Hybrid Neural Network Analysis of Short-term Financial Shares Trading

EMIL PETKOV TURKEDJIEV

PhD

2017

Hybrid Neural Network Analysis of Short-term Financial Shares Trading

EMIL PETKOV TURKEDJIEV

A thesis submitted in partial fulfilment of the requirements of the

University of Northumbria at Newcastle

for the degree of

Doctor of Philosophy

Research undertaken in the

Faculty of Engineering and Environment

April 2017

Thesis

PhD Student: Emil Petkov Turkedjiev

Mode: Part-time

Principal supervisor: Professor Krishna Busawon

Supervisor: Professor Sean Danaher

Supervisor: Dr. Wohida Aggoune-Bouras

Title: Hybrid Neural Network Analysis of Short-term Financial Shares (Day Trading)

Start Date: 21/02/2011

Date: 30/04/2017

Abstract

Recent advances in machine intelligence, particularly Artificial Neural Networks (ANNs) and Particle Swarm Optimisation (PSO), have introduced conceptually advanced technologies that can be utilised for financial market share trading analysis.

The primary goal of the present research is to model short-term daily trading in Financial Times Stock Exchange 100 Index (FTSE 100) shares to make forecasts with certain levels of confidence and associated risk. The hypothesis to be tested is that financial shares time series contain significant non-linearity and that ANN, either separately or in conjunction with PSO, could be utilised effectively. Validation of the proposed model shows that non-linear models are likely to be better choices than traditional linear regression for short-term trading. Some periodicity and trend lines were apparent in short- and long-term trading. Experiments showed that a model using an ANN with the Discrete Fourier Transform (DFT) and Discrete Wavelet Transform (DWT) model features performed significantly better than analysis in the time domain.

Mathematical analysis of the PSO algorithm from a systemic point of view along with stability analysis was performed to determine the choice of parameters, and a possible proportional, integral and derivative (PID) algorithm extension was recommended. The proposed extension was found to perform better than traditional PSO. Furthermore, a chaotic local search operator and exponentially varying inertia weight factor algorithm considering constraints were proposed that gave better ability to converge to a high quality solution without oscillations. A hybrid example combining an ANN with the PSO forecasting regression model significantly outperformed the original ANN and PSO approaches in accuracy and computational complexity.

The evaluation of statistical confidence for the models gave good results, which is encouraging for further experimentation considering model cross-validation for generalisation to show how accurately the predictive models perform in practice.

Contents

Acknowledgments

I would like to thank my principal supervisor Professor Krishna Busawon for his supervision, ideas and believing in me.

Declaration

I declare that the work contained in this thesis has not been submitted for any other award and that it is all my own work. I also confirm that this work fully acknowledges opinions, ideas and contributions from the work of others.

Any ethical clearance required for the research presented in this thesis has been approved. Approval has been sought and granted by the School Ethics Committee on 11th November 2011.

I declare that the word count of this Thesis is 44,682 words

Name: Emil Petkov Turkedjiev

Signature:

Date: 30/04/2017

Chapter 1. Introduction

1.1. Overview

This chapter introduces material that is fundamental to understanding the nature of the financial shares and machine learning approach and processes.

A stock market is a public market for securities where the organized issuance and trading of company stocks take place either through exchange or over the counter in physical or electronic forms. It is nowadays commonly known that huge amounts of capital are traded through stock markets across the world (Al Wadia, et al., 2011). However, the accurate prediction of stock market movements is highly challenging as well as being an important issue for investors, and it has received much attention from practitioners and experts in financial time series research (Chandar, et al., 2015).

This chapter covers the following:

- Financial shares and the stock market
- Aims and objectives
- Theoretical framework
- Methodology including data analysis
- Contribution to the research field

1.2. Introduction

Ordinary financial shares are issued by companies to raise share capital and entitle their holders to receive yearly profits called dividends (Lexicon.ft.com, 2017, Staff, 2017). They offer investors potential rewards for the risks they take. Furthermore, ordinary shareholders have voting rights in proportion to the number of shares they own. Ordinary shares are the subject of this research.

Major newspapers, television reports and various other information sources such as those on the Internet presently show data from the Financial Times Stock Exchange (FTSE) 100

in the UK, the Standard and Poor (S&P) 500 and Dow Jones 30 in the US and the Nikkei Dow in Japan. Such information is quoted frequently on national television and in other news reports and is now easily downloaded from web sources. Shares are traded daily from Monday to Friday (252 days per year) from 8:00am – 17:00pm on stock markets such as those in London, New York, and Tokyo. Most share and option pricing models are founded on one simple model for asset price movements involving parameters derived from historical or market data.

Figure 1-1 (https://uk.finance.yahoo.com/quote/BARC.L?p=BARC.L) gives a snapshot of the Barclays PLC share data on Friday 16th November 2012 and includes the intraday graph of the share price.

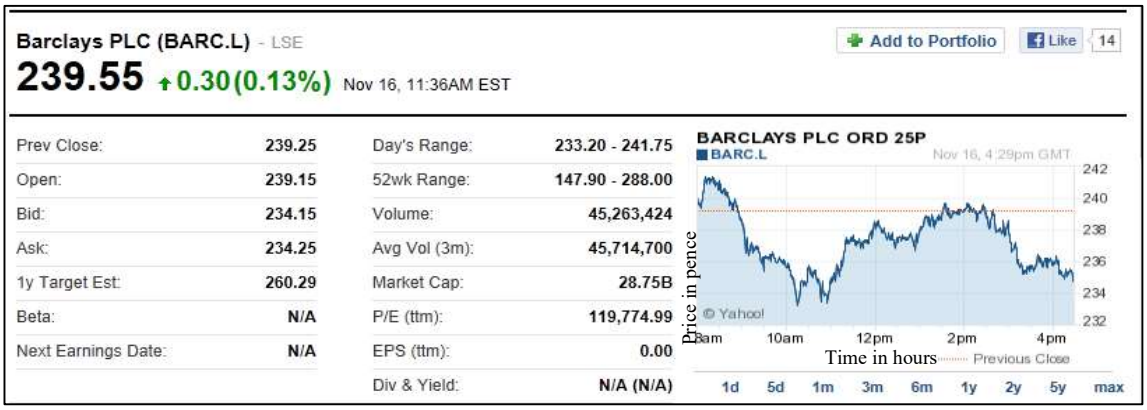


FIGURE 1-1 BARC.L 2012-11-16

An alternative short-term trading timescale period of three months from August to November 2012 is given in Figure 1-2

FIGURE 1-2 BARC.L AUGUST-NOVEMBER 2012

Figure 1-3 shows annual data from November 2011 to November 2012:

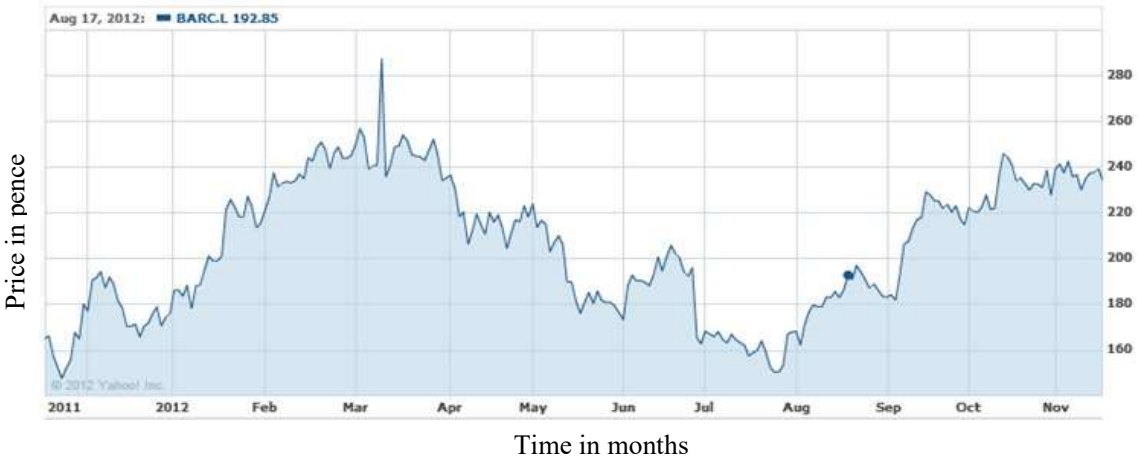


FIGURE 1-3 BARC.L FROM NOVEMBER 2011 TO NOVEMBER 2012

The stock share index is a time series representation of share prices for a certain period, such as a day, month or year, and contains a time dimension. Prediction applications with one or more time-dependent attributes are called time series problems. Time series analysis usually involves predicting numerical outcomes, such as the future price of an individual stock or the closing price of a share (Siraj, 2011). Most research into time-dependent data analysis has been statistical and limited to predicting the future value of a single variable. However, both statistical and non-statistical data mining techniques can be used for time series analysis. Typical data mining approaches use traditional linear regression and, more recently, neural networks.

The first step toward time series modelling and prediction solutions is setting up the time series prediction problem. Essentially this uses data from the recent past to construct one or more indicators which can be used as inputs to a profitable trading system. An example of time series forecasting in econometrics is predicting the opening price of a stock based on its past performance. In such predictions, neural networks are used to construct a new class of indicators which have predictive power. Most technical indicators, such as moving averages (MAs), relative strength indicators (RSIs), and directional indicators (+DI, —DL ADX, ADXR), are elements of parametric models. They are included in formulae that have been developed to measure some effect which is believed to be present in the data. Various

parameters, such as a smoothing period, are adjusted to maximize profit when incorporated into a larger system. The output of these indicators is typically used in conjunction with other indicators as one component of a trading strategy. A time series model generally reflects the fact that observations close together in time will be more closely related than observations further apart. In addition, time series models often make use of the natural one-way ordering of time, so that values for a given period will be expressed as deriving in some way from past values. Methods for time series analysis may be divided into two classes: frequency-domain methods and time-domain methods. The former include auto-correlation, cross-correlation analysis, spectral analysis, and recently, wavelet analysis. Time-domain auto-correlation and cross-correlation analyses are completed in the time domain as well (Jani, 2012).

Technical analysis and fundamental analysis are the two main approaches to the analysis of financial markets. Technical analysis looks at the price movements of a security and uses this data to predict its future price movements. Fundamental analysis, on the other hand, looks at economic factors, known as fundamentals, examining earnings, dividends, new products, research conducted and the like.

Investment theory usually states that it is impossible to "beat the market", because stock market efficiency causes existing share prices always to incorporate and reflect all relevant information. According to the Efficient Market Hypothesis (EMH), or "no arbitrage" (Harper, 2012), stocks always trade at their fair values on stock exchanges, making it impossible for investors to either purchase undervalued stocks or sell stocks for inflated prices.

As such, it should be impossible to outperform the overall market through expert stock selection or market timing, so that the only way an investor can possibly obtain higher returns is by purchasing riskier investments. In fact the EMH contradicts the basic tenets of

technical analysis by stating that past prices cannot be used to profitably predict future prices. Thus, it holds that technical analysis cannot be effective.

Traditional linear methods fail to predict breaks in trends, stock market collapses and recessions, which brings into question the core assumption that financial markets follow a purely random walk and the EMH. However, due to the non-linear, non-stationary, highly noisy and chaotic characteristics of the stock market, forecasting is always considered to be a very difficult and challenging process (Atsalakis, 2009). Different kinds of technical, fundamental and statistical measures have been proposed and used in financial forecasting, such as the simple moving average, linear regression, the Support Vector Machine (SVM) (Huang, et al., 2011) and Back Propagation Neural Network (BPNN) (Devadoss, et al., 2013). The current belief is that the market's behaviour is a result of many non-linear processes and interactions. Hence, slight differences in initial conditions can cause the market to evolve in completely different ways (Baestaens, et al., 1994). This research advances these approaches by developing a methodology and deep learning techniques with hybrid ANN and PSO models in both time and digital domains . Despite the complexity and uncertainty of the context, stock market traders continue to have an intuitive feeling that there are recurrent return and volatility patterns that can be isolated and used as the basis for trading and investments. These patterns may be observed both with individual shares and on a cross-sectional basis. The new presumption of chaos encourages investigations that might identify these patterns using improved and mainly stochastic and non-linear methods to analyse complex and dynamic economic and financial data. Common features of the analytic techniques required in this domain are pattern recognition and generalization capabilities.

For regular stock exchange dealers, 'short-term' trading means buying in the morning and selling at a profit the same day and, if possible, to do that five days a week. More moderate trading uses a spread-betting approach. It is not often that a share price moves up

significantly within one day. It is more realistic to set a time limit for short-term trading which is more likely to be successful under normal market conditions, such as two-three weeks.

1.3. Aims and Objectives

The hypothesis to be tested here is an educated guess about an outcome of some event; for example, that it is possible to predict in the short term (a day's trading) the share price with a certain level of confidence. Typically, the outcome is stated in the form of a null hypothesis which takes a negative point of view in asserting that any relationship found is due purely to chance; for example, prediction based on past data. The null hypothesis declares that the outcome would show no significant difference between models based on the linear and non-linear nature of financial share prices.

A plausible formulation of the null hypothesis for this research is that there is no significant difference in predictability between linear models such as linear regression and non-linear models like those using ANNs, and the confidence level of the predictions is not useful. The reason for pessimism rather than optimism is that there is no prior guarantee that assuming the non-linear stochastic nature of the shares is true and furthermore that using an ANN would be a successful approach. Experiments will be conducted to prove or disprove the hypothesis.

A confusion matrix for the null hypothesis is given in Table 1-1 below

TABLE 1-1 CONFUSION MATRIX

	Computed Accept	Computed Reject
Accept Null Hypothesis	True Accept	Type 1 Error
Reject Null Hypothesis	Type 2 Error	True Reject

A type 1 error occurs when a true null hypothesis is rejected. A type 2 error is observed when a null hypothesis that should have been rejected is accepted.

For the experiments in this research, a type 1 error would have one believe that prediction has significant evidential support. Likewise, a type 2 error would state that the prediction fails.

An important requirement for this methodology is that an independent dataset is used.

The primary objectives of the present research are the modelling of short-term (daily trading) changes in FTSE 100 shares (Aminian, et al., 2006, Amman, et al., 2010) by means of the use of Artificial Neural Networks (ANNs) to predict their performance with a certain confidence level (Rotundo, 2004) and associated risk levels. This research represents an advance in the use of present machine learning approaches with hybrid ANN and PSO models in both time and digital domains and a suitable methodology is developed as follows to:

1. Identify and model actual real-world trading processes.
2. Identify financial share types and trading strategies.
3. Research ANNs and time series analysis.
4. Apply ANNs to financial share prediction.
5. Investigate the ANN convergence and acceleration methods.
6. Identify alternative approaches to evaluate the findings.
7. Evaluate the results, including real share datasets.
8. Simulate the model using computational simulation.

1.4. Theoretical Framework

The results gained from the use of neural networks are difficult to interpret, and so deriving corresponding formal mathematical models such as stochastic or autoregressive models could help with the understanding of the results. Secondly, it is interesting to compare the results of a stochastic model and those of machine learning (ANN and PSO) approaches.

The theoretical framework is used to investigate the analytical mathematical modelling of financial shares via modelling through analysis and simulation. Numerous research studies that have been conducted with financial derivatives and particularly options are adopted. The assertion is that the models should not be very different. This is likely to be formulated as a null hypothesis for the methodological purposes of consistency. Furthermore, the outcomes of numerical solutions for partial differential equations are similar to those of regression and neural networks, and the parallels could be relevant.

Knowledge of the characteristics of share probability distributions provides insight into how to model investment returns, future prices and their confidence intervals. Asset returns are usually considered to be normally distributed, as shown in Figure 1-4.

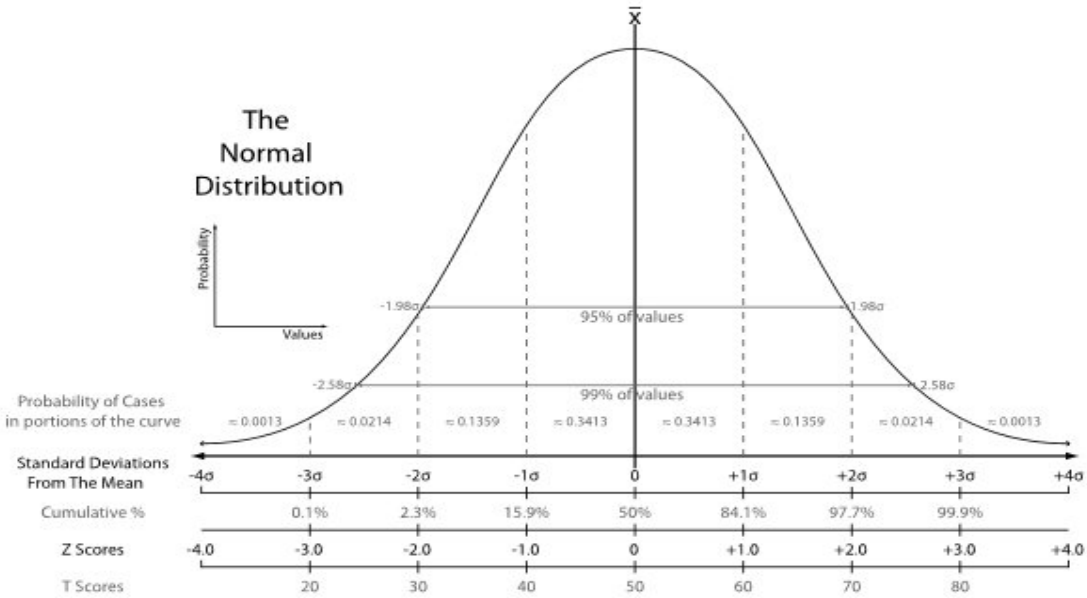


FIGURE 1-4 NORMAL DISTRIBUTION (ZUCCHI, 2017)

The log normal distribution is specific to expected stock prices. Such a distribution is non-zero and skewed to the right, although it has no theoretical upper limit but cannot fall below zero as shown in Figure 1-5. The expected price is the product of the current stock price and various rates of return which are assumed to be normally distributed. Compounding the returns creates a lognormal distribution (Zucchi, 2017).

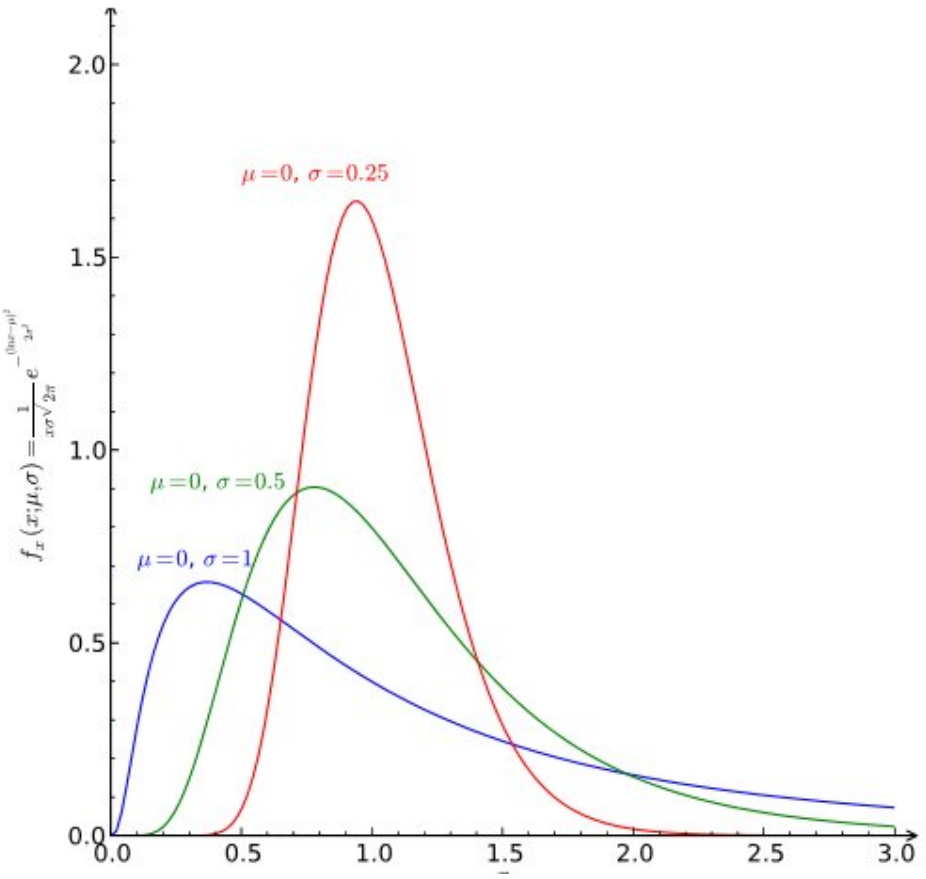


Figure 1-5 Lognormal distribution (Zucchi, 2017)

The confidence level of a model associated with uncertainty could be determined with Bayesian reasoning, certainty factors and evidential reasoning and building a fuzzy system (Kodogiannis, et al., 2002) or the operation of sets, rules, and inferences (Roiger, et al., 2003).

The unified approach to financial equities modelling is conducted via partial differential equations (PDEs) of the diffusion type, which is considered to be the best approach to the modelling of financial subjects. A popular example of this is the Black-Scholes model

(Black, 1972). The simplest case of predicting a share would be sufficient for the hypothesis. The preference is for the engineering of the numerical solution of the model, rather than an explicit analytical solution, wherever possible and appropriate. Numerical analysis is faster than exact solutions which are rare as well. The mathematical basis emphasised is the derivation and use of deterministic differential equations and associated numerical methods. This is a more intuitive approach, rather than in the terms of stochastic processes. In this way the directness of the approach is improved.

1.5. Methodology Including Data Analysis

The proposed methodology is the conceptual ANN methodology for learning and generalization. An artificial neural network is built from interconnected neurons, and two types of neural networks can be distinguished: static neural networks, which are often described as feed-forward networks; and dynamic neural (or recurrent) networks. Initially the focus is anticipated to be on static networks, for simplicity, but further investigation of dynamic networks is considered. Static feed-forward neural networks are composed of static neurons, and the output of the network is computed as soon as input values are presented and can be organised in several topologies. When not all neurons are output neurons, the network contains hidden neurons. The most general architecture is realised when the different neurons are fully connected (with no recurrent connections) to the others, and for most applications the different neurons are grouped in layers. A basic structure of a feed-forward neural network with one hidden layer is shown in Figure 1-6.

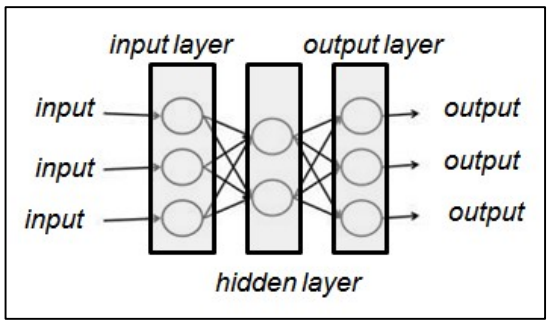


FIGURE 1-6 FEED-FORWARD NEURAL NETWORK

Designing a neural network involves defining the architecture and values of weights. The first issue is the ANN architecture. Its design is not a trivial task, and requires the number of hidden layers to be defined, how many units are in each, how many connections there are and what learning parameters are to be used. The usual procedure relies on trying different architectures with different patterns of connectivity to find the 'best' or at any rate a satisfactory model.

The second issue is finding the right values for the weights, which is often described as 'learning' or 'training the network'. The learning phase is concerned with different ways to obtain a close fit between the mapping function and the training set. During the learning phase, synaptic coefficients are computed so that the network performs a task (such as classification or time series prediction) where the required performance is defined by a training set that consists of examples with their desired output values. The learning can be viewed as an optimisation problem with the goal of minimising an error measure with respect to the weights for a given set of training samples. Aspects of the learning phase to be considered are error criteria, back-propagation, convergence and acceleration methods (weight initialisation, avoiding local minima, data sequencing, batching, momentum, learning rate control, change in the sigmoid derivative, and so on).

Nevertheless, the goal of forecasting is not to memorise the training set but to learn something about the past that can be generalised in the future when given a new example of the problem. The generalization phase is the ultimate estimation of network behaviour and its performance with the entire population of all possible examples (the universe of possible cases). In general, it is impossible to access this set, and there is a practical problem of trying to maximise performance relative to the universal dataset rather than encouraging the network to fit the 'noisy' training set too closely. Aspects of the generalization phase to be taken into account are issues of noise and over-fitting such as sample size, concurrent descent, cross-validation and regularization.

Particle Swarm Optimization (PSO) is a stochastic optimization technique. The algorithm is based on the behaviour of swarms, such as groups of birds. The PSO idea has expanded to become a common heuristic optimization algorithm with many interpretations of its concepts, issues, and applications. Despite the relative simplicity of individuals, swarm systems display complex behaviour. They are made up of numerous individuals and tend to be flexible and robust. Swarm intelligence thus provides a framework for the design and implementation of systems made up of many agents that are capable of cooperation for the solution of highly complex non-linear optimization problems and thus suitable to combine with the neural network technique. One common feature of heuristic approaches is that they use probabilistic rules to find global optimal solutions and may prove to be very effective in solving problems without modifying the shape of their cost curves. A basic concept of a particle swarm optimization algorithm is shown in Figure 1-7

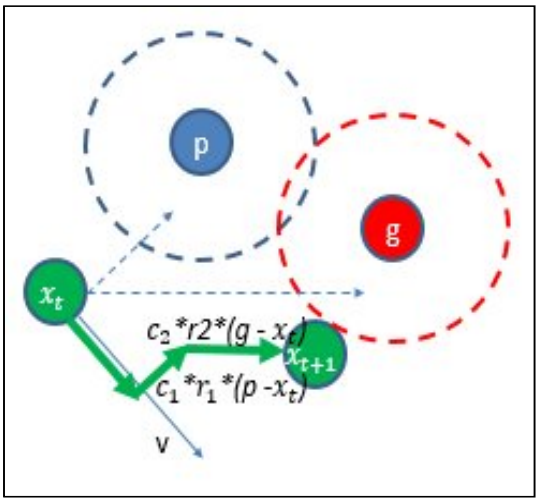


FIGURE 1-7 PARTICLE SWARM OPTIMIZATION ALGORITHM

The discrete Furrier transform and discrete wavelets transformations of time series changes the representational space from the time domain to the digital domain. In analyzing financial shares trading, this provides another way to interpret and understand data patterns. It is then possible to perform certain types of time series processing and measurement operations with much less computational effort compared to analysis in the time domain, reducing complexity and increasing the understanding of patterns in the data

and the selection of important features. A basic concept of a time-to-frequency transformation model is shown in Figure 1-8.

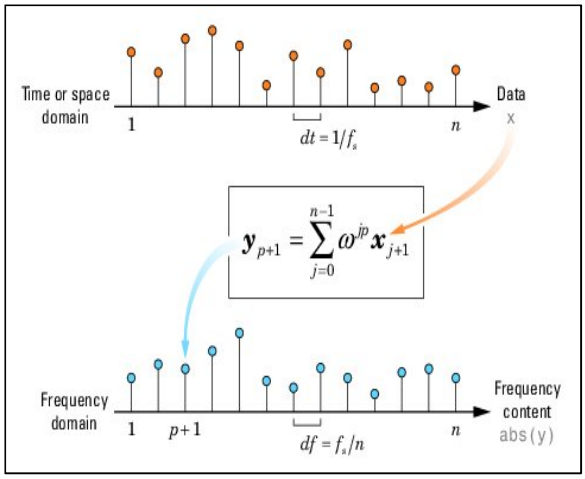


FIGURE 1-8 TIME TO FREQUENCY TRANSFORMATION MODEL

A mathematical analysis of the PSO algorithm is conducted from a system point of view in both continuous-time and discrete time settings along with a stability analysis for the choice of the parameters currently employed for the algorithm. A basic block diagram of the PSO algorithm is shown in Figure 1-9.

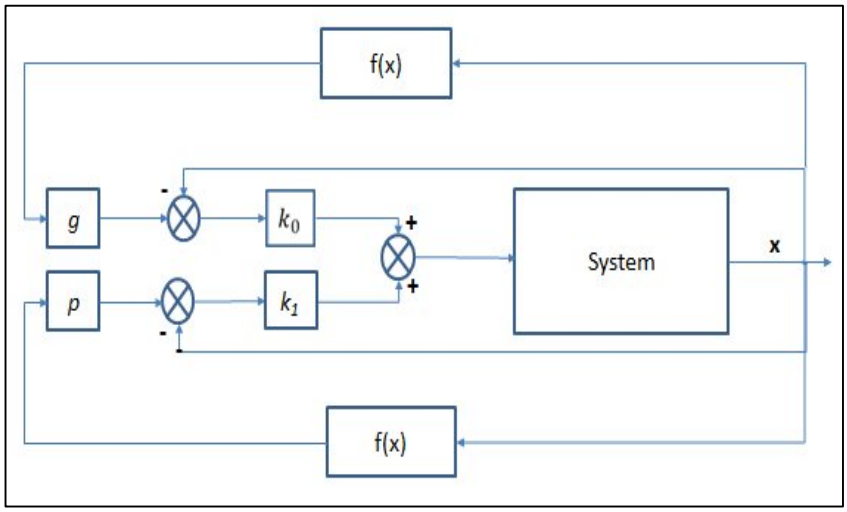


FIGURE 1-9 CLOSED LOOP FEEDBACK SYSTEM

Furthermore, a possible integral and differential extension is proposed. The analysis is carried out only in the scalar case in order to simplify the demonstration.

1.6. Contribution to the Research Field

The main contributions to the research field that have been done in this thesis are:

1. The related to the financial market modelling and trading problems and current approaches to them have been identified. Many people use different methods in this area but have not used recent machine learning (deep learning) investigations or considered the validity of these techniques.
2. Different trading strategies with the focus on short-term one-day trading have been investigated and identified trading criteria: long term (investments) and short term (speculation) trading exploring the volatility (historic and implied) and risks taken.
3. A comparison between different methods and algorithms has been done and has been proven that the approaches taken complement each other and gradually improve quality (optimum), generalization, robustness and performance.
4. Real data in real time has been collected and applied: intraday, daily, weekly and monthly trading strategies with a sustainable success.
5. A survey of the existing researched methods has been done, the literature has been researched and the most recent research trends have been selected.
6. The trend/patterns in the modelling of the financial market have been investigated.
7. Deep learning techniques have been applied to the financial domain model features to identify specific trends, periodicity, seasonal features and digital (DFT & DWT) presentations.
8. Systemic improvements of the PSO for convergence/stability and response quality with the PID algorithm have been proposed based on control theory analysis and design.
9. Hybrid PSO and ANN both technical implementation and methodology (how to apply them) have been integrated.

10. The stochastic and non-linear nature of financial shares has been proven and appropriate approaches to deal with them with ANN, PSO and control theory have been identified.

In this work, new models based on neural networks and particle swarm optimization are derived that improve prediction performance. The focus is on improvements in both the convergence and confidence levels of the results.

The poor prediction accuracy of linear regression analyses of typical financial shares daily closing prices time series, such as the shares listed in FTSE 100, National Association of Securities Dealers Automated Quotations (NASDAQ) and Dow Jones Industrial Average (DJIA), suggests that a non-linear model, such as one using multi-layer neural networks, is more likely to be a better choice. A prior assumption is that financial shares time series contain significant non-linearity and that artificial neural networks, either separately or in conjunction with other techniques such particle swarm optimization, can deal with them. The methodology used is time series analysis and forecasting in the financial domain. The research therefore makes contributions in the domain of computational geometry and statistical analysis.

A new systematic methodology comprising both simulation and theoretical mathematical approaches from control theory is derived, providing a framework to study and evaluate the models developed.

It could be seen that there is a similarity between artificial neural networks and statistical and numerical non-linear methods, implying possible convergence and mutual application to improve prediction and the computational convergence of both parametric and non-parametric models. However, this would make the suggested hypothesis concerning artificial neural networks and parametric methods less obvious.

The work considered in this thesis has been published in the following papers:

Turkedjiev, E., Busawon, K. and Angelova, M., 2013. Validation of artificial neural network model for share price. In: *UKSim2013: 15th International Conference on Modelling and Simulation*, Cambridge, UK.

Rani, C., Petkov, E., Busawon, K. and Farrag, M., 2014, November. Chaotic adaptive particle swarm optimisation using logistics and Gauss map for solving cubic cost economic dispatch problem. In *Environmental Friendly Energies and Applications (EFEA), 2014 3rd International Symposium on* (pp. 1-5). IEEE.

Rani, C., Petkov, E., Busawon, K. and Farrag, M., 2014, November. Particle swarm optimization with exponentially varying inertia weight factor for solving multi area economic dispatch. In *Environmental Friendly Energies and Applications (EFEA), 2014 3rd International Symposium on* (pp. 402-407). IEEE.

Busawon, K., Rani, R., Turkedjiev, E., and Binns, R. (submitted) Extension of particle swarm optimisation algorithm: application to economic dispatch. In *Transaction on Evolutionary Computation*

1.7. Structure of the Thesis

Chapter 1. Introduction

Introduces material that is fundamental to understanding the nature of financial shares and machine learning approaches and processes.

Chapter 2: Models, Methods and Performance Evaluation

Formalizes data mining problems and details advanced modelling methods such as autoregressive moving average, artificial neural networks and particle swarm optimization.

Chapter 3. Validation of ANN Model for Share Prices

Applies formal statistical and non-statistical methods for evaluating outcomes of linear regression, artificial neural networks and bi-linear regression models.

Chapter 4. Stochastic Share Price Model

Models the behaviour of financial shares prices, deriving an analytical continuous-time model considering stochastic calculus and Wiener processes and volatility.

Chapter 5. Time Series

Applies to the financial time series standard statistical analysis such as summary statistics, confidence intervals and particularly analysis of variance and correlation.

Chapter 6. Discrete Fourier Transform

Extends financial time series modelling in the time domain, specifically focusing on discrete Fourier transform analysis and forecasting including neural network utilization.

Chapter 7. Discrete Wavelet Transform

Similar to the discrete Fourier transform, this extends the financial time series investigation with the discrete wavelet transformation

Chapter 8. Hybrid Particle Swarm Optimization and Artificial Neural Networks

Introduces the particle swarm optimization (PSO) paradigm and using techniques and models drawn from the control theory analyzes its stability. A further proportional, derivative and integral (PID) extension of the basic algorithm is proposed. Integration with neural networks and methodology of application is proposed.

Chapter 9. CAPM and Risk Analysis

Explains the classical capital asset pricing model (CAPM) and the related risk concept.

Chapter 10. Conclusions and Future Work

Outlines the major conclusions of the study andrecommends future work.

1.8. Summary

As the existing literature suggests, neural networks have not been extensively utilised and evaluated in the field of the modelling of short-term financial stock market shares exclusively in respect of the time interval (short-term trading) and confidence and risk levels (low, medium, high) in various day trading strategy models.

The primary goal of the present research is to model short-term daily trading in FTSE 100 shares to forecast with certain levels of confidence and associated risk. The hypothesis to be tested is that financial shares time series contain significant non-linearity and that ANN, either separately or in conjunction with PSO, could be utilised effectively.

The combined application of artificial neural network modelling and optimization approaches such as particle swarm optimization can improve convergence and performance. Investigations with digital transforms such as Fourier and wavelets can produce superior results compared to time domain analysis.

Chapter 2: Models, Methods and Performance Evaluation

2.1. Overview

This chapter formalizes data mining problem and details advanced modelling methods such as the autoregressive moving average, artificial neural networks and particle swarm optimization and their performance evaluation.

Effective model development requires a train-test methodology. Depending on the objective of the application, the best model may be based on how well it interpolates (as measured by performance on the test dataset), or how well it performs in a deployed environment (using the validation dataset). To evaluate how well a model interpolates, the model training process should be periodically interrupted and the network tested and/or validated. Performance can be evaluated only in terms of the performance of the entire system. Standard technical measures such as root mean square (RMS) error and mean squared error (MSE) are common indicators of system performance.

This chapter covers the following subjects:

- datasets and data collection
- finding historical prices
- retrieving Google finance quote live
- evaluating performance
- autoregressive moving average (ARMA)
- artificial neural networks

2.2. Datasets and Data Collection

Stock index datasets are time series representations of Open, High, Low, Close, Volume and Adjusted Close prices. The data source is the Yahoo Finance website (http://finance.yahoo.com/) that gives prices of USA, European and Asian markets in downloadable Excel and "csv" spreadsheet formats. There are options for start and end

dates as well as whether the information is to be summarised by day, week or month. Examples that have been downloaded and ready for data mining are FTSE 100 (^FTSE) (from 2/04/1984 to 7/07/2011) and Barclays PLC (BARC.L) (from 01/01/2003 to 07/07/2011).

Attributes are Date represented in DD/MM/YYYY, Open, High, Low, Close and AdjClose in GBP and Volume in number integer.

2.3. Finding Historical Prices

1. Go to http://uk.finance.yahoo.com/.
2. Enter "BARC.L" in "Get Quotes" window shown in Figure 2-1.

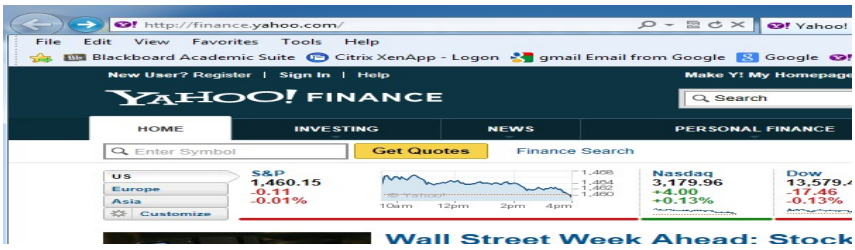


FIGURE 2-1 GET QUOTES

3. Follow "Historical Prices" menu on the left shown in Figure 2-2.

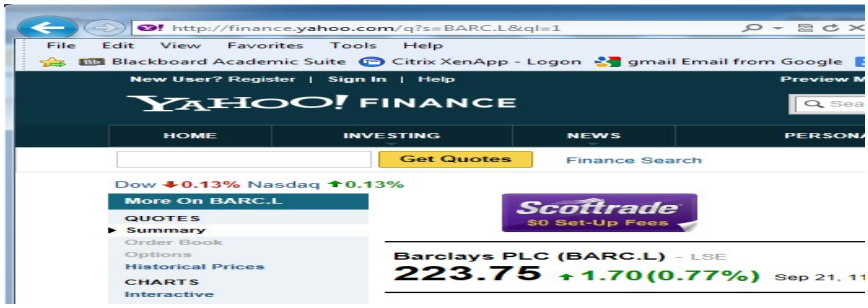


FIGURE 2-2 HISTORICAL PRICES

4. Select "Set Date Range" shown in Figure 2-3.

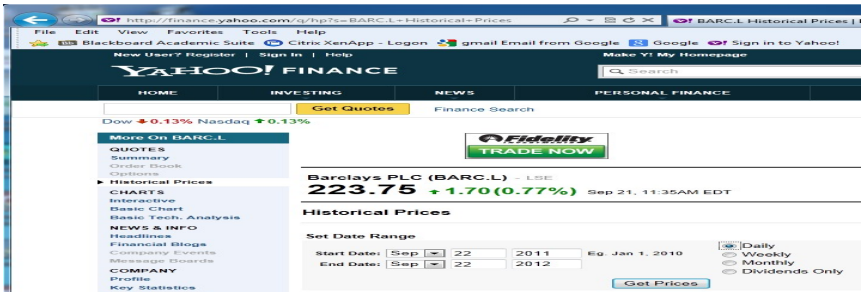


FIGURE 2-3 SET DATE RANGE

5. Follow “Download to Spread sheet” at the bottom of the page shown in Figure 2-4.

Jun 25, 2012	199.65	200.80	192.09	194.25	33,386,900	194.25
Jun 22, 2012	198.30	204.90	195.74	200.70	24,832,300	200.70
Jun 21, 2012	204.35	208.60	201.70	202.30	34,720,700	202.30
Jun 20, 2012	199.85	206.50	199.05	205.75	44,890,300	205.75
Jun 19, 2012	196.75	202.35	193.95	200.60	33,217,700	200.60

* Close price adjusted for dividends and splits.

First | Previous | Next | Last

Download to Spreadsheet

Currency in GBp.

FIGURE 2-4 DOWNLOAD TO SPREAD SHEET

6. Sorting oldest to newest in Excel: select all columns; Sort & Filter tab (Sort Oldest to Newest).

Barclays PLC long term share prices 22/09/2011 to 21/09/2012 are shown in Figure 2-5.

FIGURE 2-5 BARC.L FROM 22/09/11 TO 21/09/12

Long-term distribution 22/09/2011 to 21/09/2012 is shown in Figure 2-6.

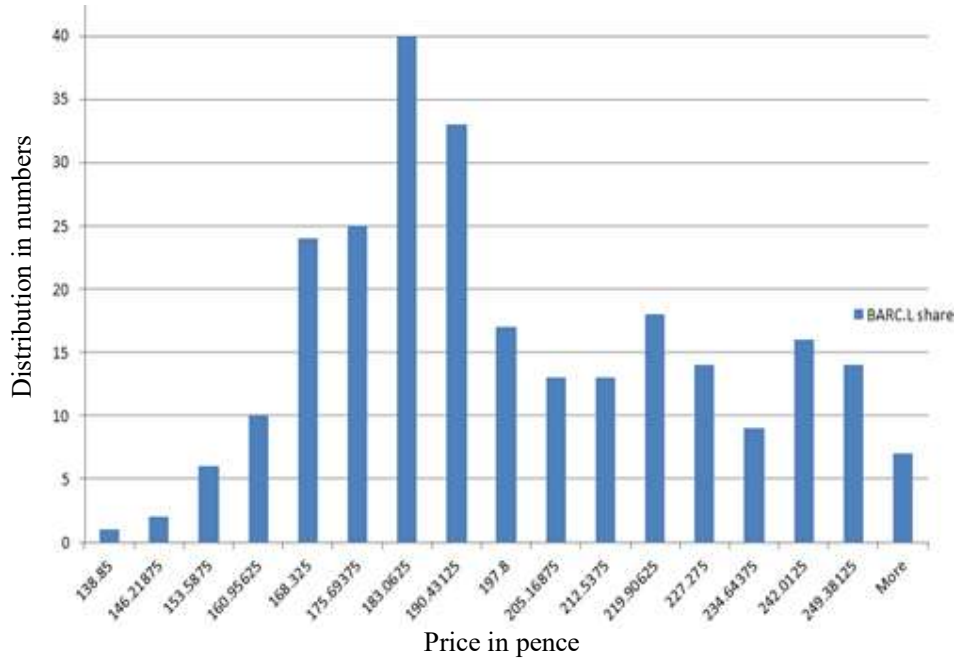


FIGURE 2-6 DISTRIBUTION OF BARC.L SHARE PRICES FOR ONE YEAR

The BARC.L share short term prices dataset from 27/08/2012 to 21/09/2012 is shown in Appendix C: Table 1 and BARC.L share closing prices dataset from 27/08/2012 to 21/09/12 is shown in Appendix C: Table 2.

