Analyzing Trading Signals

Technical Report

Name	Student Number
Seán Gill	16423306
Will Douglas	16334046
Carl Hany Waheeb	16328331
Ciarán Winters	16388803
Bryan Naughton	16442736
Moses Tijani	15188141

Module: Digital Signal Processing **Module Code:** EE401

Facilitator: Dr. Bob Lawlor

Department of Electronic Engineering,

Maynooth University.

Abstract

The primary objective of this report to lay down the schematics of Analyzing Trading Signals; a large market in the modern world. The brief provided tasked the group the Analysis of an information-bearing signal. The information-bearing signal in question is a Trading Signal/Stock Market signal. This report will provide a comprehensive insight into the workings and process that lead to the culmination of the team's final solutions.

This project is unorthodox. We collect Market Data and apply common DSP techniques coincide with common Trading Methods to obtain beautiful results. Through the sections, the schematics of our approaches will be laid out. The schematics of the sections include the Theory, Implementation, Results and Conclusion.

The scalability of this project is large. The results signify that there is plenty more research that could be carried out on in this field. A different insight to a Trading Signal can be obtained through implementing some DSP theory into practice in an innovative way.

Declaration

I declare that this report and the project it describes is our original work only. We have not plagiarized or excessively quoted the work of others, nor have colluded with others to represent collaborative work as our own. I confirm that I have appropriately cited all information derived from the published and unpublished work of others.

Signed: _____ *Student Number:* _____ *Date:* _____

Signed: _____ *Student Number:* _____ *Date:* _____

Signed: _____ *Student Number:* _____ *Date:* _____

Signed: _____ *Student Number:* _____ *Date:* _____

Signed: _____ *Student Number:* _____ *Date:* _____

Signed: _____ *Student Number:* _____ *Date:* _____

Acknowledgements

An acknowledgement to the Maynooth University Library for access to multiple Literature pieces.

An acknowledgement to Dr. Bob Lawlor for assistance throughout the Project with his great DSP knowledge.

An acknowledgement to the Engineering Department at Maynooth University for the use of the open-access labs in place.

Table of Contents

Table of Figures

1 Introduction

The primary objective of this report surrounds market data for signal analysis. There are various types of Market Data, but this report will focus on three test signals (3.5 Test Signals) with a special emphasis on Stock Markets. This section will introduce what this project is about and what it sets out to achieve.

1.1 Insight to Stock Markets

A stock market is a collection of sellers and buyers of stocks. Stocks which are also referred to as equities, produce partial control of a business. Stock Markets represent the ownerships on businesses. A precise and successful stock market is treated as a crucial location to economic evolution. This is due to the fact it grants business the capabilities to immediately approach capital from the community. Predicting the success of a stock is not easy by any means. Predicting success requires the continuous effort of Stock Analysis.

Stock analysis is the evaluation of a trading instrument, an investment sector, or the market as a whole. Stock analysts attempt to determine the future activity of an instrument, sector, or market. Once the stock analysts have found a successful analyzing technique, they will begin to invest in the stock markets.

Investment in stock markets is most often done via stockbrokerages and electronic trading platforms. Investment is usually made with an investment strategy in mind. In our case, this strategy will be making profitable decisions based on the **Signal Analysis** (stock market analysis) carried out.

1.2 Project Guidelines [1]

The following guidelines were provided to offer direction in how to go about completing the project:

Students are given some level of freedom in the selection of their DSP group project. The reason for this is to try to allow students to undertake a project which is related to their areas of interest. The ideal group size is 5. Groups of 4 and 6 are also permitted.

The purpose of these guidelines is to ensure that your chosen group project is aligned with the fundamental learning outcomes of the module.

1. Your group project must use an information-bearing signal e.g. speech and/or audio, etc..
2. Your group project must include a time-frequency analysis of the above signal
3. Your group project must filter and/or compress the above signal in some way
4. Try to integrate elements of other taught modules wherever possible
5. Make sure that your group project report includes a detailed explanation of the fundamental theory relating to all aspects of your project. The individual interview questions will be based primarily on this fundamental theory and how you applied it.
6. Sections of the course notes may be adapted as necessary. Be sure to reference all sources to avoid plagiarism.

1.3 Project Brief

At the beginning of the project, the group constructed a project brief that consisted of a Background, a Scope, the High-Level Requirements and the High-Level Overview. This brief was based on the **1.2 Project** Guidelines above. The scope that was constructed was followed but more ideas came up throughout the entirety of the project. The methods that were furthered introduced into the project are discussed in detail in their respective section.

1.3.1 Overview

The information-bearing signal in question are generated signals that simulate the behaviour of a typical market. The idea of the project is to first analyse the signal generated by the user through use of Fourier analysis or filters, ex/ Moving average filter, FIR filter, Anomaly Detection filters, etc...

By the end of the project, the aim is to be able to extract stock-markets from the Web as discrete sequences and perform all the analysing methods.

1.3.2 Project Background

The project came about through a brainstorming session. There were multiple ideas proposed including the analysis of an ECG signal. The group finally decided to settle for the Analysis of a Market-like signal for the simple fact that it entails all the learning outcomes. The scalability of the project is also endless.

1.3.3 Project Scope

The project scope defines the boundaries of the project. The following are the project elements and defines what the group will be doing. It sets limits for what will be done as part of the project and the goals that are set.

- Generation of a signal that simulates the behaviour of a Market.
- Time/Frequency Analysis of the generated signal.
- Moving Average Filter
- Anomaly Detection using filters and statistics
- Extraction of Markets from the web
- Computing the statistics based on the markets extracted
- Further Research into trading techniques

These were the goals set at the beginning of the project. These goals were crushed with an extensive amount more of Analysis Methods used.

1.3.4 High-Level Requirements

The final project must include the following:

- Ability to generate a signal that simulates the behaviour of a Market correctly.
- Ability to perform time/frequency analysis of the generated signal or the extracted signal from the web.
- Ability to perform a Moving Average Filter on the signal in question.
- Ability to perform Anomaly Detection on the signal when a signal peaks a certain percentage above the average of the signal.
- Ability to extract market data from the web as discrete sequences.
- Compute all of the DSP/Trading methods created on the signal.

The High-Level Requirements were blown out of the water. Our final solutions consist of a lot more methods that mentioned in 2.3 Our Methods.

2 Background on Existing Solutions

As this is a large field, there are multiple existing solutions out there. So, what can we do differently to the current solutions? It's first important to observe the current analysing methods that are commonly approached in Stock Analysis. Different Stock Markets behave differently. This leads to the questions, are the Analysis Methods that we propose realistic for all types of types of Stock Markets? This section will delve into the behaviour of some the different stock markets and also covering the common Stock Analysis methods.

2.1 Different Markets to Analyse

2.1.1 Crude Oil

One of the market types is **Crude Oil**. It is an extremely active market and it is acknowledged to traders worldwide. It is worth noting that the prices of oil fluctuate massively on the slightest data concerning pricing. This places it as a favorite of swing and a favorite for day traders searching for the slightest edge.

Crude Oil is without a doubt one of the most actively traded assets along with gold which will be discussed at a later stage. The decrease or increase of Crude Oil has a massive impact on many other valuable assets such as gasoline and natural gas. Furthermore, the increase and decrease of the price of the oil influences the costs of stocks, bonds and currencies worldwide. Even though the world currently is interested in the renewable energy sector, Crude Oil remains the biggest and most popular source of energy.

Ticker Symbol	CL
Exchange	NYMEX
Trading Hours	09:00 am – 2.30 pm
Contract Capacity	42,000 Gallons
Contract Months	All Months Jan – Dec
Last Trading Day	Third business day preceding the 25th calendar day of the month before the delivery month.

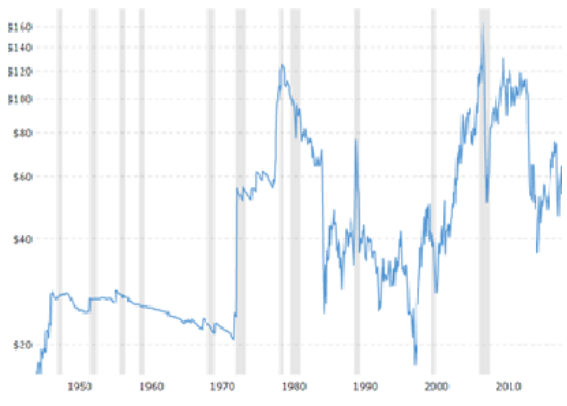


Figure 1: Crude Oil Stock Market Signal

2.1.2 A Football Club

Another market type is a **Football club**. The football world is a gigantic industry to invest in. It is largely closed to the fans. However a minority of the football clubs have shares which any fan can invest in. They are recorded on the stock exchange around the world and include even some of the biggest and renoqned clubs in the world. The most immense football club in the world that is traded on the stock exchange currently is Manchester United. They are traded on the NYSE (New York Stock Exchange) with the symbol noted as NYSE:MANU.

Figure 2: Football Club Stock Market Signal

Manchester United were firstly known as Newton Health Lancashire & Yorkshire Railway Football club and then it was changed to Manchester United in 1892. The price of Shares in the company were valued at £1 each. Manchester United exited the London Stock Exchange in 2005 and were placed on the New York Stock exchange in 2012. The price fluctuated massively since United were first found as in 2012 they were priced at $14 a share. As of 06/11/2019 the price of a share in Manchester United is $16.

2.1.3 Gold

Gold Market is another market that operates worldwide. It relates to buying and selling Gold. An important note to make when referring to the Gold Market is that the Gold spot price is the same all over the entire market, regardless of the currency. Gold trading has been used for several generations and it is ideal for transporting wealth internationally. As a result, this makes it an ideal source for long term investment. Savvy investors watch the Gold market to time their acquisitions to their best advantage.

What determines the price of Gold?

The primary benchmark for determining the market for Gold is the supply and demand. A quite thought-provoking aspect about the Gold market is that it doesn't depend on freshly-mined Gold but it depends heavily on Precious Metal markets. Gold that is recycled for personal use usually come back into the Gold market after several years and its value goes up relatively high. A small research on the Gold market demand will indicate how good and usually reliable the long-term investment of Gold is. [2]

Figure 3: Gold Stock Market Signal

2.1.4 Currency Market

Finally, the **Currency Market** is the final market that will be discussed. It combines the Foreign Currency market and the Euro-Currency Market. The primary base for the foreign exchange market doesn't exist. It is in large dealing rooms of an abundance of central banks, large international banks and some gigantic corporations. These specified rooms are linked by a means of telephone, computer and fax.

The process of trading on the Foreign Exchange Market authorizes rates of transfer for currency. The upwards or the downwards trends for the currency rate also depends on the demand. If the demand increases the price of the currency exchange increases and vice versa. Rapid rate quotes for different currencies are made available by Reuters. "A rate of change for currencies is the ratio at which one currency is exchanged for another".

An important note to outline is that the foreign exchange market does not have an arrangement or constraints and does not have a supervised board. Thus, meaning that if a crisis suddenly develops in the market, there will never be a system that will stop trading. In order to know how to trade on

the Forex Market, the Federal Reserve Bank of New York gives instructions. They highlight the 50 best strategies for trading.[3]

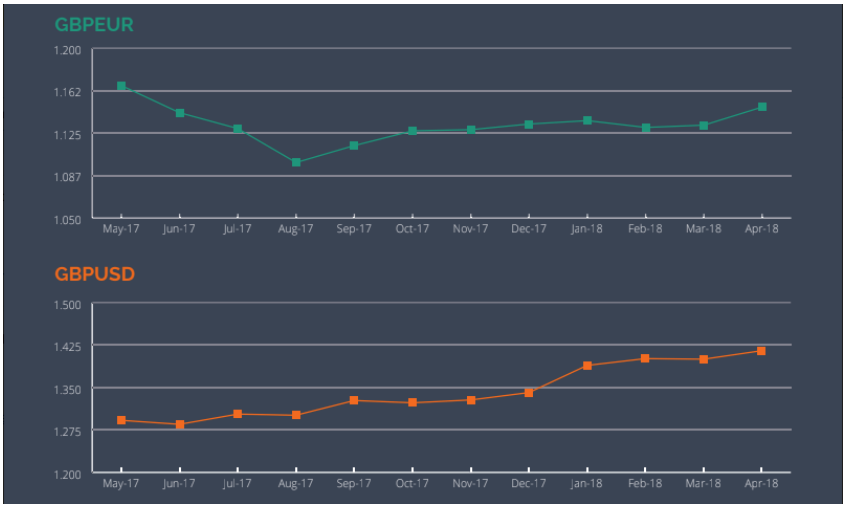


Figure 4: Currency Trading Signal

2.2 Existing Stock Analysis Methods

Stock traders uses three main types of analysis in order to analyze stock markets. The three main types they use are technical analysis, fundamental analysis and Sentimental analysis. When trading stocks there is always a huge risk. There is an ongoing argument as to which analysis is the best between all three in order to reduce this risk when inverting. Although some methods are more beneficial than others in order to make the best decision when investing in stocks you need to use all three analysis's in order to make the most strategic decision and give yourself the best possible chance to make a profit.

2.2.1 Technical Analysis

Technical analysis uses historical data of a specific stock in order to try and predict its future outcome and whether it will increase or decrease in value. Technical analysis uses charts and technical indicators in order to try and find trends and patterns in the market to predict these future outcomes. Technical indicators are mathematical calculations that are applied to a stock's historical charts. There are two types of technical indicators leading indicators and lagging indicators.[4]

Leading Indicators

Leading indicators are very beneficial as the alert the trader before a trend is about to start or before a stock is about to change in the market. This gives the trader a chance to get ahead of the pack and make some very profitable trades. Although these can be very beneficial leading indicators are not always accurate. But when used in conjunction with analysis methods they can lead to very profitable trades. Two types of leading indicators used today are stochastic and Bollinger bands. [5]

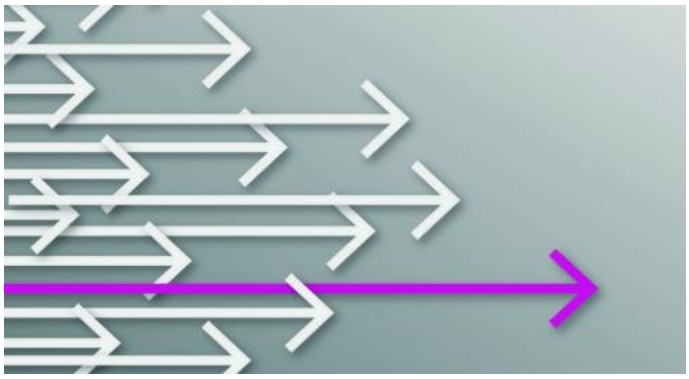


Figure 5: Leading Indicators

Stochastic Indicators

A stochastic indicator calculates the momentum of a stock. It takes the information of a stock over a specific period of time and calculates the range between the high and low-price points. It then compares where the closing price lies in this range and gives the result as a percentage from 0% to 100%. If the closing price is above 80% then this stock is said to be overbought and is expected to drop in price this is a good type to divest and sell these specific stocks. If the closing price is below 20% then this stock is said to be oversold and is expected to increase in price, this is a good time to invest and buy these specific stocks.

Figure 6: Stochastic Indicators

Bollinger Bands

Bollinger bands were introduced in the 1980's. This indicator uses the moving average filter of a signal and then puts two trading bands around this filter, one above and one below. The two trading bands are calculated by adding or subtracting one standard deviation to the moving average.

Form analysis on the Bollinger bands if the market price goes above the bands then those stocks are said to be overbought (overpriced) and expected to drop in the market. If the market price goes below the bands then those stocks are said to be oversold (under-priced) and expected to rise in the market.

If the price action of the stocks becomes volatile and erratic, then the bands expand. If there is low volatility in the stocks and they become bound in a tight trading range, then the bands contract. Period of low volatility tend to be followed by periods of high volatility and visas versa. So, if the Bollinger bands start to pinch together there is a reasonable chance that the volatility will increase

and make either a significant increase or decrease in the market. Thus, creating a trading opportunity.

Figure 7: Bollinger Bands

Lagging Indicators

Unlike leading indicators which predict when a trend is about to begin, lagging indicators follow the trends. Lagging indicators tell you about what has already happened and sends you signals after trends have started. They are used by traders all the time in order to gauge the general trend of a company and to see whether a company is currently increasing or decreasing in value.

The advantage of lagging indicators are that they are a lot more reliable than leading indicators. This is why a lot of trader would prefer to uses them. But since lagging indicators follow the trends instead of predicting trends this makes them less profitable. An example of a popular lagging indicator is a moving average filter. [5]

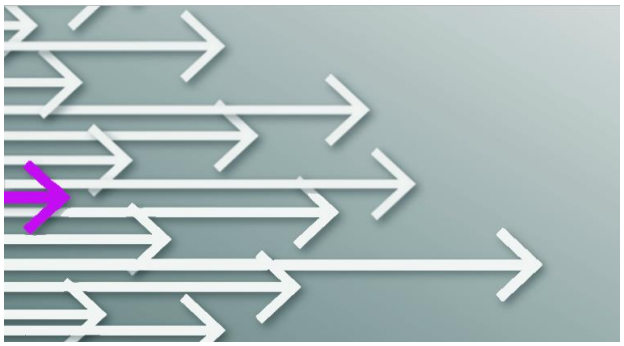


Figure 8: Lagging Indicator

Moving Average Filter

Moving average filters are the most common indicator for stock markets. A moving average smooths out the stock signal by averaging price fluctuations into a single line. It gives a representation of the general direction of the stocks from the historical data. Moving averages give us information of the trends direction, magnitude and rate of change. Moving averages work by taking a previous data over specific period and plotting the mean of the signal.

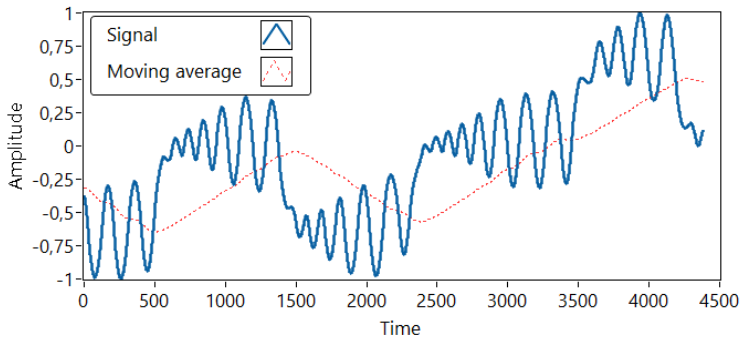


Figure 9: Moving Average Filter

2.2.2 Fundamental Analysis

Fundamental analysis is a very useful when analysing stock trades. Fundamental analysis focuses on the economy, social forces and political forces in order to try and predict future changes in the market. It uses supply and demand as an indicator which seems simple. But in reality is a lot harder. As you must look at events like the increase or decrease in unemployment rate and how this effects the economy or supply and demand of a company.

