round(corrmatrix,3),'D28:O3
1');
xlswrite('ModelPredictionData4',round(corrcolavgs,3),'D32:O
32');
xlswrite('ModelPredictionData4',round(corrrowavgs,3),'P28:P
31');
xlswrite('ModelPredictionData4',round(corrvol,3),'D36:G39')
;
xlswrite('ModelPredictionData4',round(corrvolcolavgs,3),'D4
0:G40');
xlswrite('ModelPredictionData4',round(corrvolrowavgs,3),'H3
6:H39');
```

```
xlswrite('ModelPredictionData4',round(corrmatrix2,3),'D88:O
91');
xlswrite('ModelPredictionData4',round(corrcolavgs2,3),'D92:
O92');
xlswrite('ModelPredictionData4',round(corrrowavgs2,3),'P88:
P91');
save('Lyapdata')
toc
```