A graph of BARC.L share closing prices from 27/08/12 to 21/09/12 is shown in Figure 2-7.

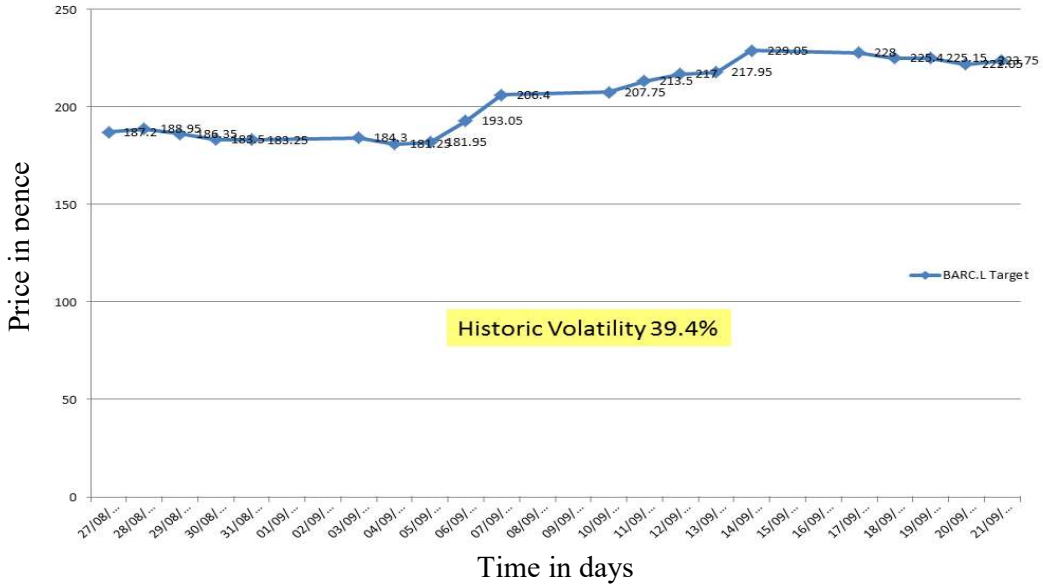


FIGURE 2-7 BARC.L SHARE CLOSING PRICES FROM 27/08/12 TO 21/09/12

The distribution of BARC.L share closing prices from 27/08/12 to 21/09/12 is shown in Figure 2-8.

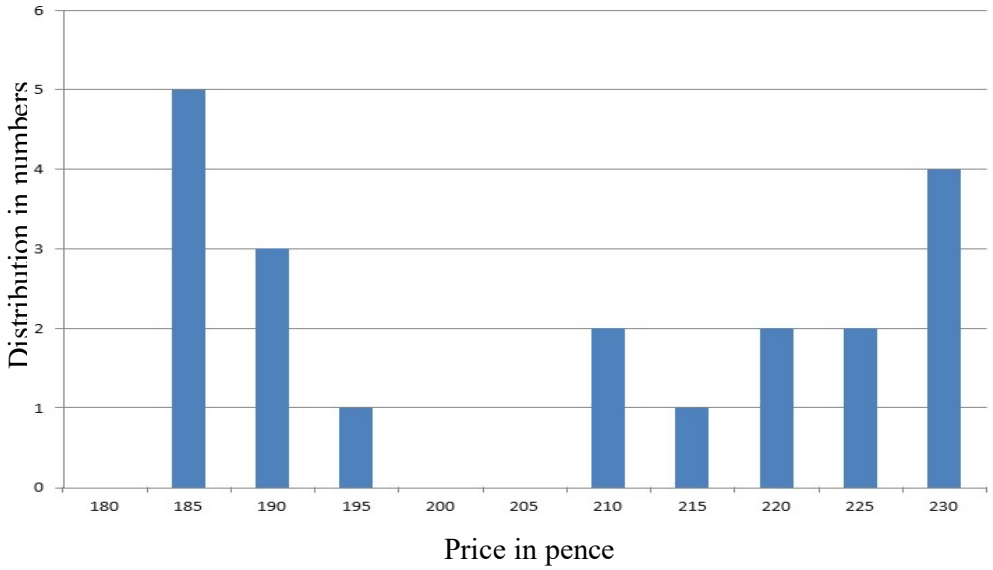


FIGURE 2-8 DISTRIBUTION BARC.L SHARE CLOSING PRICE FROM 27/08/12 TO 21/09/12

2.4. Retrieving Google Finance Quote Live

Matlab script is used to retrieve a stock quote (last trade) from Yahoo! Finance. Yahoo's quotes are delayed by 15 minutes, limiting their timeliness, where as there is no delay in Google Finance's quotes and so it is better to retrieve data from the Google site instead. The method used to get a free real-time stock quote from Google Finance in Matlab code is described elsewhere (luminouslogic, 2015).

Functions: *periodic_prices_stop.m*, *timerCallback.m*, *get_last_trade_record_google.m*, and *periodic_prices_start.m*with default of 10secondsand'BARC.L'.

Execution Sequence:

(Symbol 'BARC.L' and 10sec period are hardcoded.)

>>periodic_prices_start (Excel spreadsheet is created)

stock_symbol	date_time	last_trade
BARC.L	07/27/16 16:26:38	149.95
BARC.L	07/27/16 16:26:48	149.57

2.5. Evaluating Performance

When building a system that uses neural network indicators, a trading strategy and an evaluation system to measure the performance of the combined system must also be

developed (Deboeck, 1992, Pardo, 1992). Performance is measured in terms of system objectives, such as profit, Sharpe ratio (Sharp, 1994), which is a measure for a risk-adjusted return to variability, or maximum drawdown before a new peak for a specified period. If the current network scores higher than prior ones, it is saved as the best network. Training stops when the measured system performance is repeatedly less than the current best performance.

Standard technical measures such as RMS error may be quite poor predictors of system performance. Defining the system objectives and then integrating them into the neural network development process is essential.

Model performance is most often evaluated with some measure of test set error rate. For categorical outputs this is the ratio of test set errors to total test set instances. For numerical outputs it is the MSE or the RMS. The distribution of sample means taken from a set of independent samples of the same size is distributed normally, and so the test set error rate can be treated as a sample mean. Using the properties of the normal distribution, the error rate confidence intervals can be computed. Also, classical hypothesis testing can be used to compare test set error rates for different models. These techniques allow one to associate measures of confidence with the results. When a model fails to perform as expected, an appropriate strategy is to evaluate the effect which every component has had on model performance, such as training and test data, input attributes, learning technique, and user-specified parameters (Roiger, et al., 2003).

2.5.1 Evaluating Supervised Models with Numerical Output

- Mean Squared Error (MSE)

$$MSE = \sum(xi - \mu)^2 \quad (2\text{-}1)$$

- Root Mean Squared Error (RMS), where applying the square root reduces the dimensionality of the MSE to that of the actual error computation. The value of *t* is used as a measure of convergence with the ANN:

$$RMS = \sqrt{\sum(xi - \mu)^2} \quad (2\text{-}2)$$

- Mean Absolute Error (MAE). This is less affected by large deviations

$$MAE = |xi - \mu| \quad (2\text{-}3)$$

2.5.2 Model for Significant Model Difference

The classical model to test for a significant difference between the mean scores of a measured parameter is through the value of the ratio *P* of the absolute difference between the mean scores and the standard error for the distribution of mean differences.

$$P = \frac{|\overline{E1} - \overline{E2}|}{\sqrt{(\frac{\gamma 1}{n1} + \frac{\gamma 2}{n2})}} \quad (2\text{-}4)$$

where:

P is the significant difference score,

$\overline{E1}$ and $\overline{E2}$ are the sample means of the test dataset error,

$\gamma 1$ and $\gamma 2$ are variance scores for the respective means

the mean μ, or average value, is computed via $\mu = \frac{\sum e}{n}$,

the variance σ^2 measures the dispertion about the mean, $\sigma^2 = \sum(e_i - \mu)^2$, and

the denominator $\sqrt{(\frac{\gamma 1}{n1} + \frac{\gamma 2}{n2})}$ is the standard error for the distribution of means differences

This model is valid for independent test datasets because the distribution of differences between sample means is normal, like the distribution of means. As a result to be 95% confident that the means are different, the ratio *P* has to be greater than 2.

2.5.3 Pair wise Comparison for Model Difference

$$\gamma_{12} = \frac{1}{n-1}\sum_1^n[(ae1_i - ae2_i) - (mae1_i - mae2_i)]^2 \quad (2\text{-}5)$$

$$P = \frac{|mae1 - mae2|}{\sqrt{\frac{\gamma_{12}}{n}}} \quad (2\text{-}6)$$

where:

γ_{12} is the joint variance,

$ae1_i$ and $ae2_i$ are the absolute errors, and

$mae1_i$ and $mae2_i$ are the absolute error means.

2.5.4 Confidence Interval for Numerical Output

$$\gamma(mae) = \frac{1}{n-1}\sum_1^n(aei - m \quad)^2 \tag{2-7}$$

$$SE = \sqrt{\frac{\gamma(mae}{n}} \tag{2-8}$$

$confidenceUpperLimit(error) = mae + 2SE$

$confidenceLowerLimit(error) = mae - 2SE$

$accuracyUpperLimit = 100\% - confidanceLowerLimit$

$accuracyLowerLimit \quad 100\% - confidanceUpperLimit$

$accuracyUpperLimit$

2.6. Autoregressive Models

For generalization performance purposes, it is important to reduce attribute correlations. Correlation graphs between the normalized current value at t_i and previous values at t_{i-1}, t_{i-2}... t_{i-5} are illustrated in Figure 2-9.

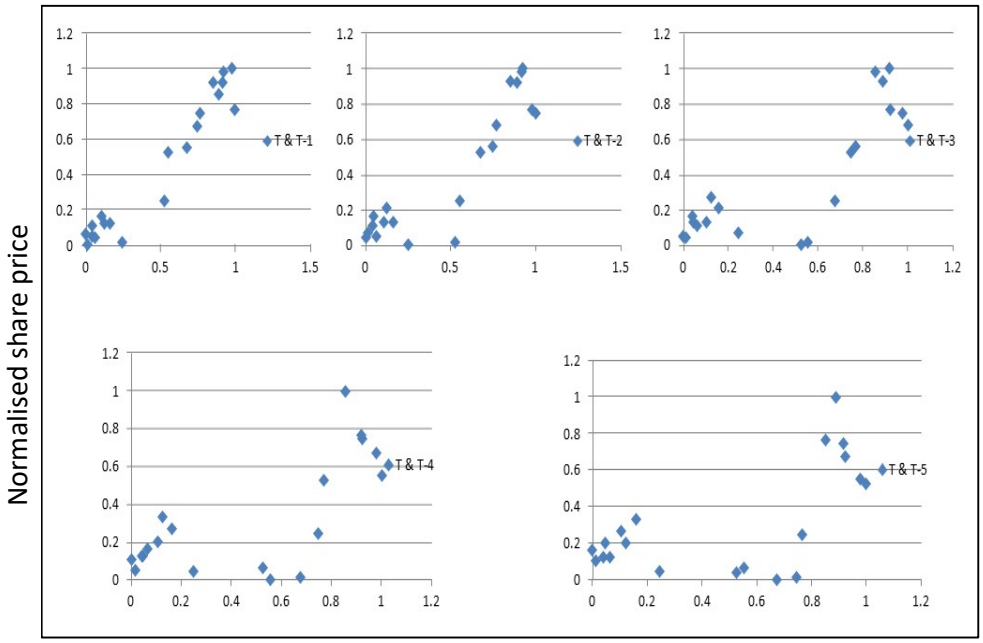


FIGURE 2-9 ATTRIBUTE CORRELATION

Correlation graphs show gradually less linear relations (correlations) between the current value at t_i and the previous values at t_{i-1}, t_{i-2}... t_{i-5}. The relationship trends are declining as well, from one when there is one-to-one value equivalence without a time difference shift between the values t_i vs. t_i to a slightly declining trend of approximately 0.6~0.8 at the furthest values at time t_i vs. t_{i-5}.

2.6.1 Autoregressive Moving Average (ARMA)

The most popular type of time-domain time–series modeling based on statistics and signal processing in econometrics are autoregressive moving average (ARMA) models, which are sometimes called Box-Jenkins models after the iterative Box-Jenkins methodology (Box, 1970) usually used to estimate them. They assume auto-correlated time series data. Given a time series of data X_t, the ARMA model is a tool for understanding and, perhaps, predicting future values in this series. The model consists of two parts, an autoregressive (AR) part and a moving average (MA) part. The model is usually then referred to as the *ARMA (p, q)* model where p is the order of the autoregressive part and q is the order of the moving average part, as in the equation below:

$$X_t = \mu + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \quad (2\text{-}9)$$

where:

μ is the mean

X is the time series

ε are white noise terms

p *is* the order of the autoregressive part

q is the order of the moving average part

φ are the autoregressive parameters

θ are moving average parameters

2.6.2 ARMA Methodology

The original ARMA modeling methodology uses an iterative three-stage modeling approach (Brockwell, et al., 1987, Pankratz, 1983), which involves the following stages:

1. Model identification and selection, making sure that the variables are stationary, identifying seasonality in the dependent series (seasonally differencing it if necessary), and using plots of the autocorrelation and partial autocorrelation functions of the dependent time series to decide which (if any) autoregressive or moving average components should be used in the model.
2. Parameter estimation, using computation algorithms to arrive at coefficients, which best fit the selected ARMA model. The most common methods use maximum likelihood estimation or non-linear least-squares estimation.
3. Model checking, by testing whether or not the estimated model conforms to the specifications of a stationary univariate process. In particular, the residuals should be independent of each other and constant in mean and variance over time. The means and variances of residuals are plotted over time and a Ljung-Box test (Ljung, et al., 1978) is performed, or plotting the autocorrelations and partial autocorrelations of the residuals is helpful to identify misspecification. If the estimation is inadequate, we have to return to step one and attempt to build a better model.

Estimating the parameters for Box–Jenkins models is a quite complicated non-linear estimation problem. The main approaches to fitting Box–Jenkins models are to use non-linear least squares and maximum likelihood estimation. Maximum likelihood estimation is generally the preferred technique (Brockwell, et al., 1987).

2.6.3 ARMA Investigations

The results with the ARMA (5,3) model with p=5 and q=3 for time-lags of 5 days for BARC.L from 27/08/2012 to 21/09/2012 with dataset in Appendix C: Table 3 are shown in Figure 2-10 and Table 2-1 The models parameters are as follows:

$$X_t = 0.056\varphi_{t-5} - 0.379\varphi_{t-4} + 0.416\varphi_{t-3} - 0.45\varphi x_{t-2}\, 1.291\varphi_{t-1} + 15.321 + 1.040\varepsilon_t + 1.091\varepsilon_{t-1} + 1.073\varepsilon_{t-2} \quad 2\text{-}10$$

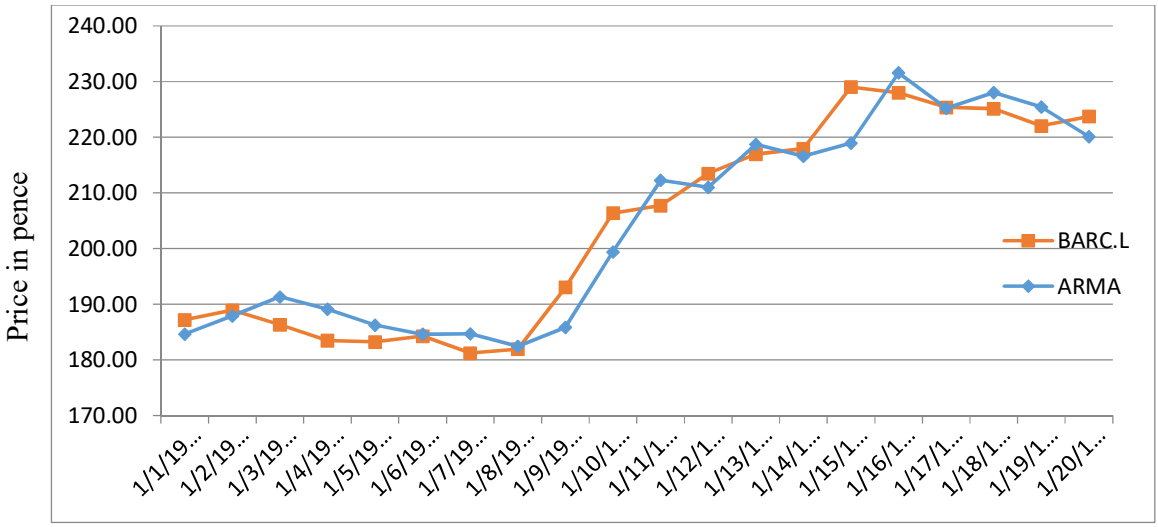


FIGURE 2-10ARMA FROM 27/08/2012 TO 21/09/2012GRAPH

TABLE 2-1 ARMA FROM 27/08/2012 TO 21/09/2012

Date	Target	ARMA(5,3)	abs(err)
8/27/2012	187.20	184.82	2.38
8/28/2012	188.95	188.11	0.84
8/29/2012	186.35	190.14	3.79
8/30/2012	183.50	188.06	4.56
8/31/2012	183.25	185.86	2.61
9/3/2012	184.30	184.34	0.04
9/4/2012	181.25	185.37	4.12
9/5/2012	181.95	180.57	1.38
9/6/2012	193.05	183.97	9.08
9/7/2012	206.40	197.52	8.88
9/10/2012	207.75	211.78	4.03
9/11/2012	213.50	212.27	1.23
9/12/2012	217.00	220.17	3.17
9/13/2012	217.95	219.41	1.46
9/14/2012	229.05	220.83	8.22
9/17/2012	228.00	232.68	4.68
9/18/2012	225.40	225.16	0.24
9/19/2012	225.15	225.70	0.55
9/20/2012	222.05	223.09	1.04
9/21/2012	223.75	219.37	4.38

The ARMA model fits well the target with the sum of absolute error errors for the last five days of 10.89 corresponding to a mean error of 2.18. The r2 value (the graphs similarity fit) for the whole data range is good at 0.94 but marginal for the last five days at 0.63, which is obvious from the graph where the last day in the model is in the opposite direction to the target original graph. The model is quite sensitive to the random autoregressive

second part, though generally without significant differences in the performance sum of absolute errors less than 10%. The model is quite robust to the order p (the order of the average part) and q (the order of the autoregressive part) as well.

2.6.4 Statistical Regression

Statistical regression is a supervised technique that generalizes a set of numerical data by creating a mathematical equation relating one or more input attributes to a single output attribute. With linear regression, we attempt to model the variation in a dependent variable as a linear combination of one or more variables, for example see Table 2-2:

$$f(x_1, x_2 \ldots x_n) = y = m_1 x_1 + m_2 x_2 + \cdots m_n x_n + b \tag{2-11}$$

where:

m_i are parameters,

b is a constant, and

x_i are independent values.

Excel support is the LINEST function (LINEST.Support.office.com, 2017)

TABLE 2-2 EXCEL LINEST FUNCTION

	A	B	C	D	E	F
1	m_n	m_{n-1}	...	m_2	m_1	b
2	se_n	se_{n-1}	...	se_2	se_1	se_b
3	r_2	se_y				
4	F	d_f				
5	ss_{reg}	ss_{resid}				

where:

m values are coefficients corresponding to each x-value

se1,se2,...,sen are the standard error values for the coefficients m1,m2,...,mn.

seb - is the standard error value for the constant b.

r2 is the coefficient of determination, which compares estimated and actual y-values, and ranges in value from 0 to 1. If it is 1, there is a perfect correlation in the sample — there is no difference between the estimated y-value and the actual y-value. At the other

extreme, if the coefficient of determination is 0, the regression equation is not helpful in predicting a y-value.

sey is the standard error for the y estimate.

F is the F statistic, or the F-observed value used to determine whether the observed relationship between the dependent and independent variables occurs by chance.

df is the degrees of freedom used to find F-critical values in a statistical table. The values found in the table are compared to the F statistic returned by LINEST to determine a confidence level for the model.

ssreg is the regression sum of squares.

ssresid is the residual sum of squares

2.6.5 Statistical Regression Investigations with LINEST

The results with Excel LINEST function for time-lags of 3 and 5 days for BARC.L target data in Table 2-5 from 27/08/2012 to 21/09/2012 Appendix C: Table 1 and Appendix C: Table 2 are shown in Figure 2-11 graphs and Table 2-5. Model parameters specified in Table 2-2 are given in Table 2-3 and Table 2-4

TABLE 2-3 LINEST 5 DAYS

0.055974	-0.37928	0.416048	-0.4547	1.291492	15.32143
0.281587	0.417605	0.440336	0.435629	0.262299	17.36956
0.944012	5.090413	#N/A	#N/A	#N/A	#N/A
47.21082	14	#N/A	#N/A	#N/A	#N/A
6116.706	362.7723	#N/A	#N/A	#N/A	#N/A
#N/A	#N/A	#N/A	#N/A	#N/A	#N/A

TABLE 2-4 LINEST 3 DAYS

-0.00017	-0.33294	1.288545	10.33216	#N/A	#N/A
0.259345	0.404207	0.246675	13.85784	#N/A	#N/A
0.938197	5.002824	#N/A	#N/A	#N/A	#N/A
80.96218	16	#N/A	#N/A	#N/A	#N/A
6079.026	400.452	#N/A	#N/A	#N/A	#N/A
#N/A	#N/A	#N/A	#N/A	#N/A	#N/A

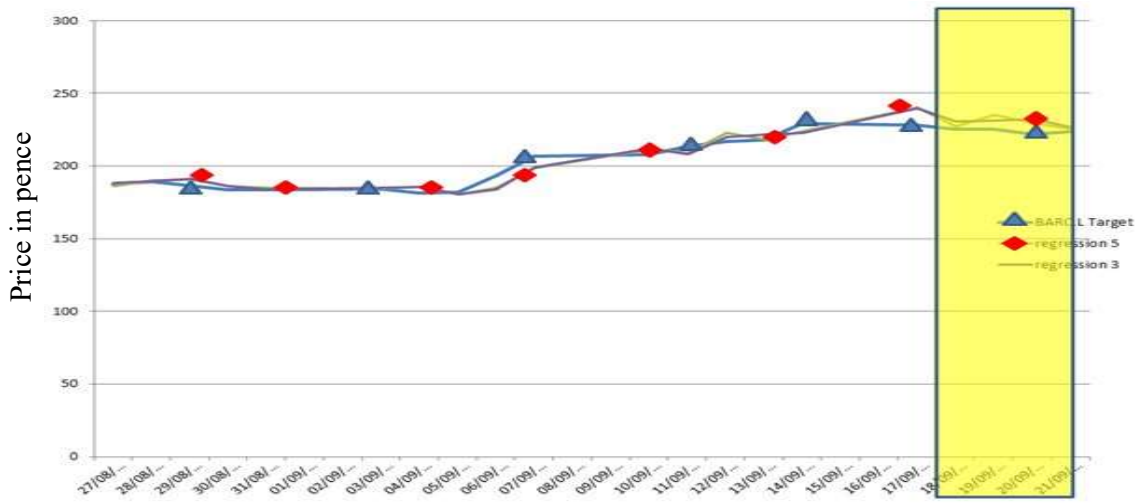


FIGURE 2-11 STATISTICAL REGRESSION CHART

TABLE 2-5 STATISTICAL REGRESSION RESULTS

Target		Regression 3		Regression 5	
27/08/2012	187.2	188.32	1.12	186.13	-1.07
28/08/2012	188.95	189.98	1.03	190.31	1.36
29/08/2012	186.35	191.45	5.10	190.89	4.54
30/08/2012	183.5	186.30	2.80	186.60	3.10
31/08/2012	183.25	184.33	1.08	185.43	2.18
03/09/2012	184.3	185.12	0.82	184.41	0.11
04/09/2012	181.25	186.00	4.75	185.73	4.48
05/09/2012	181.95	180.58	-1.37	180.73	-1.22
06/09/2012	193.05	184.08	-8.97	185.44	-7.61
07/09/2012	206.4	199.46	-6.94	198.97	-7.43
10/09/2012	207.75	212.19	4.44	211.89	4.14
11/09/2012	213.5	208.42	-5.08	209.66	-3.84
12/09/2012	217	220.32	3.32	223.13	6.13
13/09/2012	217.95	222.05	4.10	218.30	0.35
14/09/2012	229.05	222.85	-6.20	223.85	-5.20
17/09/2012	228	240.08	12.08	240.65	12.65
18/09/2012	225.4	231.09	5.69	227.66	2.26
19/09/2012	225.15	231.32	6.17	235.12	9.97
20/09/2012	222.05	232.42	10.37	229.62	7.57
21/09/2012	223.75	227.09	3.34	225.70	1.95
		output	error	output	error
		MSE	RMS	MSE	RMS
		67.01	8.19	65.11	8.06

The results show that there is no significant difference between the three and five day parameter models MSE_3=67.01 and MSE_5=65.11 and RMS_3=8.19 and RMS_5=8.06. There is an inertia (affinity) in following the trend of the time-series that is eventually compensated for at the end of the predication period which seems to be similar to the length of the model's input data; that is, three and five days in these cases. The *r2* (the graphs' similarity, with maximum value of 1) values for three and five days are close and are good for the whole data range 0.94 but are marginal for the last five days at 0.68.

2.6.6 Non-linearity Regression

One of the first types of non-linear approaches was proposed by Tong in 1983 and referred to as threshold auto regression (TAR) (Tong, 1983), which switches between different linear AR models according to pre-set thresholds. Subsequently STAR models were introduced, standing for `smooth' TAR models, where a continuous threshold indicates the proportions in which different models are used.

2.7. Artificial Neural Networks

The artificial (computing) neural network (ANN) is a relatively new advanced non-linear approach method that has become popular in this field. ANNs are adaptive artificial intelligence software systems that are inspired by how biological neural networks work. The concept originated in the social sciences, physiology and economics and is now central to understanding the behaviour of financial markets. The benefits of ANNs are consistent with Simon's 'bounded rationality' argument (Simon, 1997, Godoi, 2009), according to which market efficiency is expected to be subject to human limitations in processing information. ANNs offer an alternative for investors struggling in environments where pre-existing knowledge about an evolving situation is scarce . A well-structured neural network allows data to be used to determine both the structure and parameters of a general framework for locating evolving relationships. The generic approach with ANNs is data-driven as opposed to the traditional parameterized and/or rule-based expert systems generally used in practice.

ANNs represent a class of techniques known as nonparametric models, in contrast to parametric models. A parametric model is a formula with a form derived from some external theory, which describes the dynamics of a market. For example, the Moving Average Convergence Divergence (MACD) method of trading analysis (Staff MACD, 2017) uses the difference between two moving averages as a trading signal. The length of the fast-moving average is the first parameter and the length of the slow-moving average is

the second parameter in this parametric function. Non-parametric models, however, use a very general formula. Typically, they are capable of approximating a wide variety of relationships between input and output variables. In particular, non-linear non-parametric models developed using the back-propagation neural network are capable of approximating almost any relationship between input and output variables. Building a neural network is equivalent to constructing a mathematical formula.

2.7.1 ANN for Forecasting

The focus of this study is to assess the suitability of neural nets for classification problems and time series analysis. Classification problems in this case are concerned with positioning shares into a number of categories (concerning confidence or risk levels) with categorical classification (low-, medium- and high-risk being the most appropriate); for example, discriminating between low-risk and high-risk investments or surviving and failing companies. The use of time series, on the other hand, is concerned to generate future values for a target variable based on information on current and past values for related and environmental factors.

In the main, a neural network consists of a set of fully connected neuron nodes. The neuron itself comprises a single linear combiner with adjustable synaptic weights independent of the input values, where the output is passed further along to an activation function f(x).

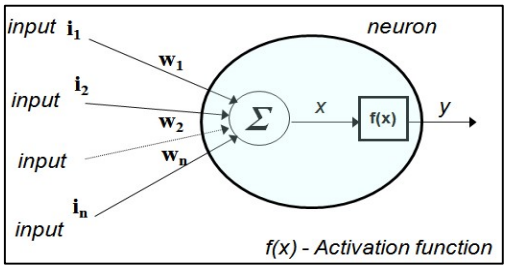


FIGURE 2-12 NEURON NODE

The activation function output propagates activity, with output of a value close to 1 only when significantly excited (Roiger, et al., 2003). The perceptron is a type of neural network (Minsky, et al., 1990) that is a binary classifier mapping its real-valued input

vector *w* to a single binary output value across an input-output matrix. It comprises a linear combiner (neuron) with adjustable synaptic weights independent of the input value threshold *b* (that does not depend on any input value) and a hard limiter step activation function (Roiger, et al., 2003).

$$X = \sum_{i=1}^{n} i_i w_i = i_1 w_1 + i_2 w_2 + \cdots + i_n w_n \tag{2-12}$$

$$f(x) = \begin{cases} 1 \; if \; X > 0 \\ else \; 0 \end{cases} \tag{2-13}$$

In multi-layered neural networks, the hidden or output layer node generally combines the input values into a single value and uses it as an input to an activation function.

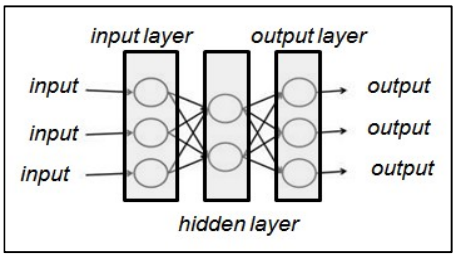


FIGURE 2-13 MULTI-LAYERED ANN

Neural network input indicators attempt to predict market trends and turning points, such as relative strength indicators or directional movement indicators, to decide what the appropriate trading position should be: *long*, *short*, or *out*. These indicators together with the ideal trading signal are used to develop a neural network indicator. Much of the data selected for input to a network is time series data. The method used to capture time-varying information is by using a sliding window on each of the data inputs. Neural network indicators usually cannot deal with the wide ranges of values found in raw data. The data must be scaled into a usable range, usually [0, +1] or [-1, +1].

The generic learning-generalization methodology could be mapped to classification problems and time series analysis, which differ mainly in the presence or absence of a temporal order between the examples.

In a classification problem, it is required to assign static patterns to classes. Choosing the statistical representation of a given pattern is a key issue in any statistical pattern-recognition problem. This is referred to as pre-processing the data, and extracting relevant and discriminant features of the pattern usually involves considerable problem-related expertise. Different choices of features lead to different patterns of the dispersion of distribution and convexity of clusters of examples in the input space, requiring boundaries of different complexity (from linear to highly non-linear). Better pre-processing simplifies the classification task.

2.7.2 Neural Network Building Procedure

In building a neural network classifier, the following stepwise procedure could be considered:

1. Data:
    - Collect a database of representative examples of the classification task to be realised.
    - Split the data into a training set and a test set.
2. Pre-processing:
    - Choose a pattern representation selecting a set of discriminant features and transform the data into appropriate inputs for the network (scaling, standardisation, etc.).
    - Choose a presentation format for the target value(s).
3. Network design, learning and evaluation:
    - Choose a network topology: number of units and connectivity (input, layers, and output).
    - Choose the activation functions to be used.
    - Choose an appropriate learning algorithm for the network.

Estimate the performance of the network (on a validation set or with an information criterion) as a function of its complexity, and eventually try to optimise the architecture (weight decay, pruning by removing sections that provide little contribution, etc.).

- Retain the 'best generalising' network and estimate its performance with the test set.

4. Use and diagnostics:
    - Study the impact of different features on the decision (heuristics).
    - Study the marginality of misclassifications.
    - Revert as necessary to step 2 with other pattern representations or a cleaned database.
    - Develop the trained network for implementation.

The design of a good classifier is highly dependent on the quality of the data available, and no classification paradigm, whether in pattern recognition, machine learning or multivariate statistics, will ever produce an adequate classifier unless the data sample available is large enough to be representative of the population with which the model will be used.

In the time series literature, a time series is defined as a series of observations x_k, and a set of real variables ordered and regularly spaced in time (t=1, 2,…,T). To investigate relationships with past values, the vector of lagged values (x_{t-1}, x_{t-2}... xt_{-n}) lies in the n-dimensional time delay space or lag space. The goal of time series analysis is to extract information from a given time series by building a mathematical model for the data. The aim is to describe the data in terms of, for example, randomness, trends, periodicity or stationarity in order to allow filtering such as smoothing, or the removal of outliers and so on, and finally to forecast future values. The model has to define an appropriate phase space (selecting a set of indicators or variables) from which forecasting or extrapolation results can be constructed. Since time series measurements typically exhibit stochastic

fluctuations and noise, the performance of the model depends on its ability to approximate the assumed underlying structure of the data while ignoring as much as possible of the noise. Neural networks can be seen as a generalisation of classical approaches to time series analysis. They bring an additional capability to model non-linear phenomena and to detect chaotic behaviour. Because of their adaptability, where they can realise a wide variety of mappings with the same topology, they are able of capturing a wide range of structures in the phase space. Interestingly all of the traditional AR models can be implemented using neural networks. A multi-layered network can be used to reproduce any relationship that is represented with a continuous non-linear function.

2.7.3 Financial Forecasting Neural Network Set-up

However, there is no statistically satisfying methodology available for time series modelling with connectionist networks. Usually, a rather heuristic framework is used since neural nets combine complex interactions among various factors. The set-up of a financial forecasting application with neural networks is usually conducted in the following way:

1. Pre-processing
    - Data acquisition
    - Data archiving
    - Data filtering
    - Indicator selection
2. Analysis and forecasting
    - Building the model
    - Learning optimization
    - Static and adaptive learning
    - Selection and testing of the model
3. Trading
    - Post-processing

- Trading scenarios
- Trading room

Decisions in the pre-processing stage relate mainly to the problem of specifying the type and number of indicators assumed to contribute to the underlying process, including the lag structure. Next, a topology for the network needs to be chosen. If feed-forward networks are used, the number of hidden units has to be specified. In order to estimate model parameters, an error criterion has to be selected together with an optimisation (learning) algorithm. Then, diagnostics tools have to be used to check the different properties of the model. Finally, the output of the network has to be interpreted and may be used as input for yet another decision-supporting system.

2.7.4 Experimental Three Layer ANN

Experiments can be constructed for supervised learning; that is, induction-based supervised concept learning. The feed-forward ANN is a simple and popular supervised learning built with perceptron components (nodes) as shown in Figure 2-14.

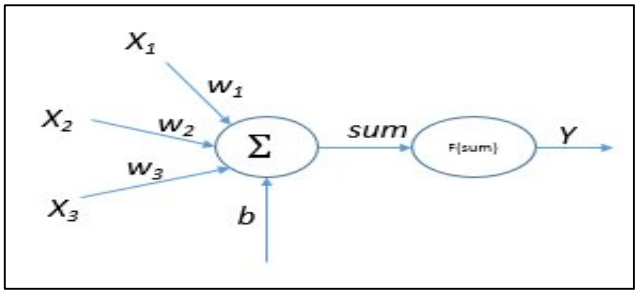


FIGURE 2-14 ANN PERCEPTRON

Here:

$$sum = \sum X_i w_i + b = X_1 w_1 + X_2 w_2 + \cdots + b \quad (2\text{-}14)$$

$$Y = F(sum) \quad (2\text{-}15)$$

The sigmoid (squashing) function shown in Figure 2-15 produces a continuous *Output* in a limited range (0, 1).

$$Y = \frac{1}{1+e^{-sum}} \quad (2\text{-}16)$$

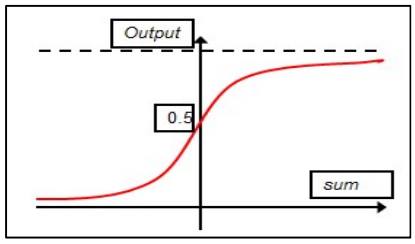


FIGURE 2-15 SIGMOID FUNCTION

There are two asymptotes associated with the sigmoid function:

- Output →1 as sum→∞
- Output →0 as sum→-∞ (2-17)

The variable x_n is the target output and the inputs are x_n, x_{n-1}, x_{n-2}, x_{n-3}, x_{n-4} and x_{n-5}. The configuration of the three-layers ANN is shown in Figure 2-16.

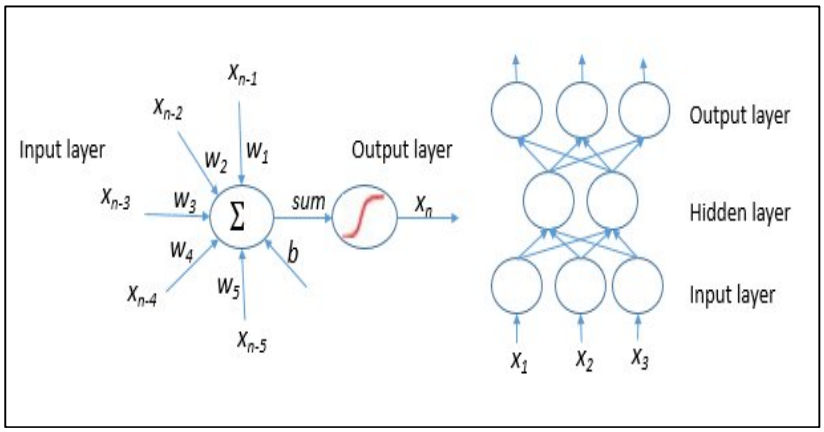


FIGURE 2-16 ANN IMPLEMENTATION

The training of a perceptron uses the following protocol:

1. Start weights at random
2. Present inputs and calculate outputs
3. Find error compared with desired output
4. Adjust weights
5. Repeat steps 2-4 until:
    - Either you have the outputs you want
    - Or the results are not getting any better
6. Then use the network to make predictions

The learning rule is as follows:

- How to adjust the weights?
  - If input > 0:
    - If the answer is too big, reduce the weight.
    - If it is too small, increase it.
  - but if input < 0:
    - If the answer is too big, increase the weight.
    - If it is too small, reduce it.
  - Only increase or decrease the weight by a little at a time

$error = output - target$

$rate = 0.2(for\ example)$

$w_{new} = w_{old} - (error * input * rate)$ for analogue inputs

$w_{new} = w_{old} - (error * (-1) * rate)$ for binary threshold inputs (2-18)

2.7.5 Three Layers ANN Results

The ANN results are shown in Figure 2-17 for data in Table 2-6.

TABLE 2-6 ANN RESULTS

Target		ANN-5	
27/08/2012	187.2	195.61	8.41
28/08/2012	188.95	195.28	6.33
29/08/2012	186.35	194.82	8.47
30/08/2012	183.5	193.73	10.23
31/08/2012	183.25	192.46	9.21
03/09/2012	184.3	191.79	7.49
04/09/2012	181.25	191.22	9.97
05/09/2012	181.95	190.81	8.86
06/09/2012	193.05	192.25	-0.80
07/09/2012	206.4	197.04	-9.36
10/09/2012	207.75	203.19	-4.56
11/09/2012	213.5	208.40	-5.10
12/09/2012	217	211.00	-6.00
13/09/2012	217.95	212.96	-4.99
14/09/2012	229.05	223.11	-5.94
17/09/2012	228	227.66	-0.34
18/09/2012	225.4	227.28	1.88
19/09/2012	225.15	224.91	-0.24
20/09/2012	222.05	219.03	-3.02
21/09/2012	223.75	219.38	-4.37
		output	error
		MSE	6.39
		RMS	2.52

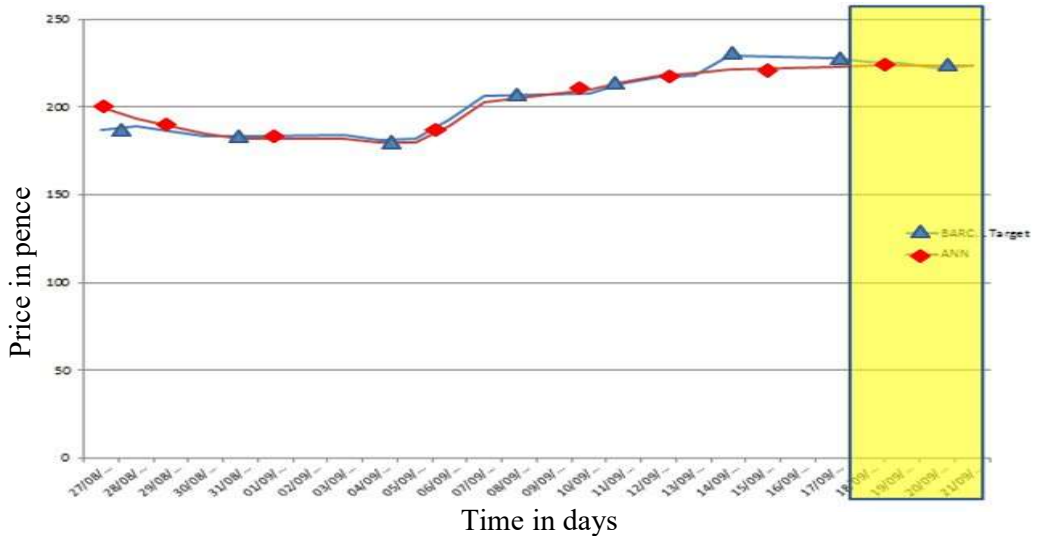


FIGURE 2-17 ANN RESULTS GRAPH

The ANN results are good for both MSE-= 6.39 (about 2.6% of the share price scale) and RMS= 2.52 (about 1% of the share price scale) errors and fitness. The *r2* (graphs similarity, the max value is 1) values are good for the whole data range at 0.91 and are acceptable for the last five days at 0. 78.