For example, in simple terms if the economy in a country is good, investors and foreign businesses will invest in that country. This will lead to more demand for that country's currency by all these business and investors thus increasing the value of the country's currency. [4]

2.2.3 Sentiment Analysis

Sentiment Analysis is the analysis of people's opinions on the market. Each individual trader has their own opinion on how the market will act next and whether stocks are bullish (optimistic and on the rise) or bearish (pessimistic and going to drop). This is important as if you feel a stock is bullish but everyone else feels its bearish then the odds aren't in your favour and you might want to take another look at the market before buying. [4]

2.3 Our Methods

Through carrying out the Literacy Background on existing solutions (2.1 & 2.2), it's clear to see that Analyzing Trading Signals is a market in itself. Current existing solutions are constantly being updated/refined to work appropriately. "Yahoo Finance" is a media property that is a part of Yahoo's network. It provides financial data and the most up-to-date analyzing methods on the website. Through harnessing the implementation of analyzing techniques that Yahoo provide and the data to accompany it, our analyzing methods can be verified. Although Yahoo provide the most up-to-date techniques, it does not include some of the innovative solutions that are implemented in this report.

Some of the analyzing methods that are implemented in this report are strict Digital Signal Processing techniques. As defined in the problem description, we must carry out a certain number of tasks on an "information-bearing signal". Generally speaking, most of the existing solutions are founded on statistics and probability as opposed to Digital Signal Processing. A greater insight to the Trading signal can be obtained from using a combination of DSP and Statistical techniques.

Multiple programming scripts (in Python & MATLAB) have been created to implement the following functions. The following is a high-level description of the functionality of the Methods used. The methods will be further discussed in their respective sections.

Table 1: Methods Researched

Method	Functionality
Time/Frequency Analysis	Analysis of the Trading signal in the discrete-time domain and the frequency domain (magnitude & phase)
Windowing Methods	Changing the spectral properties (larger – subset) to obtain a greater insight to the signal in question
Savitzty-Golay Filter	A DSP technique for smoothing the data without distorting the signal tendency.
Bollinger Bands	For Anomaly Detection. Tailoring the method using a combination of the Savitzty-Golay and current Bollinger strategy.

Moving Average Filter	Taking the average of the previous n samples. Smooths the sampled signal.
Average Anomaly Detection	Using the Moving Average Filter signal and statistics to draw conclusions on where Anomalies have occurred.
Linear Predictive Algorithm	Obtaining the m linear co-efficients and predicting the next Trading Value. (generally used for audio analysis)
Cross-Correlation	Analyzing the correlation between certain markets using the DSP technique
Data-set Narrower	Created to be able to use the stochastic indicator. Narrows a data-set down and obtains the Highest, Lowest, Open and Close prices of the defined narrowing length.
Stochastic Indicator	Shows the momentum of the Signal by comparing the closing price with previous trading ranges.
Machine-Learning	Using common machine learning techniques on all of the data generated from the techniques implemented

The methods above will be broken down in the following format:

1. **Theory** behind the Method
2. **Implementation** of the Method (what software did we use, what libraries?)
3. **Results** collected from the analysis method
4. **Conclusion** on the results collected and explaining the significance of the results.

Many of the methods require some of the results from another method. For example, the Bollinger Bands is a statistical analysis of the Savitzky-Golay filtered signal. For the confusion that could occur, all of the signals will be strictly labelled including the filtered signals that are fed in as inputs to other Analysis Methods.

3 Generating Market Data

The information-bearing signal in question are generated signals that simulate the behavior of a typical stock-market/trading signal. Generating data that accuracy tests can be performed on is essential on drawing conclusions from the current trading methods. Companies such as Corvil, Pico, Morgan Stanly, JP Morgan rely on realistic datasets that they can perform accuracy tests on. Hence, the first key part of this project is to generate controllable data that we understand all of the characteristics regarding. Once those datasets are generated, we run our Analysis Methods against the dataset. We then collect Results and draw conclusion.

With regards to the generation of data, there were 3 main approaches. Sum of Sinusoidal with Noise Data, Discrete Auto-Regressive Data and data extracted from Yahoo Finance. Using these 3 approaches will ensure that we can draw conclusive results. The generation of data was done implementing different MATLAB and Python libraries. The format of the data was maintained constant.

3.1 Specifications of Data Generated

The generated data will be in the file extension CSV. A comma-separated values (CSV) file is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format. A CSV file typically stores tabular data (numbers and text) in plain text, in which case each line will have the same number of fields.

The CSV file will consist of two rows namely **N** and **F(N).** Where N will simulate the time data-point and F(N) the current Trading Signal Value. The CSV will have the following specifications.

- **Length of Data:** 10000 data-points.
- **Step-Size:** 0.01 **Length: 100**

The behavior of **F(N)** will depend solely on the type of Data generated. Re-emphasizing the objective that **F(N)** simulates the behavior of a trading signal/stock markets.

3.2 Sinusoidal Combinations w/ Noise

The term sinusoid simply means that the sinusoid is an activity of some changing value which could be t is comparable to the sine function but may be altered in phase, frequency, amplitude or any mixture of the 3 mentioned.

The common equation for a sinusoid function is:

$$f(t) = A\sin(2\pi ft + \theta) = A\sin(\omega t + \theta)$$

- A is amplitude of the signal. This is defined as the maximum values of the function implemented.
- f is the common frequency. This is the number of cycles per unit of t.
- ω : 2πf is the angular frequency. This is outlined as the number of radians per unit of 2.
- Finally, θ ; the phase offset (in radians).

It is also worth highlighting that the sum of sinusoids with the identical frequency is also accepted as a sinusoid.

The sum of sinusoids chosen in our case is:

```
20*sin(2*pi*0.025*t) + 150*sin(2*pi*0.1*t) + 150*sin(2*pi*0.02*t);
```

As it can be clearly illustrated the amplitude values are chosen randomly as well as, the value of f and the phase offset. These values were chosen through trial-and-error to obtain a sine-wave that somewhat simulates the behaviour of a Trading Signal.

3.3 Discrete Auto-Regressive Data

In statistics, econometrics and signal processing, an autoregressive (AR) model is a representation of a type of random process; as such, it is used to describe certain time-varying processes in nature, economics, stock-markets, etc. A statistical model is autoregressive if it predicts future values based on past values. For example, an autoregressive model might seek to predict a stock's future prices based on its past performance. Autoregressive models implicitly assume that the future will resemble the past.

There is an element of random involved with the auto-regressive data. The script that can be found in the Appendix titled autoregressive_sig_generation.py is the code used to generate the signal. **High-level Overview** of the script:

1. Firstly, defining all of the dataset parameters. This includes the length, the step-size of the data-set. The starting trade value or starting **F(N)** value is initialized to be 200. The signal will initially start to rise.
2. The current trade value will to increase/rise a certain amount based on random number generators. The following conditions are set:
    a. If the **rising_state** is **True** – The next **F(N)** value will have an 80% of Rising, else it will decrease. The value that the signal increases/decreases by is a random number between (0, 20) **UNLESS** an anomaly occurs. An anomaly will happen 2% of the time and this means that the value will rise/decrease by a random amount between (100, 300).
    b. If the **rising_state** is **False** – The next **F(N)** value will have an 80% of Falling, else it will rise. The value that the signal increases/decreases by is a random number between (0, 20) **UNLESS** an anomaly occurs. An anomaly will happen 2% of the time and this means that the value will rise/decrease by a random amount between (100,300)
3. Once the entire data-set is generated based on the above conditions, the Matrix is appended to the CSV file. The file is then run through the different Analysis methods and results can be collected.

The behavior of the script beautifully simulates the behavior of a typical trading signal. As the behavior of the signal is correct, the analysis methods can be carried out on the signal and conclusions can be drawn and verified.

3.4 Extracting Market Data – Yahoo Finance

The final test signal will be an actual market data signal extracted from the Yahoo Finance website. The website provides an intuitive tool where you can extract the stock-market values as a CSV file. The extraction tool that Yahoo provide is ideal as all of the tests that we perform are on a CSV file as previously mentioned in the Specifications section. The extraction is as follows:

1. Go onto the Yahoo Finance website and search for a market of your choice. For this example, the Apple USD equity data will be extracted.
2. Once you are on the market of your choice, navigate to the historical signal tab. You can then download the data (for your given time-series range) with all of the different columns.

Date	Open	High	Low	Close*	Adj Close**	Volume
Dec 18, 2019	279.80	281.25	279.12	281.01	281.01	14,108,922
Dec 17, 2019	279.57	281.77	278.80	280.41	280.41	28,539,600
Dec 16, 2019	277.00	280.79	276.98	279.86	279.86	32,046,500
Dec 13, 2019	271.46	275.30	270.93	275.15	275.15	33,396,900
Dec 12, 2019	267.78	272.56	267.32	271.46	271.46	34,327,600
Dec 11, 2019	268.81	271.10	268.50	270.77	270.77	19,689,200
Dec 10, 2019	268.60	270.07	265.86	268.48	268.48	22,605,100
Dec 09, 2019	270.00	270.80	264.91	266.92	266.92	32,010,600

Figure 10: Historical Data Example

All the columns have their own precise meaning with regards to the Stock Market. In the Stochastic Indicator section, five of the seven sections are used for Analysis.

3.5 Test Signals

Based on the specifications the Three Test Signals were generated and extracted as a CSV file.

3.5.1 Test_Sig_1.csv

This signal follows the specifications from **3.2 Sinusoidal Combinations w/** Noise**.**

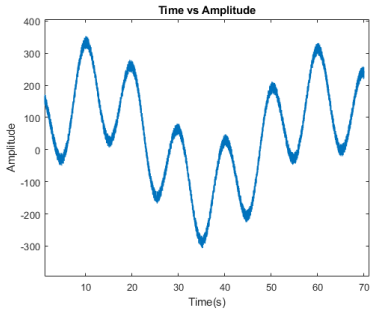


Figure 11: Test_Sig_1

3.5.2 Test_Sig_2.csv

This signal follows the specifications from **3.3 Discrete Auto-Regressive** Data**.**

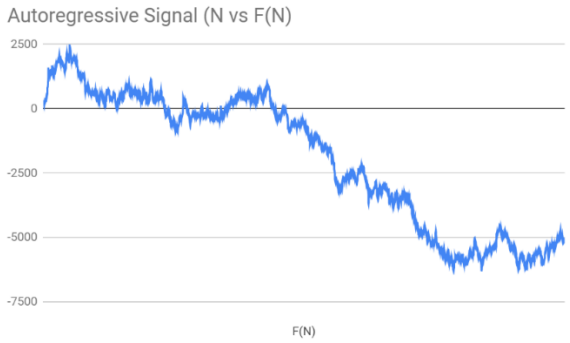
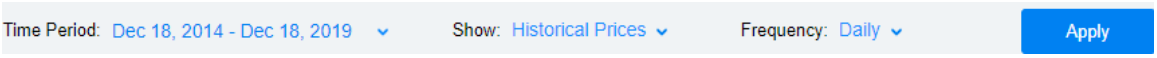


Figure 12: Test_Sig_2

3.5.3 Test_Sig_3.csv

5 years' worth of data from Yahoo Finance on the Apple Equity Symbol (or stock market) [6]

Figure 13: Test_Sig_3



4 Time/Frequency Analysis

Fourier spectrum analysis is a powerful technique that describes the distribution of power contained in a stationary data over the frequency component of the signal.[7] Many natural events are oscillatory in nature which are frequency depended, now understanding the frequency dependence can produce some powerful information.[8] Spectral analysis allows us to decompose signals into its periodic components. [9]

The history of spectral analysis in finance can be traced as early as the 1920s. The application of spectral analysis in finance was not established until Granger and Hatanaka published a compressive overview of there are two Fourier method in 1964.[10]

4.1 Fourier Spectrum Analysis

Fourier transform discovered by Joseph Fourier in 1805. he said that "any" function on the interval [0,1] can be written as a sum of sine and cosines.[11]

$$f(x) = \sum a(n)\cos(\cos(2\pi x)) + \sum b(n)\sin(2\pi n x))$$

In general, the Fourier Transform is a general and important set of mathematics. Every signal is determined by its spectrum. The Fourier Transform allows us to transform a Signal into a different space. In a nutshell, a signal can be observed and analyzed in the Time Domain and Frequency Domain through the Fourier Transform.

As stated, any continuous/discrete signal in the time domain can be represented by a sum of carefully chosen sinusoids. Sinusoids can totally be described by their amplitude, phase and frequency.[12] The objective of Spectrum Analysis is to take our **3.5 Test** Signals and observe them in the Time and Frequency Domain.

Discrete Fourier transform (DFT) is the basic building block for spectral analysis. MATLAB allows us to do this using the fast Fourier transform (FFT) function. The input of the FFT is the discrete sequence of F(N) defined in the particular test-signal. The output of the DFT is a complex value. The magnitude & phase of the complex DFT sequence can be calculated through basic elementary-level mathematics:[13]

$$\text{Where } H(f) = \sum[a_n + b_n i]$$

$$|H(f)| = \sqrt{a_n^2 + b_n^2}$$

$$< H(f) = \tan^{-1}(\frac{b}{a})$$

4.1.1 Implementation

The code used to calculate the DFT of the test signals can be seen in the Appendix in the section:

4.1.2 Results

Often when observing the frequency domain, the phase plot is disregarded. The phase is needed though if one were to reconstruct the signal as a sum of sinusoids.

For some of the plots below, the DC offset was calculated through obtaining the mean across the entire interval of the signal. We remove the DC offset through subtraction of the mean value. Another way of removing the DC-Offset is through implementation of a High-Pass Filter. Removing the DC offset, allows us to observe the natural frequency without the big impulse around the 0 frequency.

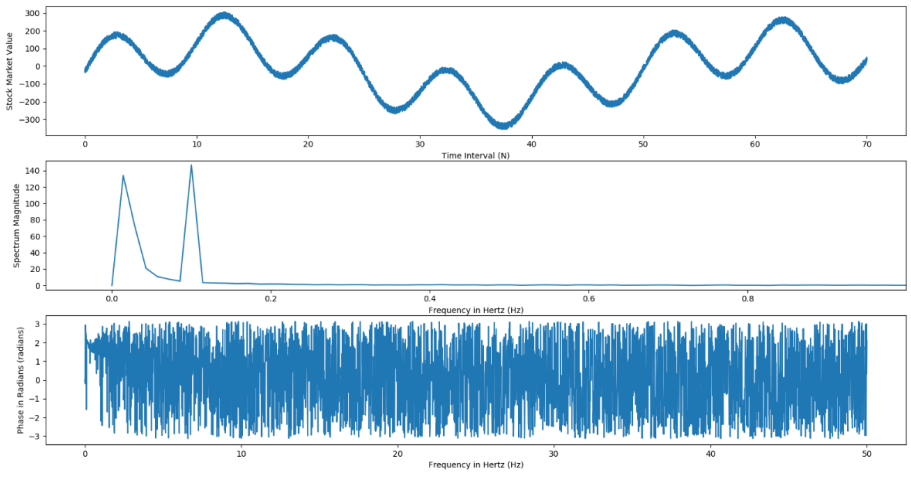


Figure 14: Test_Sig_1

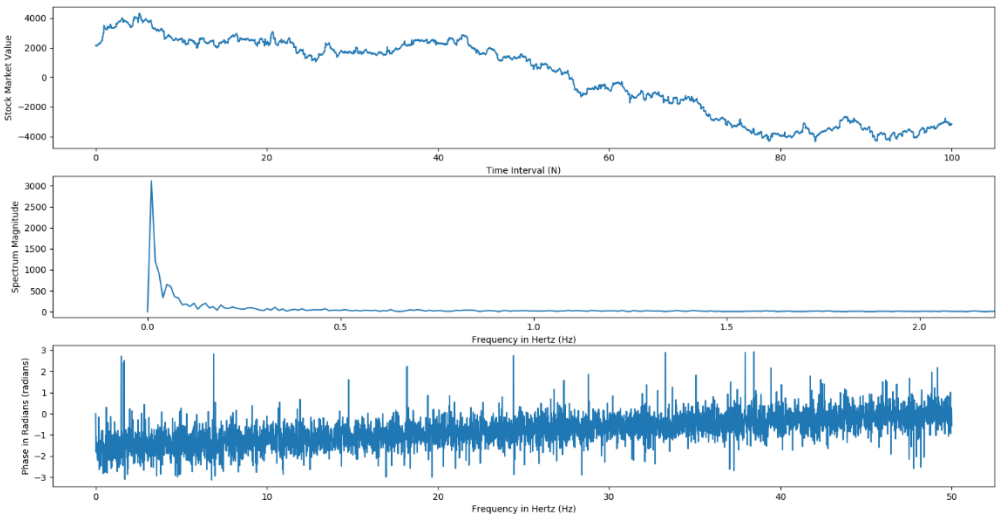


Figure 15: Test_Sig_2

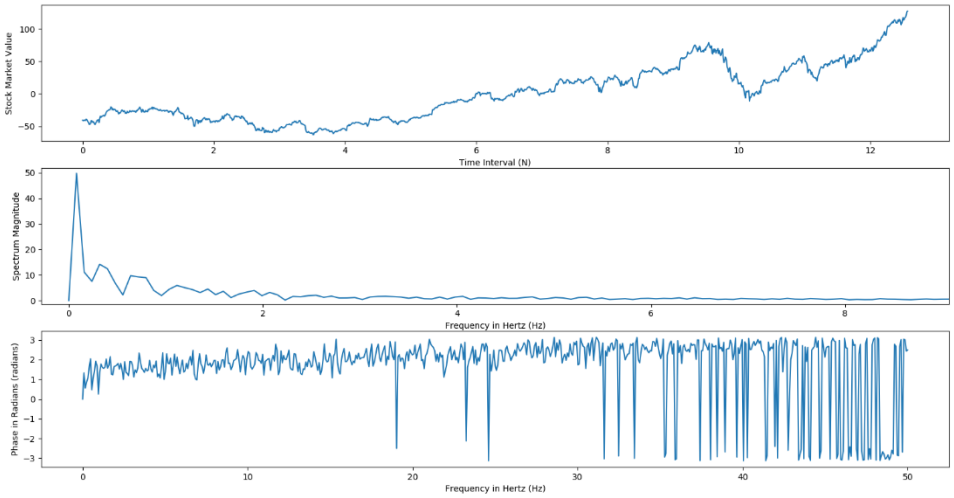


Figure 16: Test_Sig_3

4.1.3 Conclusion

It was examined whether the Fourier analysis really does provide a certain advantage for investors forecasting the future development of a stock market price. The main finding is that this method basically fails to find the existing predominant cycles. Based on an attempt to detect significant periods in stock market data, using FFT – one of the methods of the Fourier analysis – it has been found that this method is unacceptable. Moreover, it is reasonable to expect similar failures in the case of other liquid investment instruments or similar financial data series. The Fourier analysis is

still used nowadays for forecasting in finance and its benefit is under the discussion among both financial market practitioners and academicians.