2.7.6 Comparison of Three Layers ANN vs. Statistical Linear Regression

The performance of the ANN compared to linear regression is illustrated in Figure 2-18 based on the data shown in Table 2-7.

TABLE 2-7 ANN VS. REGRESSION

Target		ANN - 5		Regression - 5	
27/08/2012	187.2	195.61	8.41	186.13	-1.07
28/08/2012	188.95	195.28	6.33	190.31	1.36
29/08/2012	186.35	194.82	8.47	190.89	4.54
30/08/2012	183.5	193.73	10.23	186.60	3.10
31/08/2012	183.25	192.46	9.21	185.43	2.18
03/09/2012	184.3	191.79	7.49	184.41	0.11
04/09/2012	181.25	191.22	9.97	185.73	4.48
05/09/2012	181.95	190.81	8.86	180.73	-1.22
06/09/2012	193.05	192.25	-0.80	185.44	-7.61
07/09/2012	206.4	197.04	-9.36	198.97	-7.43
10/09/2012	207.75	203.19	-4.56	211.89	4.14
11/09/2012	213.5	208.40	-5.10	209.66	-3.84
12/09/2012	217	211.00	-6.00	223.13	6.13
13/09/2012	217.95	212.96	-4.99	218.30	0.35
14/09/2012	229.05	223.11	-5.94	223.85	-5.20
17/09/2012	228	227.66	-0.34	240.65	12.65
18/09/2012	225.4	227.28	1.88	227.66	2.26
19/09/2012	225.15	224.91	-0.24	235.12	9.97
20/09/2012	222.05	219.03	-3.02	229.62	7.57
21/09/2012	223.75	219.38	-4.37	225.70	1.95
		output	error	output	error
		MSE	6.39	MSE	65.11
		RMS	2.52	RMS	8.06

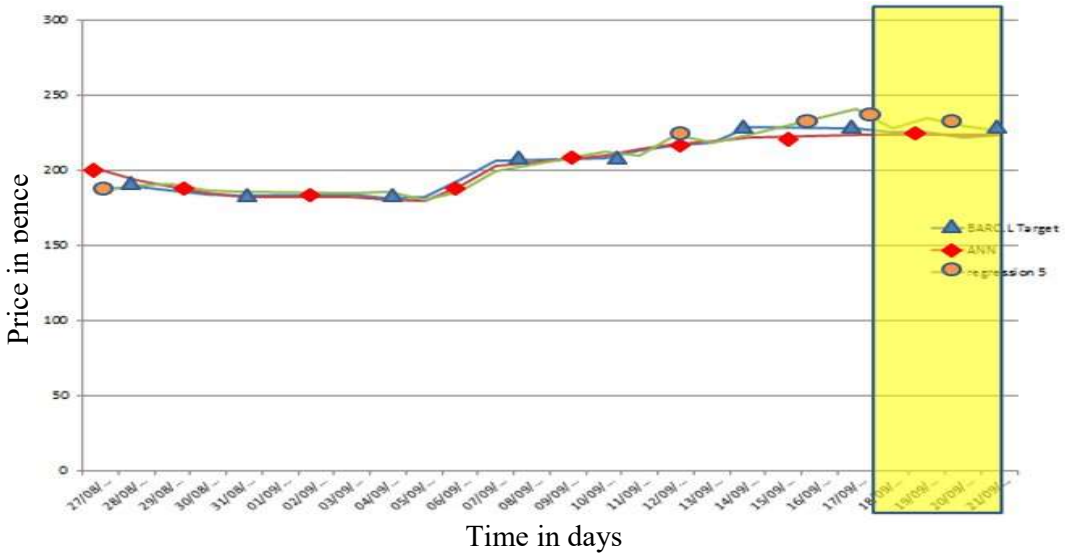


FIGURE 2-18 ANN VS. REGRESSION

There is a significant improvement in performance between linear autoregressive and neural network models, changing from $MSE_{5\text{-}AR}$=65.11 and $RMS_{5\text{-}AR}$=8.06 for the autoregressive model to $MSE_{5\text{-}ANN}$=6.39 and $RMS_{5\text{-}ANN}$ =2.52 for the later model predictions. The ARMA model performance lies between that of the statistical linear and neural network models, though it is sensitive to the randomness of the autoregressive AR component. Furthermore there is a closer ANN fit with the predictions to the actual test data, so there is an improved robustness to variations in input data trend, which makes the neural network model's overall performance superior to that of the autoregressive models. The graph similarity r2 tests results are:

	20-days	5-days
ARMA	0.947165	0.636626651
ANN	0.915519664	0.781246538
LINST	0.944012112	0.688230109

The *r2* (graph similarity, max value is 1) ARMA value is the best for the whole data range at 0.947 and is acceptable for the last five days at 0. 78.

2.8. Summary

The generic learning-generalization methodology could be mapped to forecasting time series autoregressive and neural network regression models.

Correlation between the input model parameters impacts on the generalization of the models and their future performance with unknown data. The correlation graphs between the current value at time the previous time values show a gradually less linear relationship (correlation). The relationship trends decline as well, from one to a slightly declining trend of approximately 0.6~0.8 at the furthest time values.

For models with numerical outputs, standard technical measures such as root mean square error and root mean squared error are the common measures of system performance.

Autoregressive models have a simple mathematical structure. The analysis with a sample model using three and five days input parameters does not give a significant difference in performance (errors values are similar):

- mean square error 67.01 and 65.11
- root mean square error 8.19 and 8.06.
- there is an affinity prediction graph to follow the trend of the input data which is eventually compensated for at the end of the prediction period.

The next step in time series analysis is non-linear modelling such as non-linear autoregressive models or neural networks. The experiments with the neural network model have found that there is a significant improvement in performance compared to that of the linear autoregressive model:

- mean square error from 65.11 to 6.39
- root square error from 8.06 to 2.52
- there is a closer fit between predictions and actual test data.

Overall, the non-linear models such as the neural network model are superior to the linear autoregressive models in performance and robustness. This supports the hypothesis that financial shares time series are non-linear as well.

Chapter 3. Validation of ANN Model for Share Prices

3.1. Overview

This chapter applies formal statistical and non-statistical methods to evaluate the outcomes of the linear regression, artificial neural network and bi-linear regression models.

The objective of the chapter is to justify the use of the ANN for the short-term prediction of share prices, particularly in the banking sector (Petkov, et al., 2012). The assumption is that time series data for financial shares contain significant non-linearity and that the ANN can be utilized effectively.

This chapter covers the following subjects:

- Introduction to short-term trading and long-term investment
- Experiments with linear regression, neural network and bi-linear generalized scalar regression models
- Comparison and evaluation of the experimental results

3.2. Introduction

Share values are represented by a time series of prices over a certain period, such as a day, month or year, and sampled accordingly, usually with a natural time ordering. Time series analysis normally tries to predict future numerical outcomes (such as the closing or opening prices of an individual share) based on past performance. The hypothesis tested is that data over a short period are expected to be more closely correlated than data over a longer period. This encourages the use of short-term range data.

Figure 3-1 below gives a snapshot of the intraday chart of the BARC.L share price on Wednesday 27th June 2012.

FIGURE 3-1 BARC.L 2012-06-27

Another short-term trading timescale, which justifies a one-two week trading strategy during the three months of July to September 2012, is given below in Figure 3-2. For example, prices in the period between September 5th at 181.95p and the 14th at 229.05p, prices are clearly non-linear with an eventual profit of 47.1p compared with the intraday prices of 193.05p and 196.8p, giving 3.75p profit for 27th June.

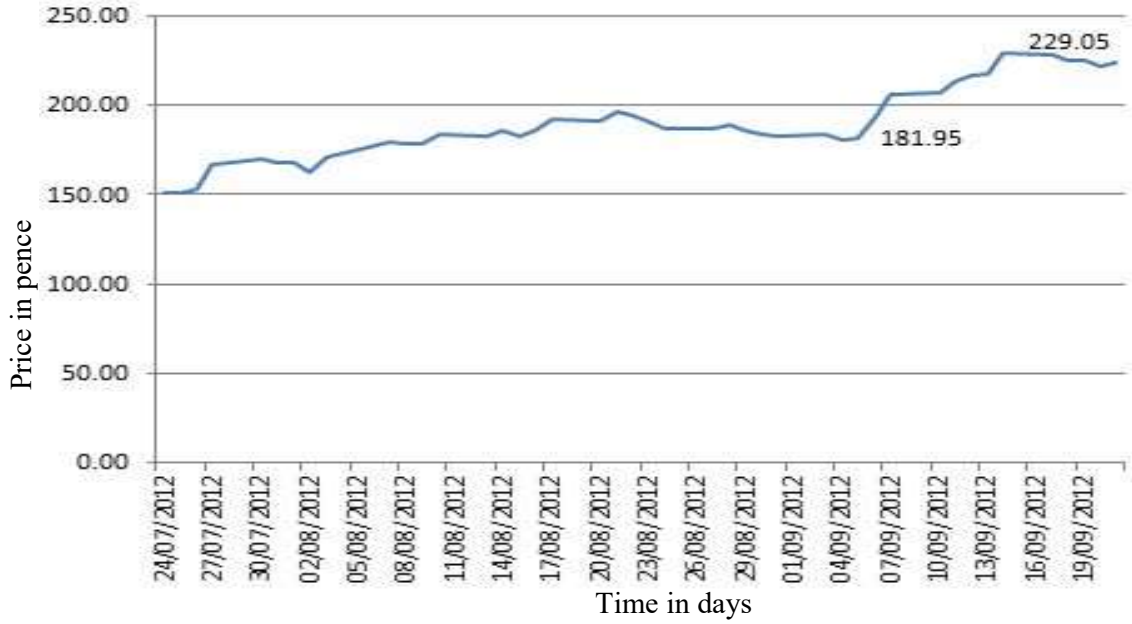


FIGURE 3-2 BARC.L JULY-SEPTEMBER 2012

In contrast, the opposite share dealing strategy is known as long-term investment. Finally, for a long-term investment period from September 2011 to September 2012 as given below in Figure 3-3, it could be argued that both linear and non-linear models could have generally comparable performance considering monthly periods such as from January to March 2012.

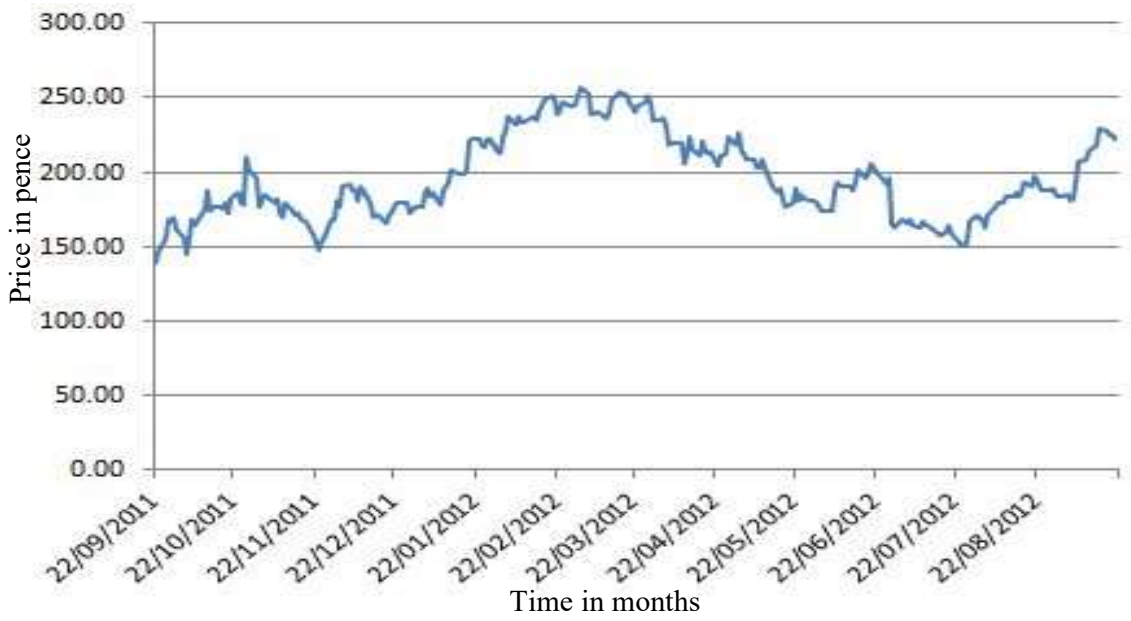


FIGURE 3-3 BARC.L SEPTEMBER 2011 TO SEPTEMBER 2011

The distribution of the share price during the twelve months' period from 22/09/2011 to 21/09/2012 is given below in Figure 3-4.

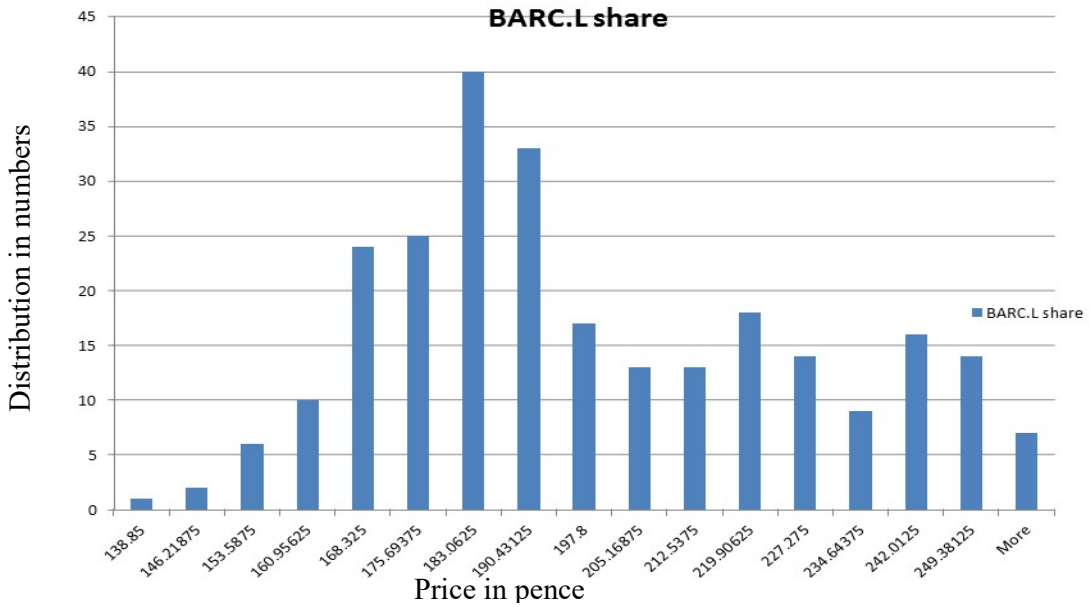


FIGURE 3-4 DISTRIBUTION BARC.L YEAR

For regular dealers, ‘short-term’ usually means daily trading, involving buying in the morning and selling at a profit on the same day. In fact, under normal market conditions, share prices are not expected to change significantly within a day. It would make more sense to aim for successful trading within one to two weeks rather than daily, which would furthermore reduce trading costs.

The underlying models commonly used by traders for analysis are known as technical or chart analysis and are based on linear regression. A novel perception is that share performance is affected by many small non-linear processes and interactions, and slightly different initial conditions could cause very diverse outcomes, hence being non-linear processes (Baestaens, et al., 1994). This encourages the investigation of stock market behaviour that might be explained by non-linear models with commonly expected features such as pattern recognition and generalization abilities. These features are genuine characteristics of artificial neural networks (ANN). Moreover, there is as yet limited , mainly for proprietary internal use, evidence that neural network models have been used or released for the general public and individual traders or available from major providers such as Yahoo Finance.

3.3. Experiments

3.3.1 Hypothesis

The original proposition is that the outcome of experiments would show a significant difference between models based on the linear or non-linear nature of changes in financial share prices. The null hypothesis is that there is no difference between the models.

3.3.2 Dataset

The share datasets that have been used are time series representations of Open, High, Low, Close, Volume and Adjusted Close share prices. The source of data is the Yahoo Finance website (http://finance.yahoo.com/) which gives the prices of US, European and Asian markets in downloadable Excel and “csv” spreadsheet formats. There are options for start

and end dates as well as whether the information is to be summarized by day, week or month.

Examples downloaded for data mining are of the Barclays PLC (BARC.L) share price. Attributes are that the date is represented in DD/MM/YYYY and Open, High, Low and Close values in pence number integers.

The share price working dataset during the one-month period from 27/08/2012 to 21/09/2012 is given in Appendix C: Table 1 and the working dataset of closing share prices used for the models with a time-lag of five days during the one-month period from 27/08/12 to 21/09/12 is given in Appendix C: Table 2. All prices are given in GB pence. The share price chart during the one-month period from 27/08/12 to 21/09/12 is given below in FIGURE 3-5.

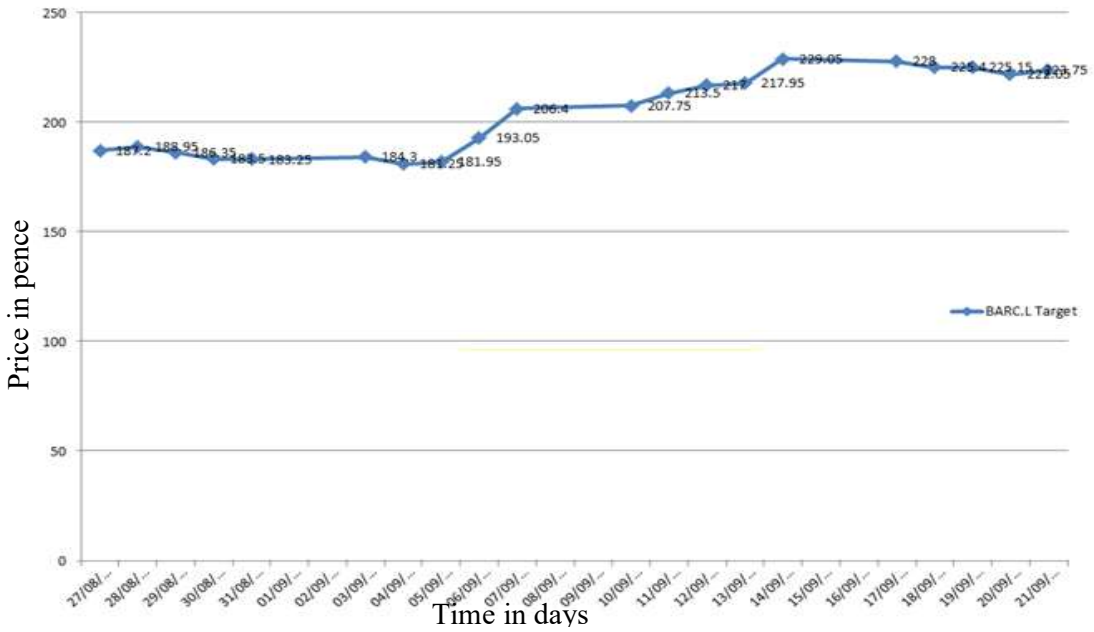


Figure 3-5 BARC.L shares in pence 27/08/12 to 21/09/12

Data from 27/08/2012 to 14/09/2012 is used to train the models, while data from 17/09/2012 to 21/09/2012 is used for model evaluation.

3.3.3 Model Evaluation

To evaluate the models' performance (Roiger, et al., 2003), their results have to be examined by checking the error rate, which for numerical outputs is measured as the mean squared error (MSE):

$$MSE = \sum(x_i - \mu)^2 \tag{3-1}$$

where x_i is the share price value of the i-th share price value of the share's time series and μ is the mean.

For the root mean squared error (RMS), applying the square root reduces the dimensionality of the MSE to that of the actual MSE error. It is generally used as a measure of convergence with the ANN:

$$RMS = \sqrt{\sum(x_i - \mu)^2} \tag{3-2}$$

The mean absolute error (MAE) is less affected by large deviations,

$$MAE = |x_i - \mu| \tag{3-3}$$

Furthermore, if the dataset has a normal bell-shaped distribution, then the confidence intervals can be computed as well (Roiger, et al., 2003), as follows:

$$f(x) = 1/\ (\sqrt{2\pi\sigma})e^{-(x-\mu)^2/2\sigma^2} \tag{3-4}$$

where μ is the mean and σ^2 is the variance, given by:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3-5}$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(\mu - x_i)^2 \tag{3-6}$$

3.3.4 Model Validation

The validation of model performance is conducted by using the significance score *P* value with instance-by-instance pair-wise comparison, given as:

$$P = \frac{|E_1 - E_2|}{\sqrt{\frac{\gamma_{12}}{n}}} \quad (3\text{-}7)$$

$$\gamma_{12} = \frac{1}{n-1}\sum_1^n[(e_{1i} - e_{2i}) - (E_1 - E_2)]^2 \quad (3\text{-}8)$$

where E_1 is the overall error for the ANN model, E_2 is the overall error for the linear model, e_{1i} is the error for instance i for the ANN model, e_{2i} is the error for instance *i* for the linear model and γ_{12} is the joint variance.

To have 95% confidence that the performance of the different models is statistically different, the significance score *P* has to be greater than or equal to 2 (Roiger, et al., 2003).

3.3.5 Linear Regression Analysis

Linear regression curve fitting generalizes a numerical dataset with an equation using one or more input data values (such as the daily close price) to a single output. It attempts to model the variation in a dependent variable y as a linear combination of one or more input variables $x_i, i = 1, 2, .., n$ with coefficients $m_i, i = 1,2, .., n$:

$$y = f(x_1, .., x_n) = m_1x_1 + .. + m_nx_n + b \quad (3\text{-}9)$$

For a time-lag of five days the model is:

$$y = f(x_{n-5}, .., x_{n-1}) = m_1x_{n-1} + m_2x_{n-2} + m_3x_{n-3} + m_4x_{n-4} + m_5x_{n-5} + b \quad (3\text{-}10)$$

where x_{n-1} the closing price at day *n-1* is, x_{n-1} is the closing price at the day before day *n-1*, and so on.

The learning algorithm that is used is the well-known learning gradient descent algorithm (Deboeck, 2003). For the coefficient m_1, for example,

$$m_{1_i} = m_{1_{i-1}} - (error_i * rate * x_{n-1_i}) \quad (3\text{-}11)$$

The calculated model for a time-lag of five days is:

$$y = 0.18 * x_{n-1} + 0.09 * x_{n-2} + 0.01 * x_{n-3} + (-0.09) * x_{n-4} + 0.11 * x_{n-5} + (-0.14) \quad (3\text{-}12)$$

The results of the linear regression for a time-lag of five days are given below in Table 3-1 where the shaded area represents the testing. The performance up to the last five days is good but in the last five days there is a big bump although at the end both graphs match at the final day21/09/2012. Further generalization such as cross-validation would eventually help forecasting. These results are worse than the statistical linear regression as in Table 2-5 due to the superior model tuning algorithm.

TABLE 3-1 LINEAR REGRESSION RESULTS FOR BARCLAYS PLC

Date	Target	Output	Error
27/08/2012	187.2	211.43	24.23
28/08/2012	188.95	174.37	-14.58
29/08/2012	186.35	200.00	13.65
30/08/2012	183.5	184.75	1.25
31/08/2012	183.25	182.87	-0.38
03/09/2012	184.3	182.07	-2.23
...	...	...	...
11/09/2012	213.5	203.44	-10.06
12/09/2012	217	217.76	0.76
13/09/2012	217.95	204.40	-13.55
14/09/2012	229.05	239.93	10.88
17/09/2012	228	247.27	19.27
18/09/2012	225.4	246.55	21.15
19/09/2012	225.15	242.09	16.94
20/09/2012	222.05	235.36	13.31
21/09/2012	223.75	226.72	2.97

The linear regression charts for a time lag of five days are given below in Figure 3-6.

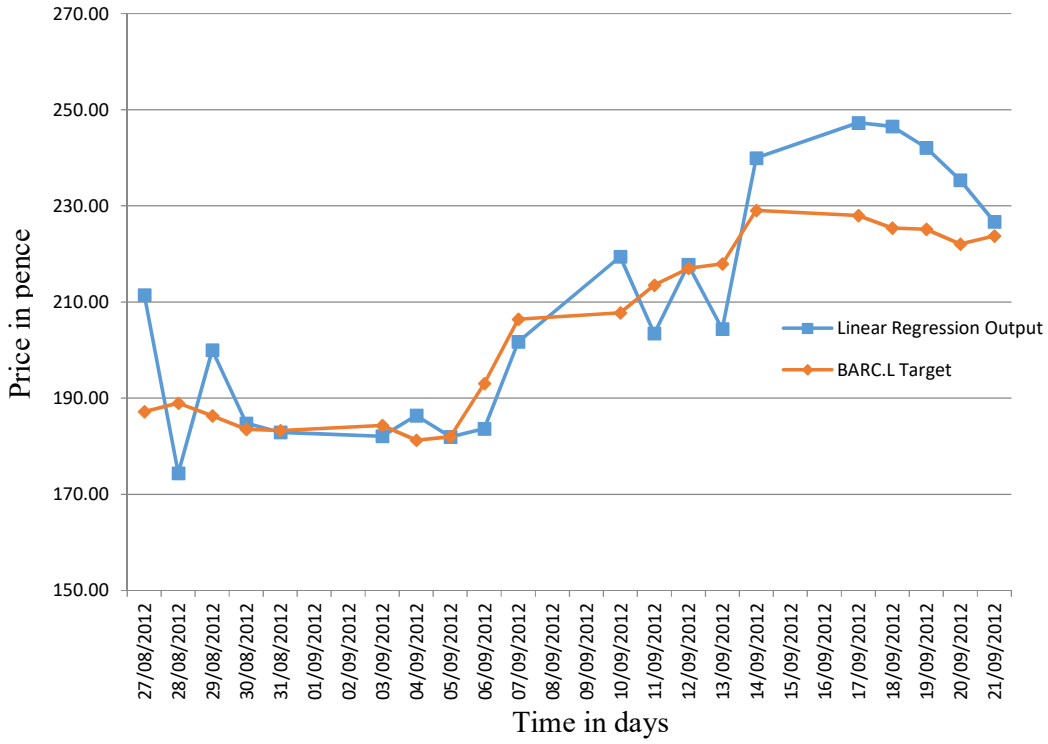


FIGURE 3-6 LINEAR REGRESSION CHART FOR BARC.L 27/08/12 TO 21/09/12

3.3.6 One Layer Artificial Neural Networks

The linear regression model results suggest that an alternative model capable of dealing with implied time series non-linearity may perform better with financial shares forecasting. The neural networks are capable of managing non-linear patterns and therefore can be constructed for supervised forecast learning. The feed-forward perceptron model is a simple and popular supervised learning model for the modelling of time series. A single layer perceptron with five inputs is shown below in Figure 3-7.

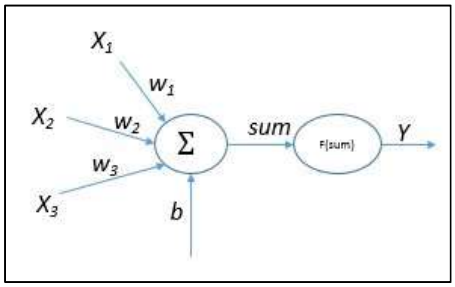


FIGURE 3-7 ANN WITH A SINGLE LAYER PERCEPTRON

The output Y is:

$$Y = \frac{1}{1+e^{-S}} \quad \textbf{(3-13)}$$

$$S = \sum_{n-5}^{n-1} w_i x_{n-i} + b \quad \textbf{(3-14)}$$

The ANN learning algorithm that is used in this study is the well-known learning gradient descent algorithm (Deboeck, 2003). It is intuitive iterative optimization algorithm to find a local minimum of a cost function using gradient descent by takings steps proportional to the negative of the gradient. For the coefficient w_1, for example:

$$w_{1_i} = w_{1_{i-1}} - (err_{\ \ i} * rate * x_{n-1_i}) \quad (3\text{-}15)$$

The calculated model for a time-lag of five days is:

$$S = 0.91x_{n-1} + 0.59x_{n-2} + 0.23x_{n-3} + (-0.10)x_{n-4} + 0.33x_{n-5} + 2.85 \quad (3\text{-}16)$$

The results for the single one-layer ANN for a time lag of 5-days are given in Figure 3-8 and Table 3-2, where the shaded area represents the testing. The graphs generally follow the same movement patterns with a closer match at the final five days, which is the opposite to the linear regression. These results are worse than the results for a three-layer ANN as in Table 2-6. Target and ANN model output charts for 5-days are given below in Figure 3-8.

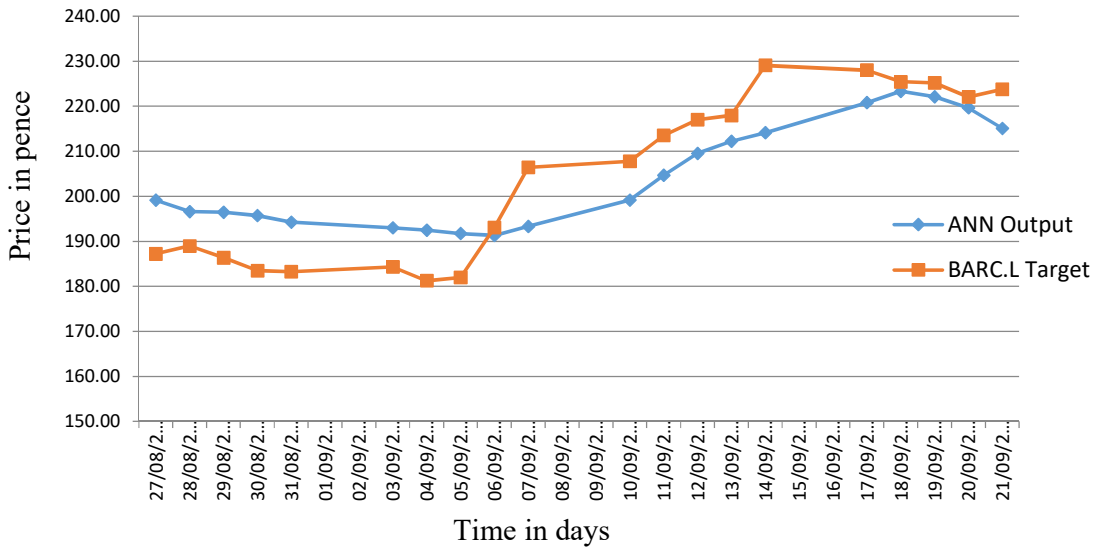


FIGURE 3-8 ANN RESULT GRAPH FROM 12/08/27 TO 12/09/21

TABLE 3-2 ANN RESULTS FOR ONE MONTH BARCLAYS PLC

Date	Target	Output	Error
27/08/2012	187.2	199.14	11.94
28/08/2012	188.95	196.60	7.65
29/08/2012	186.35	196.45	10.10
30/08/2012	183.5	195.70	12.20
31/08/2012	183.25	194.27	11.02
...	....	....	....
10/09/2012	207.75	199.16	-8.59
11/09/2012	213.5	204.65	-8.85
12/09/2012	217	209.50	-7.50
13/09/2012	217.95	212.20	-5.75
14/09/2012	229.05	214.10	-14.95
17/09/2012	228	220.77	-7.23
18/09/2012	225.4	223.27	-2.13
19/09/2012	225.15	222.08	-3.07
20/09/2012	222.05	219.65	-2.40
21/09/2012	223.75	215.07	-8.68

3.3.7 Bi-linear Regression Model

It is of specific interest to compare the ANN with an alternative non-linear type of model such as a bi-linear scalar regression model for data fitting, such as:

$$y = \sum_1^n a_i x_{n-i} + \sum_n^n \sum_n^n b_{i,j} x_{n-i} x_{n-j} + Noise \quad (3\text{-}17)$$

The learning algorithm that is used is the well-known learning gradient descent algorithm (Deboeck, 2003). For the coefficient a_1, for example:

$$a_{1_i} = a_{1_{i-1}} - (error_i * rate * x_{n_{i-1}}). \quad (3\text{-}18)$$

For the joint coefficients, both inputs are used in the learning algorithm. For the coefficient b_{12}, for example:

$$b_{1_i} = b_{1_{i-1}} - (error_i * rate * x_{n-1_i} * x_{n-2_i}). \quad (3\text{-}19)$$

The generic model includes noise as well but, for consistency with the use of the ANN model, this is omitted from the model used for the experiments. For a time lag of five days, the model is;

$$y = a_1 x_{n-1} + a_2 x_{n-2} + a_3 x_{n-3} + b_{12} x_{n-1} x_{n-2} + b_{13} x_{n-1} x_{n-3} + b_{23} x_{n-2} x_{n-3} \quad (3\text{-}20)$$

The calculated model for a time lag of five days is:

$$y = 0.18 x_{n-1} + 0.02 x_{n-2} + (-0.008) x_{n-3} + 0.017 x_{n-1} x_{n-2} + 0.01 x_{n-1} x_{n-3} + (-0.002) x_{n-2} x_{n-3} \quad (3\text{-}21)$$

The results for a time lag of 5-days are given in Figure 3-9 and Table 3-3 where the shaded area represents the testing. Overall very good performance resulted in a close graphs match over the whole period and almost overlapping at the final five days. The results are encouraging although further cross-validation generalization is recommended. So far it looks the best model fit to this dataset. The bi-linear charts are given below in Figure 3-9.

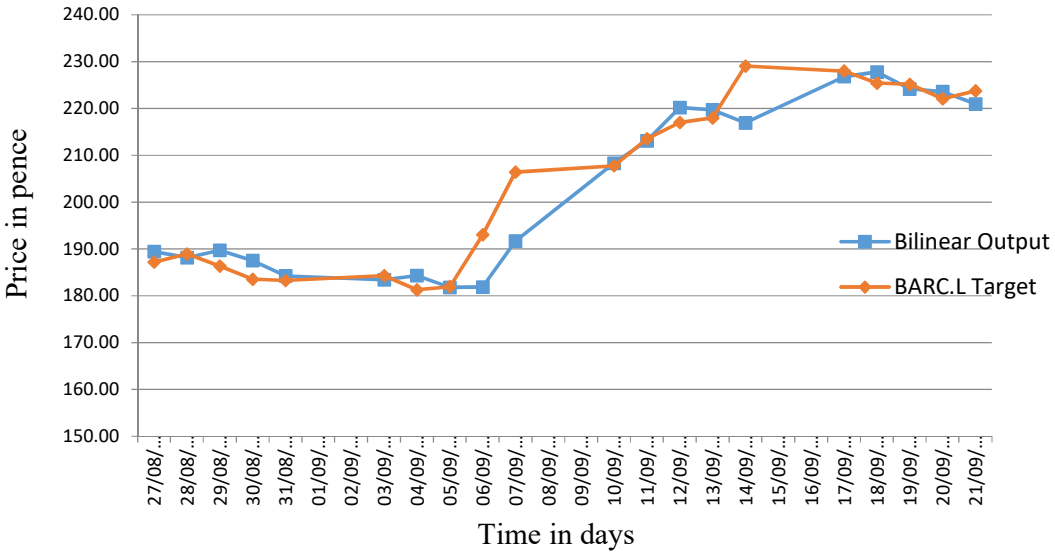


FIGURE 3-9 BI-LINEAR RESULT GRAPH FOR BARC.L

TABLE 3-3 BI-LINEAR RESULTS FOR BARCLAYS PLC

Date	Target	Output	Error
27/08/2012	187.2	189.43	2.23
28/08/2012	188.95	188.15	-0.80
30/08/2012	183.5	187.50	4.00
31/08/2012	183.25	184.24	0.99
03/09/2012	184.3	183.43	-0.87
05/09/2012	181.95	181.79	-0.16
06/09/2012	193.05	181.87	-11.18
07/09/2012	206.4	191.70	-14.70
10/09/2012	207.75	208.31	0.56
11/09/2012	213.5	213.05	-0.45
12/09/2012	217	220.18	3.18
13/09/2012	217.95	219.73	1.78
14/09/2012	229.05	216.90	-12.15
17/09/2012	228	226.79	-1.21
18/09/2012	225.4	227.79	2.39
19/09/2012	225.15	224.16	-0.99
20/09/2012	222.05	223.57	1.52
21/09/2012	223.75	220.96	-2.79

3.4. Comparison of the Experimental Results

The evaluation of the results of the linear regression model for a time lag of five days shows that MSE=258.31and RMS=16.07.

The evaluation of the results of the ANN regression model for a time lag of five days shows that MSE=29.47 and RMS=5.42.

The evaluation of the results of the bi-linear regression model for a time lag of five shows that MSE=3.65 and RMS=1.91.

These results are worse than the results for statistical linear regression (Table 2-5 MSE=65.4 and RMS=8.06) and the three-layer ANN (Table 2-6 MSE=6.39 and RMS=2.52) due to superior statistical regression tuning and layer architecture for the ANN. The point of this validation is to justify the non-linear hypothesis and the models are simplified to facilitate the interpretation of the results.

The results demonstrate significantly better performance of the ANN model compared to linear regression, as the values of MSE and RMS are respectively 9 times and 3 times better (lower). The best performance is achieved with the bi-linear model, as MSE and RMS values are 8 times and 3 times better respectively than the corresponding ANN errors and furthermore almost 70 times and 8 times better than the linear regression.

The calculated significant difference ratio P for the experimental results with instance-by-instance pair-wise comparison between ANN and linear regression is P=6.58 with a joint variance γ_{12}= 34.88, which, therefore, gives 95% statistical confidence in the comparison of the models' performance. The calculated ratio P for the experiments with instance-by-instance pair-wise comparison between the ANN and bi-linear regression model is P=3.85, which, therefore, gives 95% statistical confidence in the comparison of these models' performance.

3.5. Summary

The performance of the ANN model is compared to that of a linear regression model. Non-linearity is shown by deduction via a comparison of experimental results using the ANN and linear regression models. Furthermore, the ANN model is compared to another non-linear type of model, the bi-linear model. Experiments are conducted based on real monthly (four-week) datasets, and the performance of the models is formally evaluated.

The non-linear models are likely to be a better choice than a traditional linear regression model for short-term trading, and furthermore the bi-linear model outperforms the ANN. Experiments have been conducted with a single-layer in order to clearly compare the ANN with linear regression models. The results for mean squared error (MSE) and root mean square (RMS) are shown below:

- For the linear regression model, MSE=258.31and RMS=16.07.
- For the ANN regression model, MSE=29.47 and RMS=5.42.
- For the bi-linear regression model, MSE=3.65 and RMS=1.91.

It is expected that a multilayer ANN would improve the results further. However, the interpretation of the ANN model results is more difficult and comparison with the linear model is less straightforward.

The validation of the ANN model for share price investigations has found that non-linear models are likely to be a better choice than traditional linear regression for short-term trading. The conclusions are positive with good statistical confidence, encouraging further experimentation such as considering further generalization with experiments using cross-validation, inclusion of a noise component in the model, or other methods for choosing model parameters.

Chapter 4. Stochastic Share Price Model

4.1. Overview

This chapter models the behaviour of financial share prices, deriving an analytical continuous-time model considering stochastic calculus and Wiener processes and volatility.

It is often stated that asset prices must move randomly, which is due to the efficient market hypothesis. There are several different forms of this hypothesis with different restrictive assumptions, but they all basically say two things (Wilmott, et al., 1996):

- Past history is fully reflected in the present price, which does not hold any further information;
- Markets respond immediately to any new information about an asset.

Thus, the modelling of asset prices is really about modelling the arrival of new information which affects the price. Given the two assumptions above, a Markov process represents unanticipated changes in an asset price (Wilmott, et al., 1996). The movement of share prices is described as a random walk, which underlines the notion that future prices cannot be predicted with certainty. This implies that any model must include a degree of unpredictability.

This chapter covers the following subjects:

- Brownian motion and the Wiener process
- Analytical modelling
- Market and normal distributions
- Stochastic model predictions summary
- Volatility
- Historic volatility
- Online volatility indices

4.2. Introduction

A random walk is defined as a process relating the value of a variable S_n at time n to its previous position according to the rule:

$$S_n = S_{n-1} + W_n \tag{4-1}$$

where $W_i, i = 1, 2, \dots, n,$ are independent and identically distributed random variables.

A risk-neutral valuation is one which values an investment solely via the present value of its return. Financial share models assume that geometric Brownian motion turns all investments involving buying and selling into fair bets (Wilmott, et al., 1996). For this reason, these valuations are called risk-neutral valuations. Brownian motion means that, if *S(t)* is the price of the security at a time t, then, for any price history up to time *t*, the ratio of the price at a specified future time ($t + T$) to the price at time *t* has a log-normal distribution. The mean and variance parameters will be normal random variables with mean $t\mu$ and variance $t\sigma^2$. Black and Scholes (1973) showed, that given the assumption that price changes follow geometric Brownian motion, there is a single price that does not allow an idealized trader to follow a strategy that will result in a sure profit in all cases. There will be no certain profit (i.e., no arbitrage). Price volatility is a rate at which the price of a security increases or decreases for a given set of returns; that is, the degree of variation of a trading price series over time. In addition, the price depends only on the variance parameter σ of the geometric Brownian motion. Because parameter σ is a measure of the volatility of the security, it is often called the volatility parameter.

Share prices move up and down throughout the working day and they are recorded and updated every 15 minutes. The latest price will either have moved up or down from, or stayed the same as, the previous price. If we record the price of a commodity over a long period of time we get a graph plot such as the one shown in

Figure 4-1. This graph refers to the closing price of Barclays bank shares over a period of about 11 months in 2014-15.

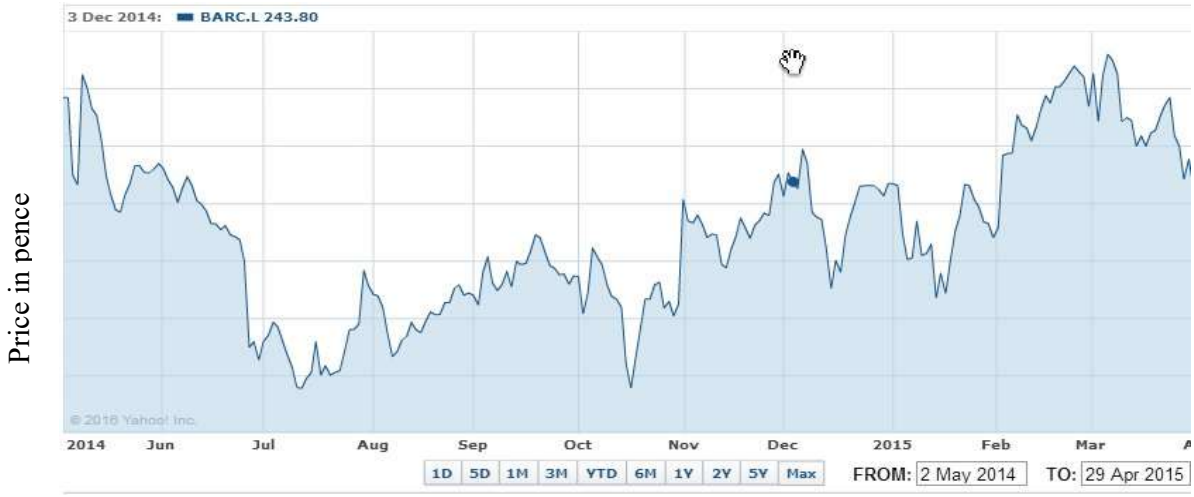


FIGURE 4-1 BARCLAYS PLC SHARE PRICE 02/05/2014 TO 29/04/2015

The jumps in its value are independent of each other and the share price moves irregularly with time with no evidence of a trend as the series progresses. A later realisation of the same share price is shown below in Figure 4-2. The graph has a similar appearance to the one above, illustrating this property of the process.

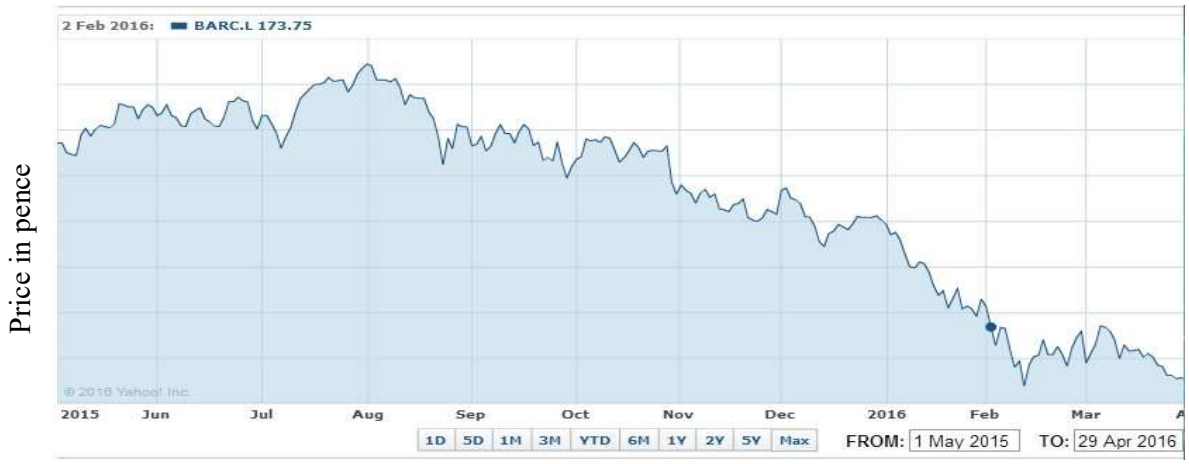


FIGURE 4-2 BARCLAYS PLC SHARE PRICE 01/05/2015 TO 29/08/2016

This means that, if S_n is known at time n, then further information before time n is not relevant to its future values. The random process has the property that its future values depend on its present value and not in any way on its past values.

We shall denote the value of the share price at time *t* as $S_t, t \geq 0$, where the value of the share at t = 0 is zero, $S_0 = 0$. If the price moves up or down, it is equally likely, then, for each value of *t,* that the random variable S_t should have a symmetrical distribution with a mean of zero. Also, the change in its value from time s to s + t denoted by $S_{s,s+t}$is given by:

$$S_{s,s+t} = S_{s+t} - S_s \tag{4-2}$$

This should have the same distribution as $S_{0,t} = S_t$ and be independent of historical values of S_t before time *t*. This means that S_t is a stationary process and is continuous.

These three conditions of symmetry, stationarity and continuity lead to the random process called Brownian motion or the Wiener process. This is formally defined as follows. The Wiener process or Brownian motion is defined as a random process $S_t, t \geq 0$, with $S_0 = 0$ such that the following conditions hold:

1. Every increment $S_{s,s+t}$ is normally distributed as$N(0, \sigma^2 t)$, where σ^2 is constant.
2. For every pair of disjoint intervals (t_1, t_2) and (t_3, t_4), the increments $S_{1,2}$ and $S_{3,4}$ are independent random variables following the above distribution.
3. S_t is continuous.

From property 1, it follows that $S_t \sim N(0, \sigma^2 t)$. The parameter σ is called the volatility parameter and is a measure of the standard deviation.

We want to model the corresponding return on the asset. Suppose that at time *t* the asset price is *S(t)*. Let us consider a small subsequent time interval *dt*, during which *S* changes to (*S* + *dS*).

The anticipated return is based on the return on money invested in a risk-free bank account. The most common model of share prices decomposes the return into: deterministic and random parts. The deterministic and predictable part gives a contribution μdt to the return dS/S, where the mean μ is a measure of the average rate of growth of the asset price, also known as drift. In simple models, μ is taken to be a constant. The deterministic part μSdt corresponds to the return on money invested in a risk-free bank where μ is the interest rate. In more complicated models μ can be a function of S and t.

The second contribution to dS/S models the random change in the asset price in response to external effects. It is represented by a random sample drawn from a normal distribution with a mean of zero and it adds a term σdW to dS/S. The parameter σ is called volatility, which measures the standard deviation of the returns. The quantity dW is the sample from a normal distribution which contains the randomness that is certainly a feature of asset prices. The random part σSdW is the random change in the asset price in response to external effects such as unexpected news. This is known as a Wiener process.

4.3. Analytical Model

Putting these contributions together, we obtain a stochastic differential equation as a mathematical representation of a simple asset price:

$$\frac{dS}{S} = \mu dt + \sigma dW \tag{4-3}$$

where S is the asset price, μ is the average rate of growth, σ is the volatility which is measured by the standard deviation of the returns, and dW which contains the randomness known as a Wiener process.

Normally, σ and μ are variable functions of time and can depend on other things as well. However, for a short period of time, σ, μ =const. Constant volatility means constant noisiness in the share price, while constant drift means a constant trend of an increase or decrease in the share price.

The stochastic differential equation is solved as follows:

$$\frac{dS}{S} = \mu dt + \sigma dW \tag{4-4}$$

Despite uncertainty in share prices, a certain determinism can be derived in a small time period using Ito's lemma (Wilmott, 1996) :

$$d(lnS) = \frac{dS}{S} - \frac{1}{2}\sigma^2 dt \tag{4-5}$$

Replacing $\frac{dS}{S}$ gives:

$$d(lnS) = \mu t + \sigma dW - \frac{1}{2}\sigma^2 dt \tag{4-6}$$

Then, integrating:

$$\int d(lnS) = \int \mu t + \int \sigma dW - \frac{1}{2}\int \sigma^2 dt \tag{4-7}$$

$$lnS = \sigma W + \left(\mu - \frac{1}{2}\sigma^2\right)t + + C \tag{4-8}$$

The initial conditions are $t = 0,\ S_0 \neq 0$ and $W_0 = 0$ which is giving $C = ln\ _0$.

Consequently

$$ln\frac{S}{S_0} = \left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W \tag{4-9}$$

therefore,

$$S = S_0 e^{\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W} \tag{4-10}$$

If we assume that volatility $\sigma = 0$, we can ignore the randomness of the asset. The accuracy of this assumption depends on the observation period and the relative percentage of variations. Usually in normal market conditions for one-two trading weeks daily trading, the variations are small compared to the price. When μ is constant, the equation above can be solved exactly to give:

$$S = S_0 e^{\mu t} \tag{4-11}$$

where S_0 is the value of the asset at $t = 0$. Thus, if $\sigma = 0$, the asset price is totally deterministic and we can predict the future price of the asset with certainty.

This equation is a particular example of a random walk, with the probability density function represented by a skewed bell-shaped curve known as the log-normal distribution. This random walk is known as a log-normal random walk.

4.4. Methodology

The results of this investigation can be summarized as follows:

1. Datasets for share prices from 2015-06-01 to 2015-10-04:
    a. 2015-06-01 to 2015-10-04, the whole dataset.
    b. 2015-06-01 to 2015-08-31,the training dataset.
    c. 2015-08-31 to 2015-10-04, the testing dataset.
    d. Share prices for selected companies were analysed: in the financial sector: BARC.L, HSBA.L, RBS.L and LLOY.L and in the retail sector: TSCO.L, MRW.L, MKS.L and SBRY.L
    e. Data is sorted from the oldest to the newest.
2. The share price ratio $R_i = \frac{S_i}{S_{i-1}}$ logarithm $ln(R_i)$ was calculated.
3. A check for normality was performed (normal distribution) of the $ln(R)$; that is, for a log-normal distribution. The Kolmogorov - Smirnof (Jistel, et al., 1997) test was performed.
4. Descriptive statistics for the training dataset from 2015-06-01 to 2015-08-31 were calculated; namely mean μ and standard deviation σ.
5. A normalization is performed for time units for a year replacing μ with m and σ with s where $m = 252\mu$ and $s = \sigma\sqrt{252}$, taking into account the number of 252 trading days in the UK stock market, where $\frac{1}{\sqrt{252}}$ is the length of one trading day measured in years. The predicted price is:

$$S_{80} = S_0 * e^{80*(m-\frac{s^2}{2})} \qquad (4\text{-}12)$$

6. The results are plotted for volatility

7. Different distributions such as Brownian motion or binomial models are tested.

4.5. Datasets and Results

Share prices and results for selected retail and financial sector companies are shown for the financial sector: BARC.L Appendix C: Figure 1, HSBA.L Appendix C: Figure 3, RBS.L Appendix C: Figure 2 and LLOY.L Appendix C: Figure 4.

And for the retail sector: TSCO.L Appendix C: Figure 5, MRW.L Appendix C: Figure 7, MKS.L Appendix C: Figure 6 and SBRY.L Appendix C: Figure 8

Stochastic model weekly and monthly predictions show not very good fits with the target and little variation, as shown for BARC.L in Figure 4-3 below

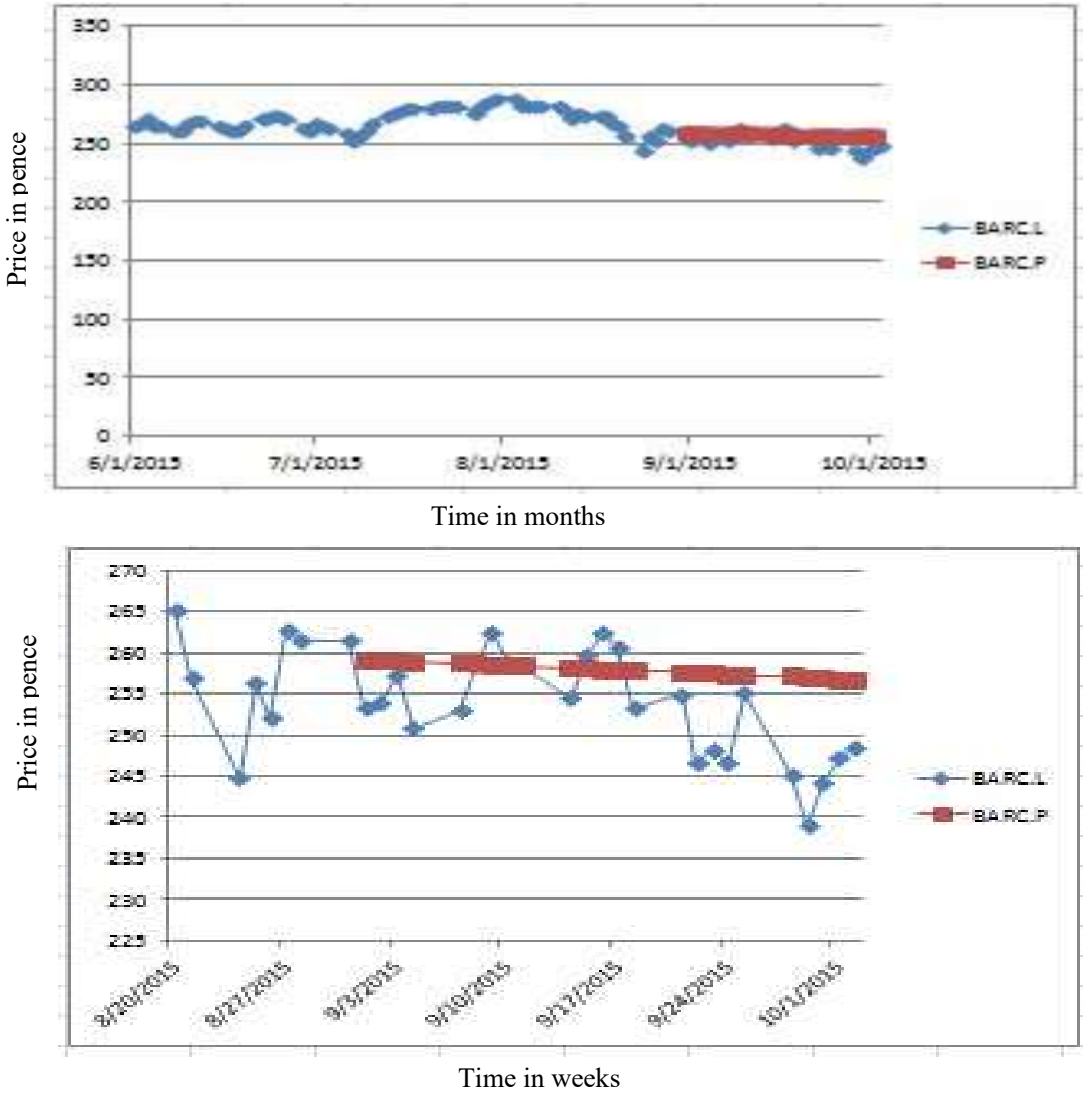


FIGURE 4-3 STOCHASTIC MODEL BARC.L WEEKLY AND MONTHLY PREDICTIONS

The stochastic model predictions are summarized in Table 4-1 and Table 4-2.

TABLE 4-1 FINANCIAL SECTOR PREDICTIONS

		HSBA.L (65 samples)	LLOY.L (65 samples)		BARC.L (65 samples)	RBS.L (65 samples)
Daily Mean	m=AVERAGE	-0.28%	-0.2%		-0.03%	-0.02%
Annualised mean	$\mu = mean * 252$	-71.78%	-52.76%		-7.22%	-5.26
Volatility	s=STDEV	1.53%	1.2%		1.68%	1.64%
Annualised volatility	$\sigma = s * \sqrt{252}$	24.30%	19.19%		26.66%	26.18
error	$\sqrt{\sum error^2}$	10.19%	7.36%		13.36%	17.45%

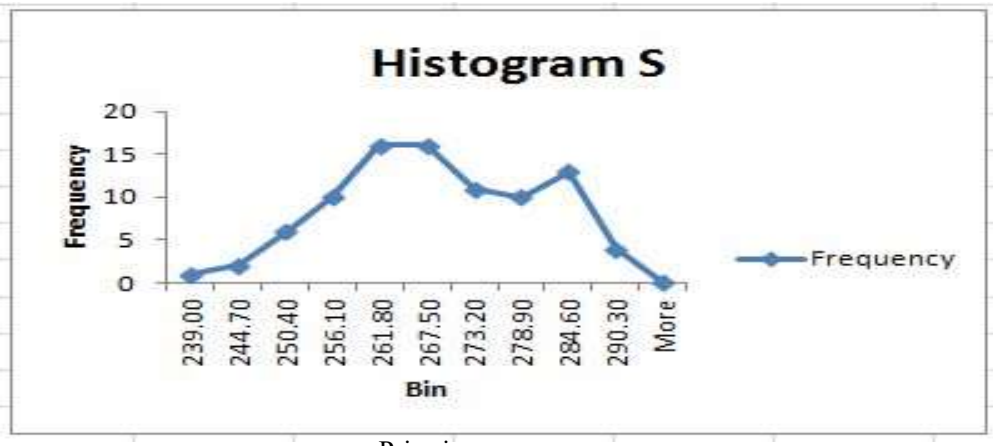
TABLE 4-2 RETAIL SECTOR STOCHASTIC MODEL PREDICTIONS

		MKS.L (65 samples)	TSCO.L (65 samples)		MRW.L (65 samples)	SBRY.L (65 samples)
Daily Mean	m=AVERAGE	-0.17%	-0.14%		-0.01%	-0.03%
Annualised mean	$\mu = mean * 252$	-42.55%	-35.69%		-2.07%	-8.38%
Volatility	s=STDEV	1.55%	1.61%		1.50%	1.48%
Annualised volatility	$\sigma = s * \sqrt{252}$	24.61%	25.48%		23.85%	23.47%
error	$\sqrt{\sum error^2}$	9.79%	10.36%		9.66%	9.45%

The analysis of the results as shown in the summary tables shows consistent market annualized volatility in the range 19.19%-26.66% and consequently the expected prediction error is similar in the range of 7.36% - 17.45% with better retail sector performance between 9.45% to 10.36%.

4.6. Market and Normal Distributions

A comparison check for normal distribution is shown for BARC.L in Figure 4-4 and for the generated normal distribution in Figure 4-5, confirming the BARC.L normal distribution hypothesis.



Price in pence

FIGURE 4-4 DISTRIBUTIONS HISTOGRAMS BARC.L

Normalized random number

FIGURE 4-5 DISTRIBUTIONS HISTOGRAMS NORMAL GENERATOR

4.7. Volatility

Annual volatility ($\tau = \frac{1}{252}$, 252 trading days) from historic data (Hull, 2008) for three months daily closing price data is calculated for AV.L (Aviva), BARC.L (Barclays), BNC.L (Banco Santander), BP.L (BP), HSBA.L (HSBC), LLOY.L (Lloyds) RBS.L (Royal Bank of Scotland), STAN.L (Standard Chartered), TSCO.L (Tesco) and VOD.L (Vodafone). The investigation is conducted mostly for BARC.L and the other shares considered are just for three months, equivalent to 66 days. The closing daily share prices are downloaded for the period 22/09/2011 to 22/09/2012. It is assumed that the length of

time interval (τ) for the three months' data is equivalent to 66 days, for two months 44 days and one month 22 days.

$$u(i) = ln\ \frac{s(i)}{s(i-1)}\Big) \tag{4-13}$$

$$\sigma = \sqrt{252} * STDEV\ [u(i) :\ u(i-\tau)] * 100\ \% \tag{4-14}$$

The charts (axis *x*, time) are presented below backwards in time, where 1 corresponds to the most recent day 21-09-2012 and the 263-rd point to 22-09-2011, as shown in Figure 4-6.

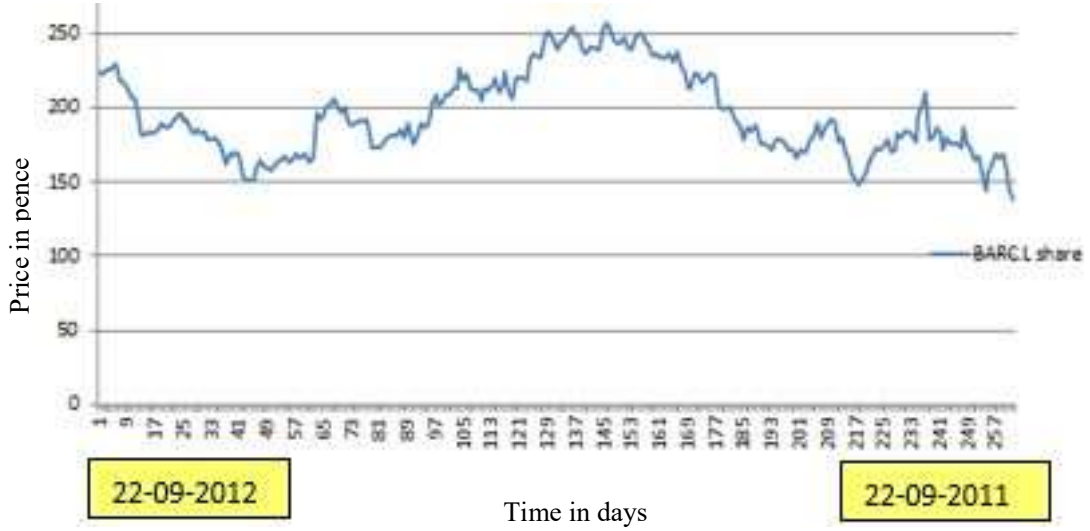


FIGURE 4-6 VOLATILITY DATASET BARC.L

The share trend chart for each day and the whole period is produced using the Excel SLOPE built-in function with a 14 days' backward window as displayed in Figure 4-7.

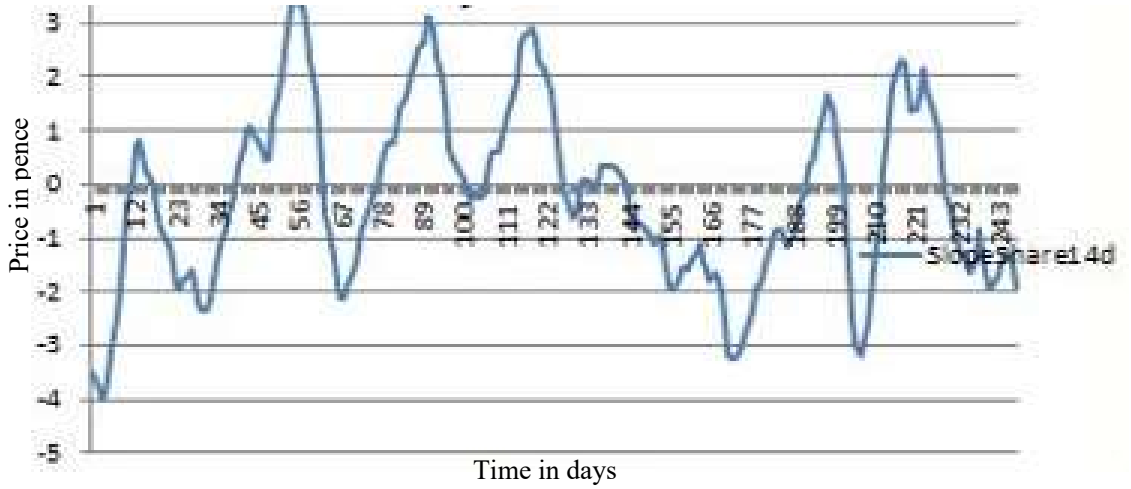


FIGURE 4-7 EXCEL SLOPE FOR 14 DAYS

4.8. Historic Volatility

The historic volatility of the Barclays share price on 22-09-2012 was 51.49%. The historic volatility chart is shown backwards for every day from 22-09-2012 to 03-01-2013 for 66 days (three months) time lag in Figure 4-8.

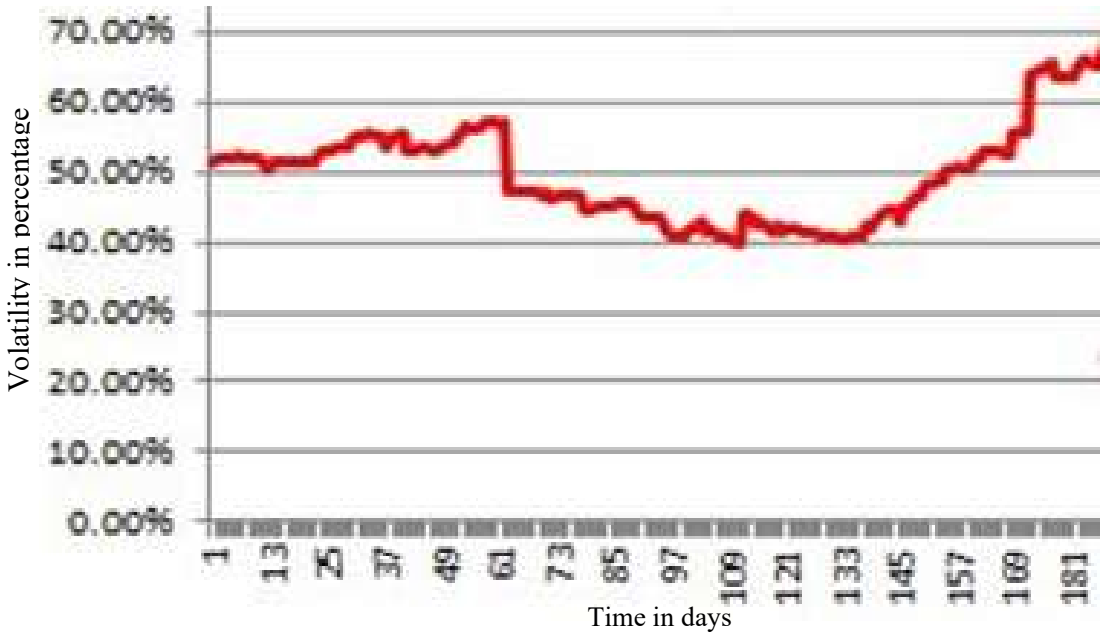


FIGURE 4-8 HISTORIC VOLATILITY BARC.L FOR 66 DAYS FROM 22-09-2012

The historic volatility slope is shown in Figure 4-9.

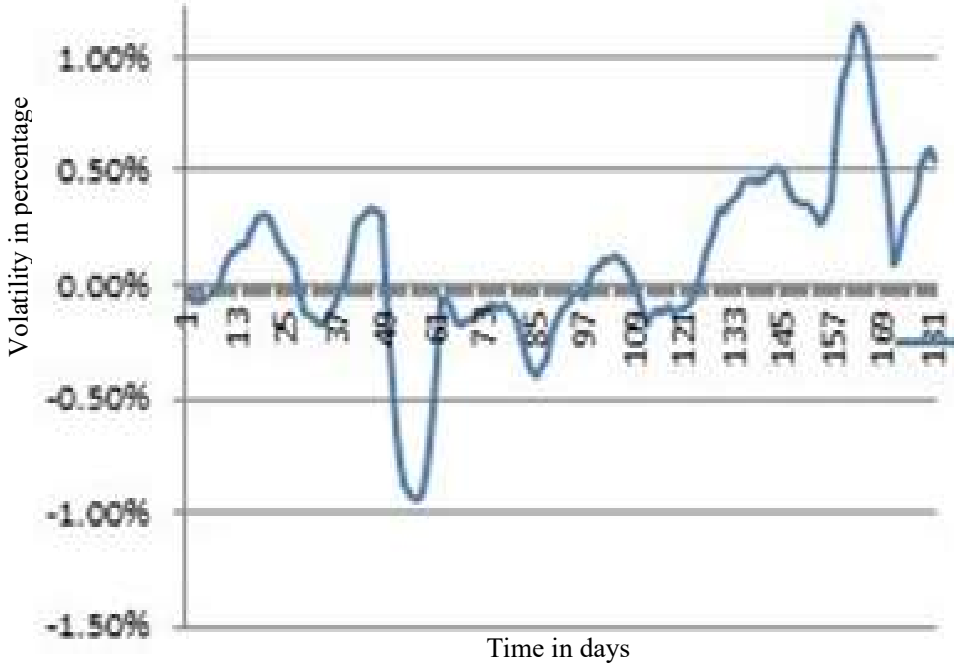


FIGURE 4-9 VOLATILITY SLOPE BARC.L FOR 66 DAYS

Volatility is calculated for each day with STDV windows of 12 days (two weeks), 22 days (one month), 44 days (two months) and 66 days (three months), as shown in Figure 4-10.

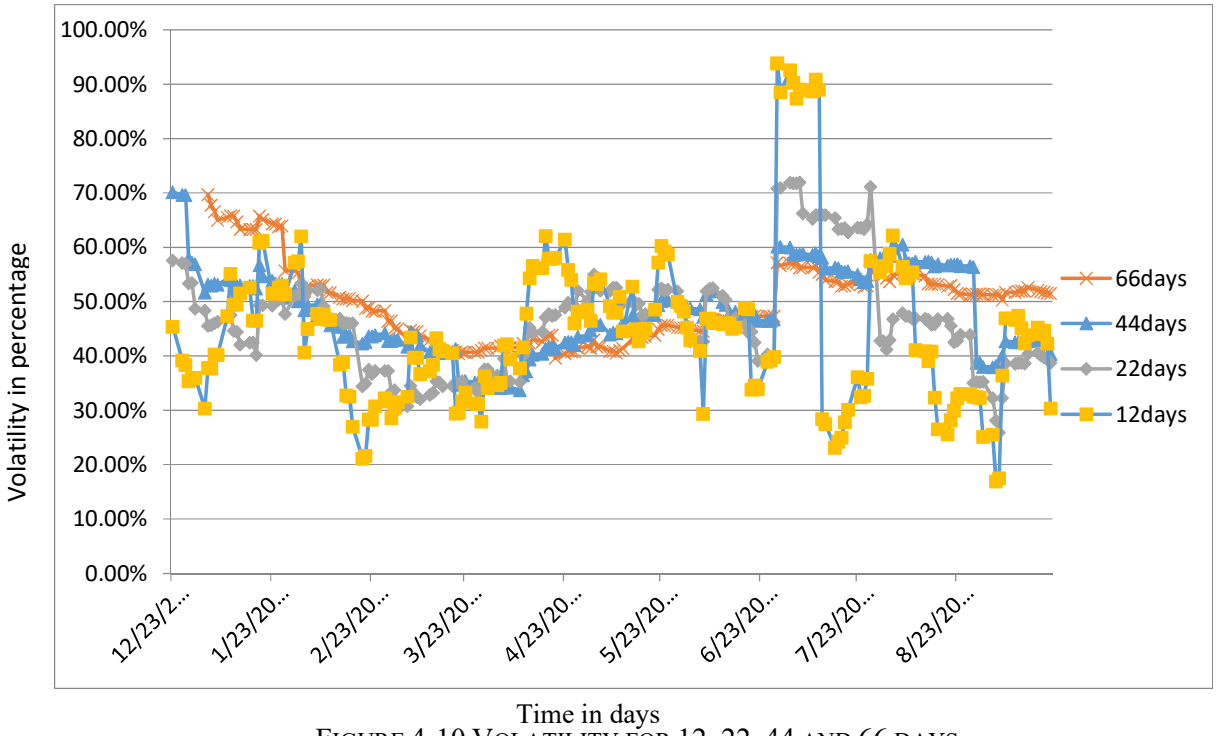


FIGURE 4-10 VOLATILITY FOR 12, 22, 44 AND 66 DAYS

The volatility trend-slopes for 44 days (two months) and 66 days (three months) with a 14 days' backward window are shown in Figure 4-11.

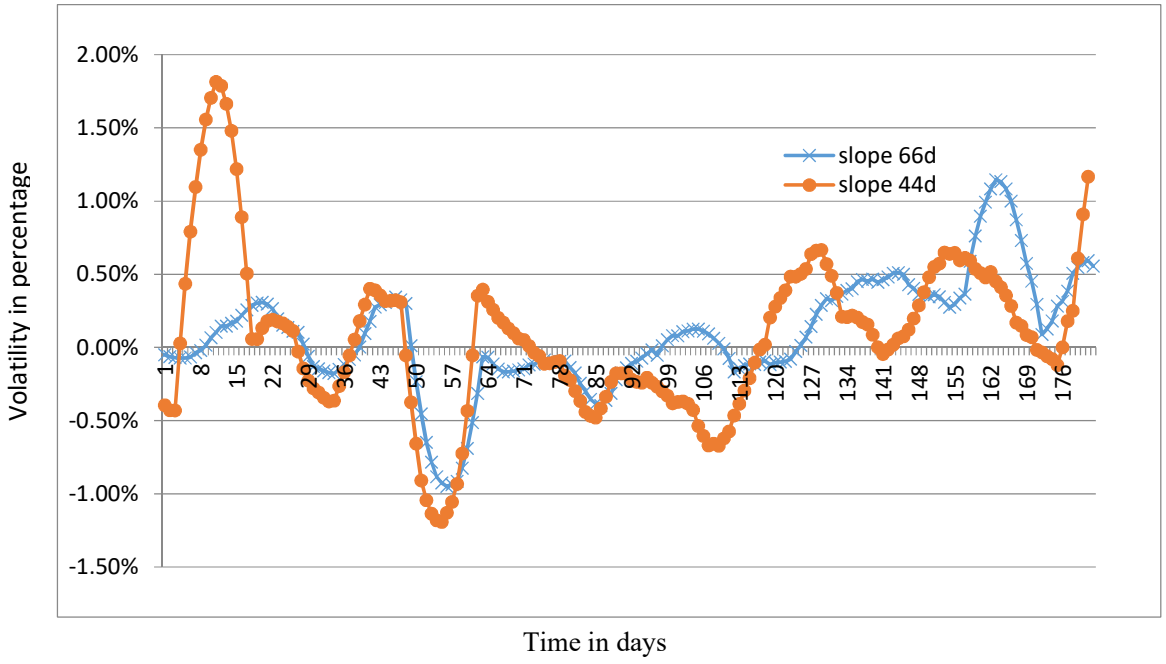


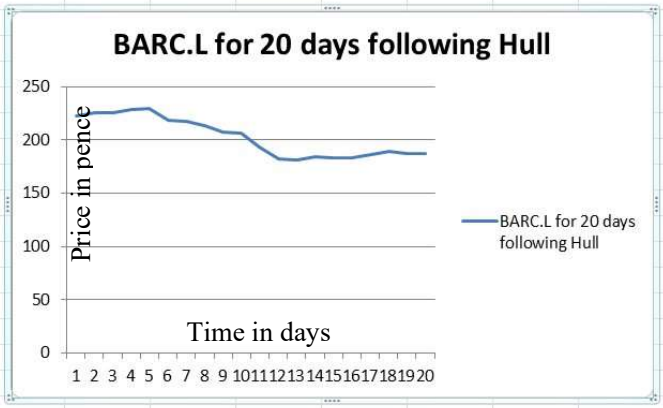
FIGURE 4-11 VOLATILITY SLOPE FOR 44 AND 66 DAYS BARC.L

The volatility of the Barclays share price for 20 days, following the example given by Hull (Hull, 2008), is still quite high at 38.50%, as shown in Table 4-3.

TABLE 4-3 VOLATILITY CALCULATIONS FOR 20 DAYS

day	closing stock price	price relative	u(i) daily return	(u(i))^2
0	223.75			
1	222.05	0.99240	-0.00763	5.82E-05
2	225.15	1.01396	0.013864	0.000192
3	225.4	1.00111	0.00111	1.23E-06
4	228	1.01154	0.011469	0.000132
5	229.05	1.00461	0.004595	2.11E-05
6	217.95	0.95154	-0.04967	0.002468
7	217	0.99564	-0.00437	1.91E-05
8	213.5	0.98387	-0.01626	0.000264
9	207.75	0.97307	-0.0273	0.000745
10	206.4	0.99350	-0.00652	4.25E-05
11	193.05	0.93532	-0.06687	0.004471
12	181.95	0.94250	-0.05922	0.003507
13	181.25	0.99615	-0.00385	1.49E-05
14	184.3	1.01683	0.016688	0.000278
15	183.25	0.99430	-0.00571	3.26E-05
16	183.5	1.00136	0.001363	1.86E-06
17	186.35	1.01553	0.015412	0.000238
18	188.95	1.01395	0.013856	0.000192
19	187.2	0.99074	-0.0093	8.66E-05
20	187.2	1.00000	0	0

tau=1/252 trading days	0.003968	
SUM[u(i)]	-0.17835	
SUM[u(i)^2]	0.012765	
STDEV(u)	0.024251	0.024251
sigma-volatility	38.50%	
standard error	6.09%	



Retail and financial institution share price charts and volatility for 66 days are shown in

Appendix C: Figure 9 for Aviva (AV.L)

Appendix C: Figure 10 for Banco Santander (BNC.L)

Appendix C: Figure 11 for BP.L (BP).

Appendix C: Figure 12 for HSBA.L (HSBC)

Appendix C: Figure 13 for Lloyds (LLOY.L)

Appendix C: Figure 14 for Royal Bank of Scotland (RBS.L)

Appendix C: Figure 15 for Standard Chartered (STAN.L)

Appendix C: Figure 16 for Tesco (TSCO.L)

Appendix C: Figure 17 for Vodafone (VOD.L)

4.9. Online Volatility Indices

Volatility indices are available online from the Financial Times and are shown in Table 4-4.

4.9.1 Financial Times, UK

http://www.ft.com/home/uk

FIGURE 4-12 FINANCIAL TIMES PORTAL

Select from the "Markets" tab dropdown menu "Markets" / "Markets Data"

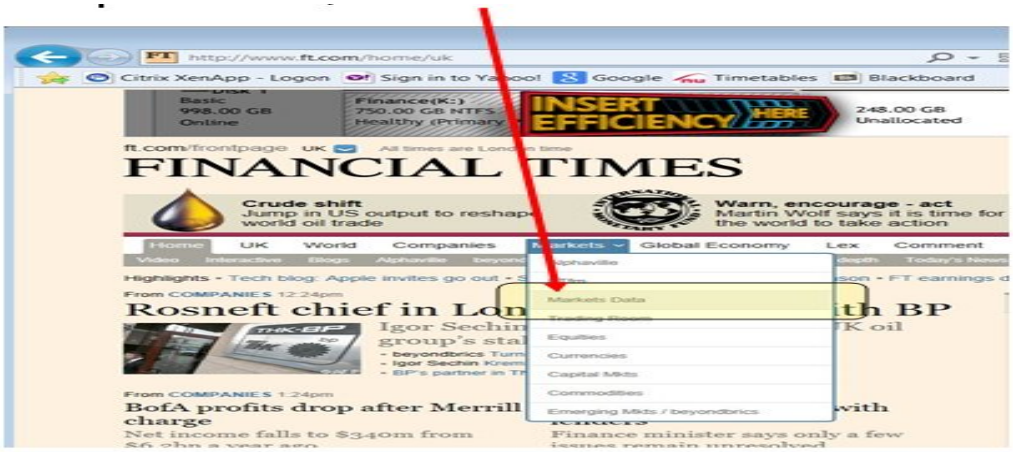


FIGURE 4-13 FT MARKETS DATA

Select "Data Archive" tab (http://markets.ft.com/research/Markets/Overview)

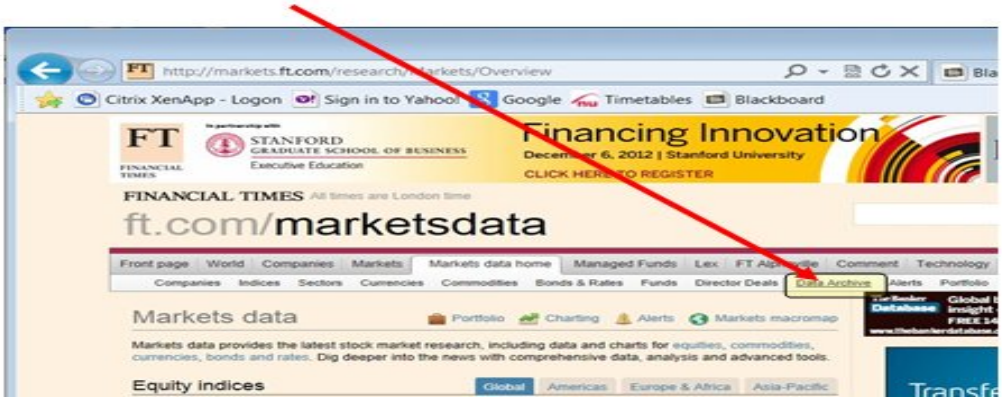


FIGURE 4-14 FT PORTAL OVERVIEW

Select the data: “Select a category”, “Select a report” and “Select a date” and “Download data” (http://markets.ft.com/research/Markets/Data-Archive)

FIGURE 4-15 FT PORTAL ARCHIVE

Equities and UK Equity Volatility Indices

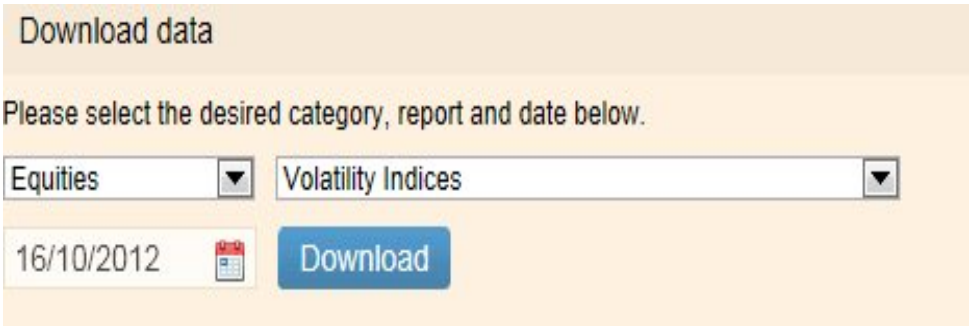


FIGURE 4-16 FT PORTAL VOLATILITY INDICES

TABLE 4-4 VOLATILITY INDICES

VOLATILITY INDICES

	Oct 16	Day Chng	Prev.	52 wk high	52 wk low
VIX †	15.22	-0.05	15.27	37.53	13.30
VXD †	13.84	-0.14	13.98	33.87	11.91
VXN †	17.17	-0.38	17.55	37.19	13.79
VDAX ‡	16.62	-0.45	17.07	31.78	16.22

† CBOE. VIX: S&P 500 index Options Volatility, VXD: DJIA Index Options Volatility, VXN: NASDAQ Index Options Volatility, ‡ Deutsche Borse. VDAX: DAX Index Options Volatility.