Why does FFT method fail? One possible explanation is non-stationarity of the data series, which could also be connected to the presence of too much "financial" noise in the development. This problematic represents one of the limitations of the research and it is recommended as a possible way of further studies in the area.

Another limitation is that the research does not take into consideration a spectrum variability during the time, so a possible way how to possibly improve speculative results is to test and assess the agent operating on frequencies based on generating a spectrogram.[14]

5 Spectral Estimations

Spectral estimations are ways of describing the distribution of power contained in a signal. These estimations are on finite sets of data.[15] This section will focus on the nonparametric and parametric methods.

The nonparametric method makes no assumptions about the correlation structure of the data which it estimates the PSD, Power Spectral Density, directly from the signal itself and it assumed that the signal is calibrated. With the simplest method that will be covered in this section being the periodogram and the welch method.

5.1 Non-Parametric Methods

Spectral estimation allows us to describe the distribution of power contained in a signal, the periodogram is one of the most common nonparametric estimates for computing the PSD. [16]

5.1.1 The Periodogram

The periodogram is a nonparametric estimate of the power spectral density (PSD) of a wide-sense stationary random process. The periodogram is the Fourier transform of the biased estimate of the autocorrelation sequence.[17] For a signal, x_n, sampled at F_s samples per unit time, the periodogram is defined as:

$$\hat{P}(f) = \frac{\Delta t}{N} \left| \sum_{n=0}^{N-1} x_n e^{-j2\pi fn} \right|^2, \quad -1/2\Delta t < f \leq 1/2\Delta t,$$

where Δt is the sampling interval. For a one-sided periodogram, the values at all frequencies except 0 and the Nyquist, $1/2\Delta t$, are multiplied by 2 so that the total power is conserved.

The modified periodogram multiplies the input time series by a window function. A suitable window function is nonnegative and decays to zero at the beginning and end points. Multiplying the time series by the window function tapers the data gradually on and off and helps to alleviate the leakage in the periodogram. See Bias and Variability in the Periodogram for an example.

If h_n is a window function, the modified periodogram is defined by:

$$\widehat{P}(\omega) = \frac{1}{2\pi N} \left| \sum_{n=0}^{N-1} h_n x_n e^{-j\omega n} \right|^2, \quad -\pi < \omega \leq \pi.$$

With MATLAB implementing the following theory, multiple different results were gathered for different window functions. Different windowing methods exhibit different results. The following windowing methods were implemented. [18]

Parameters for Windowing Functions: M – Number of points in output window,

Hamming Window

The Hamming window is a taper formed by using a raised cosine with non-zero endpoints, optimized to minimize the nearest side lobe.

The Hamming window is defined as

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{M-1}\right) \qquad 0 \leq n \leq M-1$$

It was recommended for smoothing the truncated autocovariance function in the time domain. Most references to the Hamming window come from the signal processing literature, where it is used as one of many windowing functions for smoothing values. It is also known as an apodization (which means "removing the foot", i.e. smoothing discontinuities at the beginning and end of the sampled signal) or tapering function.

Rectangular Window

A rectangular window or Dirichlet window, this is equivalent to no window at all.

Blackman Window

The Blackman window is a taper formed by using the first three terms of a summation of cosines. It was designed to have close to the minimal leakage possible. It is close to optimal, only slightly worse than a Kaiser window.

The Blackman window is defined as:

$$w(n) = 0.42 - 0.5\cos(2\pi n/M) + 0.08\cos(4\pi n/M)$$

The "exact Blackman" window was designed to null out the third and fourth sidelobes, but has discontinuities at the boundaries, resulting in a 6 dB/oct fall-off. This window approximates the "exact" window, which does not null the sidelobes as well, but is smooth at the edges, improving the fall-off rate to 18 dB/oct.

Most references to the Blackman window come from the signal processing literature, where it is used as one of many windowing functions for smoothing values. It is also known as an apodization (which means "removing the foot", i.e. smoothing discontinuities at the beginning and end of the sampled signal) or tapering function. It is known as a "near optimal" tapering function, almost as good (by some measures) as the Kaiser window.

Bartlett Window

The Bartlett window is very similar to a triangular window, except that the end points are at zero. It is often used in signal processing for tapering a signal, without generating too much ripple in the frequency domain.

The Bartlett Window is defined as:

$$w(n) = \frac{2}{M-1}\left(\frac{M-1}{2} - \left|n - \frac{M-1}{2}\right|\right)$$

Most references to the Bartlett window come from the signal processing literature, where it is used as one of many windowing functions for smoothing values. Note that convolution with this window produces linear interpolation. It is also known as an apodization (which means "removing the foot", i.e. smoothing discontinuities at the beginning and end of the sampled signal) or tapering function. The Fourier transform of the Bartlett is the product of two *sinc* functions.

Hanning Window

The Hann window is a taper formed by using a raised cosine or sine-squared with ends that touch zero.

The Hann function is defined as:

$$w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{M-1}\right) \qquad 0 \le n \le M-1$$

The window was named for Julius von Hann, an Austrian meteorologist. It is also known as the Cosine Bell. It is sometimes erroneously referred to as the "Hanning" window, from the use of "hann" as a verb in the original paper and confusion with the very similar Hamming window.

Most references to the Hann window come from the signal processing literature, where it is used as one of many windowing functions for smoothing values. It is also known as an apodization.

5.1.2 Results of Periodogram

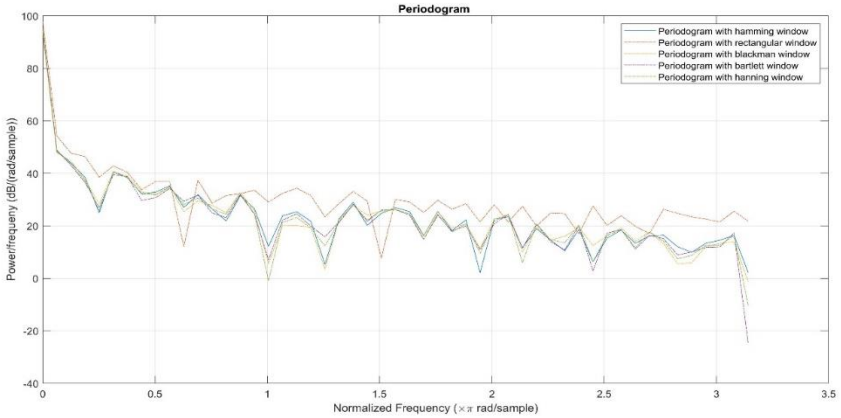


Figure 17: Periodogram w/ Numerous Methods Compared

The periodogram provided us the power spectral density estimate of our synthetic financial data. it showed how the total variance of the signal is distributed over the frequency and comparing the window like the rectangular, hamming, Henning, Bartlett and Blackman windows to get a visual representation of how different window effect our signals.

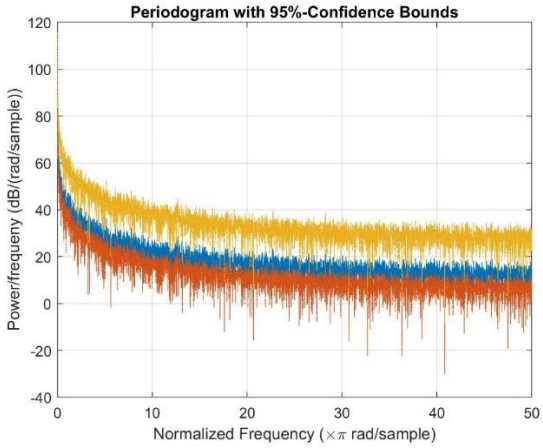


Figure 18: Periodogram w/ 95% Confidence Bounds

for a 95% confidence bound with the periodogram for our PSD estimate.

The periodogram is the basis for variety of advance estimation techniques for the estimation of the PSD .The periodogram is not a consistent estimator for the PSD because the variance of the behaves like the power spectrum squared , which is independent of L this means as L gets big variance doesn't decrease. If the periodogram was a consistent estimator, the variance would converge toward zero as L goes to infinity.

Which leads us to our next method which tackles the limitations of the periodogram, the welch method which is the averaging of the periodogram in other to reduce the variance of the power spectrum density estimator.

5.1.3 Welch's Method

Welch's method computes an estimate of the power spectral density by dividing the data into overlapping segments, computing a modified periodogram for each segment and averaging the periodograms.[19]

Denote the m th windowed, zero-padded frame from the signal x by

$$x_m(n) \triangleq w(n)x(n+mR), \quad n = 0, 1, \ldots, M-1, \; m = 0, 1, \ldots, K-1,$$

where R is defined as the window *hop size*, and let K denote the number of available frames. Then the periodogram of the m th block is given by

$$P_{x_m,M}(\omega_k) = \frac{1}{M} \left|\mathrm{FFT}_{N,k}(x_m)\right|^2 \triangleq \frac{1}{M} \left|\sum_{n=0}^{N-1} x_m(n) e^{-j2\pi nk/N}\right|^2$$

as before, and the Welch estimate of the power spectral density is given by

$$\hat{S}_x^W(\omega_k) \triangleq \frac{1}{K} \sum_{m=0}^{K-1} P_{x_m,M}(\omega_k).$$

Using the pwelch function pxx = pwelch(x,window,noverlap,nfft) looking into the parameter we see that window specifies the length of the window that were going to use if no window it set. pwelch would divide the signal into 8 segment and then multiplies these segment using hamming

window where the length is equal to the segment length as the default window but we can also compare different windows and how the effect on our power spectral density plot as shown.

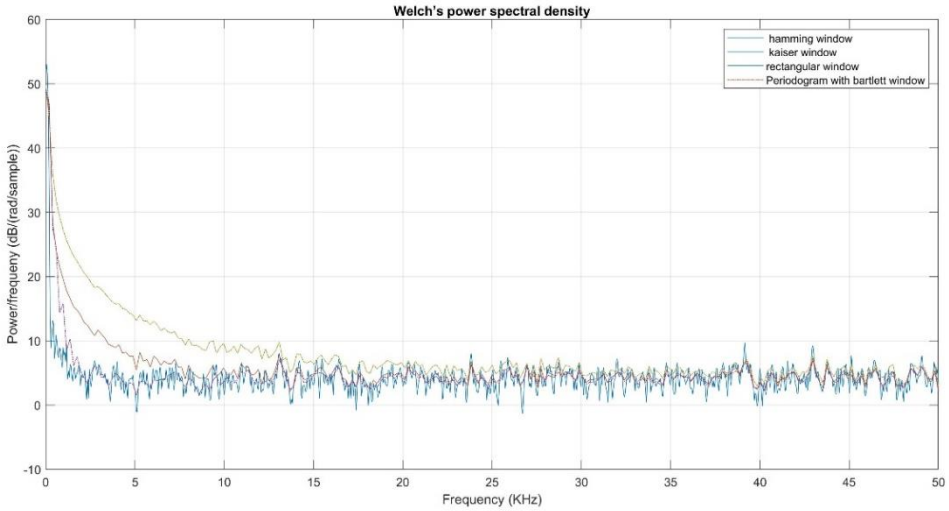


Figure 19: Welch's Power Spectral Density

Fundamentally we can see that the welch method is a much better estimator. There is a fundamental trade off that exists in the welch method between spectral resolution and statically stability. Maximizing the block L in order to maximize the resolution.

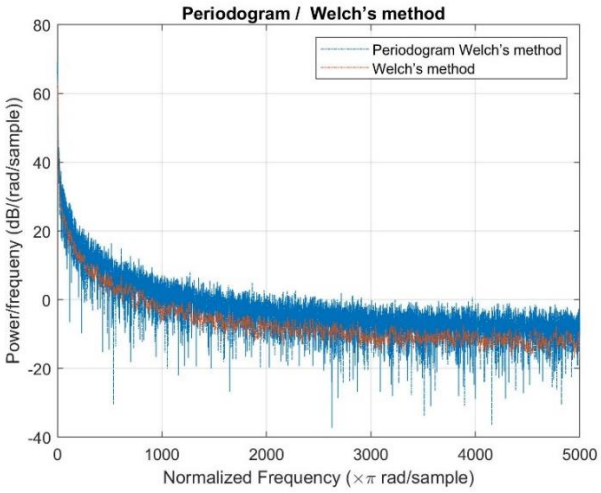


Figure 20: Periodogram/Welch's Method

5.2 Parametric Methods

We now covered the nonparametric method using the periodogram which is not a consistent estimator for the PSD and the welch method where now going to jump into parametric method, the yule walker equations.

5.2.1 The Yule Walker method

The yule walker also called autocorrelation or windowed method equations describe a very important property of auto-regressive model.[20] yule walker equations relate auto-regressive model parameters to auto-covariance of a random process. The auto-regressive model views a random signal as the output of a linear time invariant system in response to white noise input. The yule walker equation relates auto-regressive model parameters to the-coefficients of the linear time invariant system to the auto-covariance of the random-process. For the derivation we're going to assuming that our process is zero mean, meaning that the auto-covariance and the autocorrelation are the same quantity were going to start with the definition of the autoregressive model given as:

$$\sum_{k=0}^{N} a_k x[n-k] = w[n] \; ; \; a_0 = 1$$

Where x is the time series and w[n] is the white noise which is the input the liner time invariant system. To get the yule walker equations we multiply both side $x[n-\gamma]$ and take an expectation.

$$\sum_{k=0}^{N} a_k \, E\{ \, x[n-k].\, x[n-\gamma]\} = E\{w[n]x[n-\gamma]$$

The first (left side) expectation involves the data times a shifted version of data which is the definition of the autocovariance function $r_{xx}[\gamma - k]$. The right-side we got the present value of the white noise W[n] while γ is positive x[n- γ] represents the past values of the output, W[n] is independent from the time samples to time samples we would see that a past value of the output is unrelated to the present value of the input we can summarize this relationship that state:

$$\sum_{k=0}^{N} a_k . r_{xx}[\, \gamma - k] = \begin{cases} 0 & \gamma > 0 \\ \sigma^2 & \gamma = 0 \end{cases} \; ; \; a_o = 1$$

When $\gamma > 0$ we have

$$for\ \gamma > 0 : \sum_{k=0}^{N} a_k . r_{xx}[\ \gamma - k] = \ -r_{xx}$$

we can write this relationship in matrix vector form

$$\begin{vmatrix} \gamma = 1 \\ \vdots \\ \gamma = N \end{vmatrix} \begin{bmatrix} -r_{xx[0]} & \cdots & r_{xx[1-N]} \\ \vdots & \ddots & \vdots \\ r_{xx[1-N]} & \cdots & r_{xx[0]} \end{bmatrix} \begin{vmatrix} a_1 \\ \vdots \\ a_N \end{vmatrix} = - \begin{vmatrix} r_{xx[1]} \\ \vdots \\ r_{xx[N]} \end{vmatrix}$$

This is the yule walker equation which comprises of set of N linear equations and N unknow parameters. Therefore, we nowhave a system that relates the autocovariance sequence of the signal (r_{xx}) to the autoregressive parameters a_1 through a_N with N equation and N unknows [21]. We can see that if we know the r_{xx} autocovariance autocorrelation of the process we can solve for the a's, sigma squared can be obtain by considering the case when γ is equal to zero.

We can represent yule-walker equations in a compact from, where R is the matrix of the correlation coefficients, a is the vector for the autoregressive parameters and for the vector on the right-hand side of correlation coefficients.

$$R.a = -r \ \ where\ a = \ -R^{-1}\, r$$

Once we have a we can now substitute value of as into the L equal to zero case given us the value of sigma squared.

$$\sigma^2 = \sum_{k=0}^{N} a_k . r_{xx}[-k]\ ; for\ \ \gamma = 0$$

using the aryule method for the autoregressive power spectral density estimate which is implementing the yule walker method.

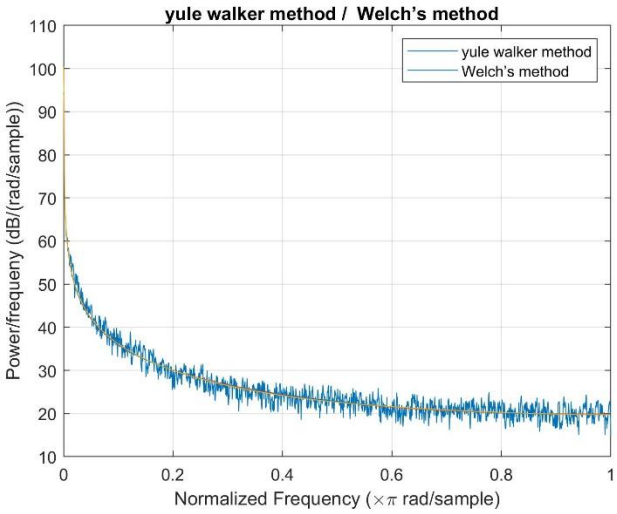


Figure 21: Yule Walker Method/Welch's Method

An estimate of the PSD using the yule-walker method with autoregressive model and Comparing to the welch method. It ascertained that the estimated yule-walker method is as expected with massive error decrease.

In conclusion as we have seen the power spectrum represents the power as a faction of frequency telling us where the energy is distributed. There two main method for power spectrum estimation the nonparametric and parametric method. Nonparametric dose not assume any model generated the data meaning that the nonparametric is based on using the DFT. The parametric method is based on the uses of model allowing us to assume that the data is generated in a certain way. there different types of system that could be used like autoregressive, the moving average and the autoregressive moving average. The parametric could misleading if the model is wrong but when the model is right the parametric gives a higher quality estimate with less data that would be required by a nonparametric method.

6 Filtering Methods

There were two main filtering methods implemented for the project, a Moving Average Filter and the Savitzky-Golay Filter.

6.1 Moving Average Filter

6.1.1 Theory

The Moving Average filter (MA filter) is an application of the Low Pass Finite Impulse Response filter, which is commonly used for smoothing of a discrete signal. To smooth the signal the MA filter takes **L** samples of the discrete signal at the time point specified and produces a single output, the average of the samples. We have implemented the Simple Moving Average filter as our formula.

The Moving Average filter can be summarized as having the following three important functions:

1. It takes **L** input points, computes the average of these input points to produce a single output point.
2. Due to the calculation of a single output point along the discrete signal, we can be sure that there is a certain amount of delay in every cycle.
3. The filter acts as a Low-Pass Filter, which entails poor frequency domain response and good time domain response.[21]

6.1.2 Implementation

The Simple Moving Average formula:

$$y[n] = \frac{1}{L}\sum_{k=0}^{L-1} x[n-k]$$

Where,

- **L** is the point size of our moving average filter.
- **x** is the input samples.
- **y** is the averaged output.