4.9.2 VIX Chicago Board Exchange Market Volatility Index

VIX (http://en.wikipedia.org/wiki/VIX) is a trademarked ticker symbol for the Chicago Board Exchange Market Volatility Index, which is a popular measure of the implied volatility in the S&P 500 index. It is often referred to as the fear index, representing one measure of implied stock market volatility over the next 30-day period. Figure 4-17 shows the value of the CBOE volatility index on 31st December from 1985 to 2012.

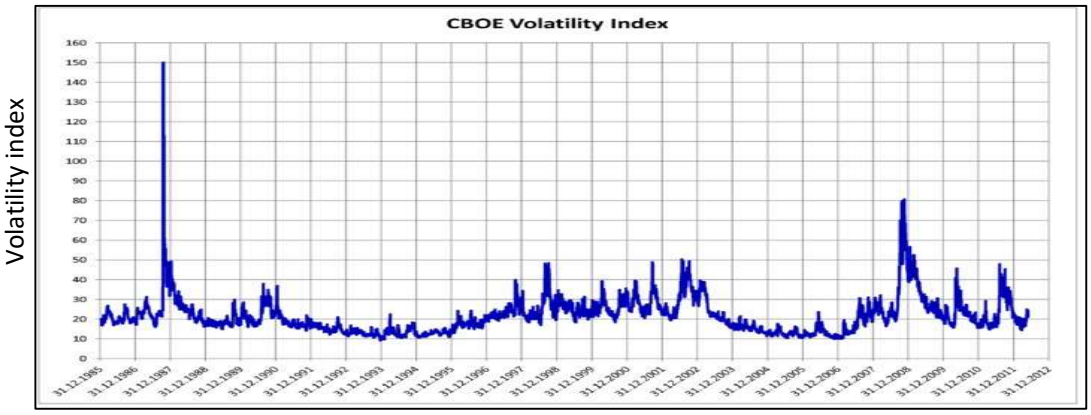


FIGURE 4-17 VIX VOLATILITY INDEX

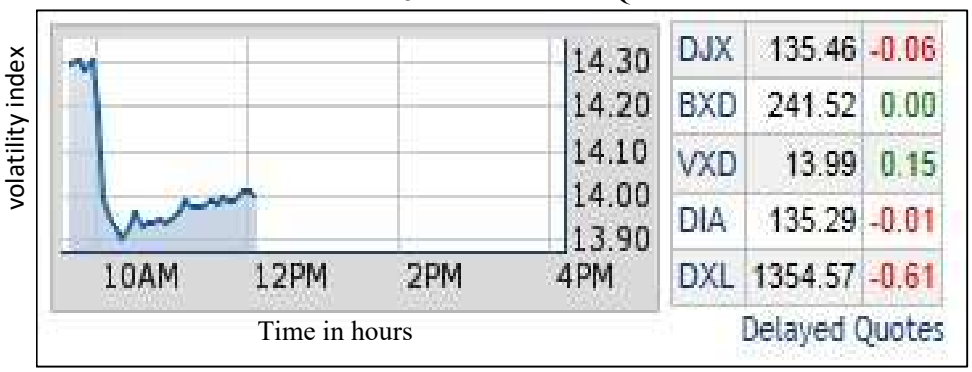
The VIX is quoted in percentage points and translates, roughly, to the expected movement in the S&P 500 index over the next 30-day period, which is then annualized. For example, if the VIX is 15, this represents an expected annualized change of 15% over the next 30 days; thus it can be inferred that the index option markets expect the S&P 500 to move up or down 15%/√12 = 4.33% over the next 30-day period. The index is priced with the assumption of a 68% likelihood (one standard deviation) that the magnitude of the change in the S&P 500 in 30-days will be less than 4.33% (up or down).

The VIX can be used to calculate implied volatility, because volatility is one of the factors used to calculate the value of these indices. Higher or lower volatility of the underlying security makes it more or less valuable, because there is a greater or smaller probability that the security will be above the market value. Thus, a higher index price implies greater volatility, all other things being equal.

4.9.3 VXD

The VXD (http://wiki.fool.com/VXD) is the CBOE DJIA volatility index (CBOE, 2015), meaning that it is an index established by the Chicago Board Options Exchange to measure investor sentiment about near-term volatility in the Dow Jones Industrial Average as shown in Table 4-5 and Figure 4-18.

TABLE 4-5 VXD DELAYED QUOTES



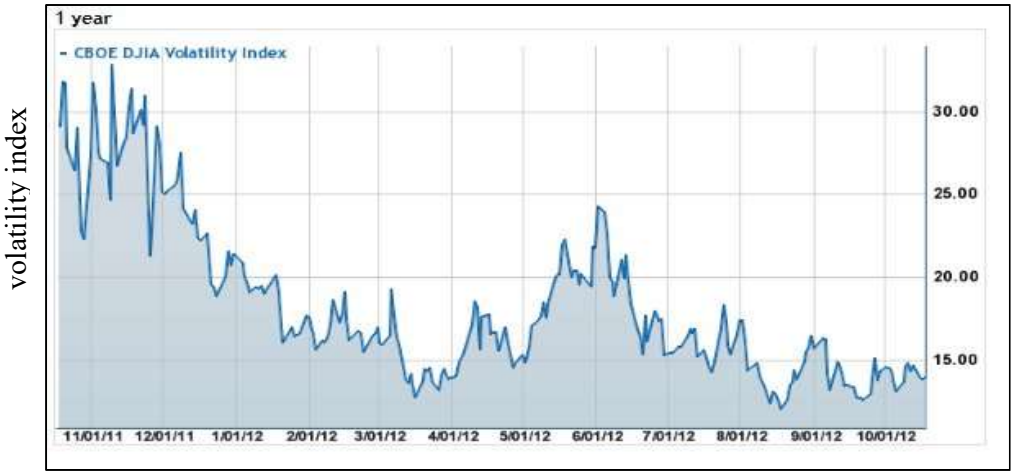


FIGURE 4-18 CBOE DJIA VOLATILITY INDEX

The CBOE DJIA volatility index (VXD) is based on the real-time prices of options on the Dow Jones Industrial Average SM (DJIA, with an options ticker of DJX), and is designed to reflect investors' consensus view of future (30-day) expected stock market volatility.

The Fund Evaluation Group (FEG) (2007) issued a new study entitled "Evaluation of Buy Write and Volatility Indexes: Using the CBOE DJIA Buy Write Index (BXD) and the CBOE DJIA Volatility Index (VXD) for Asset Allocation and Diversification Purposes." This paper studied the 109-month period from October 1997 to November 2006, and presented several findings on the 9-year performance of the VXD Index:

- The volatility index can reduce portfolio volatility. Including a small (10%) allocation to the CBOE DJIA Volatility Index (VXD) could have reduced the

volatility of an all-stock security by about 26%, without materially affecting returns.

- Low correlation and diversification. The VXD and the DJIA were inversely correlated (-0.62). VXD increased more during market declines, reacting more to stock market declines than to stock market advances), indicating that the VXD has potential as a diversification tool.
- Impact on risk-adjusted returns. The inclusion of a small (5%) allocation to the VXD Index boosted risk-adjusted returns for a stock-oriented portfolio, and lowered the risk-adjusted returns for a fixed-income-oriented security.

4.10. Summary

Stochastic model weekly and monthly predictions show not very good fits (not following the target variations) and not very good error performance. The results generally show a simple straight line trend across the target share prices chart, as shown for BARC.L in Figure 4-3.

The analysis of the results as shown in the summary Table 4-1 and Table 4-2 shows consistent market annualized volatility in the range 19.19%-26.66% and consequently the expected prediction error is similar in the range of 7.36% - 17.45% with better retail sector performance between 9.45% to 10.36%.

Evaluation of the volatility indices shows that the volatility index can reduce portfolio volatility, that there is low correlation and diversification and lowered risk-adjusted returns for a fixed-income-oriented security. Furthermore, inclusion of the index can increase the risk-adjusted returns for a stock-oriented portfolio, and lower the risk-adjusted returns for a fixed-income-oriented security.

Chapter 5. Time Series

5.1. Overview

This chapter applies standard statistical analysis such as summary statistics, confidence intervals and particularly analysis of the variance and correlation to financial time series.

Visualising data is convenient for data analysis and provides insight into experimental or simulated data. Data for the 2013 yearly period is relatively flat, without drastic changes. A first look at the BARC.L share price graph indicates a slight polynomial hill and major monthly oscillations, as shown in Figure 5-1. A one-month dataset (January 2013) is shown in Appendix C: Table 4 and the graph is shown in Figure 5-2 which shows an exponential slope and some harmonics as well. Both the polynomial hill and exponential slope could be approximated with a linear gradient. These visual observation encourage curve-fitting and regression models explorations.

This chapter covers the following areas:

- Probability distribution
- Statistical analysis
- Analysis of variance (ANOVA)
- Correlation analysis
- Time Series analysis
- Confidence intervals of trend coefficients
- Components model
- Curve fitting and regression
- Time-share price curve regression (fitting) for one attribute
- Multiple attribute regression with neural networks

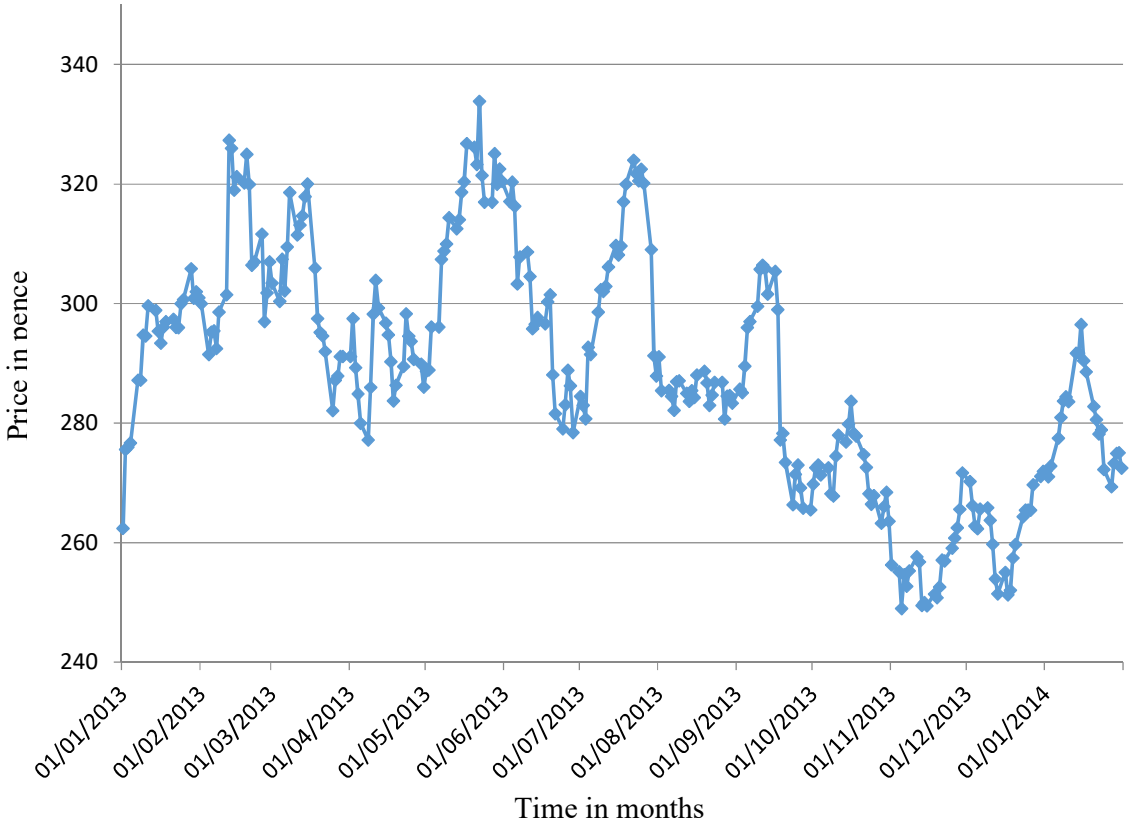


FIGURE 5-1 BARCLAYS PLC SHARE PRICE IN PENCE FOR 13 MONTHS

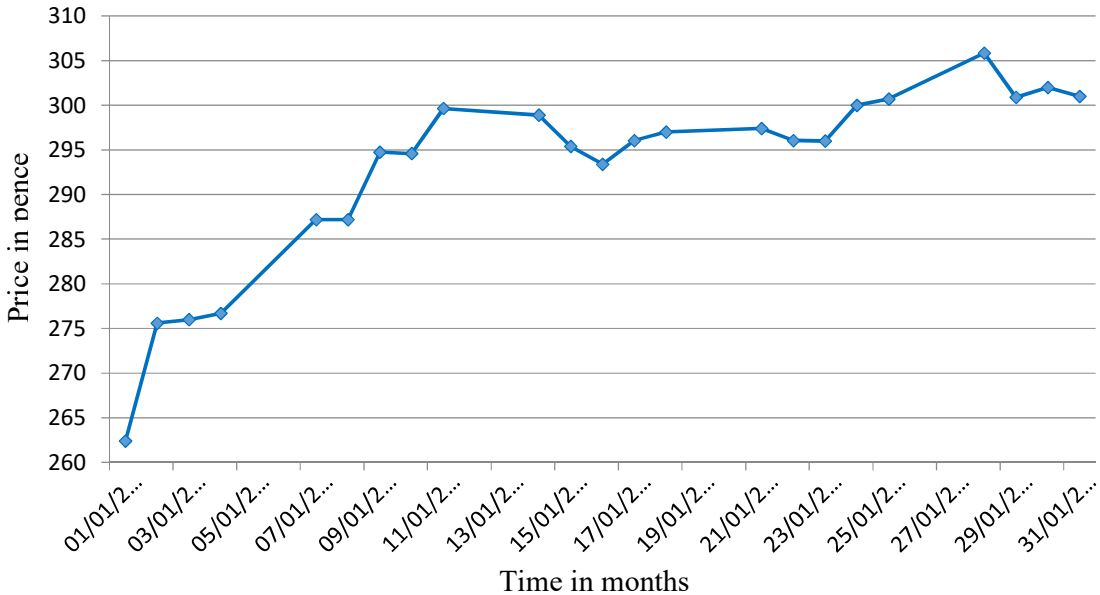


FIGURE 5-2 BARC.L SHARE PRICE IN PENCE FOR ONE MONTH

5.2. Probability Distribution

In probability theory, a probability density function (pdf), or the density of a random variable, is a function that describes the relative likelihood of this random variable taking a

given value. The share price is suggested to have a normal distribution, as shown in Figure 5-3 and Appendix C: Table 5.

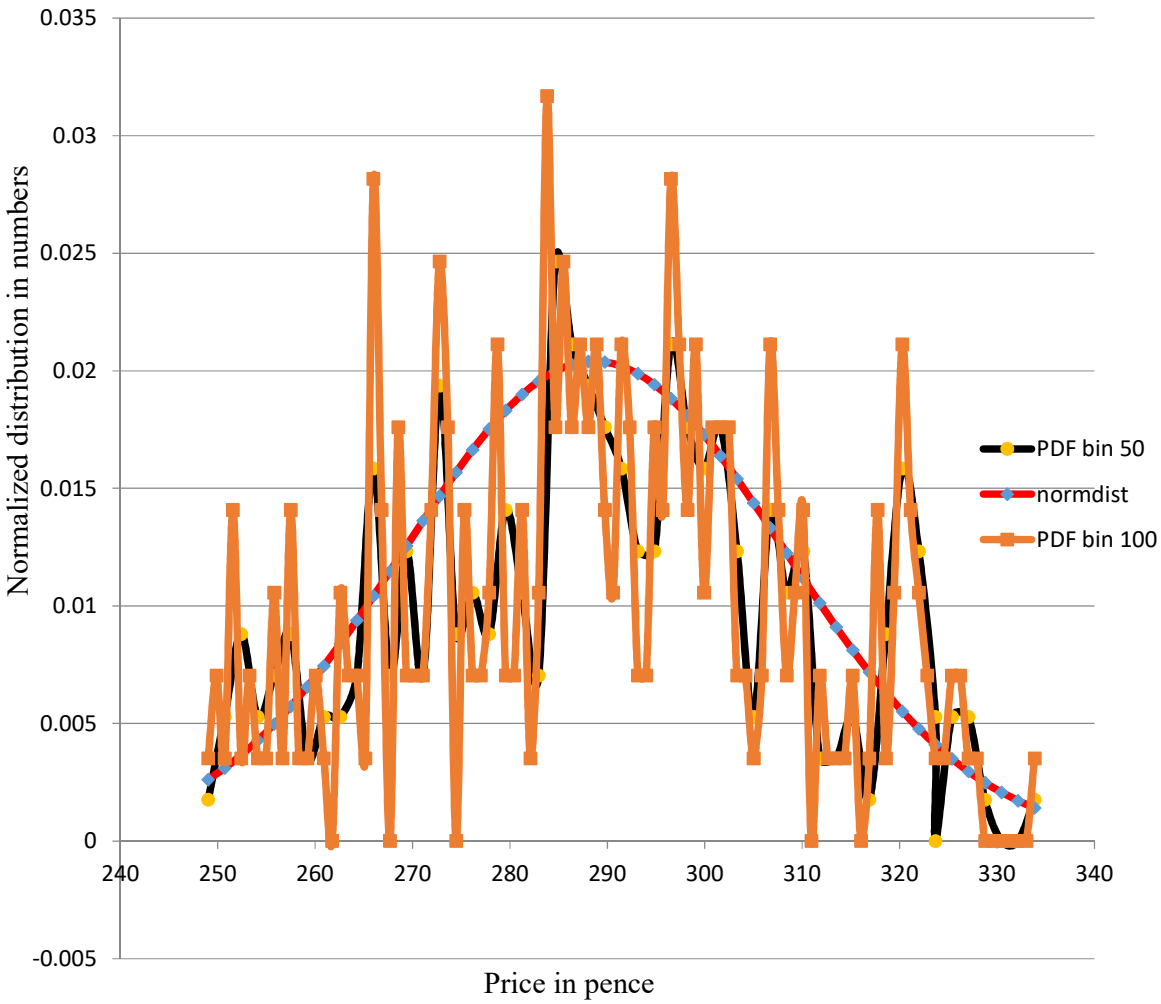


FIGURE 5-3 BARCLAYS PLC PROBABILITY DISTRIBUTION FUNCTION

The pdf is derived from the BARC.L histogram with fifty and hundred bin ranges. Bin ranges define the data frequencies or the number of data samples within the sub-range (bin). For example, for a bin of 100, the entire data range from 249.0 to 333.85 is divided into 100 sub-ranges, with the first from 249.0 to 249.8485.

The more precise the bin is in relation to the dataset size (in this case 284 samples), the closer the chart would be to the actual distribution (see Figure 5-3). Scaling is required for a bin of fifty sub-ranges, and the scaling coefficient is 0.5 which reflects the number of sub-ranges. The probability is calculated by dividing the frequency by the size of the

dataset. The bin acts like a frequency filter, although it still generally shows the nature of the distribution.

5.3. Statistical Analysis

Summary descriptive statistics help to understand the nature of a dataset, with measures such as central tendency and dispersion of the data.

The purpose of taking a random sample from a population such as share prices and computing these statistics is to approximate the mean of the population. How well the sample statistics estimates the underlying population values is always an issue.

A confidence interval provides a range of values, which is likely to contain the parameter of interest for the population. Confidence intervals are constructed at a confidence level, such as 95 %, selected by the user. This means that, if the same population is sampled on numerous occasions and interval estimates are made on each occasion, the resulting intervals would bracket the true population parameter in approximately 95 % of cases. Confidence stated at the $1-\alpha$ level can be thought of as the inverse of a significance level, α. A confidence interval is a range of values that is likely to contain an unknown population parameter most frequently to bind the mean, with a range of values so defined that there is a specified probability that the value of a parameter lies within it. If a random sample is drawn many times, a certain percentage of the confidence intervals will contain the population mean. This percentage is the confidence level (95.%).

The kurtosis characteristic represents the relative peakedness or flatness of a distribution compared with the normal distribution. Positive kurtosis indicates a relatively peaked distribution. Negative kurtosis indicates a relatively flat distribution.

Skewness indicates the degree of asymmetry of a distribution around its mean. Positive skewness indicates a distribution with an asymmetric tail extending toward

values that are more positive. Negative skewness indicates a distribution with an asymmetric tail extending toward values that are more negative.

BARC.L descriptive statistics of the closing price over the long term from 01/01/2013 to 31/01/2014 are shown in Table 5-1.

TABLE 5-1 DESCRIPTIVE STATISTICS OF BARCLAYS PLC SHARE PRICE

Mean	288.641	
Standard Error	1.160	Standard Deviation/Count
Median	288.650	The number in the middle of a set of numbers.
Mode	287.200	The most frequently occurring, or repetitive, value in an array or range of data.
Standard Deviation	19.550	SQRT(VAR)
Sample Variance (VAR)	382.219	$\frac{\sum(x-\bar{x})^2}{(n \quad 1)}$
Kurtosis	-0.681	Characteristic representing the relative flatness of a distribution compared with the normal distribution.
Skewness	-0.004	Characteristic indicating the degree of asymmetry of a distribution around its mean.
Range	84.850	Maximum minus Minimum
Minimum	249.000	Minimum value
Maximum	333.850	Maximum value
Samples	284	Number of samples in the set
Confidence Level (0.95, 95%)	2.274	Value used to construct a confidence interval for a population mean.
Confidence Level (90.0%)	1.908	

A kurtosis of -0.681 indicates a relatively flat distribution. As can be seen from the pdf, the maximum is 0.0317 and this is quite symmetrical with a skewness of -0.004 slightly to the negative (left side) of the mean of 288.650. Furthermore, the median and the mean are almost the same, with the mode close as well. All of these indicate a relatively deterministic process.

The mean is in the centre of the range ± 2.274 (2.2% of the range); for example, the share price mean of 288.641 is at the centre of the range 288.641 ± 2.274, which is the range of population means. Before running any statistical test, the alpha level has to be determined first, which is also called the "significance level" or the probability of making a wrong decision. The traditionally common 95% confidence interval of 2.274 is not much different from 1.908, which is the 90% interval, which relaxes the experimental accuracy. Interestingly the 99% confidence interval of 2.988 is a significant accuracy expectation although quite close to the traditional and relaxed intervals. This all reassures us of the investigation's statistical providence.

Short-term dataset analysis is used to determine if there are differences from the long-term dataset. Data from a period of three weeks (15 days) are analysed, as is intended for use in modelling a dataset time-window.

Appendix C: Table 4 shows the BARC.L share price from 13/01/2014 to 31/01/2014 with the descriptive statistics in Appendix D: Table 1

Appendix C: Table 7 shows the BARC.L share price from 23/12/2013 to 10/01/2014 with the descriptive statistics in Appendix D: Table 2.

Both 15-day datasets are quite similar statistically but differ significantly from the long-term standard deviation, particularly in the case of ~7.5 versus 19.5 and in the confidence level range of ~18% versus 2.7%. Short periods have a more deterministic nature with less wide distribution and confidence level/range percentage. This gives a case for the

selection of the short- term time-window of 10-15 days for the modelling of short-term trading. Table 5-2 shows the comparative statistics for the BARC.L share prices over short and long periods.

TABLE 5-2 BARCLAYS SHORT AND LONG PERIODS COMPARATIVE STATISTICS

	13/01/2014-short	23/12/2013-short	1/1/2013-long
Mean	273.29	281.12	288.641
Standard Error	1.83	2.23	1.160
Median	271.95	278.9	288.65
Mode	265.45	#N/A	287.20
Standard Deviation	7.09	8.65	19.50
Sample Variance	50.37	74.90	382.219
Kurtosis	-1.17	-1.23	-0.681
Skewness	0.43	0.41	-0.004
Range	20.05	27.15	84.850
Minimum	264.35	269.35	249.00
Maximum	284.4	296.5	333.80
Count	15	15	284
Largest	284.4	296.5	333.85
Smallest	264.35	269.35	249.00
Confidence Level/Range	0.19	0.17	0.027
Confidence Level(95.0%)	3.93	4.79	2.274

5.4. Analysis of Variance (ANOVA)

In experiments, some differences are expected among the different samples for better model training. ANOVA is a simple analysis of variance on data for two or more samples. The analysis provides a test of the hypothesis that each sample is drawn from the same

underlying probability distribution against the alternative hypothesis that the underlying probability distributions are not the same for all samples. ANOVA compares the statistical differences among two datasets checking for the variation in the datasets via comparing the amount of variation between datasets with the amount of variation within datasets. It calculates the *F*-ratio, which is used to obtain the probability *P*-value. A significant *P*-value (usually taken as $P>0.05$) suggests that datasets are significantly different, allowing the null hypothesis to be rejected. The null hypothesis is that the samples are statistically the same (for example, all population means are equal).

The *F* ratio separates the variation in the datasets into two parts, between-datasets and within-dataset, called the sums of squares. The between-dataset variation MS_B (or between sums of squares *BSS*) is calculated by comparing the mean of each dataset with the overall mean of the data. The within-group variation MS_W (or within sums of squares *WSS*) is the variation of each observation from its dataset mean. The *F* ratio is:

$$F = \frac{MS_B}{MS_W}. \qquad (5\text{-}1)$$

If the average difference between groups is similar to that within groups, the *F* ratio is approximately one. As the average difference between groups becomes greater than that within groups, the *F* ratio becomes larger than 1. To obtain a *P*-value, the *F* ratio can be tested against the *F* distribution of a random variable with the degrees of freedom associated with the numerator and denominator of the ratio.

If the *P*-value is less than the supplied alpha and the obtained F-value is greater than the critical *F* value, this implies that the null hypothesis (of statistically the same samples datasets) should be rejected.

Examples of BARC.L share prices for two fifteen days periods(13/1/2014 and 9/1/2014 are shown in Appendix C: Table **8**. A summary of the ANOVA results is shown in Table 5-3.

TABLE 5-3 ANOVA RESULTS SUMMARY BETWEEN AND WITHIN GROUPS

Groups	Count	Sum	Average	Variance
Column 1	15	4216.9	281.12	74.9003
Column 2	15	4237.35	282.49	66.1579

ANOVA

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	13.94	1	13.94	0.1976	0.6600	4.1959
Within Groups	1974.81	28	70.52			
Total	1988.75	29				

Since $P(0.66)>alpha(0.05)$ and $F(0.19)<F$-critical, the null hypothesis that the datasets are statistically the same is accepted.

Examples of BARC.L share prices for two fifteen days periods from 13/1/2014 and 20/12/2013 are shown in Appendix C: Table 9. A summary of the ANOVA results between and within the groups is shown in Appendix C: Table 11.

TABLE 5-4 ANOVA SIXTEEN DAYS STARTING DATE SHIFT 13/01/2014 AND 20/12/2013

ANOVA SUMMARY

Groups	Count	Sum	Average	Variance
Column 1	15	4216.9	281.1267	74.90031
Column 2	15	4075.55	271.7033	53.28088

ANOVA

Source of variation	SS	df	MS	F	P-value	F crit
Between Groups	665.9940833	1	665.9941	10.39145	0.003208	4.195971707
Within Groups	1794.536667	28	64.0906			
Total	2460.53075	29				

Since $P(0.003)<alpha(0.05)$ and $F(10.39)>F$-critical, the null hypothesis that the datasets are statistically the same is rejected.

Tests for datasets with different starting dates show that statistically independent samples must not overlap. To reject the null hypothesis that the datasets are statistically the same, there should be sixteen day's difference between the starting date of the two dataset samples; see row day=16. More similar results allow the hypothesis for the same datasets to be accepted. The results are summarized in Table 5-5 below.

TABLE 5-5 TESTS FOR STATISTICAL INDEPENDENCE FOR DIFFERENT STARTING DATES

Day	Average-1	Average-2	VAR-1	VAR-2	P >alpha	alpha	F<F-critical	F-critical	Result
0	281.13	281.13	74.9	74.9	0.99	0.05	1.20E-14	4.19	accept
1	281.13	281.87	74.9	69.4	0.81	0.05	0.05	4.19	accept
2	281.13	282.49	74.9	66.1	0.66	0.05	0.19	4.19	accept
13	281.13	275.12	74.9	65.29	0.059	0.05	3.86	4.19	accept
14	281.13	275.12	74.9	65.29	0.059	0.05	3.86	4.19	accept
15	281.13	273.3	74.9	50.3	0.011	0.05	7.34	4.19	reject
16	281.13	271.7	74.9	53.2	0.003	0.05	10.39	4.19	reject

5.5. Correlation Analysis

Working with a multivariable problem requires an assessment of the correlation coefficients between variables before conducting any modelling. In the case of share prices, the variables considered are opening, highest, lowest and closing share prices. This reduces the dimensionality of the analysis and avoids using highly correlated (or linear) independent variables, which can produce poor results. A correlation coefficient such as the Pearson product-moment correlation coefficient is a measure of the extent to which two measurement variables "vary together". The correlation coefficient is scaled so that its value is independent of the units in which the two measurement variables are expressed. For example, if the two variables are opening and closing share prices, the value of the

correlation coefficient is unchanged if the closing price is converted from pence into pounds. Figure 5-4 shows the scatter plot for Open, High and Low ranks to Close. The value of any correlation coefficient must be between -1 and +1, and a correlation analysis can examine each pair of variables to determine whether they tend to move together, so that large values of one variable tend to be associated with large values of the other (positive correlation), or if small values of one variable tend to be associated with large values of the other (negative correlation), or if values of the two variables tend to be unrelated with a correlation near to zero. The BARC.L two trading weeks dataset from 01/01/2013 to 14/01/2013 is shown in Appendix C: Table 10.

The correlation coefficient between opening and closing prices shows a very high positive correlation of 0.976725 in the fourth row of the first column, which means that these variables move together. Overall, the high positive correlations of approximately 0.98 between the Open, High, Low and Close share prices allows the number of independent variables to be reduced and to use just one, such as the closing share price. Open, High and Low share prices are positively correlated and in a regression analysis, for example, they should be excluded from the curve-fitting. The Pearson product-moment correlation matrix is shown in Table 5-6.

TABLE 5-6 CORRELATION MATRIX PEARSON PRODUCT-MOMENT

	Open	High	Low	Close
Open	1			
High	0.98455263	1		
Low	0.97800967	0.972681403	1	
Close	0.976725	0.986709797	0.980223731	1

An alternative method is the Spearman rank correlation. The Spearman correlation coefficients between two variables will be high when observations have a similar rank, or identical for a correlation of 1. The ranks indicate relative positions of the observations

within the data for the strength of relationships between the two variables, and a correlation is low when observations have a dissimilar (or fully opposed for a correlation of -1) rank between the two variables. The Spearman coefficient is given by the following formula (Bourg, 2006):

$$r = 1 - \frac{6\sum d^2}{N(N^2-1)} \qquad (5\text{-}2)$$

where *d* is the difference between the ranks in corresponding values for both variables and *N* is the number of data points.

TABLE 5-7 SPEARMAN RANK COEFFICIENTS

Share Prices				Rank			
Open	High	Low	Close	Close	Open	High	Low
262.4	262.4	262.4	262.4	257	256	261	248
272.85	277.08	269.6	275.6	207	216	212	218
274.65	278.89	272.83	276	206	210	206	200
275.15	278.56	273.5	276.7	205	208	208	198
281	288.46	276.04	287.2	148	193	166	190
285.6	295.48	282.5	287.2	148	166	134	158
289	298.15	288.8	294.75	114	139	119	116

For example, in Table 5-7 the rank of the first row Open 262.4 has the rank 256.

TABLE 5-8 SPEARMAN RANK CORRELATION COEFFICIENTS

	Open	High	Low	Close
Open	1	0.98386	0.9751	0.975163
High	0.98386	1	0.969012	0.984064
Low	0.9751	0.969012	1	0.977824
Close	0.975163	0.984064	0.977824	1

The scatter plot for Open, High and Low ranks to Close is shown in Figure 5-4.

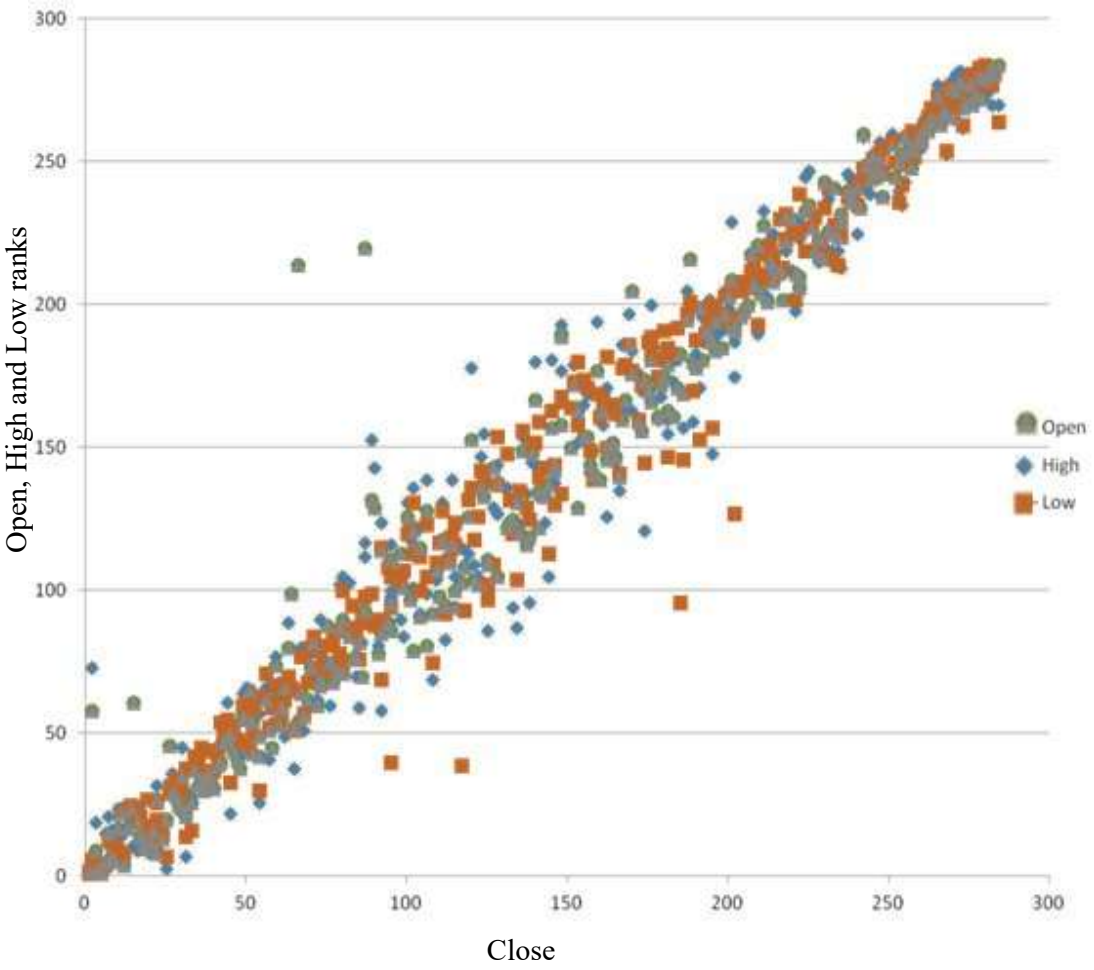


FIGURE 5-4 SCATTERED PLOT OPEN, HIGH AND LOW RANKS TO CLOSE

The scatter plot and Spearman correlation coefficients for closing and opening prices show a linear relationship between the variables. For example, the coefficient for closing and opening prices is 0.975163, in row one, column four of Table 5-8. The conclusion from the Pearson and Spearman correlation results is to use one variable such as Close share price. Open, High and Low share prices as shown in Table 5-6 and Table 5-8 are correlated and in the regression analysis they should be excluded from curve fitting.

To train the model, a dataset has to be constructed selecting datasets from the population where the columns are the datasets commencing from the date on the top in ascending order. For example in Appendix C: Table 11, the first column shows the dataset starting from 13/01/2014 to 31/01/2014 the bottom values. There is an overlap in the first six columns, which is highlighted.

A correlation matrix for 13th January is shown in Table 5-9.

TABLE 5-9 CORRELATION MATRIX FOR 13 JANUARY

	10-Jan	9-Jan	8-Jan	26-Dec	24-Dec	20-Dec	18-Nov
13-Jan	0.88	0.75	0.55	-0.82	-0.85	-0.80	-0.93

This correlation matrix indicates positive correlations of 0.88, 0.75 and 0.55 between dates close to each other (10th, 9th and 8th January) and negative correlations with further away samples of -0.82, -0.85, -0.80 and -0.93 (26th, 24th, 20th December and 18th November). The scatter plot of the samples confirms that data for the dates closer to each other have a declining slope moving together and the further away dates have generally a positive slope.

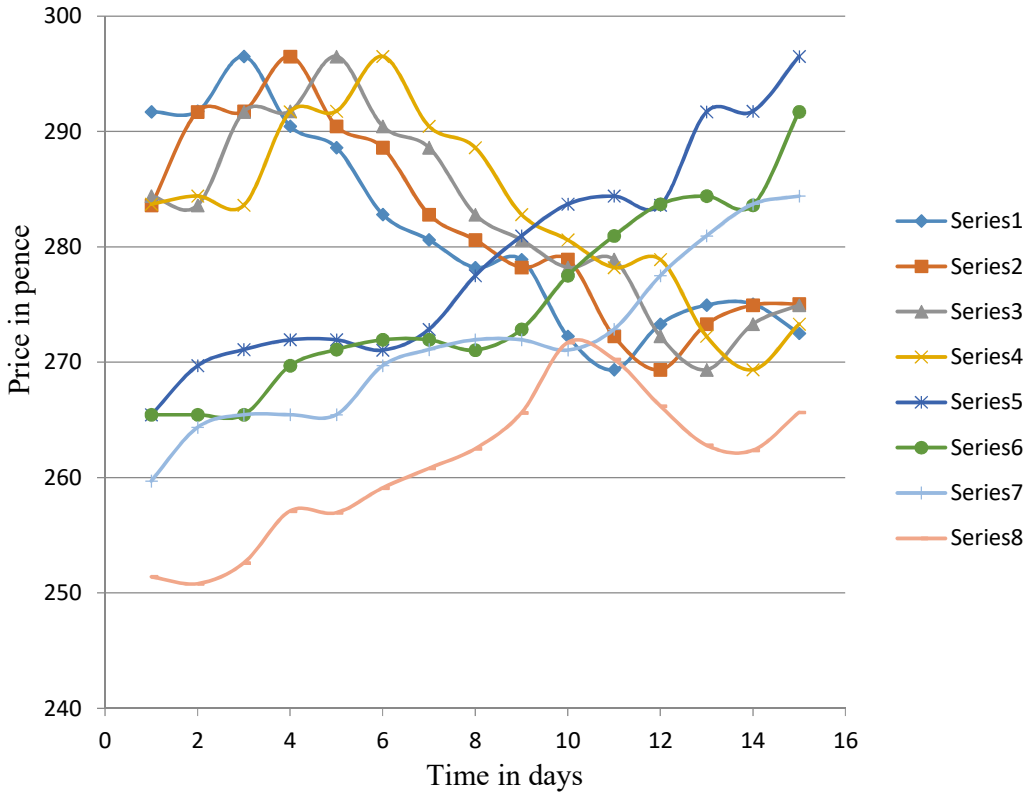


FIGURE 5-5 SHARE PRICE CHART OF THE CORRELATION DATASET SERIES

It is constructive to generate uncorrelated samples for the model by random value selection from the whole dataset, as shown in Appendix C: Table 12. The randomly assembled dataset samples look uncorrelated in the scatter plot shown in Figure 5-6. Uncorrelated

samples add robustness and predictive capability to the derived model and could be further investigated with Monte Carlo, ANN, regression and curve-fitting.