Moving Average filters are categorized as an **L-point Moving Average filter**, where **L** is the sample size.[22]

For example, if we have a 4-point Moving Average Filter, the resultant output **y[n]** is:

$$y[n] = \frac{1}{4}(x[n] + x[n-1] + x[n-2] + x[n-3])$$

50-point Moving Average FIR Filter

A 50-point Moving Average FIR Filter was used that takes the current and previous 49 samples of input and calculates the average.

We have implemented two methods of gathering the average of each point:

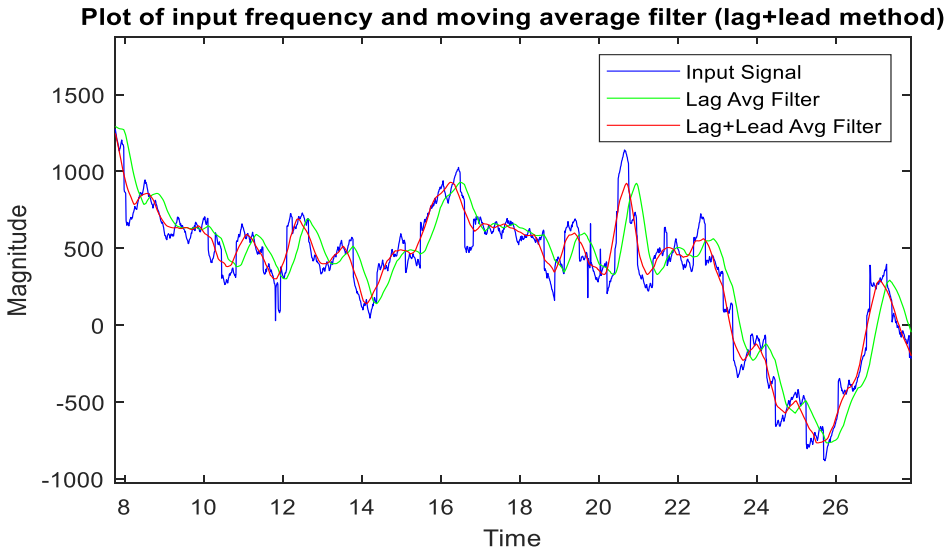


Figure 22: 50-Point Moving Average filter w/lagging and lag+lead method compared

Fig.22 shows the Lagging method and the Lag+Lead method on a single plot for ease of understanding.

1. **The Lagging Method**

The lagging method outputs the average of the sample points that have all already occurred before itself. This means for a sample size of **L**, we start our range of samples at time **(t – L)** and finishes at **(t)**. In **Fig.22** we can see that the 'Lag Avg Filter' is correctly calculating the average of the signal with the points previous and displays the current average at each point.

Following the previous example of a 4-point Moving Average Filter, we can write as:

$$y[n] = \frac{1}{4}(x[n] + x[n-1] + x[n-2] + x[n-3])$$

2. The Lag+Lead Method

The lagging method is the correct implementation of the Simple Moving Average but to display this in a much more viewer-friendly we devised the 'Lag+Lead Method', essentially this method time-shifts the Simple Moving Average by, **(-L/2)**, this allows us to perform a number of operations on the moving average filter.

- We can display it in line with the Input Signal.
- We can perform error analysis on each point.

Displaying these results alongside each other is essential to our analysis of each signal.

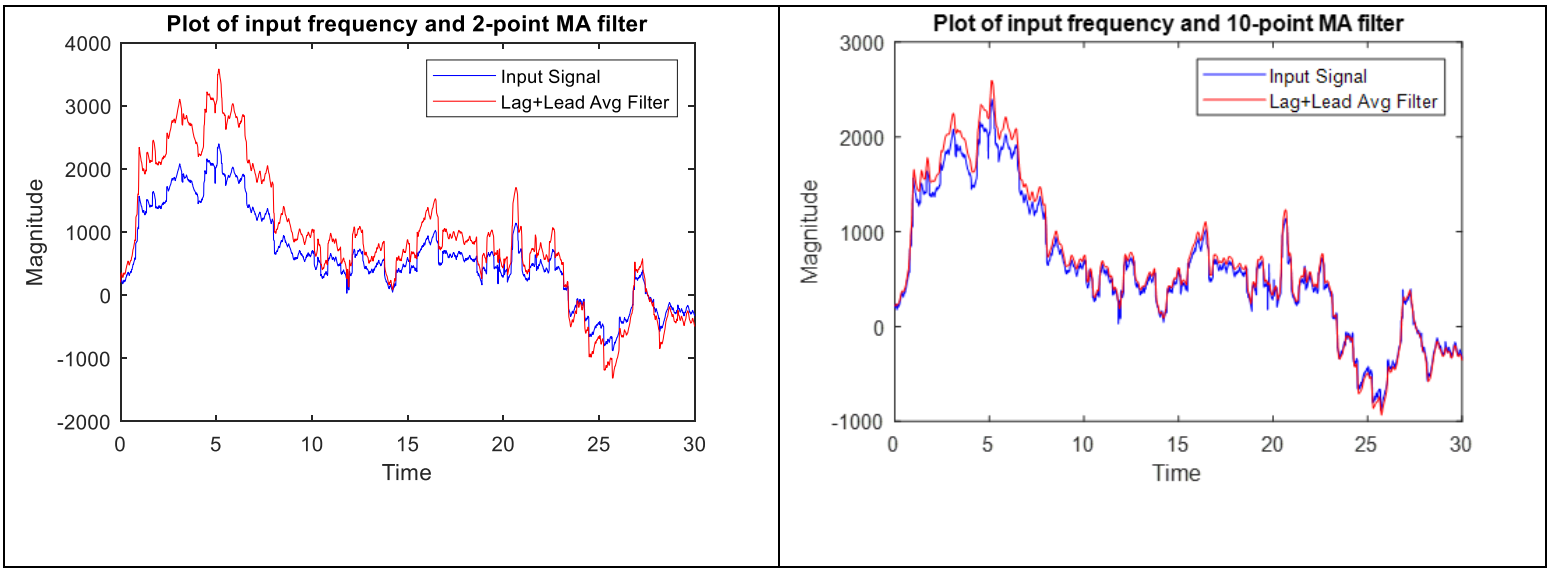
Following the previous example of a 4-point Moving Average Filter, we can write as:

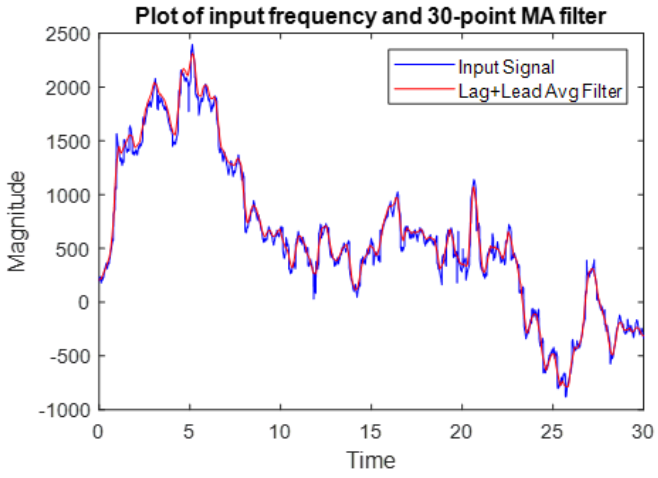
$$y[n] = \frac{1}{4}(x[n+2] + x[n+1] + x[n] + x[n-1])$$

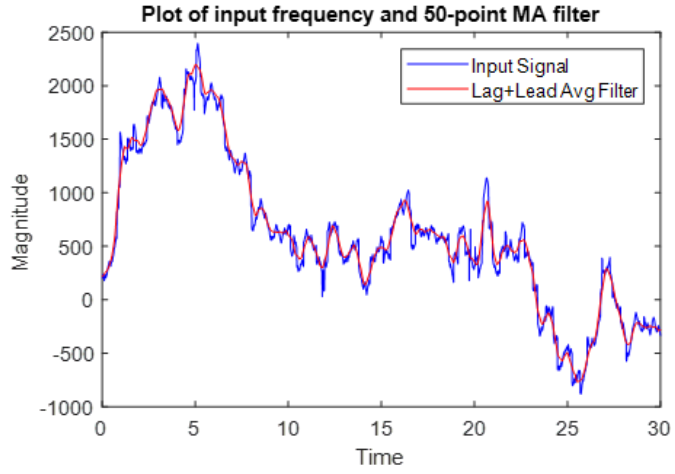
6.1.3 Results

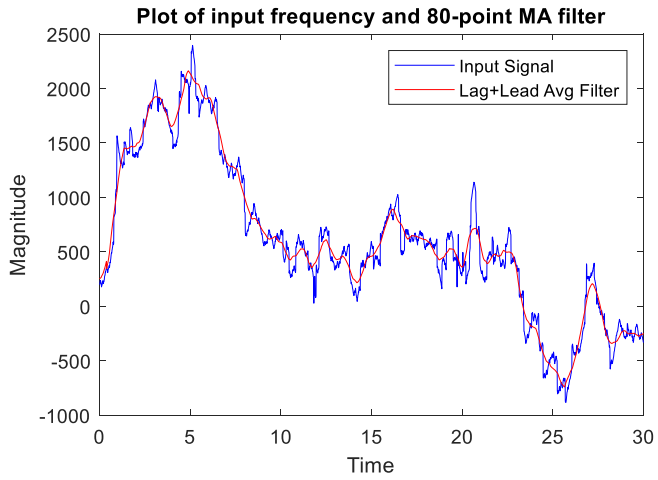
The L-Point Average Filters were investigated by varying the number of point sizes to explore the subsequent effect which this had on the signals. The data used was kept constant to allow for an honest examination of the effects of differing the value of L.

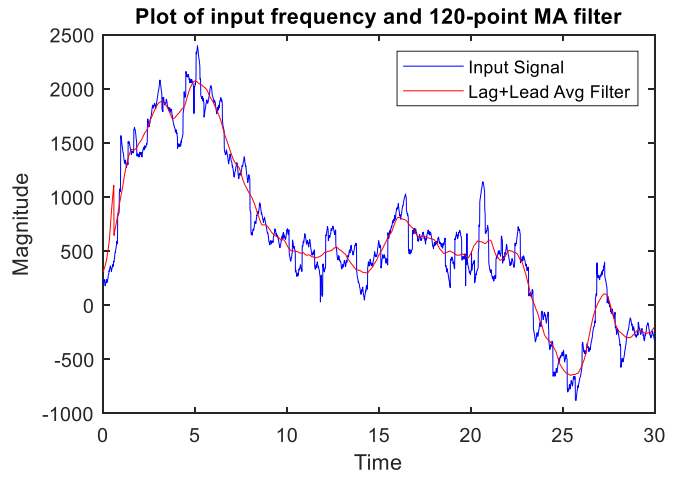
Table 2: Plot of input frequency w/ varying values of L-Point MA filters

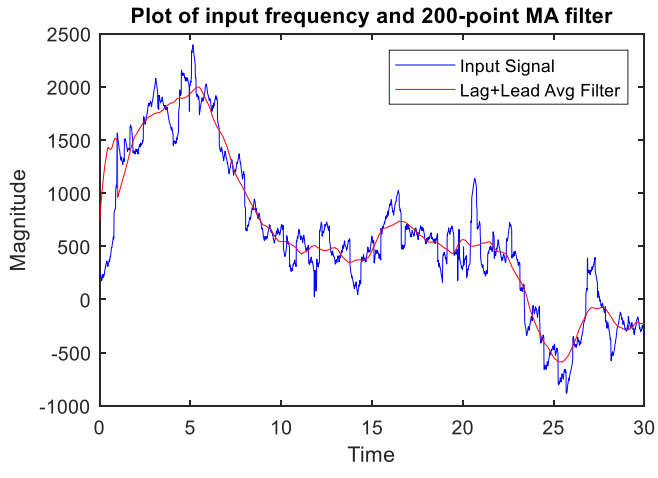



Plot of input frequency and 30-point MA filter
Input Signal
Lag+Lead Avg Filter
Magnitude
Time


Plot of input frequency and 50-point MA filter
Input Signal
Lag+Lead Avg Filter
Magnitude
Time


Plot of input frequency and 80-point MA filter
Input Signal
Lag+Lead Avg Filter
Magnitude
Time


Plot of input frequency and 120-point MA filter
Input Signal
Lag+Lead Avg Filter
Magnitude
Time


Plot of input frequency and 200-point MA filter
Input Signal
Lag+Lead Avg Filter
Magnitude
Time

When the L is varied it effects the data in which it takes into account for the calculation of each point in the moving average. More of both the past and future data is used when the value of L is larger and less then when it is smaller. The value of L changes the smoothing of the average curve as evidenced by the above experiments. As the value set for L increases the signal tends to cancel out more of the noise and creates a smoother signal. As when L is set to a small value like 2, the moving average almost follows the original signal with a slight offset.

As can be seen from the experiments above when L is set to a very high value like 120 points, this results in the beginning of the signal to be slightly distorted although it does provide quite a smoothed overall resulting signal.

The most suitable values for L seem to be 50 & 80, both of these signal disregard quite a bit of noise but also have a good balance in not completely losing the shape of the original signal as some of the smaller fluctuations are still present.

6.2 Savitzky–Golay Filter

6.2.1 Theory

The Savitzky-Golay Filter (SGF) was first introduced in 1964 by Abraham Savitzky and Marcel E. J. Golay therefore resulting in the name, the aim of the filter is to lessen the amount of noise and to smoothen the chosen signal or data. It was settled upon that the Savitzky-Golay filter would be used to analyze the market data, as it would be capable of eliminating the unwanted noise which originates from the small variations in the value of the data, therefore in turn providing a clearer signal in terms of market analysis, as over a large period of time the smaller changes in the data's values are deemed to be unnecessary.[23]

There is one disadvantage of the SGF, this is that it cannot be used at the beginning of a signal as it needs both previous and following data points to calculate the essential coefficient used in the calculation of a particular point. To overcome this problem an offset can be introduced, usually there is no offset, offset = 0 and then if the offset is set to offset = -1 this will then return the point previous to the point in question therefore allowing for the calculation of values at the beginning of a signal, varying the level of the offset to achieve the desired point.

6.2.2 Implementation

The method behind the SGF involves using the points surrounding the point in question, creating a polynomial to acquire 2n+1 points, which are bordering the point which is to be smoothed. As a result, the value chosen for n must be an odd positive integer number.

The equation used in the calculation of the points incorporates a normalization factor and a number of different coefficients depending on the number of data points which were chosen to carry out the smoothing.[24]

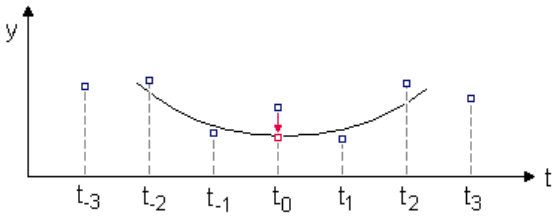


Figure 23: SGF Illustraion

Where,

- y_t is the output of the equation.
- h is the normalization factor, which can be found for each number of data points chosen in the figure below.
- np is the number of data points chosen in which to carry out the SGF.
- a_i is the coefficient
- x_{t+I} is the inputted value of the array at the position x+t.

$$y_t = \frac{1}{h}\left[\sum_{i=-\frac{np-1}{2}}^{\frac{np-1}{2}} a_i x_{t+i}\right]$$

It can be noted that the positive values for $a_i = -a_i$ as the coefficients are found to be symmetric. A table of the first eleven sets of coefficients and normalization factor can be found in the figure below. This particular list only incorporates a certain amount of data points, a further list can be found online providing the variables for when there is more data points used.

Table 3: SGF NP Co-efficients Table

NP	h	a0	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12
5	35	17	12	-3	0	0	0	0	0	0	0	0	0	0
7	21	7	6	3	-2	0	0	0	0	0	0	0	0	0
9	231	59	54	39	14	-21	0	0	0	0	0	0	0	0
11	429	89	84	69	44	9	-36	0	0	0	0	0	0	0
13	143	25	24	21	16	9	0	-11	0	0	0	0	0	0
15	1105	167	162	147	122	87	42	-13	-78	0	0	0	0	0
17	323	43	42	39	34	27	18	7	-6	-21	0	0	0	0
19	2261	269	264	249	224	189	144	89	24	-51	-136	0	0	0
21	3059	329	324	309	284	249	204	149	84	9	-76	-171	0	0
23	805	79	78	75	70	63	54	43	30	15	-2	-21	-42	0
25	5175	467	462	447	422	387	343	287	222	147	62	-33	-138	-253

An important factor in the input values into the SGF is that the data must be at equal intervals which is more than often the case for market data, which are taken at even intervals to provide honest data.

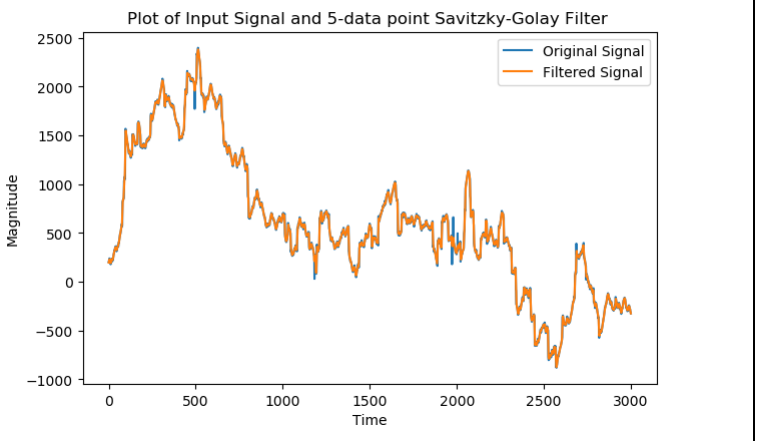
The function used to utilize the SGF was taken from SciPy.org. The parameters of the function are explained below; savgol_filter(*x*, *window_length*, *polyorder*)[25] x is the data which is to be filtered, in this case it is an array of the chosen market data.

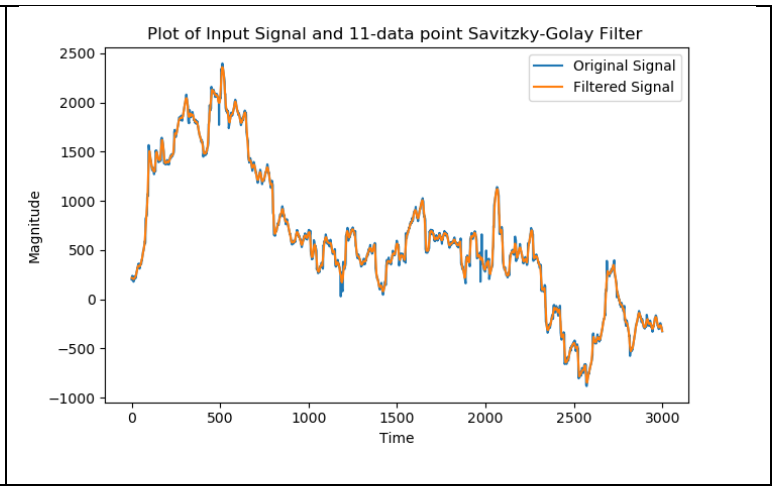
- window_length is the number of coefficients.
- polyorder is the order of the polynomial chosen.
- All of the rest of the parameters are set to the defaults as they had no positive impact on the analysis of the signals, some of these include, the order of the derivative data and axis which is the axis which the inputted array is plotted upon.