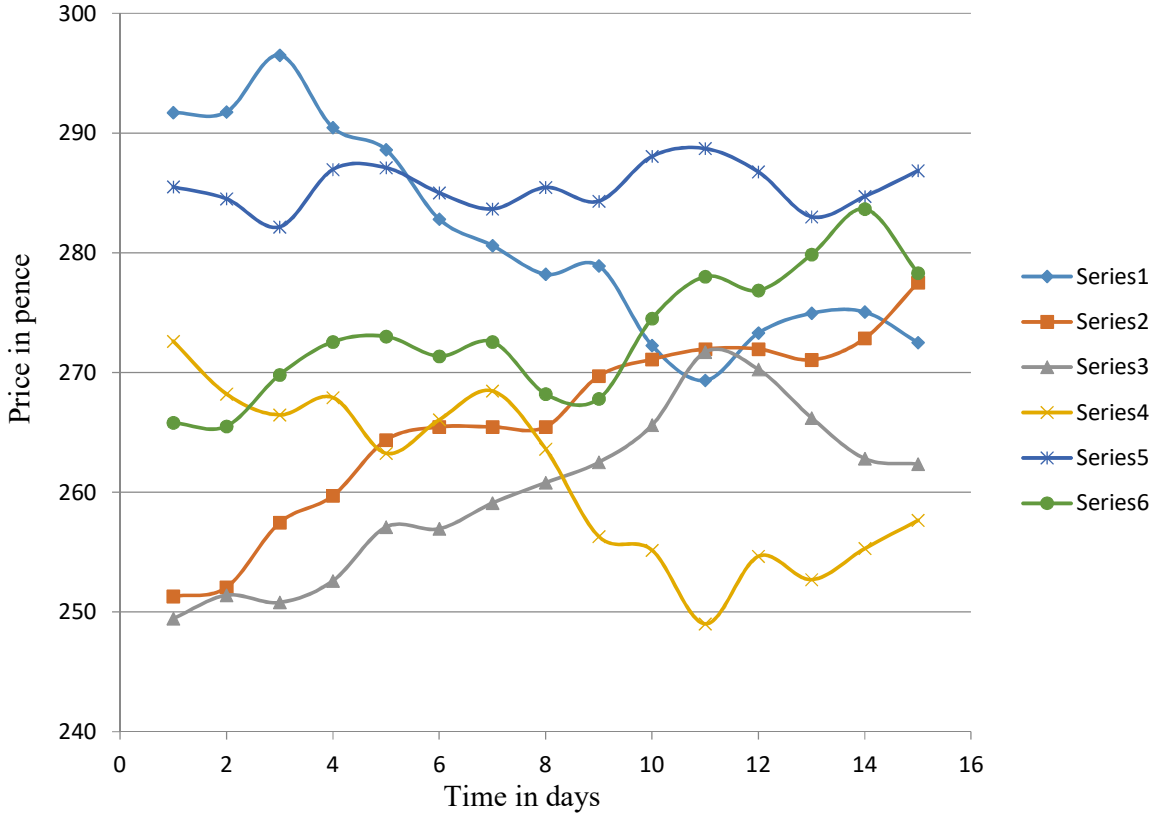


FIGURE 5-6 RANDOM SELECTION DATASET SHARES CHART

5.6. Time Series Analysis

Time series analysis can be used to study applications for stock market prediction, and it includes the pre- and post-processing of data, visualizing results, and making forecasts. Fourier analysis allows time series data to be transformed into the frequency domain, and back again, for further analysis.

The most common visualization of a time series display of continuous data over time is a line chart with markers, set against a timescale. This is ideal for showing trends in data at equal intervals. The data is distributed evenly along the horizontal axis (time), and all value data (share price) is distributed evenly along the vertical axis and displayed with markers to indicate individual data values as in Figure 5-7. Line charts are useful to show trends

over time or ordered categories, especially when there are many data points and the order in which they are presented is important.

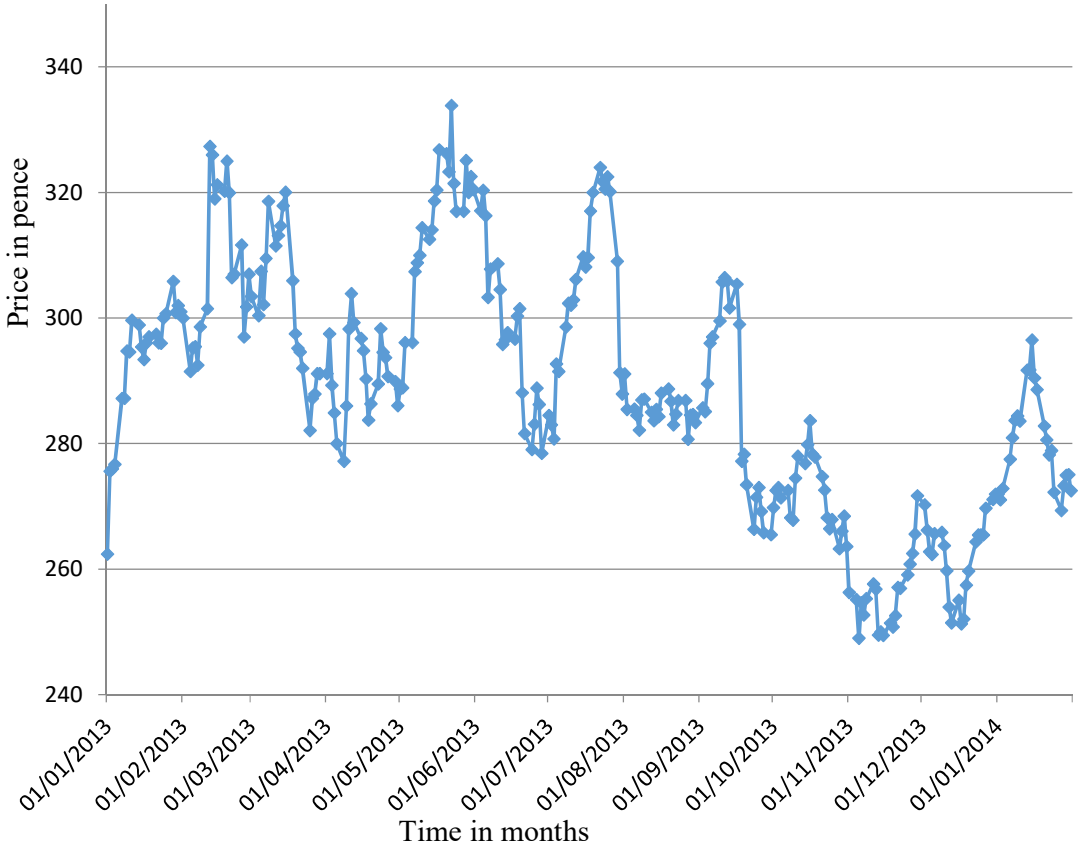


FIGURE 5-7 BARCLAYS PLC SHARES CLOSING PRICE IN PENCE

There are many different trends in such data, although a common type of representation in financial analysis is a linear trend line. To add a trend to a chart, first the type of regression to be used should be chosen, such as linear, logarithmic, polynomial, power, exponential, or moving averages. The type of data determines the type of trend line to be used.

A linear trend line usually shows that something is increasing or decreasing at a steady rate, and is used with simple linear datasets, as in:

$$y = mx + b \tag{5-3}$$

where m is the slope and b is the intercept.

A logarithmic trend line is a best-fit-curved line that is most useful when the rate of change in the data increases or decreases quickly and then levels out, as in:

$$y = c\ln(x) + b \tag{5-4}$$

where c and b are constants, and ln is the natural logarithm function.

A polynomial trend line is a curved line that is used when data fluctuates. It is useful, for example, for analysing gains and losses in a large dataset. The order of the polynomial can be determined by the number of fluctuations in the data or by how many bends hills and valleys appear in the curve. An order 2 polynomial trend line generally has only one hill or valley. An order 3 line generally has one or two hills or valleys, and order 4 generally has up to three:

$$y = b + c_1x + c_2x^2 + c_3x^3 + \cdots + c_6x^6 \tag{5-5}$$

where b and c_i are constants.

A power trend line is a curved line that is best used with datasets that compare measurements that increase at a specific rate;

$$y = cx^b \tag{5-6}$$

where c and b are constants.

An exponential trend line is a curved line that is most useful when data values rise or fall at increasingly higher rates.

$$y = ce^{bx} \tag{5-7}$$

where c and b are constants, and e is the base of the natural logarithm.

A moving average trend line smoothes out fluctuations in data to show a pattern or trend more clearly. A specific number of data points is averaged, and the average value is used as a point in the trend line. If the *period* is set to 2, for example, then the average of the first two data points is used as the first point in the moving average trend line. The average of the second and third data points is used as the second point in the trend line, and so on:

$$F_t = \frac{A_t + A_{t-1} + \cdots + A_{t-n+1}}{N} \tag{5-8}$$

The moving average is thus a sequence of averages computed from parts of a data series. The number of points in a moving averages trend line equals the total number of points in the series less the number that is specified for the period.

To give data closer to the forecast priority over that which is further away, weighted moving averages can be used:

$$Y_{n+1} = \propto Y_n + (1-\propto)S_n \qquad (5\text{-}9)$$

where Y_n is the smoothed value, S_n is the actual share value and,$\propto$ is a damping (smoothing) factor.

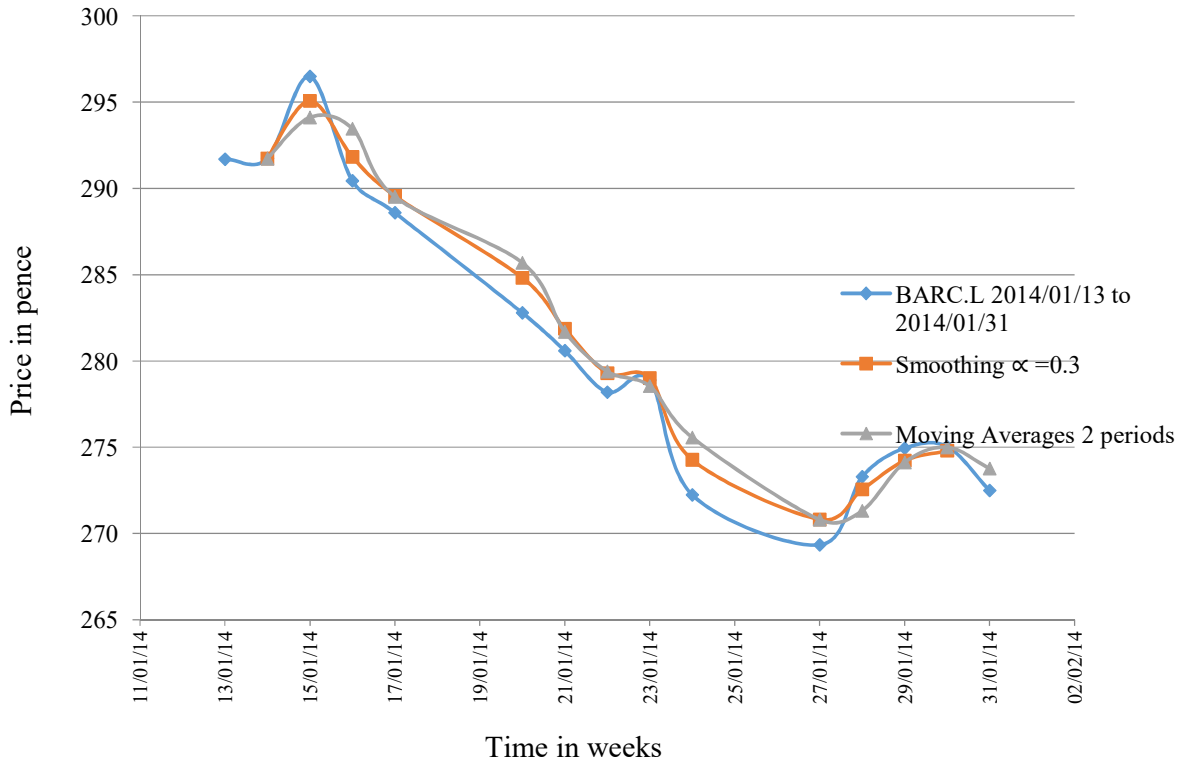


FIGURE 5-8 BARCLAYS PLC SHARE PRICE WITH SMOOTHING AND MOVING AVERAGES

Some smoothing of data could be useful for removing randomness (noise) in a time series, and so some trade-offs have to be accepted to distinguish between information and noise, and to define the level of zooming in the data required.

Figure 5-9 shows examples of share price time series data for 15-days commencing on 13-Jan (Series 1), 17-Dec (Series 2), 15-Nov (Series 3), 22-Oct (Series 4), 5-Aug (Series 5),

and 27-Sep (Series 6). A common feature is a slope with about one or two hills or valleys indicating a polynomial of the second order.

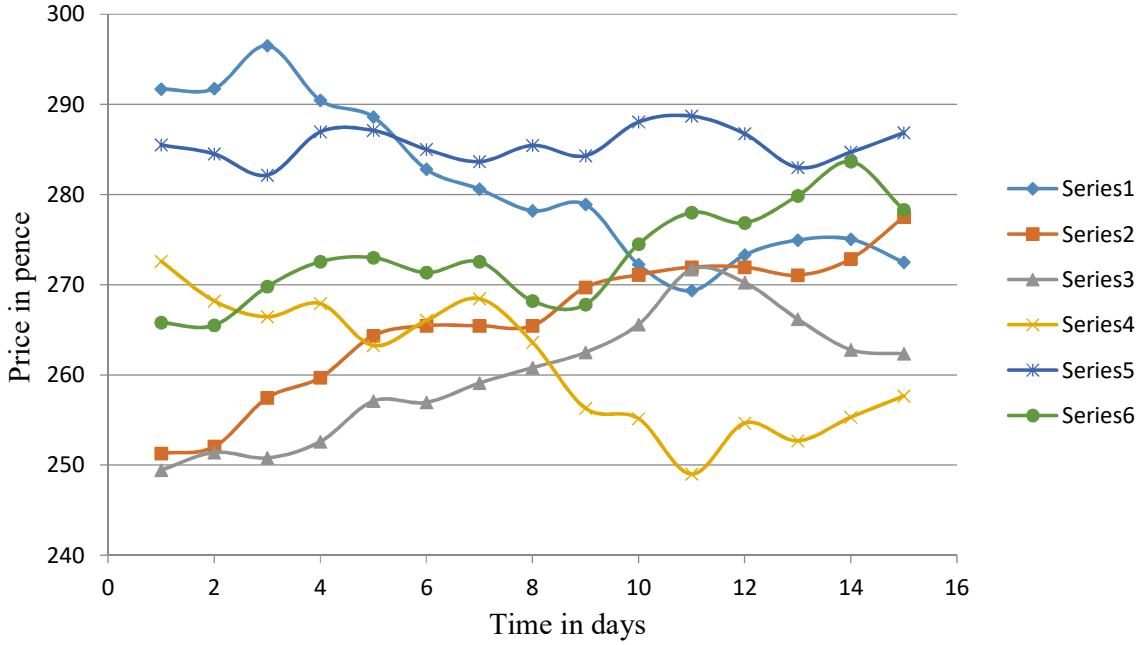


FIGURE 5-9 FIFTEEN DAYS TIME SERIES AT DIFFERENT STARTING DATES

The coefficient of determination R^2 measures how well the model fits the data; that is, how the observed outcomes are replicated by the model based on the proportion of total variation in outcomes explained by the model. In general, a model fits the data well if the differences between the observed values and the model's predicted values are small and unbiased. How well the model equation describes the data (the 'fit'), is expressed as a correlation coefficient, R^2 (R-squared). The closer R^2 is to 1.00, the better is the fit.

If μ is the mean of the observed data then:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{5-10}$$

The total sum of squares is proportional to the variance of the data:

$$SS_{tot} = \sum_i (y_i - \mu)^2 \tag{5-11}$$

The sum of squares of residuals is also called the residual sum of squares:

$$SS_{res} = \sum_i (y_i - f_i)^2 = \sum_i {e_i}^2 \tag{5-12}$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{5-13}$$

where y_i is the observed data and f_i is the model data.

Usually, financial time series are complex combinations of trend lines, harmonics frequencies and random components. The first step in the discrimination of the components would be to identify trends in the time series. Trend lines are used to graphically display trends in data and to help analyse problems of prediction, and such analysis is named regression analysis. By using regression analysis, a trend line in a chart can be extended beyond the actual data to predict future values. Furthermore, removing it from the time series for further analysis is referred to as data centralization and is recommended before applying further stochastic analysis and eventual modelling such as with an ANN.

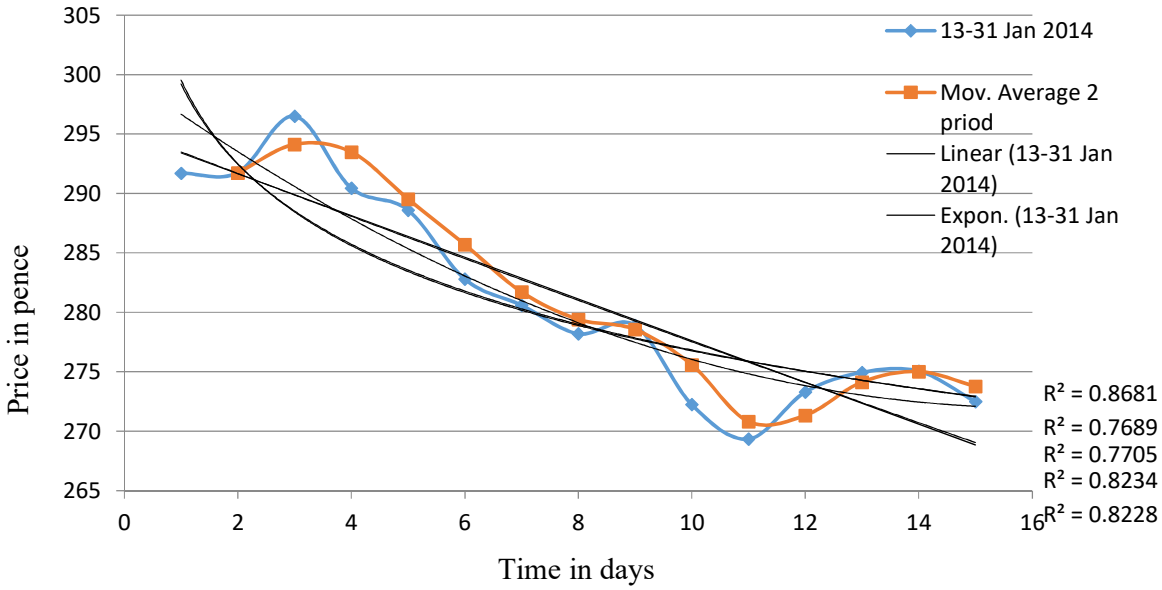


FIGURE 5-10 MOVING AVERAGES, LINEAR TREND LINES

TABLE 5-10 R^2 FOR DIFFERENT TYPE OF TREND LINES

	Linear	Exp.	Log	Power	Poly-2	Mov-2
R^2	0.8228	0.8234	0.7705	0.7689	0.8681	0.9496

A special case is the trend with an ANN. A two-layered ANN is tried, as shown in Figure 5-11, where *S1* is the output of the first layer and *S2* is the output of the second layer. The results are shown in Appendix C: Table 13.

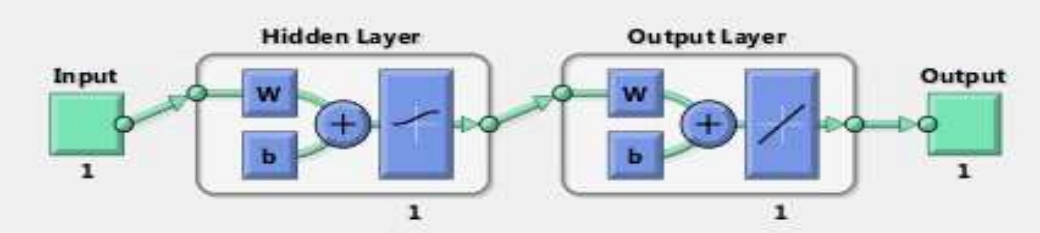


FIGURE 5-11 TWO-LAYER ANN

The coefficients of the first (hidden) layer are *K*=-0.1 and *b*=0.25. The output layer coefficients are *k2*=4 and *b*=-2, which were obtained via manual tuning.

k1	-0.1	k2	4
b1	0.25	b2	-2

The graphs shown in Figure 5-12 indicates almost linear trends for both the first and second ANN layers with fitness values of *R2*=0.832361 and 0832393 respectively, which are close to the exponential (0.8234) and linear (0.8228) trends.

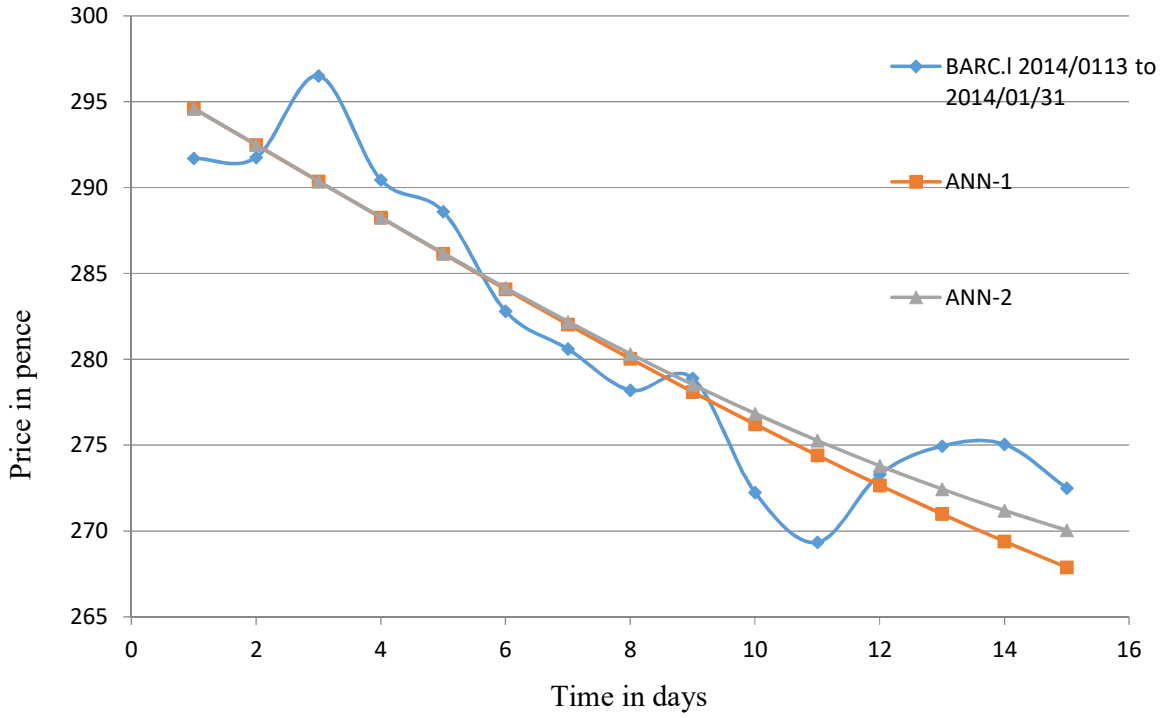


FIGURE 5-12 ANN FIRST AND SECOND LAYERS

Removing the bias or trend from a time series is generally required before processing the data using forecasting methods. The standard approach is to subtract the trend of the data

from each data item. Examples are shown in Appendix C: Table 14 for the BARC.L share prices from 13/01/2014 to 31/01/2014 and in Figure 5-13 and Figure 5-14.

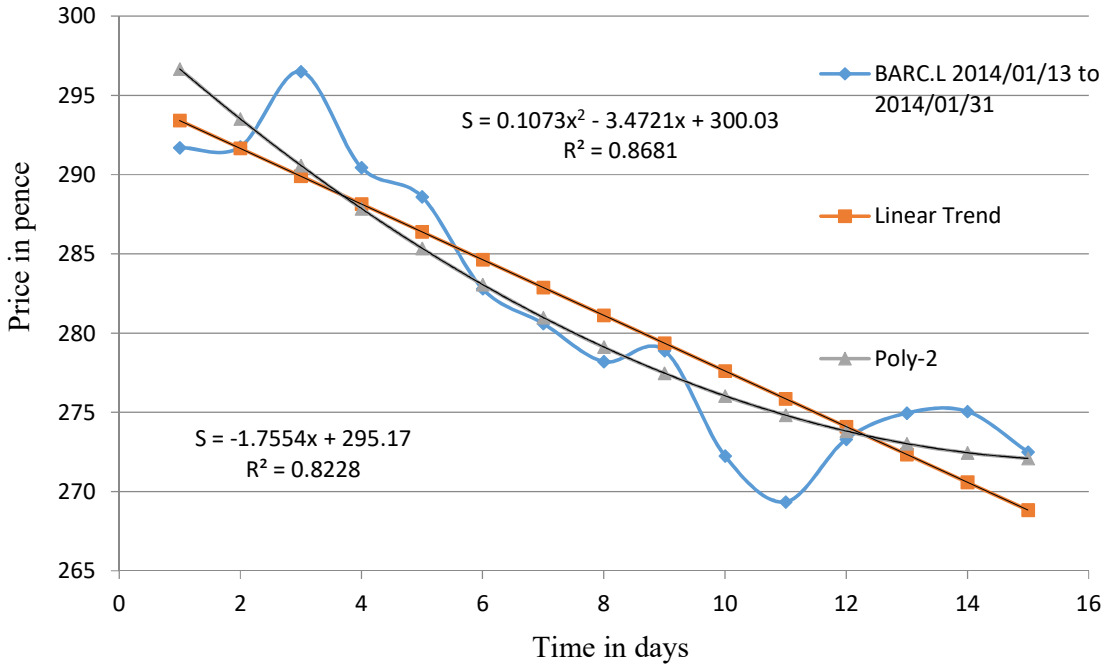


FIGURE 5-13 BARCLAYS LINEAR AND POLY-2 TREND 2014-01-31

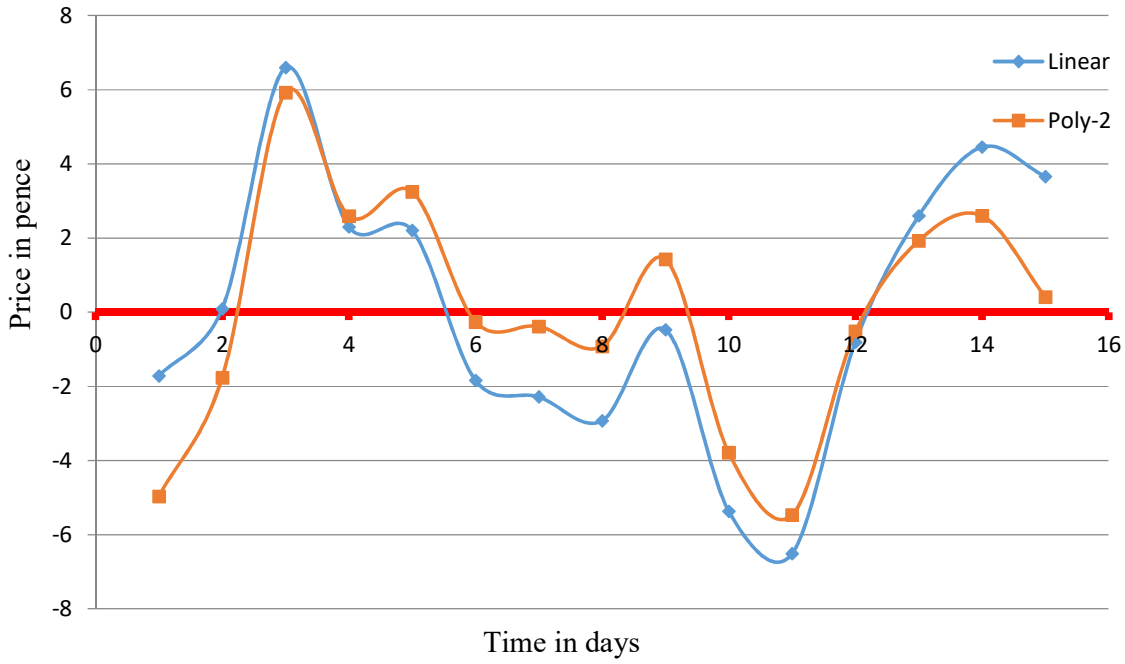


FIGURE 5-14 BARCLAYS PLC CENTRED SHARE PRICES

5.7. Confidence Intervals of Trend Coefficients

Confidence intervals for trend coefficients can be used to estimate the robustness of a model by calculating the boundary range of the trend approximation function coefficients and calculating the percentage of the range of the last value. Generally, the longer the range then the distribution and eventual forecasting error are wider.

The algorithm's operation can be summarized as follows,

1. Find the coefficients of the modelling function *Y=f(X)*, such as a polynomial *Y=p(X)* of degree *n* that fits the data.
2. (*nlinfit*) Estimate coefficients for the non-linear regression of the responses in *Y* on the predictors in *X* using the model specified by a modelling function *Y=f(X).*
3. (*nlparci*) Calculate the 95% confidence intervals (*ci)* for the non-linear least squares parameter estimates *beta.* Before calling *nlparci*, use *nlinfit*to to fit a non-linear regression model and get the coefficient estimates *beta*, residuals *resid*, and estimated coefficient covariance matrix (sigma).

The algorithm functions are defined as follows.

polyfit

p = polyfit(x,y,n) finds the coefficients of a polynomial *p(x*) of degree n that fits the data, *p(x(i))* to *y(i)*, in a least squares sense. The result, *p,* is a row vector of length *n+1* containing the polynomial coefficients in descending powers:

$$p(x) = p_1 x^n + p_2 x^{n-1} + \cdots + p_n x + p_{n+1} \tag{5-14}$$

nlinfit

beta = nlinfit(X,Y,modelfun,beta0) returns a vector of estimated coefficients for the non-linear regression of the responses in *Y* on the predictors in *X* using the model specified by the modelling function. The coefficients are estimated using iterative least squares estimation, with initial values specified by *beta0.*

nlparci

ci = nlparci(beta,resid,'covar',sigma) returns the 95% confidence intervals *ci* for the non-linear least squares parameter estimates (*beta*). Before calling *nlparci*, *nlinfit*is is used to fit a non-linear regression model and to derive the coefficient estimates (*beta*), residuals (*resid)*, and estimated coefficient covariance matrix (*sigma*).

The following examples illustrate the action of the algorithm:

5.7.1 The last (most recent) 15 samples trend with coefficients confidence level

The trend function is *Price=At*+B

A=-1.50 is in the range of [-2.03 to -0.97] at the 95% confidence level.

B=292.53 is in the range of [287.88 to 297.19] at the 95% confidence level.

The last trend price of 270.03 confidence level in the range [257.45 to 282.61] is 9.32%.

The last trend residual is 2.47 (0.91%).

5.7.2 All samples trend with coefficients confidence level

The trend function is *Price=At+B*

A=-0.15 is in the range of [-0.17 to -0.13] at the 95% confidence level

B=310.24 is in the range of [306.70 to 313.77] at the 95% confidence level

The last trend price 267.20 confidence level range [257.55 276.84] is 7.22%

The last trend residual is 5.30 (1.95%).

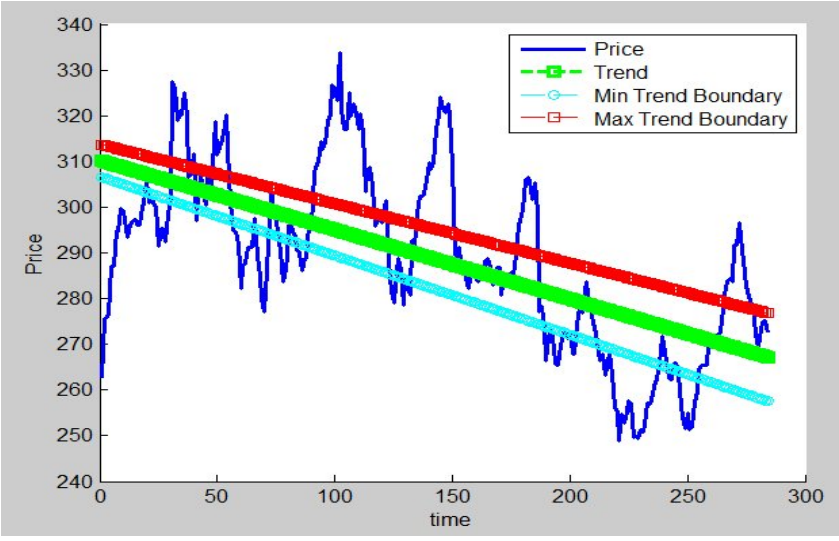


FIGURE 5-15 MINIMUM AND MAXIMUM LINEAR TREND BOUNDARIES OF THE 284 SAMPLES

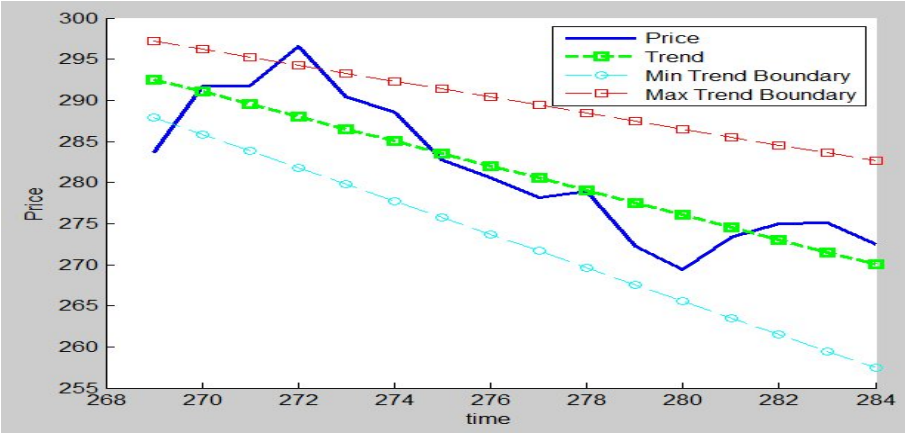


FIGURE 5-16 MINIMUM AND MAXIMUM BOUNDARIES OF THE MOST RECENT 15 SAMPLES

5.8. Components Model

A time series could be composed of several components: long term trend (*LTT*), short-term trend (*STT*), cyclical (*CYCL*) and irregular (*IRR*) components (Bourg, 2006) . There is a seasonal (*SEA*) component that could be considered as well, although in the banking sector this is not very obvious and generally could be covered by a cyclical component:

$$S = LTT + STT + CYCL + IRR + SEA \quad (5\text{-}15)$$

The first step would be to find the long-term trend (*LTT*), such as a linear trend over a period one-year long. The "Linear" column is the linear trend calculated for this point and "Centred" is the difference between the actual share price and the linear trend values. The fitness R^2=0.4053 is not very high, as is apparent from the plot. This value of " Centred" will be the input to the second level of de-trending in a short period of fifteen days or three weeks.

$$Linear = -0.1516t + 310.24$$

$$R^2 = 0.4053$$

An example of a centred (Centered = Close-Linear) long term BARC.L share prices dataset is shown in Appendix C: Table 15 for one year from 01/01/2013 to 31/01/2014 and the original and centred graphs are shown in Figure 5-17 and Figure 5-18.

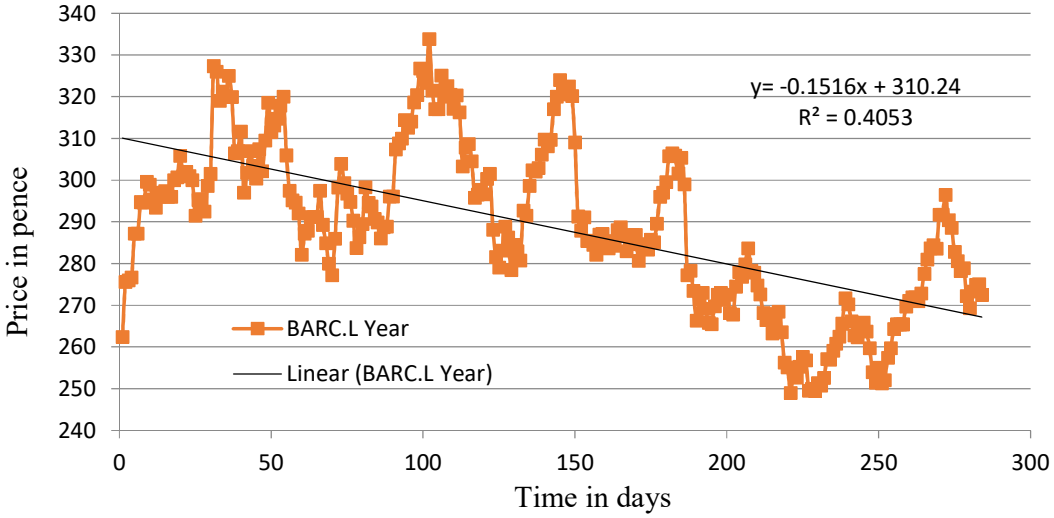


FIGURE 5-17 SHARE PRICES LONG-TERM TIME SERIES WITH TREND LINE

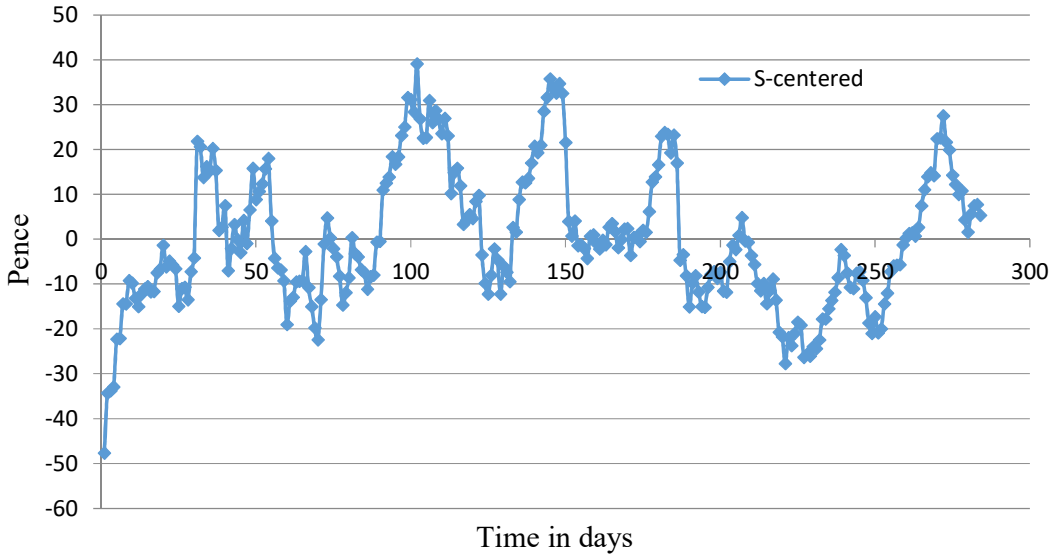


FIGURE 5-18 CENTRED SHARE PRICES LONG-TERM TIME SERIES

The second component, short term trend (*STT*), for fifteen days is de-trending the output of the long-term trending (*LTT*), and there is a much better fitness, of $R^2 = 0.6813$

$$Linear = -1.3485t + 23.073 \quad (5\text{-}16)$$

$$R^2 = 0.6813 \quad (5\text{-}17)$$

An example of a centred (Centered= *Scntr - Linear)*) short- term BARC.L share price dataset is shown in Appendix C: Table 15 for two weeks from 10/01/2014 to 29/01/2014, and the original and centred graphs in Figure 5-19 and Figure 5-20.

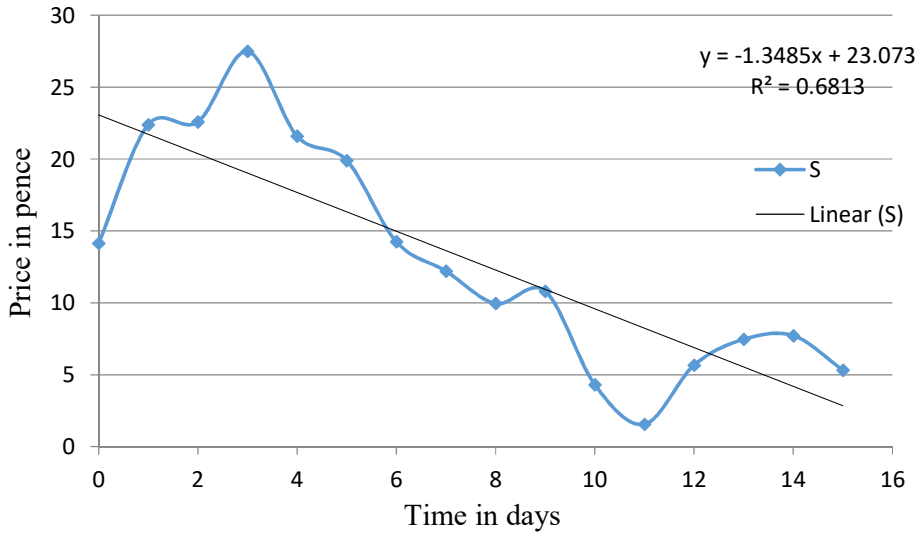


FIGURE 5-19 SHORT TERM 15 DAYS SHARE PRICE WITH LINEAR TREND

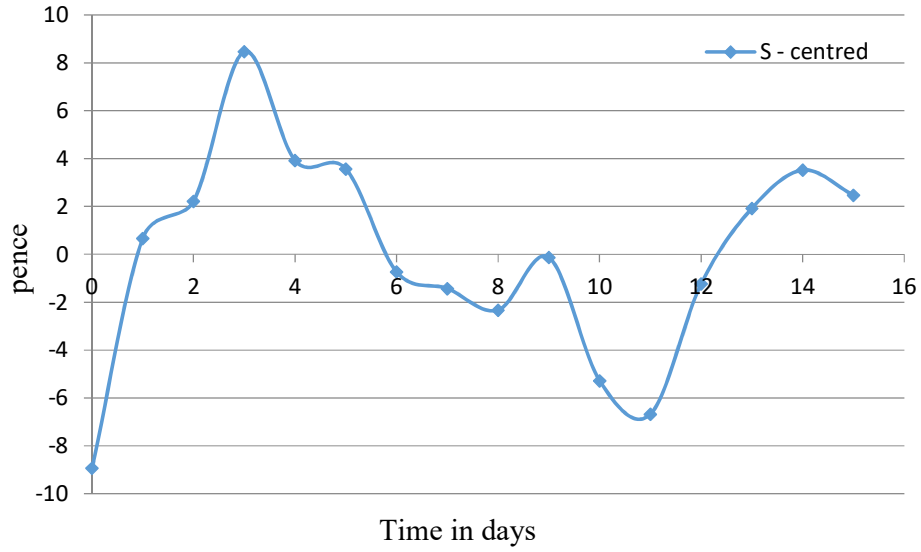


FIGURE 5-20 CENTRED SHORT TERM 15 DAYS SHARE PRICES

The third component, cyclic (*CYCL*), is assumed to be the sum of the frequencies *F1*, *F2* and *F3* and is used to approximate the *S*-centred short-term time series. Three sinusoidal frequencies are considered. To find the amplitude (*A*), period (*T*) and phase (*P*) of each frequency, the centred series is manually analysed. For example, the period of the first frequency *F1* is about 10 days long, its amplitude is 7 and shift (phase) is about -1. The values are found manually by changing variables and checking the error.

An example of a centred cyclic short term BARC.L share prices dataset is shown in Appendix C: Table 17 for two weeks from 10/01/2014 to 31/01/2014 and the original and centred graphs are shown in Figure 5-23 and Figure 5-21, Figure 5-22 and Figure 5-24. The three harmonics parameters using the EXCEL solver tool and further manual tuning (amplitude, period and phase) are shown in Table 5-11.

TABLE 5-11 HARMONICS PARAMETERS: AMPLITUDE (A), PERIOD (T) AND PHASE (P)

	F1	F2	F3
A	7	6	4
T	10	12	6
P	-1	-8	-1

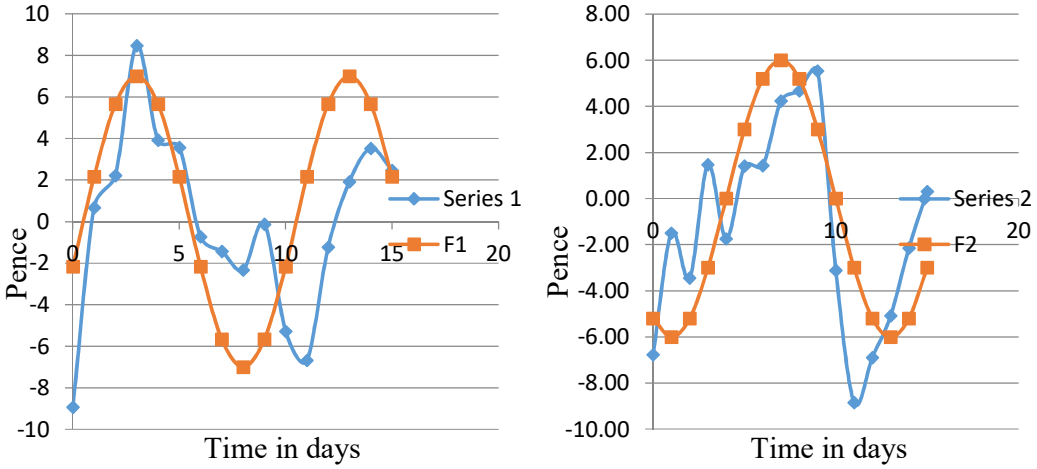


FIGURE 5-21 CENTRED WITH THE FIRST AND THE SECOND HARMONICS

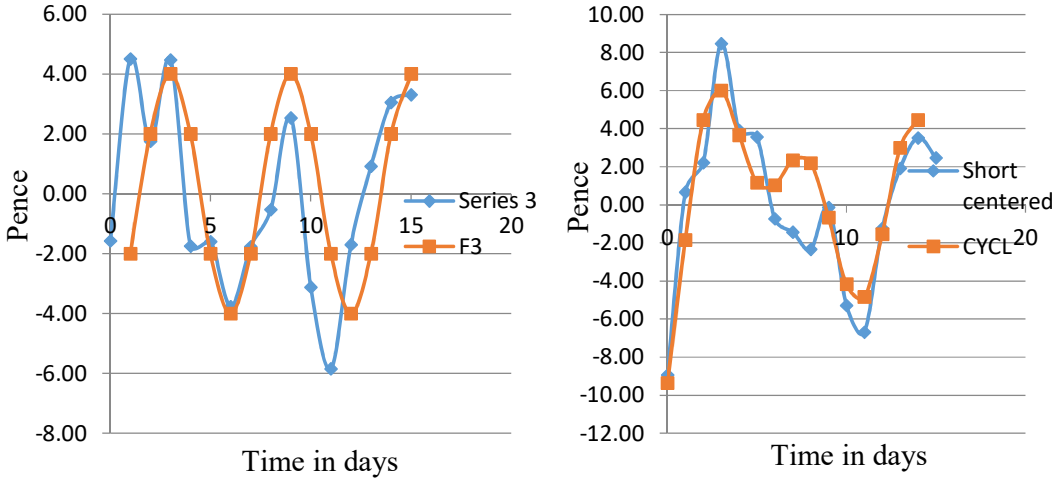


FIGURE 5-22 CENTRED WITH THE THIRD HARMONIC AND CYCL (ALL HARMONICS)

The composite approximation combining long and short linear trend and all three cyclic CYCL harmonics is quite good, with a value of R^2=0.93 which is particularly close to the forecasting point at the end of the period.

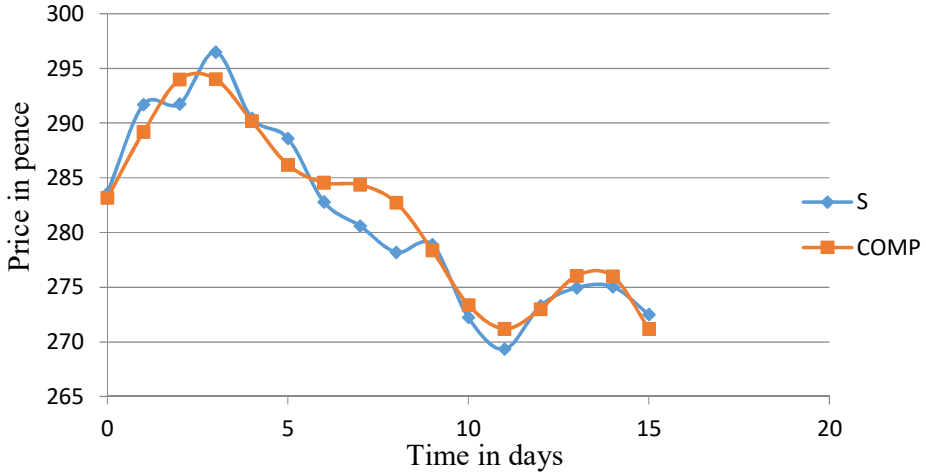


FIGURE 5-23 COMPOSITE APPROXIMATION AND ACTUAL SHARE TIME SERIES CHART

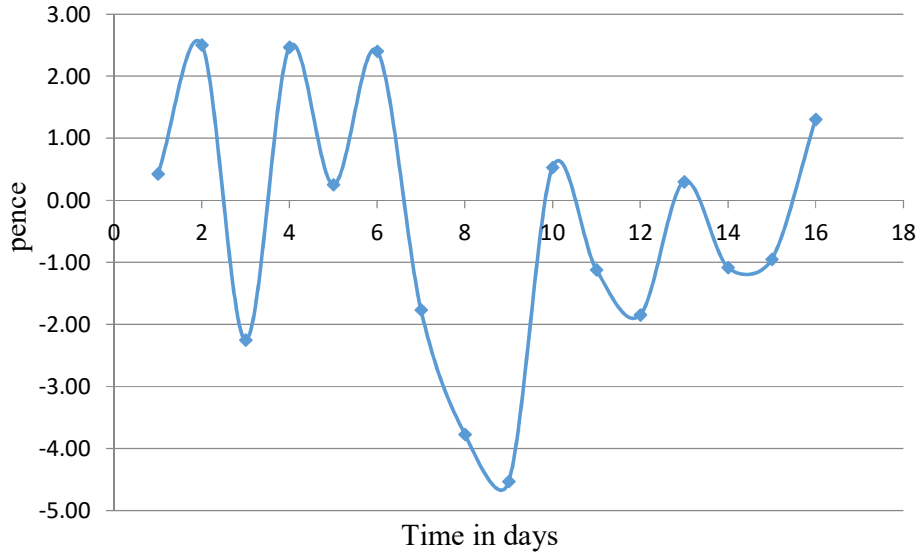


FIGURE 5-24 CENTRED WITH COMPOSITE CYCL APPROXIMATION

There is a commonly accepted belief that prices on Monday and Friday are low with a peak on Wednesday. To check this hypothesis for weekly variation in the time series, as to whether or not there is a weekly pattern in Monday, Tuesday, Wednesday, Thursday and Friday between weeks, the weekly (seasonal) indices should be computed. Finding the average-percentage is a commonly used method. The testing dataset consists of weekly

time series for weeks beginning on 06/01/2014, 13/01/2014, 20/01/2014, 27/01/2014 as shown in Table 5-12 and Figure 5-25.

TABLE 5-12 DATASET FOR FOUR WEEKLY PATTERN TIME SERIES

	06/01/2014	13/01/2014	20/01/2014	27/01/2014
Mon	277.5	291.7	282.8	269.35
Tue	280.95	291.75	280.6	273.3
Wed	283.7	296.5	278.2	274.95
Thu	284.4	290.45	278.9	275.05
Fri	283.6	288.6	272.25	272.5
Weekly Average	282.03	291.8	278.55	273.03

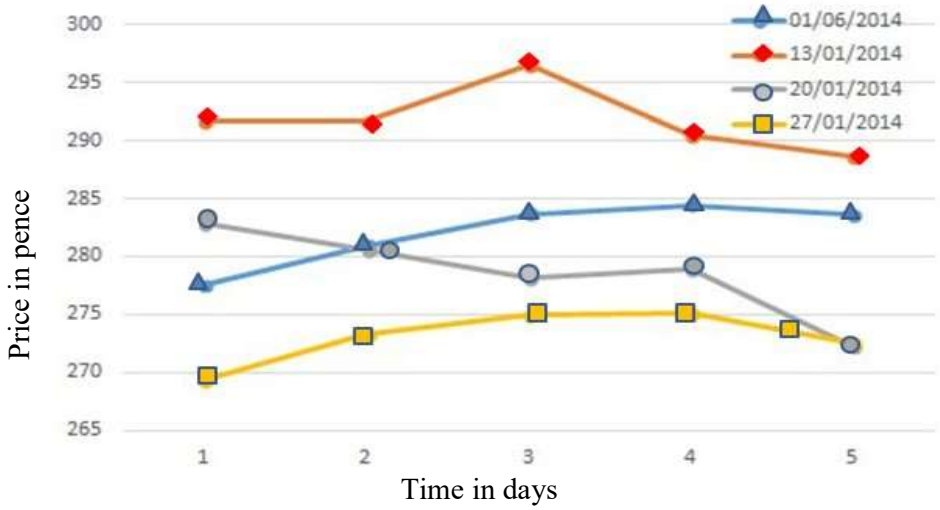


FIGURE 5-25 WEEKLY PATTERN TIME SERIES CHARTS

The horizontal axis shows weekdays 1-Mon, 2-Tue, 3-Wed, 4-Thu and 5-Fri. and there is no obvious pattern.

In the seasonal average-percentage method, for weekdays, the average percentage indices are calculated by dividing the day value by the average for the week, as shown in Table 5-13, and the value for Monday 6/01/2014 is 277.5. The index is 277.5/282.03 = 0.9839. The weekly seasonal indices for the five week days are shown in Table 5-14 and the graphs in Figure 5-26.

TABLE 5-13 AVERAGE-PERCENTAGE INDEXES

	06/01/2014	13/01/2014	20/01/2014	27/01/2014
Mon	0.9839	0.9997	1.0153	0.9865
Tue	0.9962	0.9998	1.0074	1.0010
Wed	1.0059	1.0161	0.9987	1.0070
Thu	1.0084	0.9954	1.0013	1.0074
Fri	1.0056	0.9890	0.9774	0.9981

The diagram of the weekday indices shows some weekly patterns confirming the suggestion that there is a peak in midweek, the so-called “hump day”.

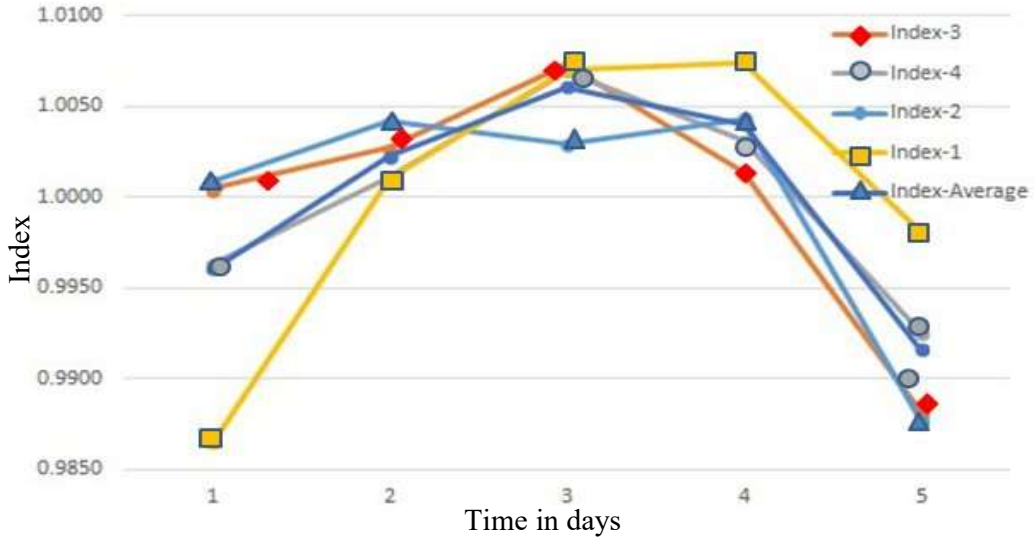


FIGURE 5-26 WEEKDAY INDICES

TABLE 5-14 WEEKLY SEASONAL INDICES

	06/01/2014	13/01/2014	20/01/2014	27/01/2014	Index-2	Index-3	Index-4
Mon	0.9839	0.9997	1.0153	0.9865	1.0009	1.0005	0.9963
Tue	0.9962	0.9998	1.0074	1.0010	1.0042	1.0027	1.0011
Wed	1.0059	1.0161	0.9987	1.0070	1.0029	1.0073	1.0070
Thu	1.0084	0.9954	1.0013	1.0074	1.0043	1.0013	1.0031
Fri	1.0056	0.9890	0.9774	0.9981	0.9877	0.9882	0.9925

The seasonal weekday's index is the average of the corresponding weekdays indices for one, two, three and four weeks. For example, Index-3 for Mondays is the average of Monday indices for the weeks 13/01/2014, 20/01/2014, 27/01/2014. The average index is the generalized average of these indices. In conclusion, there is some pattern in the weekly prices but this in fact is negligible and can be ignored.

5.9. Curve Fitting and Regression

A common approach when analysing data is to fit a curve through the data. This is the process of trying to find a curve which represents a model equation best representing the sample of data; or, more specifically, the relationship between independent and dependent variables in the dataset. When the results of the curve-fitting are to be used for making new predictions of values of a dependent variable, this process is called regression, which can be used to fit the curve to interpolate a set of data. Curve-fitting can be used to predict parameters of some known model (formula) given a set of observed data. The fitting could be linear or non-linear. To measure the degree of fit, *R2* is used as the coefficient of determination, which is a number that indicates the proportion of the variance in the dependent variable that could be predicted by the independent variable. The value of *R*-squared is a measure of the goodness of fit of the trend line to the data and a value of 1 is a perfect fit. Linear curve-fitting generally superimposes a trend line over the dataset. Multiple regression is used when the dependent variable is affected by more than one independent variable.

Neural networks include a large class of different architectures. In many cases, the issue is to approximate a static non-linear, mapping f (x) with a neural network ANN(x). The most common neural network used in function approximation is the Multilayer Layer Perceptron (MLP). An MLP consists of an input layer, several hidden layers, and an output layer. A node *i*, also called a neuron, in an MLP network includes a summer and a non-linear activation function.

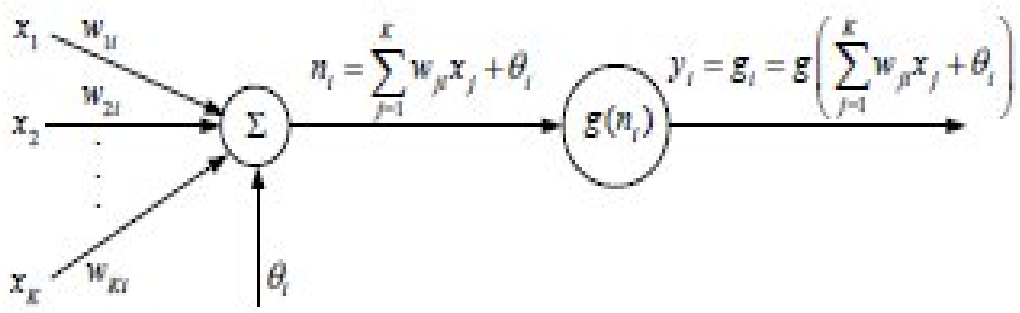


FIGURE 5-27 MLP ARCHITECTURE

The inputs '*x'* to the neuron are multiplied by weights '*w'* and summed together with the constant '*bias'* term. The result is the input to the activation function '*g'*. The activation function as a sigmoid function (or *tanh*) is most commonly used in simulating human brain neuron activation.

$$tanh(x) = \frac{1-e^{-x}}{1+e^{x}} \quad y_i = g_i = g(\sum_{j=1}^{k} w_j x_j + \theta_i) \tag{5-18}$$

An MLP network is formed by connecting several nodes in parallel and series. A multilayer perceptron network with one hidden layer is shown in Figure 5-28. The same activation function *g* is used in both layers.

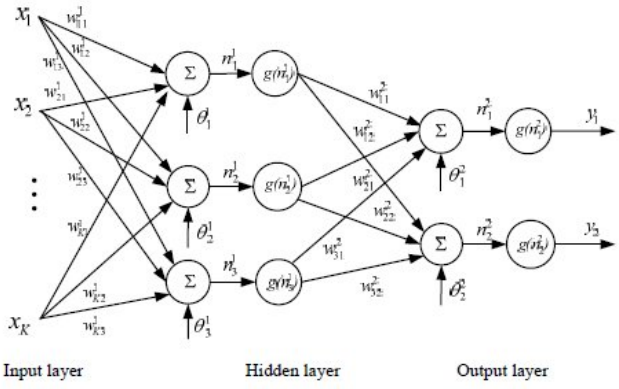


FIGURE 5-28 MLP ONE HIDDEN LAYER ARCHITECTURE

The output of the MLP network is:

$$y_i = g(\sum_{j=1}^{3} w_{ji}^2 g(n_j^1) + \theta_j^2) = g(\sum_{j=1}^{3} w_{ji}^2 g(\sum_{k=1}^{K} w_{kj}^1 x_k + \theta_j^1) + \theta_j^2) \tag{5-19}$$

It can be concluded that an MLP network is a non-linear parameterized map from input to output spaces. The parameters are the weights and the biases. Activation functions are

usually assumed to be the same in each layer and are known in advance. In the example in Figure 5-28 the same activation function is used in all layers. Given input-output data, finding the best MLP network is formulated as a data-fitting problem, where the parameters to be determined are the weights and biases.

First, the designer has to determine the structure of the MLP network architecture in terms of the number of hidden layers and neurons (nodes) in each layer. The activation functions for each layer are also chosen at this stage; that is, they are assumed to be known. The unknown parameters to be estimated are the weights and biases.

Many algorithms exist for determining network parameters. In the neural network literature, these algorithms are called learning or teaching algorithms, whereas in system identification studies they are termed parameter estimation algorithms. The most well-known are the back-propagation and Levenberg-Marquardt algorithms. Back-propagation is a gradient-based algorithm, which has many variants.

The procedure for teaching algorithms for multilayer perceptron networks can be summarized as follows:

a. The structure of the network is first defined. In the network, activation functions are chosen and the network parameters, weights and biases, are initialized.
b. The parameters associated with the training algorithm such as error goal, and maximum number of epochs (iterations), are defined.
c. The training algorithm is called.
d. After the neural network has been determined, the results are first tested by simulating the output of the neural network with the measured input data. The outcomes are compared with the measured outputs. Final validation must be carried out with independent data.

The MATLAB commands used in the procedure are *newff*, *train* and *sim*. The MATLAB command *newff* generates an MLPN neural network, which is called the net. The default

algorithm for the command *newff* is Levenberg-Marquardt (More, 1978), *trainlm*. Default parameter values for the algorithms are assumed and hidden from the user. They need not be adjusted in the first trials. Initial values of the parameters are automatically generated by the command. It should be noted that their generation is random, and therefore the answer might be different if the algorithm is repeated. After initializing the network, network training is initiated using the *train* command. To test how well the resulting trained MLP net approximates the data, the *sim* command is applied. The trained model is then used on a new test dataset with the *sim*.

5.10. One Attribute Regression with Neural Networks

The ANN for time-share price curve-fitting models the relationship between time and price; that is, the dependent variable of price and the independent variable of time. The ANN models fits time-price data to the training curve and then applies it to new samples. First the model is trained with a dataset of fourteen consecutive samples from day 265 (06/01/2014) to day 278 (23/01/2014) of price and time data and it is then tested with a dataset from day 270 (13/01/2014) to day 283 (30/01/2014). There is an overlap between the datasets from day 270 (11/01/2014) to day 278 (23/01/2014) and the actual forecasting of five new samples to the model from day 279 (24/01/2014) to day 283 (30/01/2014). The new forecast values are compared with the actual values in the period outside the training dataset.

5.10.1 Perceptron

The perceptron model architecture with one attribute and one hidden layer is shown in Figure 5-29 and the perceptron algorithm in Figure 5-30. The working dataset with training and testing subsets is given in Appendix C: Table 18. The graphs for the training and testing in Figure 5-31.

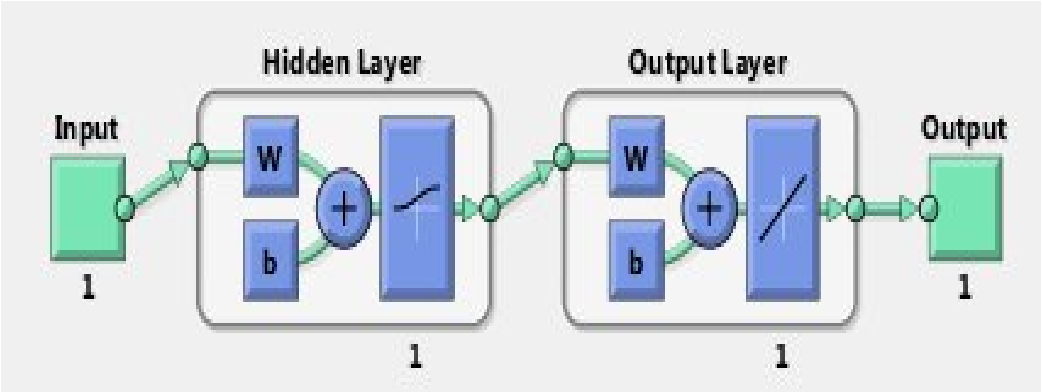


FIGURE 5-29 PERCEPTRON ARCHITECTURE ONE ATTRIBUTE AND ONE HIDDEN LAYER

FIGURE 5-30 PERCEPTRON ALGORITHMS

The following code is used:

net = newff(input_training_set,output_training_set,[1]);

net.layers{1}.transferFcn='logsig'; % transfer function for the neurons in the 1st hidden layer is log sigmoid.

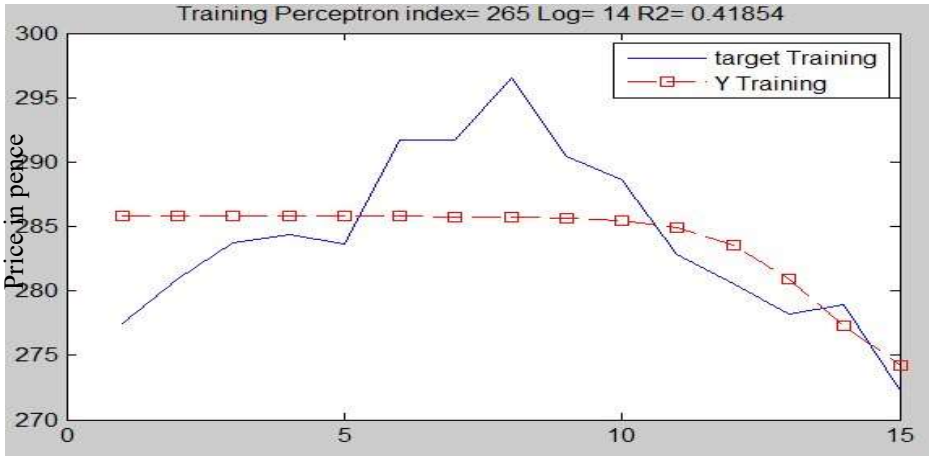
net.layers{2}.transferFcn='purelin'; % transfer function for the neurons in the output layer is linear

net = train(net,input_training_set,output_training_set); % train the network with the training samples.

Y_testing = sim(net,input_testing_set); % compute the output of the trained network

The MATLAB command for the ANN training is

net = train(net,input_training_set,output_training_set); % train the network with the training samples.



The MATLAB comand for the testing is

Y_testing = sim(net,input_testing_set); % compute the testing output of the trained network

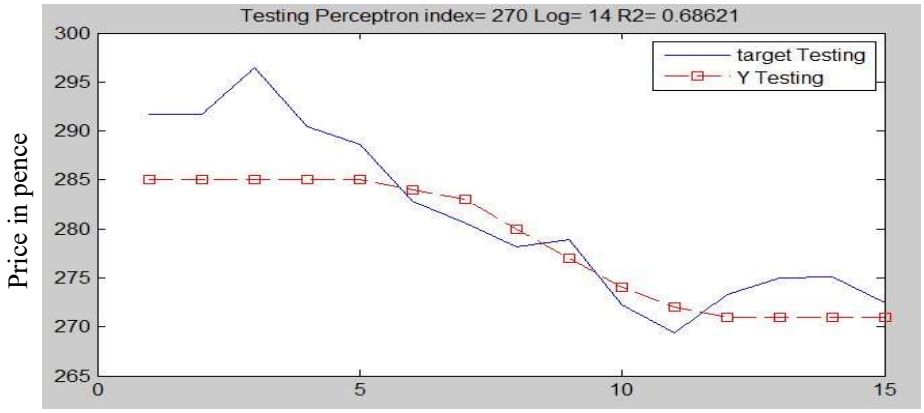


FIGURE 5-31 TRAINING AND TESTING WITH PERCEPTRON

The performance graph for the training of the model are shown in Figure 5-32.

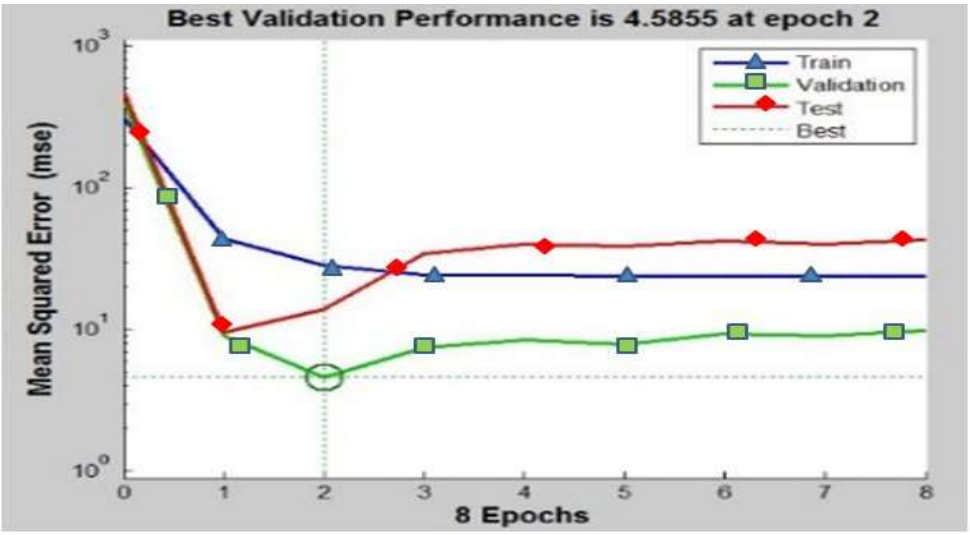


FIGURE 5-32 PERFORMANCE GRAPH PERCEPTRON

The testing and training performance (R^2) with the perceptron model are not very good at 0.41 and 0.68 respectively in terms of graphs matching.

5.10.2 MLPN

For an MLPN model with one attribute and 10 neurons in the first hidden layer and 5 in the second layer, as shown in Figure 5-33. The working dataset with training and testing subsets is given in Appendix C: Table 19.

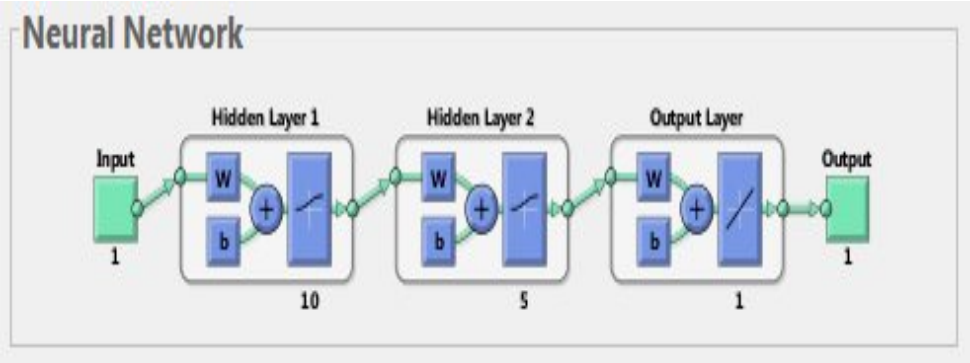


FIGURE 5-33 *MLPN* ONE ATTRIBUTE WITH TWO LAYERS [10, 5]

FIGURE 5-34 *MLPN* ALGORITHMS

The following code is used:

net = newff(input_training_set,output_training_set,[10 5]);

net.layers{1}.transferFcn='logsig'; % transfer function for the neurons in first hidden layer is log sigmoid.

net.layers{2}.transferFcn='logsig'; % transfer function for the neurons in second hidden layer is log sigmoid.

net.layers{3}.transferFcn='purelin'; % transfer function for the neurons in the output layer is linear.

net.trainParam.epochs = 40; % set to 40 the number of times the training samples will be used to train the network

net = train(net,input_training_set,output_training_set); % train the network with the training samples.

% once the training is performed the network can be used to forecast

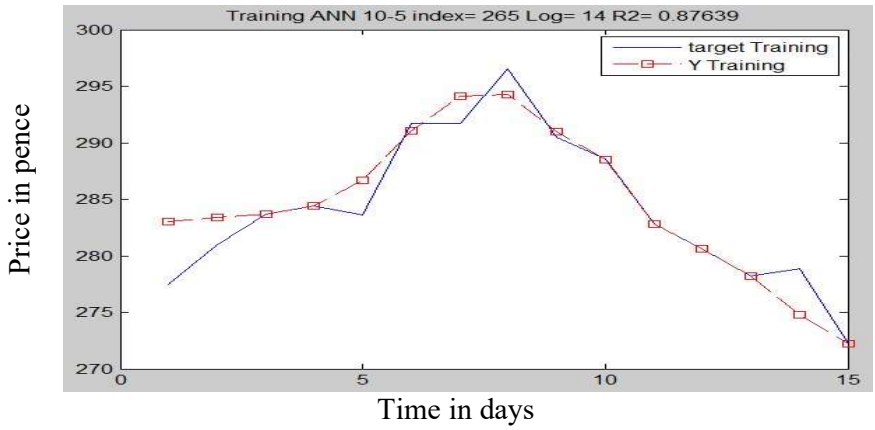
Y_training = sim(net,input_training_set); % compute the output of the trained

Y_testing = sim(net,input_testing_set); % compute the testing output of the trained network

The training and testing charts are shown in Figure 5-35.

The MATLAB command for the ANN training is:

net = train(net,input_training_set,output_training_set); % train the network with the training samples.



The MATLAB comand for the testing is:

Y_testing = sim(net,input_testing_set); % compute the testing output of the trained network

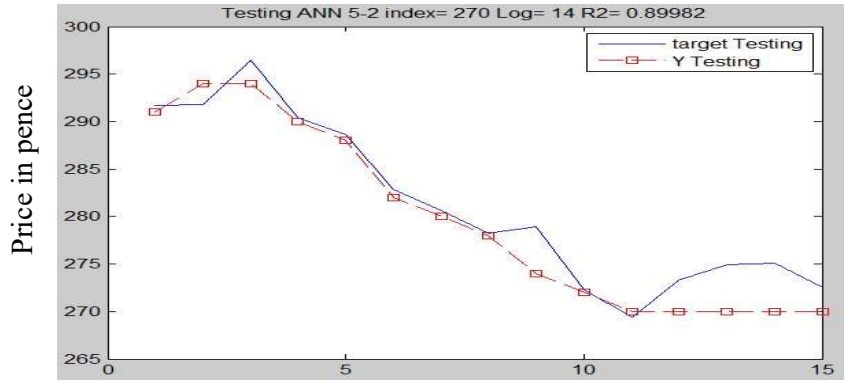


FIGURE 5-35 *MLPN* TRAINING AND TESTING CHARTS

The performance graph for the training of the model are shown in Figure 5-36.

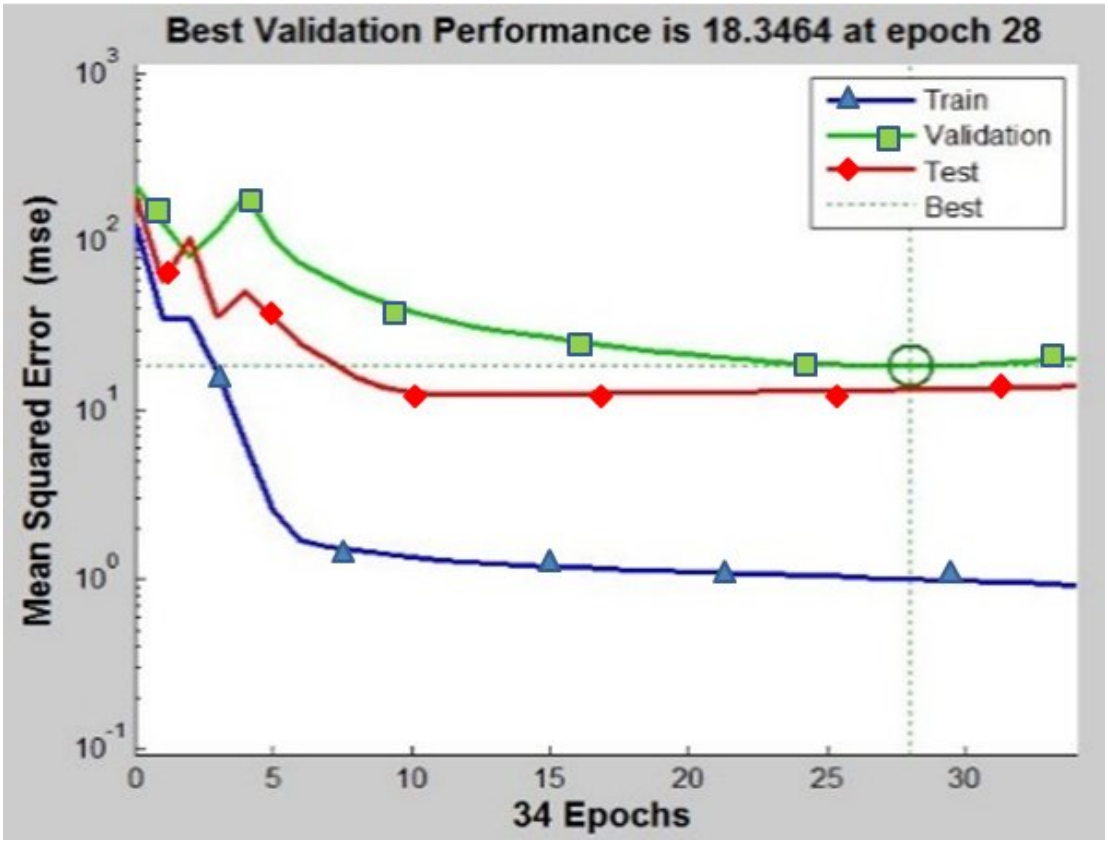


FIGURE 5-36 *MLPN* PERFORMANCE TRAINING

There is a significant improvement in performance (R^2) with the perceptron model with values of 0.41 and 0.68 to *MLPN* values of 0.87 and 0.89.

5.11. Multiple Attribute Regression with Neural Networks

The ANN for multiple attribute regression models the time series share price data by transforming a one-dimensional time series of price and time into an m-dimensional model where $price_i = f(price_{i-1}, price_{i-2}, .., price_{i-w})$. Here the index is time and w is the number of attributes, and the composed input dataset has $[m, m]$ dimension $\overline{price} = \overline{\overline{price}}(\overline{price_{i-1}}, \overline{price_{i-2}}, \ldots \overline{price_{i-w}})$ and uses vector curve-fitting to model $price_i from price_{i-1}, price_{i-2}, .., price_{i-w}$.

An illustrative simple example of how to compile a dataset with four multiple is shown in Table 5-15, Table 5-16 and Table 5-17.

TABLE 5-15 SHARE PRICE TIME SERIES FROM 01/01/2013 TO 10/01/2013

Time	Date	Close
1	01/01/2013	262.4
2	02/01/2013	275.6
3	03/01/2013	276
4	04/01/2013	276.7
5	07/01/2013	287.2
6	08/01/2013	287.2
7	09/01/2013	294.75
8	10/01/2013	294.6

The multiple attributes dataset with a window-lag of four is shown in Table 5-16.

TABLE 5-16 FOUR ATTRIBUTES MODEL

Attr. 1 P(t-4)	Attr. 2 P(t-3)	Attr. 3 P(t-2)	Attr. 4 P(t-1)	target P(t)
P1	P2	P3	P4	P5
P2	P3	P4	P5	P6
P3	P4	P5	P6	P7
P4	P5	P6	P7	P8

The actual values are shown in Table 5-17.

TABLE 5-17 SHARE PRICES WITH FOUR ATTRIBUTES

Attr. 1	Attr. 2	Attr. 3	Attr. 4	target
P(t-4)	P(t-3)	P(t-2)	P(t-1)	P(t)
262.4	275.6	276.0	276.7	287.2
275.6	276.0	276.7	287.2	287.2
276.0	276.7	287.2	287.2	294.75
276.7	287.2	287.2	294.75	294.6

The experimental dataset is for training from 23rd December 2013 to 24th January 2014 and testing from 30th December 2013 to 31st January 2014 as shown in Appendix C:

Table 20. The transformed multiple attributes training dataset is shown Appendix C: Table 21 and the testing set is shown in Appendix C: Table 22.

5.11.1 Perceptron Model

Experiments are conducted with a perceptron model with multiple attributes, and with one hidden layer only, as shown in Figure 5-37. The algorithm used in the perceptron is shown in Figure 5-38.

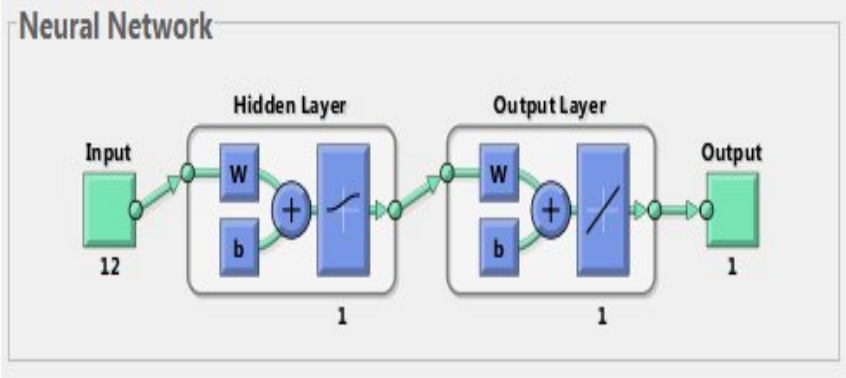


FIGURE 5-37 PERCEPTRON MODEL MULTIPLE ATTRIBUTES

FIGURE 5-38 ALGORITHMS FOR PERCEPTRON MODEL MULTIPLE ATTRIBUTES

The following code is used:

net = newff(input_training_set,output_training_set,[1]);

net.layers{1}.transferFcn='logsig'; % transfer function for the neurons in the 1st hidden layer is log sigmoid.

net.layers{2}.transferFcn='purelin'; % transfer function for the neurons in the output layer is linear

net = train(net,input_training_set,output_training_set); % train the network with the training samples.

Y_testing = sim(net,input_testing_set); % compute the output of the trained network

The target and model training and testing graphs are shown in Figure 5-39. The performance graphs are shown in Figure 5-40.