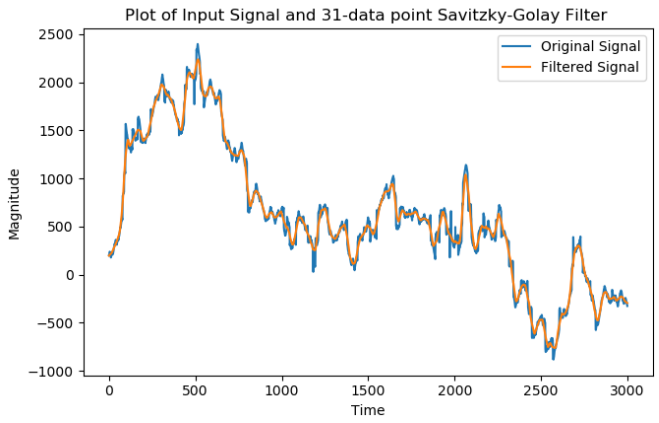
6.2.3 Results

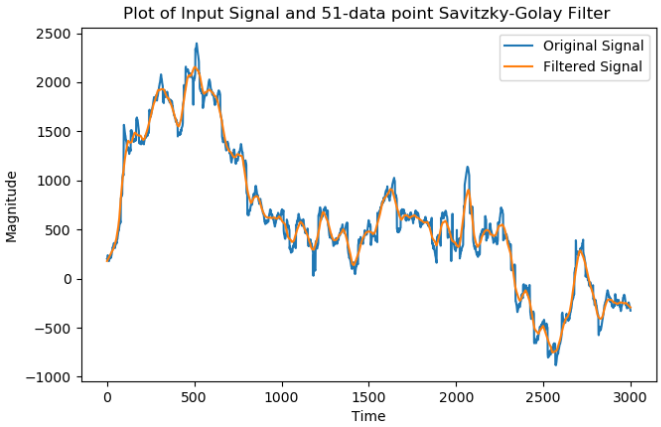
The same data which was used for the Moving Average Filter above is also used in the SGF and another examination was carried out to understand how the filter work when varying the different variables in its calculation. The number of data points is initially varied going from 5 all the way up to 501 to provide a broad spectrum of results to allow for more in depth analysis.

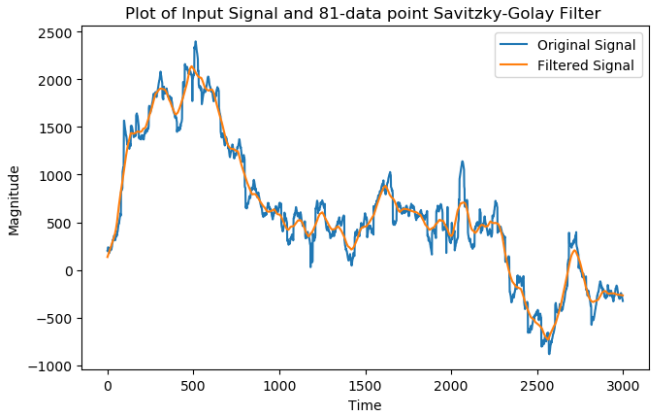
Table 4: Plot of input frequency w/ varying values of bits for Savitzky-Golay filter

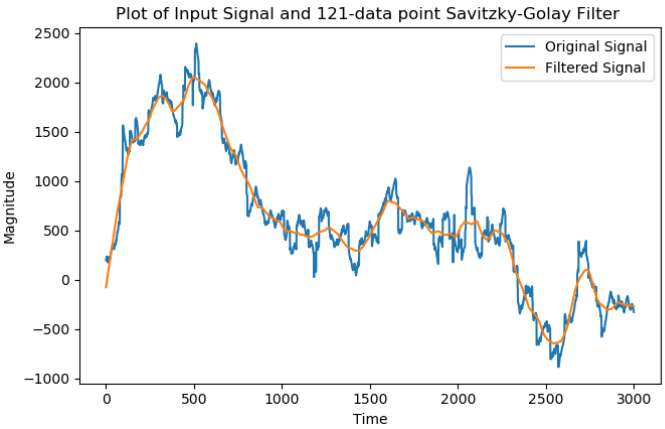


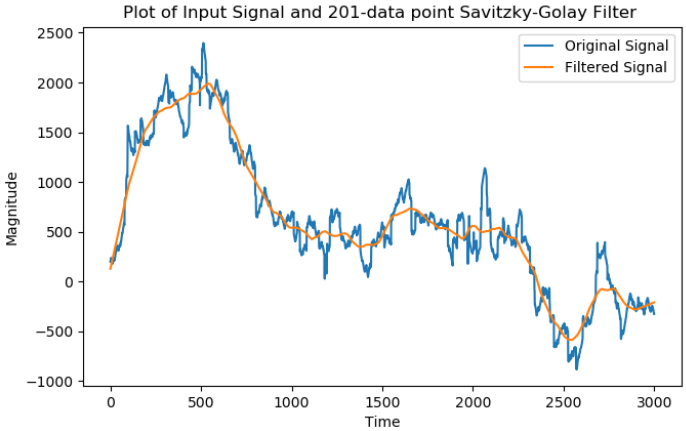


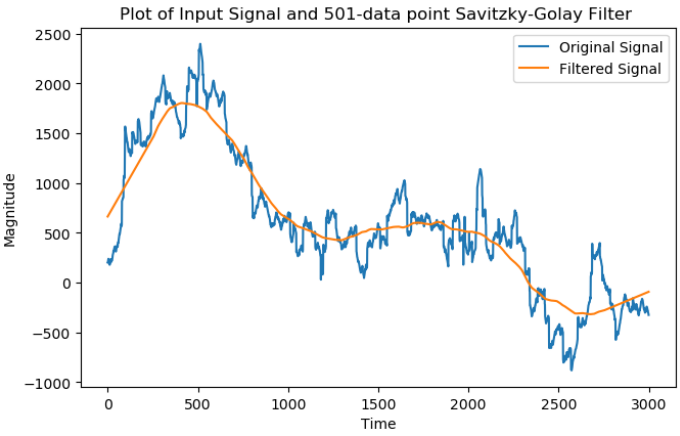












As the number of data points increases the level of noise in the resulting signal is in turn decreased and oppositely the smoothness of the signal increased. The reasoning behind this is as when the

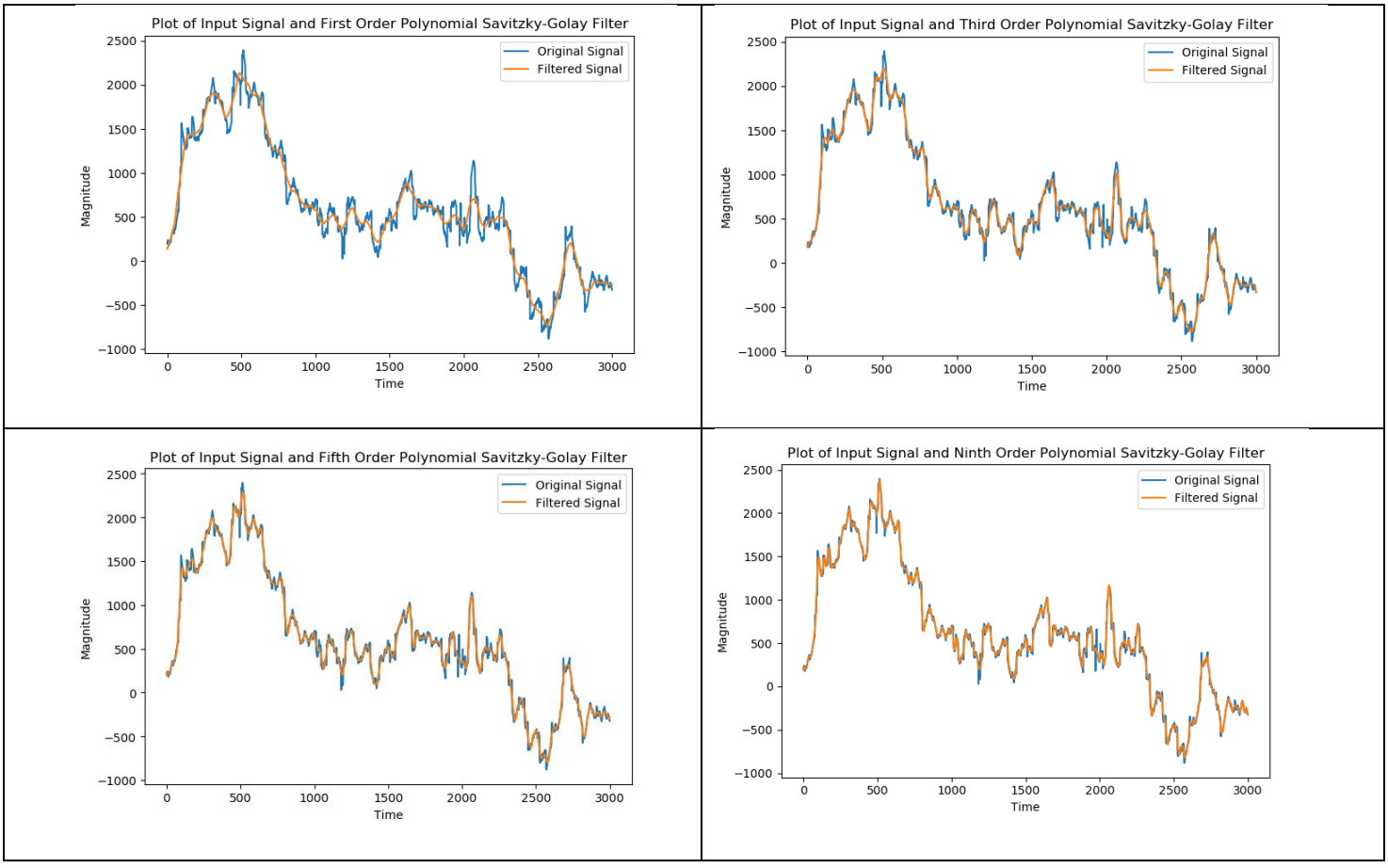
number of data points is increased the calculation takes into account more of the data so therefore provides a smoother result because it takes into account more of the future and past data and is able to create a less rigid signal.

Similarly, to the Moving Average Filter the SGF when the number of data points is very low the output signal is almost identical to the input signal.

The most suited value for the used data seems to be either the 51, 81 or 121 data point SGFs. These dismiss a lot of the unnecessary small fluctuations and noise but also maintain a good shape of the signal over this period.

Varying the order of the polynomial is another variation which was analyzed for the SGFs. As a result of changing the polynomial order this incorporates different normalization and coefficient values. The number of data points is kept constant at 81 for these tests.

Table 5: Plot of input signal w/ varying orders of polynominals



There is one key aspect that can be taken away from the results above, that the order of the polynomial has the opposite effect to that of increasing the number of data points. The output becomes more alike to the input the greater the order of polynomial therefore cancelling out the effect of increasing the number of data points to smoothen out the signal.

6.3 Conclusion

Both methods used for filtering have mainly similar properties. As previously explained both signals become smoother when the number of points used increases, therefore the number of data points to be used should be altered dependent on the task in question, small values for L are more suited to signals with a small amount of data points or when specific time intervals are under examination. The larger values are more ideal for getting an overview of discrete data as a whole and analyzing the performances and behavior over large periods.

One major advantage which SGF has over the Moving Average Filter is that when the number of data bits is very high, for example around the value of 120, the SGF has no sign of distortion while the Moving Average method does. This would make it more difficult to achieve an understanding of the signal at its early stages in terms of the Moving Average Filter. The SGF also has no offset effect on the output signal when the number of data points is low unlike the Moving Average Filter.

Overall the two methods of filtering are very identical and as a result they are both used in the area of anomaly detection as can be seen in section 7.

7 Anomaly Detection

Anomaly detection is the identification process of unexpected events in a given data set, which deviate from the norm. It is a critical step in data analysis in order to identify technical errors, significant change or a potential opportunity depending on what data is being analyzed. All anomalies should be investigated carefully. Before deciding if these data points should be eliminated from the data, we should first try to understand why they appear and if similar values will continue to appear. Thus, before any anomalies can be detected we must first characterize what normal behavior is.

There are several different methods of identifying anomalies, the ones we will be concentrating on are:

- Using the Box-Plot method
- Using the Bollinger Bands

We will be comparing the results obtained through both methods against each other.

7.1 Using the Box-Plot Method

7.1.1 Theory

The box plot method is useful for describing the behavior of the data in the middle and outer ends of the distributions. This method uses the mean of the data set and the lower (Q1) and upper (Q3) quartile ranges which are the 25^{th} and 75^{th} percentiles respectively, to construct fences that can determine if a mild outlier or an extreme outlier has been identified. The interquartile range is (Q3 – Q1). This method was is an adapted version of the standard method of creating a bell curve and distributing the data within it.[26]

7.1.2 Implementation

The calculation of the box plot method involves initially finding the moving average, which we decided on using results obtained from the Simple Moving Average Filter calculated in section 6 – Filtering Methods.

The fences are calculated as follows:

Where, **σ** is the chosen multiple of the standard deviation:

- lower inner fence: Q1 – σ/2*IQ
- upper inner fence: Q3 + σ*IQ
- lower outer fence: Q1 – σ/2*IQ
- upper outer fence: Q3 + σ*IQ

Where, Q1 is lower quartile, Q3 is upper quartile and IQ is the interquartile range.

The standard deviation (σ) is a statistic that measures the dispersion of a data set with regards to its mean.

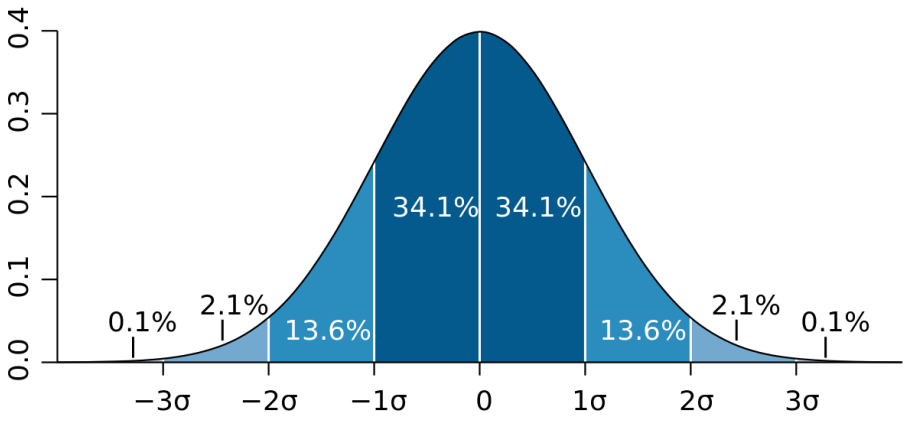


Figure 24: Stadard Deviation Bell Curve

We can vary the value of the standard deviation we use in our calculation, to include more or less of our data within the inner fences. This ability to move data between the inner fences, outer fences and finally outside of the fences altogether allows us to investigate thoroughly what is an anomaly, based on our given data set.

Outlier detection criteria

Any data point between both inner fences is not an outlier. Data points between the inner fence and the outer fence is considered mild outlier. Any data point outside the outer fence is a considered an extreme outlier, or in our case an anomaly.

The designed figure below visualizes the criteria for outlier detection.

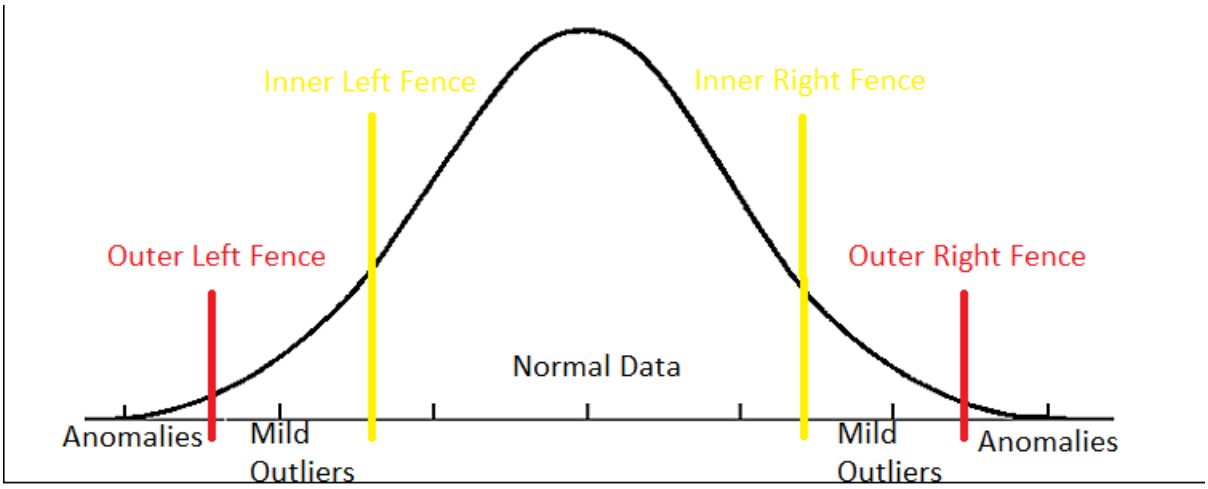


Figure 25: Box Plot Method w/ Data Categorized Illustration

7.2 Using Bollinger Bands

7.2.1 Theory

The name, Bollinger Bands comes from the creator, John Bollinger who was a well-known trader. As explained previously in section 2 – Literary Background on Existing Solutions, Bollinger Bands are a set of bands which surround the signal and are commonly used as an analysis tool in trading when detecting possible anomalies and are practiced to assess the degree of volatility, which is a measure of how liable the data is to change quickly and unpredictably.

The Bollinger Bands use the standard deviation of the moving average of a signal in the calculation of each of the upper and lower bands. Standard deviation itself is a measure of volatility, as it is an amount which describes how much a certain value varies from the mean. Along with the area of volatility there comes security, security plays a big role in identifying possible successful markets as it encourages more investment when there's less risk involved.

The Bollinger Bands can also be useful in the discovery of various opportunities which may provide a higher likelihood of success, an example of this is when a 'Squeeze' occurs. The squeeze is a period when the bands are close in proximity and is a sign of volatility in the near future and maybe a possible investment opportunity. The behavior of the bands depends on the volatility, they stretch when the volatility increases and compress when there's a decrease in volatility.

Bollinger bands can be easily adjusted to different sets of data and this makes it a valuable tool when it comes to market analysis. The bands do not provide an indication when a change may occur and this can be seen as a possible drawback.

7.2.2 Implementation

The calculation of Bollinger Bands involves initially finding the moving average in this case in the Savitzky-Golay Filter previously calculated in section 6 – Filtering Methods is used as the moving average.

The bands can be set however many standard deviations away from the moving average as the user likes and can be easily adjusted depending on their preference of accuracy. The higher the number of standard deviations will return a lower number of errors outside the bands as they have a greater distance between them. The standard deviation was modified so that the variations in the outcomes could be investigated. The standard normal distribution table was used to gauge the level of accuracy expected.

There are two equation calculations involved in obtaining both the upper and lower bands and are as follows.

$$\sigma = \sqrt{\frac{\sum |x - \mu|^2}{N}}$$

N = Number of values in data set

x = Each value in data set

μ = Mean of all values in data set

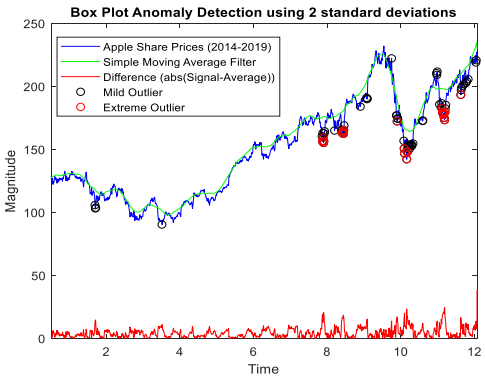
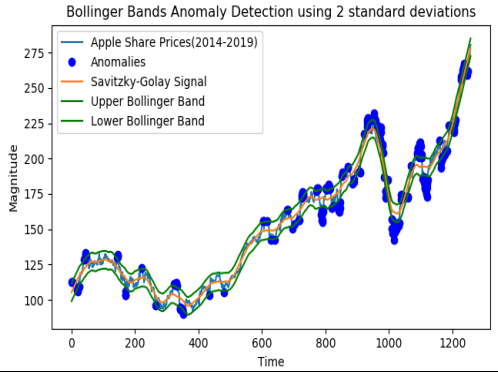
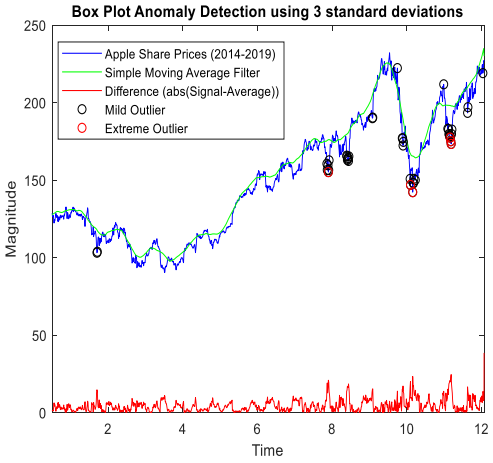
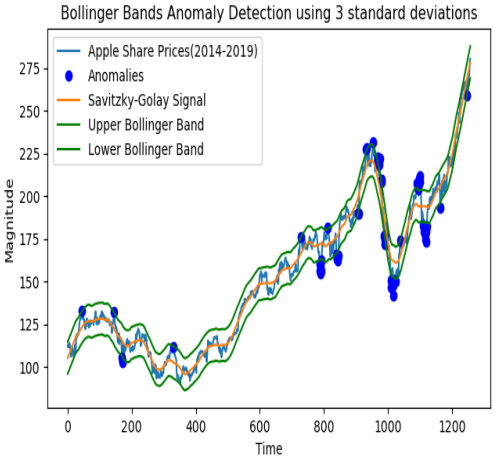
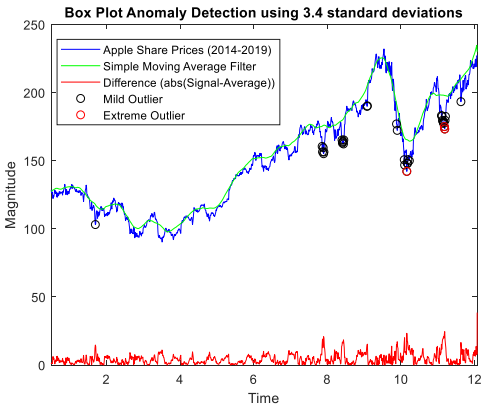
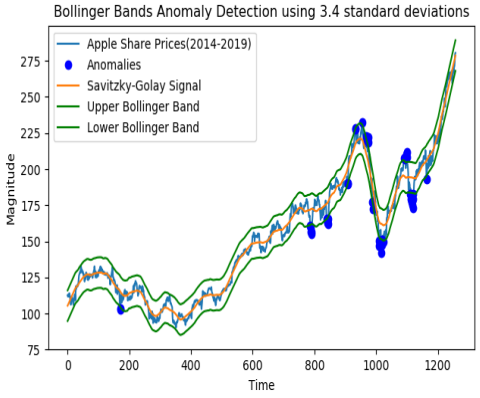
Upper Band = SGF + x(σ)

Lower Band = SGF - x(σ)

7.3 Results

7.3.1 Anomaly Detection Test 1

Table 6: Setting L to 50 for Box Plot and Number of bits to 51 for Bollinger Bands results

	
	
	
Note: Box plot figures were plotted on MATLAB while Bollinger band figures were plotted on Python, they were plotted along same time scales so, (Boxplot time 2 == Bollinger Band time 200 and so on)	

Box Plot Method Results

An anomaly is represented on the Box Plot method by a red hollow circle. If you view where the red circles appear along the 'Apple Share Prices' and then look down to the 'Difference' signal, you can see that the points of greatest amplitude line up with the extreme outliers. The Difference signal is the error between the moving average and the original signal.

We can also see that Mild Outliers are identified with black hollow circles, this represents areas where the data is not quite an anomaly, but you could expect that an anomaly is likely to occur around this point.

The number of anomalies detected from largest to smallest is:

1. 2 standard deviations
2. 3 standard deviations
3. 3.4 standard deviations

This is expected as we know that:

1. 2 standard deviations will account for about 95% of the data
2. 3 standard deviations will account for about 99.7% of the data
3. 3.4 standard deviations will account for about close to but not equal to 100%

Bollinger Bands Results

An anomaly is represented on the Bollinger Bands method by a blue circle. We can see quite clearly that every time a data point exceeds either the Upper or Lower Bollinger Band it is marked as an anomaly, this is known as a 'Breakout' when the data exceeds the bands, has no effect or indication of how the signal will behave next. Breakouts can provide a focus on key dates in a company's share value. Many various Breakouts and are mainly as a result of influential factors which affect the value.

We can see how volatile the signal is at a given time by the spreading of the Bollinger Bands. A squeeze can be identified towards the end of the sampling time, highlighting a period of time where the level of volatility is high.

The number of anomalies detected from largest to smallest is:

1. 2 standard deviations
2. 3 standard deviations
3. 3.4 standard deviations

Comparing the Results

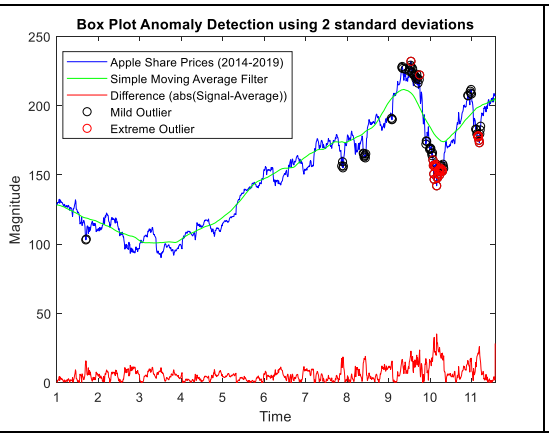
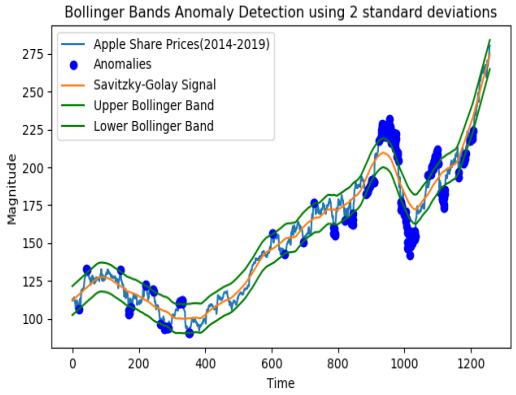
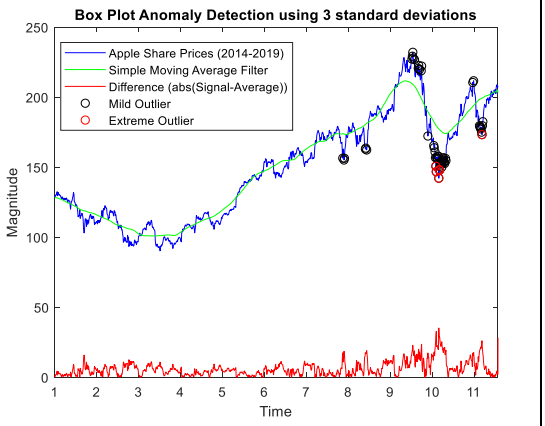
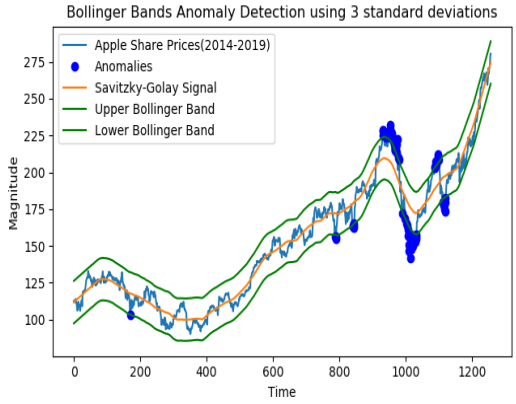
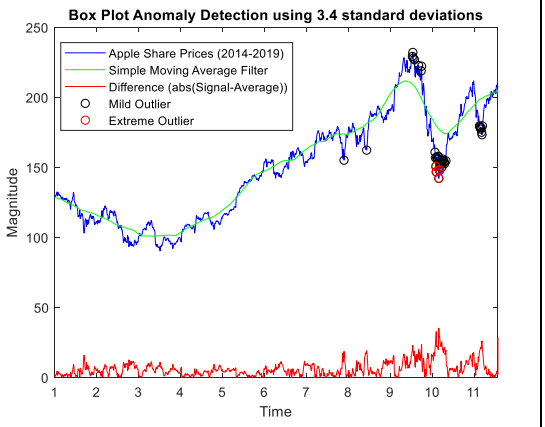
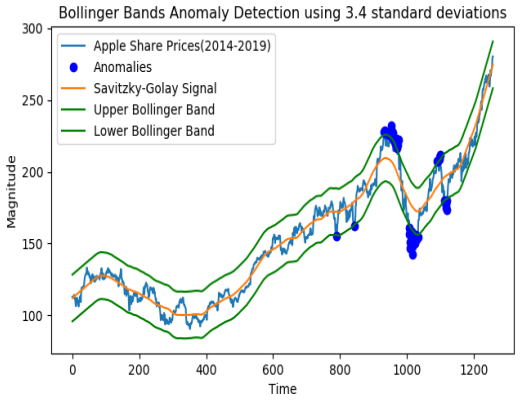
Although both methods utilize different methods of identifying an anomaly, we can see that as we increase the number of standard deviations, they both have identified smaller amounts of data as an anomaly. We also see that areas of highest concentration of anomalies in lower standard deviations is very likely to result in an anomaly detected at the highest standard deviation. This is useful for allowing us to make several interpretations on what an actual anomaly is.

From observation we can see that Box Plot method's anomalies are almost all located underneath the moving average filter, whereas the Bollinger Bands does not appear to have any preference on if the anomalies are located outside the Upper or Lower Bands.

We will carry need to carry out another test to determine if these are the defining differences and similarities between both methods.

7.3.2 Anomaly Detection Test 2

Table 7: Setting L to 100 for Box Plot and the number of bits to 101 for Bollinger Bands

	
	
	
Note: Box plot figures were plotted on MATLAB while Bollinger band figures were plotted on Python, they were plotted along same time scales so, (Boxplot time 2 == Bollinger Band time 200 and so on)	

Box Plot Method Results

We have already detailed how everything in the Box Plot method is represented, look to section – 7.3.1.1 for reminder.

The number of anomalies detected from largest to smallest is:

1. 2 standard deviations
2. 3 standard deviations
3. 3.4 standard deviations

This is expected as we know that:

1. 2 standard deviations will account for about 95% of the data
2. 3 standard deviations will account for about 99.7% of the data
3. 3.4 standard deviations will account for about close to but not equal to 100%

7.3.2.2 Bollinger Bands Results

We have previously detailed how everything in the Bollinger Band method is represented, look to section – 7.3.1.2 for reminder.

The number of anomalies detected from largest to smallest is:

1. 2 standard deviations
2. 3 standard deviations
3. 3.4 standard deviations

Comparing the Results

After concluding in section – 7.3.1.3 that Box Plot and Bollinger Band's anomaly results that the Bollinger Band's had no bias for which side an anomaly was detected and that the Box Plot method had a bias for selecting anomalies to be below the moving average, we can see a change in this conclusion as a result of the results gathered for the 100-point Moving Average filter.

We can see that now the Box Plot method is now detecting similar levels of anomalies on both sides of the moving average filter. This is a result of increasing the sample size for the calculating of the moving average filter, the increase in sample size has allowed the moving average curve to have a greater level of smoothing and thus it will be able to detect better if a signal is continually on the rise. From this we can see that for signals that exhibit characteristics similar to this signal should be tested with various different parameters before making a definite conclusion.

We see that form observation, that the Bollinger Bands results are still very similar to the results gathered in the previous section of – 7.3.1.2. As Bollinger Bands is a tried and tested method in industry, we are happy to conclude that our method of implementation of the Bollinger Bands behaves similar to the what is currently implemented in the financial industry.

The Bollinger Bands method is a recognized method of detecting anomalies in data sets similar to what we investigated (i.e Apple Share Prices). The Box Plot method is an experimental method designed by the project team as our own way of detecting anomalies within a set of data. We are very happy with the performance of the Box Plot method and from utilizing both methods we can accurately identify where an anomaly occurs.

7.4 What can we do with our Identified Anomalies?

Above in section – 7.3 we investigated and identified several anomalies using both methods.

We have taken our detected anomalies and done several internet searches to determine if there is anything notable about our results. We have detailed findings for a few of the anomalies which are listed below.

3rd January 2019[27]

"Share prices have fallen on all the world's major stock markets after poor trading figures from the tech giant Apple and a sharp drop in US factory[28] orders prompted fears of a global economic slowdown in 2019.

Apple shares lost more than 9% of their value on Wall Street after the company admitted in its first revenue warning to investors since 2002 that it was being badly affected by weaker growth, especially in China." – 3rd January 2019

On 3rd January 2019 Apple shares lost more than 9% of their value the day after Tim Cook CEO made a press release admitting to encountering challenges in China, more specifically the region of Greater China.

3rd June 2019[29]

"If you've felt that Apple's release of iOS 13 and iPadOS 13 has been exceptionally troubled, you're not alone. iOS 13 has had more updates—eight in total, if you count the iOS 13.2.1 update for the HomePod—in its first two months than any of its predecessors." – 25th November 2019

Apple hosted their annual WWDC conference and announced the launch of IOS 13 in the coming days much to great public anticipation. The software launched was buggy and should not have been launched at all, it was nowhere near public release.

This caused a lot of investors to lose faith, resulting in a considerably sharp drop in the stock share prices. They have now completely overhauled their Software Testing methods to not run into a similar problem with IOS 14.

7th February 2018[30]

"Someone just posted what experts say is the source code for a core component of the iPhone's operating system on GitHub, which could pave the way for hackers and security researchers to find vulnerabilities in iOS and make iPhone jailbreaks easier to achieve." – 7th February 2018

On 7th February 2018 source code for the iPhone was posted anonymously to the website GitHub. On 8th February the stock value for Apple had already fallen 18%. A leak of this scale will have long-lasting effects on Apple, as the source code allows hackers and "jailbreakers" to bypass all security present on an iPhone and install black market apps directly onto the iPhone among other opportunities.

7.5 Conclusion

The Box Plot method was extremely useful in identifying anomalies although we can see that more work will have to be done on it before we can develop a model that can we are confident in accurately identifying all anomalies.

The Bollinger Bands method was also extremely useful in determining when an anomaly occurs. We see that similarly to our own experimental method we should always carry out a number of tests on the same signal before we can be confident on where we have identified an anomaly.

We have successfully identified and verified several anomalies in Apple's stock prices by correlating our gathered results with news articles present online. We have confirmed that real-world events contribute heavily to the value of a stock price, for example it would be expected that Applw do not have their iPhone source code leaked again but we can see that it took a number of months for them to gain back the investors trust, perhaps they will never truly regain that lost trust. We are confident that the experiments we carried out will be useful for the identification of anomalies in other stock markets.

8 Extended Analysis Methods

Further digital signal processing & trading methods were used to provide further analysis of the test signals. Some of these methods included the use of a Linear Predictor and Stochastic indicator. The idea at the end of the project was to feed all of the results collect into Machine Learning model. This part of the project did not get fully implemented.

This section will delve into the further analysis methods that were used.

8.1 Stochastic Indicator

8.1.1 Theory

As mentioned above in section 2 of the report a stochastic indicator calculates the momentum of a stock. In order to build a stochastic indicator, you must obtain the lowest price of each day the highest price of each day and the closing price of each day. It calculates the highest price point and lowest price point for the specified period of days. The time period chosen will depend on how sensitive you want your indicator to be the usual time periods are 5 days, 9 days or 14 days with a period of 5 days being the most sensitive.

It then calculates where the closing price lies within the range of the highest and lowest price point and plot's it as a percentage between 0% and 100%. This is called the "%K line". A "%D line" is then calculated using the data from the "%K line". The "%D line" is a three-day moving average filter of the "%K line".

The "%K line" and "%D line" are then plotted together this is shown in the graph below. In the graph below the "%K line" is plotted in green and the "%D line "in red. If these two lines intersect above the 80% range, then the price is said to be overbought and is predicted to drop in price as shown below in the graph. This is when the indicator would suggest for you to sell your stocks. If these two lines intersect below the 20% range, then the price is said to be overbought and is predicted to drop in price as shown below in the graph. This is when the indicator would suggest for you to buy stocks.

Figure 26: Stochastic Indicator

8.1.2 Implementation

1. This stochastic indicator was implemented using python. For the implementation five months of a real stock market data was obtained from yahoo finance. The data was extracted from yahoo finance and stored on an excel spread sheet at shown below. The data was then read into python from this excel sheet.

	A	B	C	D
1	Date	High	Low	Close
2	2017-06-01	153.13211	149.43564	152.33211
3	2017-06-02	153.43234	150.89876	152.57234
4	2017-06-03	155.43522	152.98758	153.22312
5	2017-06-04	160.43213	154.12323	154.65321

Figure 27: Example of data used from Yahoo Finance

Example of data used from yahoo finance

2. The next step was to calculate the highest high value and our lowest low value every 14 days. This was done using a rolling max function and a rolling min function. A window size of 14 was set for these rolling functions, the window size determines the amount of variable or "days" the function stores for each iteration. The rolling function works by extracting 14 days' worth of data from an excel file then calculates the max and min points over these 14 days and moves on to the next 14 days and repeated the process.

3. The "%K line" is calculated by plotting the closing price of every day in relation to the lowest and highest price from the previous 14 days. Its plotted as a percentage with 0%

(being the lowest price from the previous 14 days) to 100% (being the highest price from the previous 14 days). The following formula is used.

$$\%K = \frac{(Closing\ price) - (Lowest\ price\ of\ previous\ 5\ days)}{(Highest\ price\ of\ previous\ 5\ days) - (Lowest\ price\ of\ previous\ 5\ days)}$$

4. The %D line is calculated using a rolling mean function with a window size of 3. Each value of the %D equals the mean value of the previous three values of %K. The %D line is essentially a simple moving average filter of %K.

8.1.3 Results

From our stochastic indicator there was 10 points where the "D line" crossed through the "K line" and was either above the 80 percent line (green line) or below the 20 percent line (yellow line) . These are the points where the stochastic indicator predicted there would be a change in the market. These 10 points are labelled from A to J in the graph below.

From the 10 points the indicator predicted 6 out of 10 times correctly. The correct predictions are labelled B, C, E, F, G and H on the diagram. The "K line" and the "D line" intersect and crossover each other above the 80% line for points B, E, F and G on the diagram this means the stocks are said to be overbought (overpriced) and predicted to drop in price which they do. This is clearly seen on the graph below.

The "K line" and the "D line" intersect below the 20% line at point C on the diagram this means the stocks are said to be oversold(underpriced) and are predicted to drop in price, this prediction is correct as shown on graph.

The indicator predicted 4 out of 10 incorrectly. These incorrect predictions are labelled A, D, I and J on the graph below. This shows that the stochastic indicator isn't always reliable but when analyzing and trading stocks there is no indicator that is wright all the time and there is always a risk when trading on the stock market.

Figure 28: Intersection Points of Stochastic Indicator marked on both Closing Price and Stochastic Indicator Plots

8.1.4 Conclusion

This stochastic indicator can be a very useful tool if used correctly. For the Data above the Stochastic indicator predicted the movement of the price correctly 60% of the time. These results could be improved if multiple indicators were used along with the stochastic indicator. Overall the results for the implementation and analysis of the stochastic indicator was a success.

8.2 Linear Prediction

It is a numerical method where the anticipated rates of a discrete-time signal are predicted as a linear function of past models.

In the digital signal processing module (EE401) and in general, linear prediction is often denoted as Linear Predictive Coding (LPC). As a result, linear prediction can be described as a subdivision of filter theory. This is simply because it filters out past data to predict future data. In system analysis, perhaps linear prediction is considered as a subset of numerical modelling or expansion.

A very basic example of Linear Prediction for anticipating data from a plot is the following:

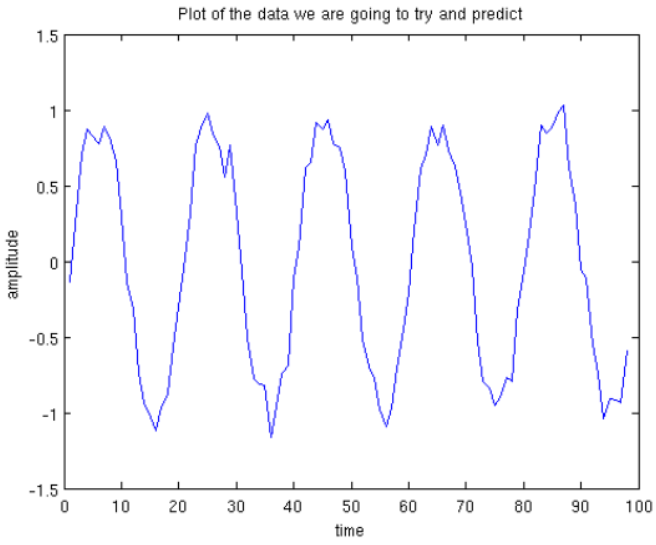


Figure 29: Plot of a Simple Example Market Data

So predicting the future value from this data set, it is easily anticipated that the future value is going to be around x = -0.3. This is simply because the last few points on the graph are on an upwards rise and more than likely it will continue to rise so that is why it is anticipated to be at -0.3. Linear predictive Coding is a good method for determining previous data in the time series to predict the future samples.

The most frequent portrayal of Linear Predictive Coding is:

$$\widehat{x}(n) = \sum_{i=1}^{p} a_i x(n-i)$$

Where $\hat{x}(n)$ is the predicted signal value,

$x(n-i)$ the previous observed values.

And a_i is the predictor coefficients.

The error generated by this estimate is:

$$e(n) = x(n) - \hat{x}(n)$$

Where $x(n)$ is the true signal value.

These mathematical statements are accurate for all types of (one-dimensional) linear prediction. The contrasts are found in the way the predictor coefficients a_i are selected. Further for multi-dimensional signal, the error metric is generally characterized as:

$$e(n) = \|x(n) - \hat{x}(n)\|$$

Where $\|\cdot\|$ is applicable for a selected vector norm. The estimated figures such as the $\hat{x}(n)$ are commonly used within Kalman filters, to predict present and previous signal values.

Predicting the Parameters:

The highly frequent preferred optimization of parameters a_i the root mean square criterion which is also known as autocorrelation principle. During this approach the expected value is decreased. The forecasted value of the squared error is expressed as $E[e^2(n)]$ is produces the following mathematical statement:

$$\sum_{i=1}^{p} a_i R(j-i) = R(j),$$

For $1 \leq j \leq p$ where R is the autocorrelation of signal x_n characterized as:

$$R(i) = E\{x(n)x(n-i)\}$$

Where E is the expected value of the data. The above equations are known as the normal equations or Yule-Walker equations which was heavily discussed in EE401.

8.2.1 Conclusion

Through implementing a Linear Predictor in python, plenty of results were observed. At the end of the day, the linear predictor is confined around the principle of linear regression. It's simple to understand why the linear predictor is not a good form of predicted the next sample. We limited the predictor to the following to collect results:

Could the predictor predict if the next value was going to rise/fall?

The results collected for the three test signals:

- Test_sig_1: 78%
- Test_sig2: 54%
- Test_sig3: 64%

The results collect are all positive expected values. Therefore majority of the time, the predictor predicts the next value accordingly. The reasoning behind the stronger performance in Test_sig_1 is because of periodicity of the signal. The signal used in Test_Sig_1 is a sum of sinusoids with noise added. This is the reason why the predictor predicts much better for this signal.

The other results are gathered from essentially taking the general trend of the graph. If the graph is tending to rise in the long run, then the predictor will make a reserved prediction on the next-value in the upward direction.

8.3 Machine Learning Techniques

Once all the Analysis Methods, it was thought to use a Machine Learning technique known as gradient boosting to determine if the next stock-market value was to rise or fall. The gradient boosting code that was intended to be implemented was XGBoost. XGBoost is a commonly used machine learning technique.

The idea of using XGBoost was:

1. Get all the data regarding the Methods Implemented **(2.3Our** Methods**)** onto the one CSV.
2. There would be around 11 columns of statistical data drawn from the signal.
3. XGBoost would be then ran on the data-set.
4. XGBoost would draw a prediction on if the next value was a rise or decrease based on all the data. The value outputted would be binary, 1 for rise and 0 for fall.

Unfortunately, the Machine Learning side of the project did not fully get implemented due to Time-Constraints. This aspect of the project is mentioned in the Future Work section of this report.

Conclusion

This report aspires to increase the level of awareness of the extensive detail that can be taken when analyzing market data. This is not the first report written regarding analyzing market data, and will not be the last. Market data analysis can be a very problematic and sophisticated method nowadays specifically when it includes implementing techniques like Anomaly detection, Autocorrelation, Linear Predictive Coding, Stochastic Indicator or other techniques mentioned previously. Throughout this report, there were guidelines outlining the theory behind the technique used, how it was implemented (in code), analyzing the results obtained and presenting a unique conclusion to each technique. This is very advantageous as there is a huge demand for effective analysis that will aid managers determine the size of a project or a business that they will invest in. They use this to solve problems and obtain information on competitors.

Future Work

Following on from our conclusion, we have determined that the research carried out and the results obtained in this report has strongly indicated the need to carry out further work in this project. We have identified numerous paths this project can take and given more time or handed to another research group it would be very likely we could implement some extremely beneficial work in the line of predicting an information signal's behaviour.

Originally, we set out to with the intention of developing a model that could predict certain aspects of a signal, for example when the signal is likely to contain anomalies, what the expected future behaviour of the signal in question is. We would like the further develop the research that has already gone into this project by gathering more results on similar signals available across the Stock Market.

Implementing Machine Learning algorithms would provide even more comprehensive results on data that we gathered. We would feed the algorithm results obtained throughout the analysis of the signal and see what conclusion the machine lands on. Software used for machine learning includes XG Boost, a capstone part of this project would entail utilizing this software and then further deciding where to go after the completion of that.

All the existing methods that were implemented could be combined together to create a single prediction model. This prediction model could be tested on real time stock market signals and the results and accuracy analysed.

A deeper look into different stock market indicators or different signal analysing methods could be another aspect of future work. There is a numerous number of indicators and different analysing methods that are used on stock markets today that were not mentioned in this report. Some of these could be implemented and the performance of each method or indicator analysed and compared. The limitations of this project are endless.

References

[1] Project Brief, Designing in September 2019.

[2] https://www.apmex.com/education/price/understanding-the-gold-market

[3]https://www.streetdirectory.com/travel_guide/36954/investment/what_is_currency_markets.html

[4][Online]. Available: https://www.babypips.com/learn/forex/the-big-three.

[5] [2] [Online]. Available: https://www.visualcapitalist.com/12-types-technical-indicators-stocks/.

[6] https://finance.yahoo.com/quote/AAPL?p=AAPL

[7] https://www.sciencedirect.com/topics/engineering/spectral-estimation

[8] https://www.sciencedirect.com/topics/engineering/fourier-transform

[9] https://www.sciencedirect.com/topics/engineering/spectral-estimation

[10] https://pdfs.semanticscholar.org/0a8d/81915d545e0fdc1030d49fd0638b9c21eeb0.pdf

[11] http://www.math.ubc.ca/~mikel/FourierTransformNotes.pdf

[12] https://www.sciencedirect.com/topics/engineering/sinusoidal-signal

[13] http://www.math.ubc.ca/~mikel/FourierTransformNotes.pdf

https://www.sciencedirect.com/topics/engineering/sinusoidal-signal

[14]https://www.researchgate.net/publication/303849241_Fourier_Analysis_for_Stock_Price_Forecasting_Assumption_and_Evidence

[15] https://edoras.sdsu.edu/doc/matlab/toolbox/signal/spectra6.html

[16] https://edoras.sdsu.edu/doc/matlab/toolbox/signal/spectra5.html#6587

https://edoras.sdsu.edu/doc/matlab/toolbox/signal/spectra6.html

[17] http://coolwiki.ipac.caltech.edu/index.php/What_is_a_periodogram%3F

[18] DSP EE401 Notes

[19] https://www.dsprelated.com/freebooks/sasp/Welch_s_Method.html

[20] https://www.gaussianwaves.com/2014/05/yule-walker-estimation/

[21] https://www.investopedia.com/terms/s/sma.asp

[22] https://www.gaussianwaves.com/2010/11/moving-average-filter-ma-filter-2/

[23] http://www.users.waitrose.com/~robinjames/SG/briefDescription.html

[24] http://www.statistics4u.info/fundstat_eng/cc_filter_savgolay.html

[25] **https://docs.scipy.org/doc/scipy-0.16.1/reference/generated/scipy.signal.savgol_filter.html**

[26] https://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm
[27] https://www.theguardian.com/technology/2019/jan/03/apple-shock-profit-warning-sends-european-shares-sliding

[28] https://www.apple.com/ie/newsroom/2019/01/letter-from-tim-cook-to-apple-investors/

[29] https://tidbits.com/2019/11/25/ios-13-bugs-cause-apple-to-overhaul-software-testing/

[30] https://www.vice.com/en_us/article/a34g9j/iphone-source-code-iboot-ios-leak

Appendix

Generating Signals

"""

autocorrelation_sig1.py: This script generates an autocorrelated signal based on the previous rising state. The rising

state will have a low probability of change (20% for the rising_state to inverse).

The output will be in the form of a comma-separated CSV file. The behaviour of the data is very similar to Market

Data and can be seen when you use a simple plotting function.

"""

from numpy import random, arange

Defining the Parameters

length, step = 10, 0.01

N, F_N = [], []

start_trade_value = 200

rising_state, rising_states = True, []

```
print("Generating Autocorrelated Data...\n")

print("Length: ", length, " Step-Size: ", step)

print("Initial Trade Value: ", start_trade_value, "\n")


# Declaring Trading signal

F_N.append(start_trade_value)

for i in arange(0,length+step,step):

    N.append(round(i, 2))


    # Defining the Rising/Falling value

    F_N_Deviation_Low = random.choice([random.uniform(-20, 0), random.uniform(-300, -100)], p = [0.95, 0.05])

    F_N_Deviation_High = random.choice([random.uniform(0, 20), random.uniform(100, 300)], p = [0.95, 0.05])


    rising_state = random.choice([rising_state, not rising_state], p = [0.80, 0.20])

    rising_states.append(rising_state)

    print("Trade is Rising: ", rising_state)


    if rising_state == True:

        F_N.append(F_N[int(i/step)]+F_N_Deviation_High)

    else:

        F_N.append(F_N[int(i / step)] + F_N_Deviation_Low)
```

```
# Deleting last element to ensure N & F_N and same length
del F_N[-1]


# Appending the results to a CSV
f = open("trading_data_sig1.csv", "w")
for i in range(len(N)):
    f.write("{},{}\n".format(N[i], F_N[i]))
f.close()
```

Spectral Analysis

```
%% loading  our signals into matlab
filename = 'test_sig_2.csv';
X = csvread(filename);
% figure;
col1 = X(:, 1);
col2 = X(:, 2);
% plot( col2);


%% computing the FFT of our signal


%  FS=10000;      % sampling frequency
%  Ts=1/FS;      %sampling period
```

```
%  dt=0:Ts:5-Ts; %duration of signal

% % nfft = length(col2);%length of the time domain signal

% % nfft2=2^nextpow2(nfft);%resoution of the signal

% % ff=fft(col2,nfft2);%repersenting our magnitude and phase

% % fff=ff(1:nfft2/2);%to remove the symmetrical FFT

% % figure;

% % T=(abs(fff));

% % plot(T) %the absolute part

% % title('Fast Fourier Transform Magnitude  ')

% % xlabel('Frequency (Hz)')

% % ylabel('Amplitude')

% % figure;

% % plot(angle(fff))

% % title('Fourier Transform Phase ')

% % xlabel('Frequency (Hz)')

% % ylabel('Amplitude')

%

 %% computing the spectrogram

% figure;

% spectrogram(col1,500,[],[],FS,'yaxis');

 %% comparing different window using the periodogram

% figure;
```

```
% [pxx , p ]=periodogram(col2,hamming(length(col2)),100);

% plot(p,10*log10(pxx))

% hold on

% [pxc, w]=periodogram(col2,rectwin(length(col2)),100);

% plot(w,10*log10(pxc),'-.')

% hold on

% [pxb, b]=periodogram(col2,blackman(length(col2)),100);

% plot(b,10*log10(pxb),'-.')

% hold on

% [pxm, m]=periodogram(col2,bartlett(length(col2)),100);

% plot(m,10*log10(pxm),'-.')

% hold on

% [pxm, m]=periodogram(col2,hanning(length(col2)),100);

% plot(m,10*log10(pxm),'-.')

% grid on

% xlabel('Normalized Frequency (\times\pi rad/sample)')

% ylabel('Power/frequeny (dB/(rad/sample))')

% title('Periodogram ')

% legend('Periodogram with hamming window','Periodogram with rectangular window
','Periodogram with blackman window'...

% ,'Periodogram with bartlett window','Periodogram with hanning window');
```

```
%% comparing different window using the pwelch method

%  len = 500 ;

%  w = kaiser(len,2.5);

%  b = bartlett(len);

% pwelch(col2,w,[],[],100);

% hold on

% h = rectwin(len);

% pwelch(col2,h,[],[],100);

% [pwe ,f]=pwelch(col2,[],[],[],100);

% plot(f,10*log10(pwe))

% hold on

% [pwc, f]= pwelch(col2,w,[],[],100);

% plot(f,10*log10(pwc),'-.')

% hold on

% [rec , f] = pwelch(col2,h,[],[],100);

% plot(f,10*log10(rec),'-.')

% hold on

% [bef , f] = pwelch(col2,b,[],[],100);

% plot(f,10*log10(bef),'-.')

% grid on

% xlabel('Frequency (KHz)')

% ylabel('Power/frequeny (dB/(rad/sample))')
```

```
% title('Welch's power spectral density')

% legend(' hamming window',' kaiser window ','rectangular window '...

% ,'Periodogram with bartlett window');

%% comparing the periodogram to the

%  figure;

% [pxb, b]= pwelch(col2,[],[],[],FS);

% [pxm, m]= periodogram(col2,[],[],FS);

% plot(m,10*log10(pxm),'-.')

% hold on

% plot(b,10*log10(pxb),'-.')

% grid on

% xlabel('Normalized Frequency (\times\pi rad/sample)')

% ylabel('Power/frequeny (dB/(rad/sample))')

% title('Periodogram /  Welch's method  ')

% legend('Periodogram Welch's method ','Welch's method ');

%% computing the aryule the computing method

figure

pwelch(col2);

hold on ;

N= [2,3,4,5,6,7,8,9,10,11,12,];


for i=1:11
```

```
[d,p] = aryule(col2,N(i));

[H, w]=freqz(sqrt(p),d);

hp = plot (w/pi,20*log10(20*abs(H)/(20*pi)));

xlabel('Normalized Frequency (\times\pi rad/sample)')

ylabel('Power/frequeny (dB/(rad/sample))')

title('yule walker method /  Welch's method  ')

legend('yule walker method ','Welch's method ');

end
```

Spectral Analysis (python)

```
"""
Fast Fourier Transform (FFT) to compute the spectral density of a signal. The
spectrum represents the energy associated
to frequencies (encoding periodic fluctuations in a signal). It is obtained
with a Fourier transform, which is a
frequency representation of a time-dependent signal. A signal can be
transformed back and forth from one representation
to the other with no information loss.
"""

# Necessary Libraries
import numpy as np
from scipy import fftpack, signal
import pandas as pd
import matplotlib.pyplot as plt

# Reading the generated data signal in CSV format
df = pd.read_csv('test_sig_1.csv', header=0)

# Setting Number of Samples and Period
N = len(df)
T = (df.iloc[-1][0]-df.iloc[0][0])/(N-1)
print("Numbers of Samples (N):", N, " |  Sampling Interval (T):", T)

# Obtaining the Values as a Numpy Array from the Dataframe
x = df['N'].values
y = df['F(N)'].values
DC_offset = y.mean()
print("Current DC Offset:", DC_offset)

# Removing the DC offset from the signal
y = y - DC_offset
```

```
#sos = signal.butter(4, 0.0001, 'high', output='sos')
#y = signal.sosfilt(sos, y)

# Computing Fast Fourier Transform of Signal N, F(N)
xf = fftpack.fftfreq(N) / T
yf = fftpack.fft(y) * (2.0/N)

# Setting the plotting parameters
xf_plot, yf_magplot, yf_phaseplot = np.linspace(0.0, 1.0/(2*T), num=N//2),
np.abs(yf[:N//2]), np.angle(yf[:N//2])

# Plotting Time Domain & Frequency Domain (Magnitude and Phase)
fig, (time, freq, phase) = plt.subplots(3, 1)
time.plot(x, y), freq.plot(xf_plot, yf_magplot), phase.plot(xf_plot,
yf_phaseplot)
time.set_xlabel('Time Interval (N)'), time.set_ylabel('Stock Market Value')
freq.set_xlabel('Frequency in Hertz (Hz)'), freq.set_ylabel('Spectrum
Magnitude')
phase.set_xlabel('Frequency in Hertz (Hz)'), phase.set_ylabel('Phase in
Radians (radians)')
plt.show()

# Periodogram with different window methods
f_s = 1/0.01
windows = ['hamming', 'boxcar', 'blackman', 'bartlett', 'hanning']
for w in windows:
    f, P = signal.periodogram(x, f_s, window=w, scaling='spectrum')
    plt.semilogy(f, P, label=w+' function')
plt.legend()
plt.show()

# Reconstructing Sinusoidal Signal
y_recon = fftpack.ifft(yf) / (2.0/N)

# Comparing the original VS reconstructed
plt.plot(x, y_recon, label='Reconstructed Signal'), plt.plot(x, y,
label='Original Signal')
plt.legend()
plt.show()
```

Anomaly Detection

%Moving Average Filter method 1+2

---

data = csvread('Z:/EE401 Group Project/Discrete_Autocorrelated_1.csv'); %input frequency

```
%split into separate arrays

time = data(:,1);

magnitude = data(:,2);


%creating moving average filter

window_size = 50;

t = 0;

moving_average = zeros(size(magnitude)); %gets size of magnitude array and fills with zeroes


%filling correct elements for each row of respective of time

%need a for loop to fill in elements


%Fill in with window_size ending on current time

for i = 1:length(magnitude)

   total = 0;              %set total

   if i >= window_size     %ensure greater index val is not < 0

      for j = (i-window_size):i

         if j >= 1

            total = total + magnitude(j);

         end

      end

      calculation = (total)/window_size;
```

```
    end
    if i < window_size
        for k = 1:i   %this gives the average of the data contained before the first full window
            total = total + magnitude(k); % correct total val
        end
        calculation = (total)/i;    %total/(current window size)
    end
    moving_average(i) = calculation;    %sets the row val to calculation
end

figure(1)
plot(time,magnitude,'-b') %normal plot
hold on
plot(time,moving_average,'-g') %moving average filter

%Get all figure description
title('Plot of input frequency and moving average filter (lagging method)')
xlabel('Time'), ylabel('Magnitude')
legend('Input Signal','Moving Average Filter')

moving_average2 = zeros(size(magnitude)); %gets size of magnitude array and fills with zeroes
```

```
%filling correct elements for each row of respective of time
%need a for loop to fill in elements


%Moving_average filter 2 (Centering around the current time)
%Fill in with window_size ending on current time
for i = 1:length(magnitude)
   total = 0;              %set total
   if i >= window_size/2     %ensure greater index val is not < 0
      for j = (i-window_size/2):(i+window_size/2)
         if j >= 1
            if j < (length(magnitude)-window_size/2)
            total = total + magnitude(j);
            end
         end
      end
      calculation = (total)/window_size;
   end
   if i < window_size/2
      for k = 1:i   %this gives the average of the data contained before the first full window
         total = total + magnitude(k); % correct total val
      end
      calculation = (total)/i;    %total/(current window size)
```

```
    end

    moving_average2(i) = calculation;    %sets the row val to calculation

end


figure(2)

plot(time,magnitude,'-b') %normal plot

hold on

plot(time,moving_average,'-g') %moving average filter

hold on

plot(time,moving_average2, '-r') %second moving average filter with outputs average centered around current time


%Get all figure description

title('Plot of input frequency and moving average filter (lag+lead method)')

xlabel('Time'), ylabel('Magnitude')

legend('Input Signal','Moving Average Filter','Lag+Lead Avg Filter')



---------------------------

%Links to help

%https://www.gaussianwaves.com/2010/11/moving-average-filter-ma-filter-2/

%https://www.youtube.com/watch?v=7Rz_ITRIADg
```

Moving average filter method 3

```
data = csvread('Z:/EE401 Group Project/Discrete_Autocorrelated_1.csv'); %input frequency



time = data(:,1);

magnitude = data(:,2);


%creating moving average filter

window_size = 50;

t = 0;

moving_average3 = zeros(size(magnitude)); %gets size of magnitude array and fills with zeroes


%filling correct elements for each row of respective of time

%need a for loop to fill in elements


%This method didn't end up being used but still included in appendix as it furthers understanding of moving average filter

%This filter
```

```
%Fill in with window_size ending on current time
for i = 1:length(magnitude)
   total = 0;              %set total
   if i >= window_size     %ensure greater index val is not < 0
      for j = i:i+window_size
         if j <= length(magnitude)
            total = total + magnitude(j);
         end
         if j > length(magnitude)
            total = total + magnitude((length(magnitude)));
         end
      end
      calculation = (total)/window_size;
   end
   if i < window_size
      for k = i:i+window_size   %this gives the average of the data contained before the first full window
         total = total + magnitude(k); % correct total val
      end
      calculation = (total)/window_size;    %total/(current window size)
   end
   moving_average3(i) = calculation;    %sets the row val to calculation
```

```
end


figure(1)
plot(time,magnitude,'-b') %normal plot
hold on
plot(time,moving_average,'-r')
hold on
plot(time,moving_average2, '-k')
hold on
plot(time,moving_average3,'-g') %moving average filter


%Get all figure description
title('Plot of input frequency and moving average filter (lagging method)')
xlabel('Time'), ylabel('Magnitude')
legend('Input Signal','Moving Average Filter','lag+lead','Lead')



---------------------
Anomaly detection
---------------------


% Anomaly detection
```

```
%call moving average filter

run C:/Users/16334046/Downloads/Moving_average_filter.mlx %set this to where your moving average fiilter is created


% start identifying mean at each point

% all things created in last file


ts_moving_average = zeros(1,length(magnitude));

%get moving average time shifted

for j = 1:length(moving_average)-window_size

    ts_moving_average(j)=moving_average(j+window_size/2);

end


%create difference matrix

difference_matrix = abs(ts_moving_average - magnitude); %essentially the error between average and original signal


% move everything to the DC line

% for ease of analysis if the signal went under DC line

dc_magnitude = magnitude+abs(min_mag);

dc_moving_average = ts_moving_average+abs(min_mag);
```

```
figure(1)

plot(time,dc_magnitude,'b')

hold on

plot(time,dc_moving_average,'g')

hold on

plot(time, difference_matrix,'r')


%determining outliers

%must determine quartile ranges for outliers

%mean

middle_point = transpose(zeros(1,length(magnitude)-window_size));

for j = 1:length(middle_point)

    middle_point(j)=difference_matrix(j);

end

sorted_middle_point = sort(middle_point);


median_matrix = length(sorted_middle_point)/2;

median_lower_index = floor(median_matrix);

median_upper_index = median_lower_index + 1;


median =
(sorted_middle_point(median_lower_index)+sorted_middle_point(median_upper_index))/2;
```

```
%lower quartile index, upper quartile index
lower_quartile_index = floor((length(sorted_middle_point)/100)*25);
upper_quartile_index = floor((length(sorted_middle_point)/100)*75);

%lower + upper quartiles
lower_quartile = sorted_middle_point(lower_quartile_index);
upper_quartile = sorted_middle_point(upper_quartile_index);

%interquartile range
interquartile_range = upper_quartile - lower_quartile;

%lower inner fence % standard deviations are wrote here!!
lower_inner_fence = lower_quartile - 1.5*(interquartile_range);
%upper inner fence
upper_inner_fence = upper_quartile + 1.5*(interquartile_range);

%lower outer fence
lower_outer_fence = lower_quartile - 3*(interquartile_range);
%upper outer fence
upper_outer_fence = upper_quartile + 3*(interquartile_range);
```

```
mild_outlier_matrix = transpose(zeros(1,length(middle_point)));

extreme_outlier_matrix = transpose(zeros(1,length(middle_point)));


%now check data that lies outside inner fences

for i = 1:length(middle_point)

   if middle_point(i) < lower_inner_fence || middle_point(i) > upper_inner_fence

         mild_outlier_matrix(i) = middle_point(i);

   end

end


%get index values for mild outliers

mild_outlier_index_list = find(mild_outlier_matrix)


%now check data that lies outside outer fences

for i = 1:length(middle_point)

   if middle_point(i) < lower_outer_fence || middle_point(i) > upper_outer_fence

         extreme_outlier_matrix(i) = middle_point(i);

   end

end


%get index values for extreme outliers

extreme_outlier_index_list = find(extreme_outlier_matrix);
```

```
%now plot graph with the outliers

figure(2)

plot(time,magnitude,'b')

hold on

plot(time,ts_moving_average,'g')

hold on

%work out how to plot certain values

plot(time(mild_outlier_index_list),magnitude(mild_outlier_index_list),'ok')

hold on

plot(time(extreme_outlier_index_list),magnitude(extreme_outlier_index_list),'or')


%describe the plot

xlim_lower = window_size/100

xlim_upper = (length(magnitude)-window_size-1)/100

xlim([xlim_lower xlim_upper])

title('Signal vs Moving Average Filter w/ outliers marked')

xlabel('Time'), ylabel('Magnitude')

legend('Input Signal','Moving Average Filter','Mild Outlier','Extreme Outlier','Location','best')
```

Bollinger Bands

```
"""
```

Fast Fourier Transform (FFT) to compute the spectral density of a signal. The spectrum represents the energy associated

to frequencies (encoding periodic fluctuations in a signal). It is obtained with a Fourier transform, which is a

frequency representation of a time-dependent signal. A signal can be transformed back and forth from one representation

to the other with no information loss.

"""

```
# Necessary Libraries
import numpy as np
from scipy import fftpack, signal
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import savgol_filter

# Reading the generated data signal in CSV format
df = pd.read_csv('test_sig_3.csv', header=0)

# Setting Number of Samples and Period
N = len(df)
T = (df.iloc[-1][0]-df.iloc[0][0])/(N-1)
print("Numbers of Samples (N):", N, " |  Sampling Interval (T):", T)
```

```
# Obtaining the Values as a Numpy Array from the Dataframe

x = df['N'].values

y = df['F(N)'].values

print("Current DC Offset:", y.mean())


# Computing Fast Fourier Transform of Signal N, F(N)

xf = fftpack.fftfreq(N) / T

yf = fftpack.fft(y) * (2.0/N)


# Setting the plotting parameters

xf_plot, yf_magplot, yf_phaseplot =  np.linspace(0.0, 1.0/(2*T), num=N//2), np.abs(yf[:N//2]), np.angle(yf[:N//2])


# Reconstructing Sinusoidal Signal

y_recon = fftpack.ifft(yf) / (2.0/N)


# Run the given data through the Savitzky-Golay Filter

filtered = savgol_filter(y, 101, 1)


# Create an array to place the different anomalies outside the bands

anom = []
```

```
array1 = []

# Outlier array to get the specific points where the anomalies occur

outlier_array = []


# Run through the signals to find the error in difference between SGF and input signal

for i in range(len(filtered)):

    array1.append(abs(filtered[i] - y[i]))


# Find the standard deviation of the previously calculated error

std_1 = np.std(array1)

# Get the two bands

ball1,ball2 = filtered+(3.4*std_1),filtered-(3.4*std_1)


# Run through the filtered and add elements into the anomaly array

for i in range(len(filtered)):

    if y[i] > ball1[i]:

        anom.append(y[i])

        outlier_array.append(i)

    elif y[i] < ball2[i]:

        anom.append(y[i])

        outlier_array.append(i)

    else:
```

```
    anom.append(float('nan'))


print (outlier_array)

plt.plot(y, label='Apple Share Prices(2014-2019)')

plt.plot(anom,'bo', label='Anomalies')

plt.plot(filtered, label='Savitzky-Golay Signal')

plt.plot(ball1, label='Upper Bollinger Band',color='green')

plt.plot(ball2, label='Lower Bollinger Band' ,color='green')

plt.title('Bollinger Bands Anomaly Detection using 3.4 standard deviations')

plt.xlabel('Time')

plt.ylabel('Magnitude')

plt.legend()

plt.show()
```

Stochastic Indicator

```
"""
A stochastic indicator used on trading stocks
"""

# Modules used
import pandas as pd
import matplotlib.pyplot as plt

#downloads data into  a DataFrame
df = pd.read_csv('GOLD.csv', header=0)

#Create the "Lowest_14" column in the DataFrame
#" Lowest_14" is set to the lowest value for every 14 values
df[Lowest_14] = df['Low'].rolling(window=14).min()

#Create the " Highest_14" column in  the DataFrame
```

```
#" Highest_14" is set the highest value for every 14 values
df[Highest_14] = df['High'].rolling(window=14).max()

#Create the "%K" column in the DataFrame
df['%K'] = 100*((df['Close'] - df[Lowest_14]) / (df[Highest_14] - df[Lowest_14]) )

#Create the "%D" column in the DataFrame
df['%D'] = df['%K'].rolling(window=3).mean()

# Plots the close price on the first subplot
plt.subplot(211)
plt.plot(df['Close'])
# Add title and axis names
plt.title('Closing Price')
plt.grid()

# Plots %K & %D on the second subplot
plt.subplot(212)
plt.plot(df['%K'], label='%K')
plt.plot(df['%D'], label='%D')
# Add title and axis names
plt.title('Stochastic Oscillator')
plt.legend()
plt.grid()
plt.show()
```

Correlation

"""

signal_analysis_convolution.py - This script calculates the convolution of two trading signals

"""

Necessary Libraries

import numpy as np

from scipy import fftpack, signal

```
import pandas as pd
import matplotlib.pyplot as plt
"""
sig1_df = pd.read_csv('trading_data_sig2_A.csv', header=0)
sig2_df = pd.read_csv('trading_data_sig2_B.csv', header=0)

sig1_y = sig1_df['F(N)']
sig2_y = sig2_df['F(N)']

sig_conv = signal.convolve(sig1_y, sig2_y)
sig_corr = signal.correlate(sig1_y, sig2_y)

fig, axs= plt.subplots(2,2)
(signal1, signal2), (conv, corr) =axs
signal1.plot(sig1_y)
signal2.plot(sig2_y)
conv.plot(sig_conv)
corr.plot(sig_corr)
plt.show()
"""
df = pd.read_csv('Discrete_Autocorrelated_1.csv', header=0)
y = df['F(N)'].values
```

```
df = pd.read_csv('LPC_Results.csv', header=0)

actual, predicted = df['Actual'].values, df['Predicted'].values


predicted_temp = []

for i in range(900):

    predicted_temp.append(0)


predicted_temp = predicted_temp + predicted.tolist()

y = np.append(y, actual)


#predicted = np.append(predicted_temp, predicted)

#y_predicted = np.append(, predicted)

plt.plot(y, label='Actual Signal')

plt.plot(predicted_temp, label='Predicted Signal')

plt.legend()

plt.show()
```

Linear Predictor

```
import numpy as np

from scipy import fftpack, signal

import pandas as pd

import matplotlib.pyplot as plt
```

```
def lpc(y, m):
    "Return m linear predictive coefficients for sequence y using Levinson-Durbin prediction algorithm"
    # step 1: compute autoregression coefficients R_0, ..., R_m
    R = [y.dot(y)]
    if R[0] == 0:
        return [1] + [0] * (m-2) + [-1]
    else:
        for i in range(1, m + 1):
            r = y[i:].dot(y[:-i])
            R.append(r)
        R = np.array(R)
    # step 2:
        A = np.array([1, -R[1] / R[0]])
        E = R[0] + R[1] * A[1]
        for k in range(1, m):
            if (E == 0):
                E = 10e-17
            alpha = - A[:k+1].dot(R[k+1:0:-1]) / E
            A = np.hstack([A,0])
            A = A + alpha * A[::-1]
```

```
        E *= (1 - alpha**2)

    return A

# Importing Necessary Libraries

from pandas import read_csv

from signal_analysis_functions import lpc

import numpy as np

import matplotlib.pyplot as plt


# Importing the Trading Signal and obtaining the numpy array

df = read_csv('Discrete_SW_2.csv', header=0)

x, y = df['N'].values, df['F(N)'].values


# Importing the future values of Discrete_Autocorrelated_1: 9.0-10.0

df_extended = read_csv('appended_sw.csv', header=0)

y_future = df_extended['F(N)'].values


# Plotting the Original Signal

plt.plot(y)

plt.show()


predicted_values = []

for i in range(len(y_future)):
```

```
    # Obtaining the LPC coefficients for m samples
    m = 5
    lpc_coefficients = -1*lpc(y, m)

    # Cleaning up LPC coefficients array
    lpc_coefficients = np.delete(lpc_coefficients, 0)
    lpc_coefficients = np.flip(lpc_coefficients)
    lpc_coefficients = np.round(lpc_coefficients, 8)
    print("The LPC coefficients: ", lpc_coefficients)

    # Taking last m samples and predicting next value using co-efficients
    last_values = y[-m:]
    print("These last 5 values: ", last_values)
    next_value = np.sum(np.multiply(lpc_coefficients, last_values))
    predicted_values.append(next_value)
    print("Predicted Value", next_value)
    print("")

    y = np.append(y, y_future[i])

import pandas as pd
pd.DataFrame(predicted_values).to_csv("predicted.csv", index=False)
```