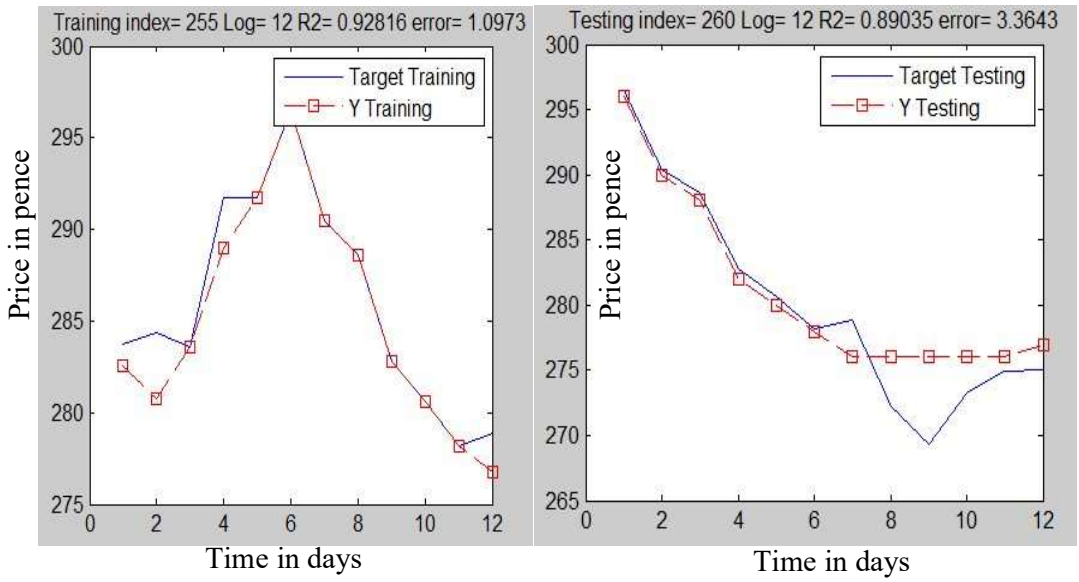


FIGURE 5-39 TRAINING AND TESTING PERCEPTRON WITH MULTIPLE ATTRIBUTES

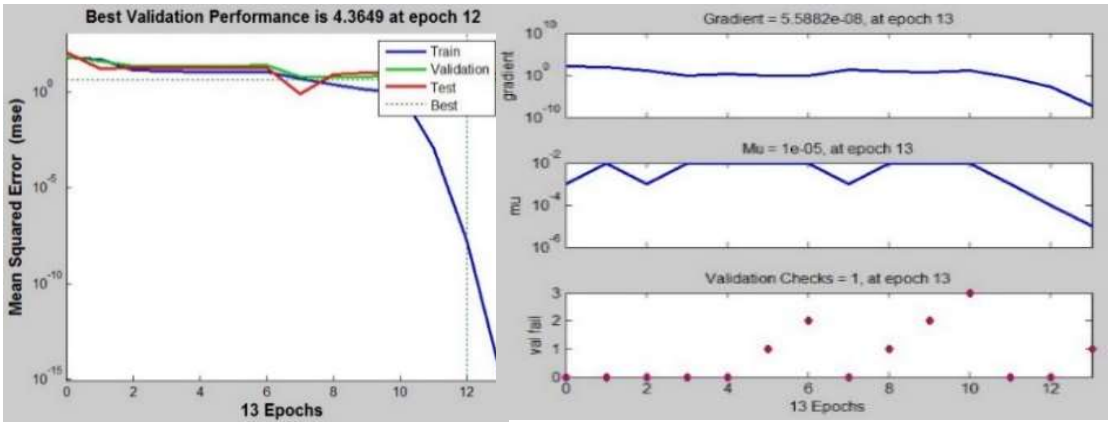


FIGURE 5-40 PERCEPTRON PERFORMANCE WITH MULTIPLE ATTRIBUTES

The results for training and testing performance are 0.92816 and 0.89035 respectively and these are better than with one attribute at 0.41 and 0.68. A disadvantage is that the forecasting requires the previous day's values and so only one day could be predicted, although it could be used to produce a forecast with one attribute and to validate it with multiple attributes.

5.11.2 MLPN Model

An MLPN model with multiple attributes and two hidden layers (10 neurons in the first hidden layer and 5 in the second layer), is shown in Figure 5-41. The algorithm used in the perceptron is shown in Figure 5-42 .

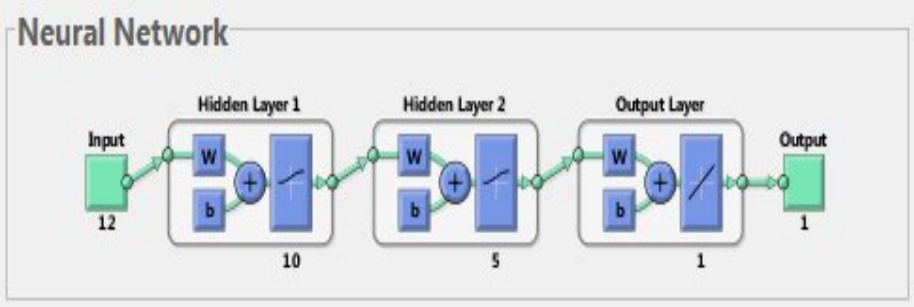


FIGURE 5-41 MULTIPLE ATTRIBUTES MULTIPLE NEURONS LAYERS [10, 5]

FIGURE 5-42 ANN ALGORITHMS

The following code is used:

net = newff(input_training_set,output_training_set,[10 5]);

net.layers{1}.transferFcn='logsig'; % transfer function for the neurons in first hidden layer is log sigmoid.

net.layers{2}.transferFcn='logsig'; % transfer function for the neurons in second hidden layer is log sigmoid.

net.layers{3}.transferFcn='purelin'; % transfer function for the neurons in the output layer is linear.

net.trainParam.epochs = 40; % set to 40 the number of times the training samples will be used to train the network

net = train(net,input_training_set,output_training_set); % train the network with the training samples.

Y_training = sim(net,input_training_set); % compute the output of the trained network

% once the training is performed the network can be used to forecast

Y_testing = sim(net,input_testing_set); % compute the output of the trained network

The target and model training and testing graphs are shown in Figure 5-43. The performance graphs are shown in Figure 5-44.

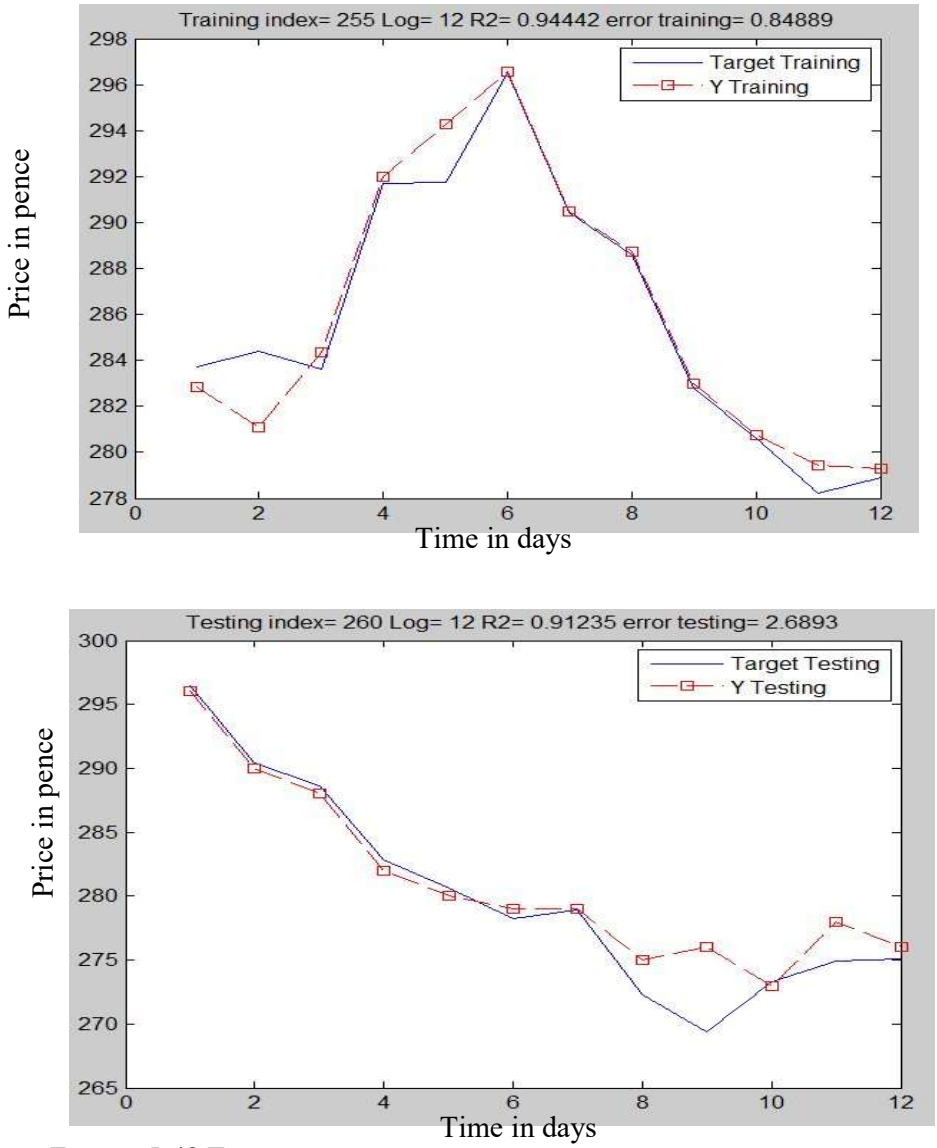


FIGURE 5-43 TRAINING AND TESTING MULTIPLE ATTRIBUTES AND LAYERS

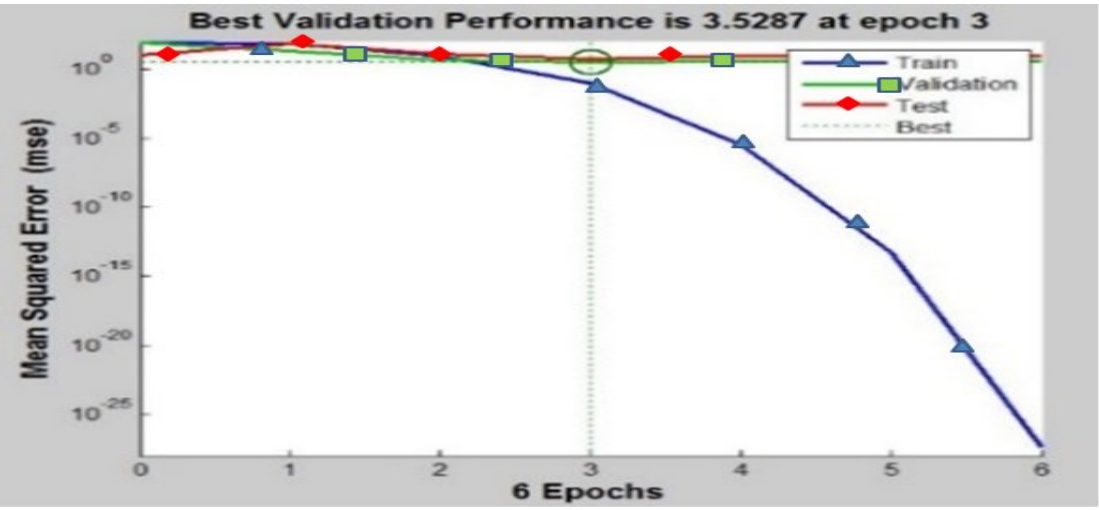


FIGURE 5-44 PERFORMANCE MULTIPLE ATTRIBUTES AND MULTIPLE LAYERS

The training and testing performance values are 0.9444 and 0.9124 respectively and these are better than with one hidden layer, whose performance values are 0.92816 and 0.89035.

5.12. Summary

Some periodicity was apparent in short-term and long-term trading. Decomposition of the share time series revealed trends, harmonics and seasonality in the banking share sector, which helps with generalization and improving the performance of the model.

Statistical probability analysis of the share prices for various time intervals suggests that price time series generally have a normal distribution. Furthermore, short-term datasets have a more deterministic nature than long-term, which justifies stochastic model assumptions and gives a case for the selection of short-term trading.

Statistical independence tests of the datasets with different starting dates show that to have fully statistically independent samples, there should be a shift about two trading weeks between their starting date which is confirmed by the correlation analysis as well. The correlation analysis also show that share open price, close price and the rest of the available daily data move together, which allows the number of independent variables to be reduced and just one, such as the closing share price, can be used.

The results of the experiments with neural networks with multiple attribute models show notably better performance than common practice regression models.

Chapter 6. Discrete Fourier Transform

6.1. Overview

This chapter extends the financial time series modelling in the time domain, specifically focusing on the discrete Fourier transform (DFT) analysis, forecasting and validation. This includes a contribution to knowledge with multiple DFT features composition for neural network utilization for the short-term trading. Furthermore a novel fitting methodology is proposed for its application as well.

Spectral analysis, also known as frequency domain analysis, decomposes a time series into a spectrum of cycles of different lengths and it can be used for the analysis of a time series, filtering and forecasting.

This chapter covers the following:

- Experimental definition
- Experimental definition of the algorithm in Excel
- Experimental definition in MATLAB
- Forecasting exploration
- DFT and ANN for regression dataset
- Standard error of the Mean (SEM) calculations
- Coefficient of determination (R-square) calculations
- Fast and inverse Fourier transforms
- DFT investigations
- Time domain
- Power distribution frequencies

6.2. Introduction

The Discrete Fourier Transform (DFT) decomposes a time series into sine and cosine components. The DFT of *f(t)* denoted by *F(u)*, is given by:

$$F(u) = \sum_{t=0}^{N-1} f(t) e^{-i\,2\pi\,u\,\frac{t}{N}} \quad (6\text{-}1)$$

The frequency domain is simply the coordinate system spanned by *F(u),* with *u* as the frequency variable. This is analogous to the time domain, which is the coordinate system spanned by *f(t)*. The transform is complex. If *R(u)* and *I(u)* represent the real and imaginary components of *F(u),* the frequency spectrum is defined as:

$$|F(u)| = \sqrt{R^2(u) + I^2(u)} \quad (6\text{-}2)$$

The phase angle of the transform is defined as:

$$\varphi(u) = \tan^{-1}\left(\frac{I(u)}{R(u)}\right) \quad (6\text{-}3)$$

The discrete Fourier transform allows us to manipulate time series data either in the frequency or time domains. DFT components represent the input dataset as the sum of the trigonometric sine-cosine functions. The main motivation for its use is to extrapolate and forecast beyond the input dataset. A Fourier transform produces the same number of frequency bins, or bands, as in the time series samples. The time series has to be centred in order to be de-trended and its FFT algorithm is simplest by far if *N* is an integral power of 2 (2, 4, 8, 16, ...). The latter is the only requirement of the most popular implementation of this algorithm (Radix-2 Cooley-Tukey) (Cooley, 1965) where the number of points in the series should be a power of 2.

The Fourier transform produces complex numbers, where *DFT z = x + yi*. The power spectrum represents the distribution of the frequency content of the time series. The power *p* of a frequency band is the absolute value (modulus) of the corresponding complex number $p = \frac{|z|^2}{n^2}$. The series amplitude is symmetrical around the $\frac{N}{2}$ component. From the power plot, the major contributing frequencies (maximum plot peaks) can be identified. The corresponding periods are $T = \frac{i}{n}$, where i is the frequency bin on the plot peak and n is the size of the time series. For example, if there are 32 samples in the time series and the plot peak is at frequency bin (sample number) four, then the period will be 8 (8=32/4).

The example in Figure 6-1 displays an example of the magnitude and phase plots of the DFT for an example of a time series.

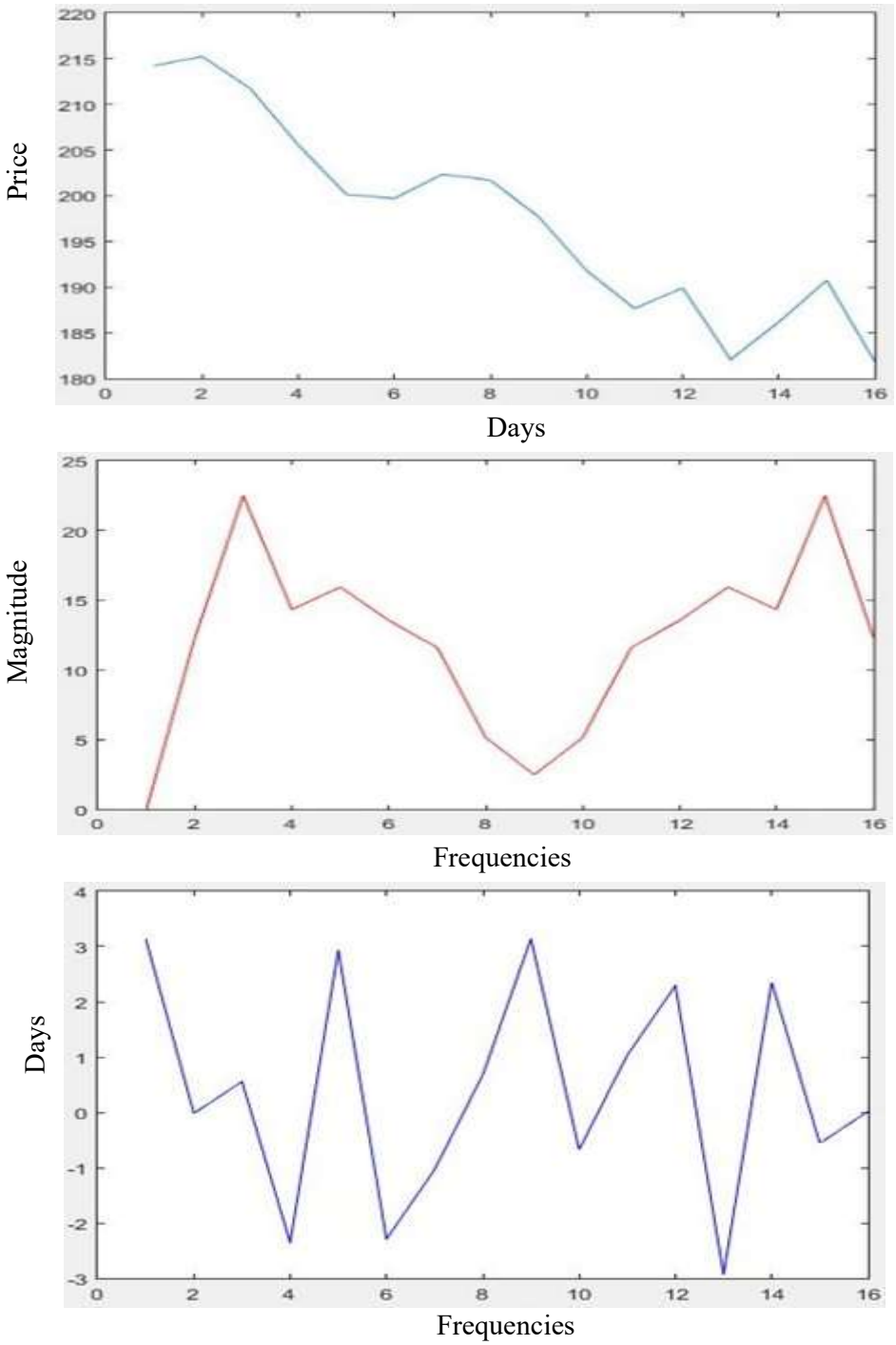


FIGURE 6-1 DFT PRICE, MAGNITUDE AND PHASE

The further away from the start of the time series period a share is, the higher its corresponding frequency will be, such as for sample number 4 of 32 samples time series it is 4/32 (period 8) and for sample number 16 it is 16/32 (period 2). The magnitude variable

measures the change in values between neighbouring prices; for example, the change from 240.53 to 250.13 is 9.6. The phase measures the shift from the origin at the corresponding frequency, such as 2 sample periods.

The results show that the magnitude spectrum plot contains components of all frequencies, but that their magnitudes become smaller for higher frequencies. Hence, low frequencies contain more time series information than higher ones. Share price time series are low-frequency signals. The display of the phase plot does not yield much new information about the structure of the share price time series in the time domain; however, the phase in formation is needed to reconstruct the original time series.

The main properties of the DFT are as follows:

- Completeness (it is invertible).
- De-correlation (DFT coefficients are not correlated).
- Energy compaction (the energy in financial share price time series is mostly grouped in low frequencies in the DFT domain).
- Invariance (the spectrum is invariant to shifting).
- Robustness (DFT coefficients could be robust against many time series processing operations such as noise).

The main disadvantage of Fourier extrapolation is that it merely repeats the series with a period *N*, where *N* is the length of the original time series.

The Discrete Cosine Transform (DCT) is an alternative digital transformation. It represents a time series as a sum of cosine functions of varying magnitudes and frequencies, as follows:

$$F(u) = \frac{2}{N} C(u) \sum_{t=0}^{N-1} f(t) \cos\left(\frac{\pi t}{N}(t + \frac{1}{2})\right) \quad (6\text{-}4)$$

$$c(0) = \frac{1}{\sqrt{2}} C(t) = 1, if t \neq 0 \quad (6\text{-}5)$$

The values of the DCT are real (negative and positive). The DCT has the property that, for a typical time series, most of the significant information about it is concentrated in just a few coefficients of the DCT. The example in Figure 6-2 displays an example of the magnitude and phase plots of the DFT for the time series example.

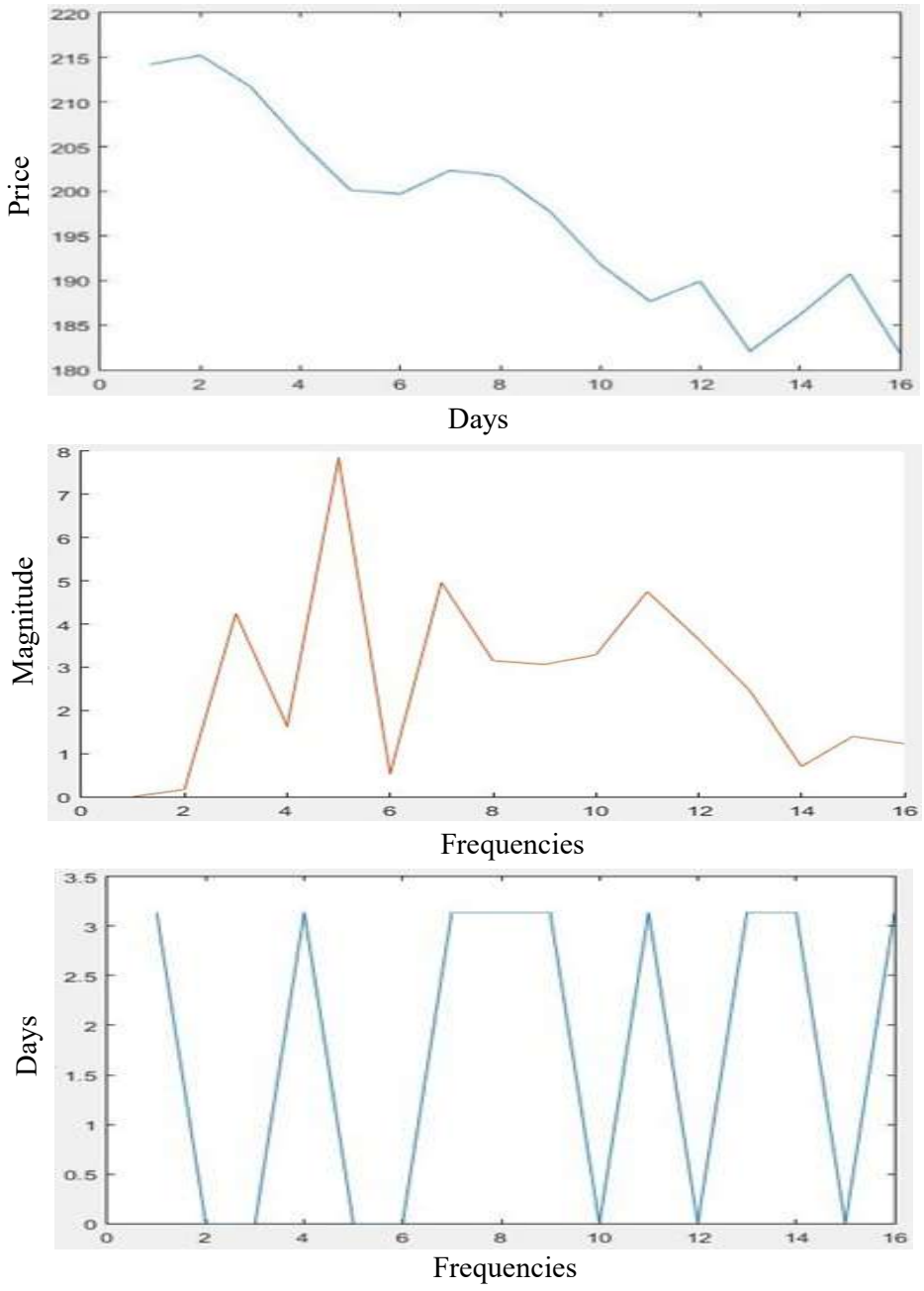


FIGURE 6-2 DCT PRICE, MAGNITUDE AND PHASE

The main properties of the DCT are as follows:

- Completeness (the DCT is invertible).

- De-correlation (it removes redundancy between neighbouring time domain points).
- Energy compaction (the energy of the image is concentrated in the low frequency region.
- Robustness, the DCT coefficients can be robust against many processing operations such as noise.

6.3. Experimental Definition

The following time series samples are taken every day for the closing price of a share for a total of N=16 samples. The data making up this time series is shown in Appendix C: Table 23 and the share price graphs in the original Figure 6-3 and centred Figure 6-4. These are a portion of a bigger time series of 153 samples as shown in Figure 6-5.

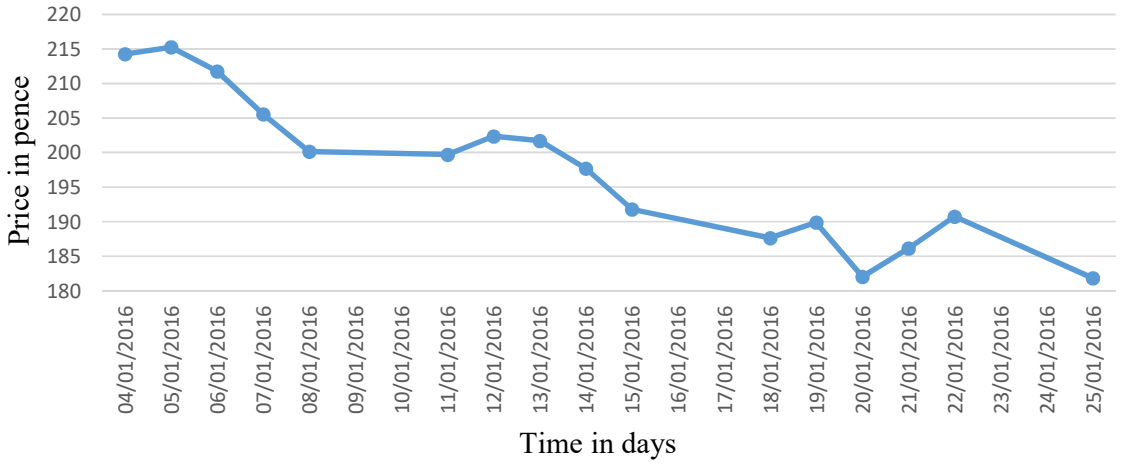


FIGURE 6-3 TRADING SHARE PRICES 16 SAMPLES

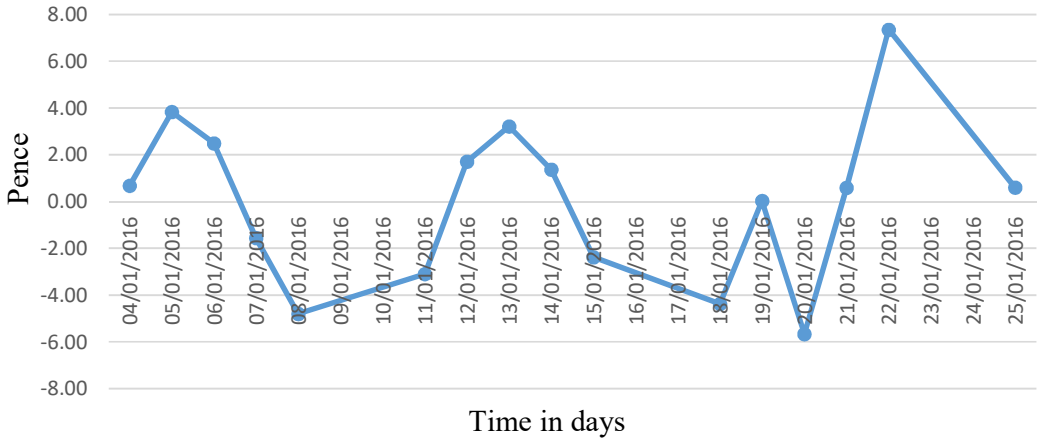


FIGURE 6-4 TRADING SHARES CENTRED 16 SAMPLES DATASET

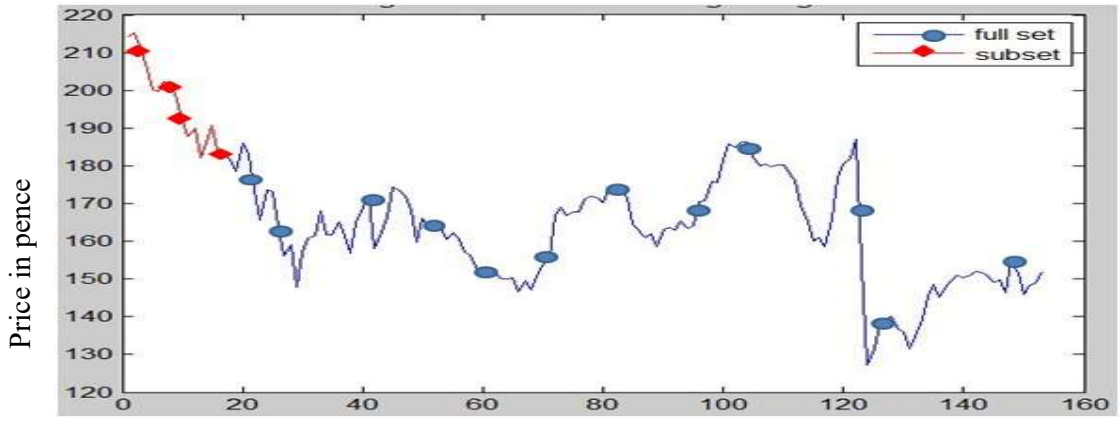


FIGURE 6-5 TRADING SHARES FULL 153 SAMPLES TIME SERIES

The first column in Appendix C: Table 23, labelled "Date" contains the date at which each sample was taken. The second column, labelled "Frequency cycle", contains the frequencies in standard units of the sample per cycle - index/cycle. This information will be used in the model as a time parameter. The third column, labelled "n", contains an index identifying the sample number. The fourth column, labelled "Frequency trading", contains the frequencies in normalised yearly cycle - index/252, where the trading day is the time span for which a particular stock exchange is open and it is accepted that there are 252 trading days in one year. Finally, the fifth column, labelled "Y", contains the sampled vertical ordinate. If the time series exhibits a significant trend, it is recommended to de-trend it first.

A Fourier transform produces the same number of frequency bins, or bands, as time series samples, and so there are 16 frequency bins in this time series. The size of the range of the time series should be a power of 2 (for example, 2^1, 2^2, 2^3, 2^4, and so on). If the time series data is not a power of 2, then the end of the series should be padded to the next power of 2. Time series data is often padded on one end before being transformed, due to the so-called wraparound effect that can distort results when convolving data. The padded values of the time series should be 0 if the original time series is centred or de-trended. If the time series is not centred, then the average value of the time series ordinates can be used as the pad value.

The column DFT(Y) in Appendix C: Table 24 contains the discrete Fourier transform of the time series generated with the Fourier analysis tool in MS Excel. These are complex numbers, and looking, for example, at 05/01/2016, the second sample in the time series, it appears like this: 12.2802009957215-0.228725990422478i. At this point, this is the frequency domain version of the original time series. It can be manipulated in many different ways to suit the type of analysis being conducted.

To generate a power spectrum of the time series, which is used to analyse its frequency content, the power contained in each frequency band has to be calculated. To compute the power contained in each frequency band p_i, the absolute value of a complex number DFT_i (modulus $|DFT_i|$ Excel function IMABS) to the power of two is divided by the size N of the time series to the power of two, $p_i = \frac{{DFT_i}^2}{N^2}$. For example, the power of the second frequency band of 0.0625 sample/cycles is 0.589279891. The use of a factor of 2 for all intermediate frequency power calculations is due to the periodicity of the Fourier transform. The power is calculated for each frequency band up to the highest frequency, the Nyquist frequency (0.5 sample/cycles). The calculations are shown in Appendix C: Table 24. The column "*power(Y)*" contains the power spectrum for this time series. The plot of the power spectrum is shown in Figure 6-6.

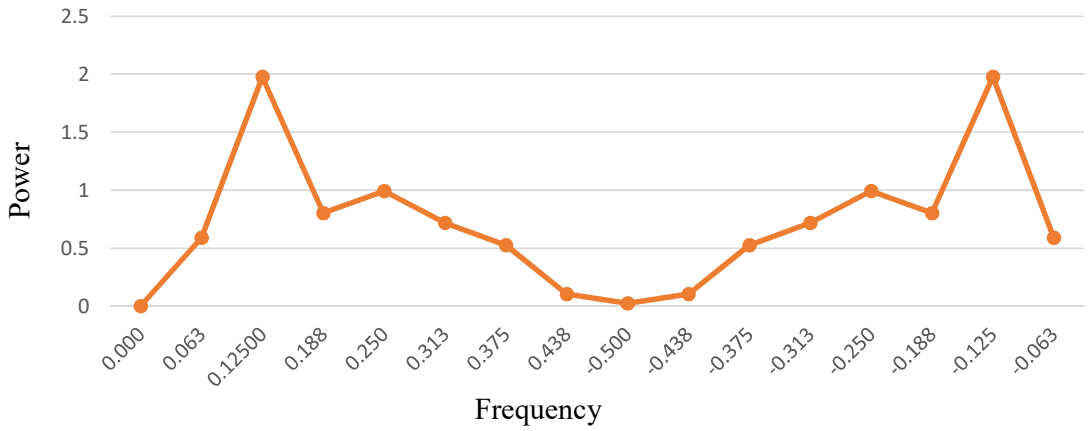


FIGURE 6-6 POWER FREQUENCY DISTRIBUTION (PFD) OF THE CENTRED PLOT

There is a strong component of 1.977 at a frequency of about 0.125. Furthermore, there is a clear contribution of 0.993 at a frequency of 0.250, although this is not as strong as the one

at 0.125, which should be considered in an eventual frequency approximation model for this share price time series example.

To conduct further analysis, a frequency filter can filter the data in the frequency domain to isolate the 0.125 component of the original series. Firstly, an appropriate filter has to be constructed and applied and then an inverse Fourier transform is performed to show the resulting filtered series in the time domain. A suitable filter for this example would be a band-pass filter centred at a frequency of 0.125 samples per cycle. This is a standard Gaussian band-pass filter with f_0=0.125 and $\varphi = 0.07$ as shown in Figure **6-7**. PFD of the centred and Gaussian filtered time series is shown in Figure 6-8. Finally, the filtered and the original share price are shown in Figure 6-9.

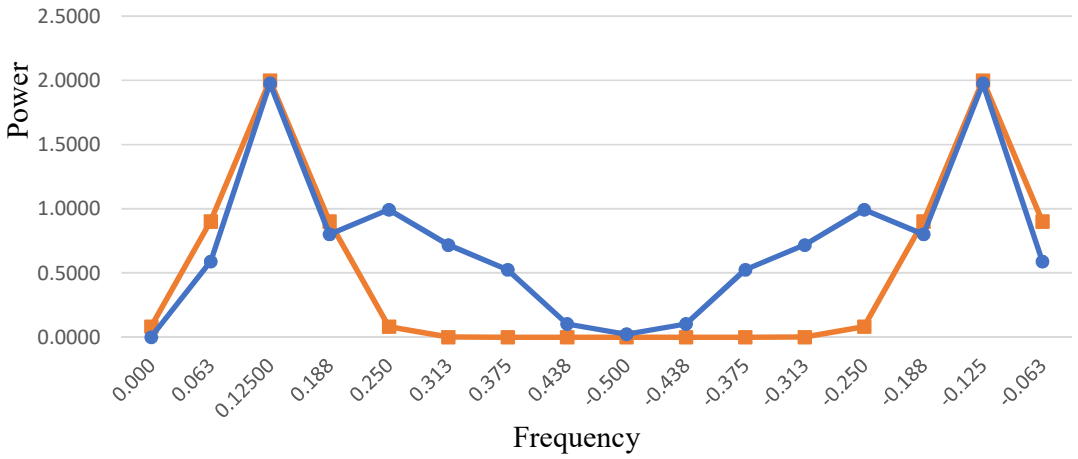


FIGURE 6-7 PFD OF THE CENTRED AND GAUSSIAN FILTERED TIME SERIES

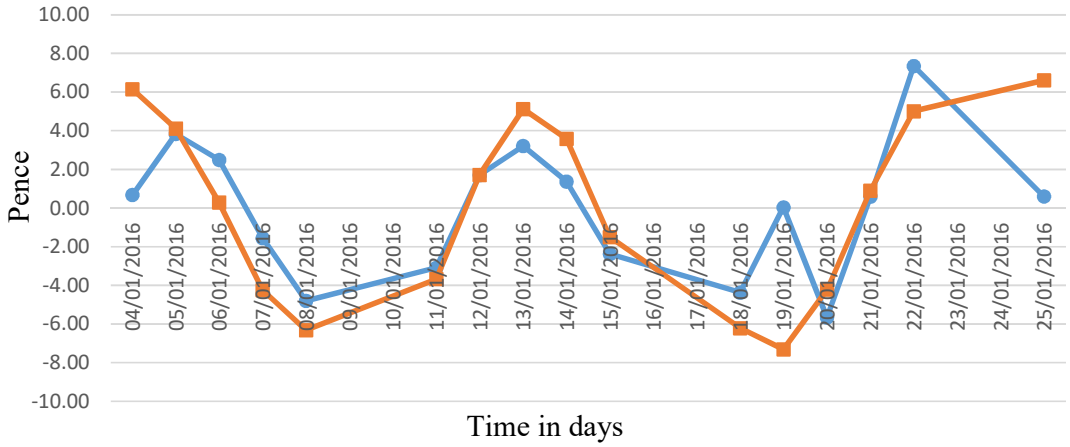


FIGURE 6-8 CENTRED AND FILTERED TIME SERIES CHART

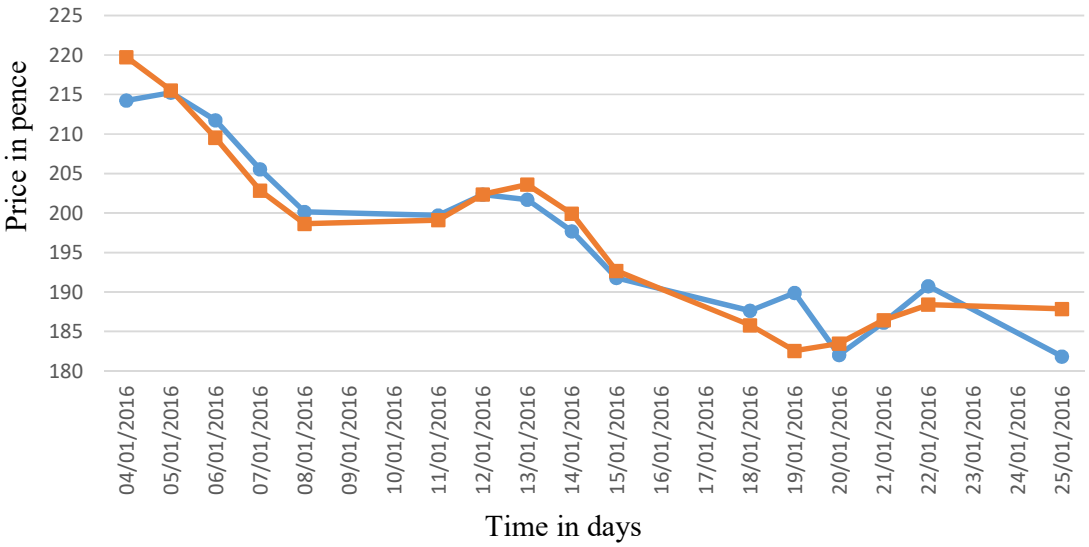


FIGURE 6-9 THE FILTERED ORIGINAL TIME SERIES CHART

The graph shows that the filter removes the high frequencies from the original signal and is a low band-pass filter. The resulting time series have less small changes in the amplitude as the original signal is smoothed.

6.4. Experimental Definition of the Algorithm in Excel

The discrete Fourier transform may be used to identify periodic structures in time series data. Suppose that a physical process is represented by the function of time, *h(t)*. The function is sampled at N times, *h(k)*. From these measurements, complex amplitudes are determined, which satisfy the following equation:

$$H_n = \sum_{k=0}^{N-1} h_k e^{ik\frac{2\pi n}{N}} \quad (6\text{-}6)$$

The sampled function then has the discrete Fourier expansion

$$h_k = \frac{1}{N}\sum_{n=0}^{N-1} H_n e^{-i\ \frac{2\pi k}{N}} \quad (6\text{-}7)$$

This equation can be cast in a familiar form with $\frac{2\pi}{N} = \frac{2\pi}{T_0} k \frac{T_0}{N} = \omega_0 k\Delta t = \omega_0 t_k$:

$$h_k = \frac{1}{N}\sum_{n=0}^{N-1} H_n e^{-inw_0 t_k} \quad (6\text{-}8)$$

The right-hand side is the discrete analogue to the complex form of the Fourier expansion:

$$h(t) = \sum_{n=-\infty}^{\infty} c_n e^{in\ _0 t_k} \quad (6\text{-}9)$$

where the complex coefficients are given by:

$$c_n = \frac{1}{T_0}\int_0^{T_0} h(t)e^{-in\omega_0 t}\,dt \qquad (6\text{-}10)$$

The Excel data analysis package has a Fourier analysis routine, which calculates complex coefficients from time series data. The routine requires that the number of samples in the time series data be of a power of 2.

To select Fourier analysis, click on Data in the Excel menu bar, and select Data Analysis. In Data Analysis, select Fourier Analysis, and a simple dialog box appears. Make sure that the "Inverse" box is not checked (selected).

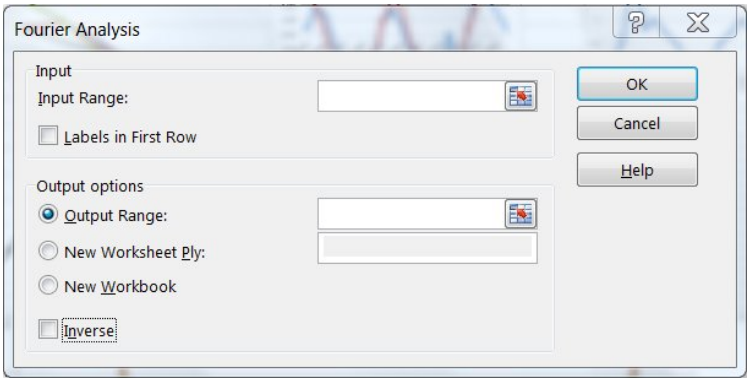


FIGURE 6-10 EXCEL DATA ANALYSIS, FOURIER ANALYSIS MENU

For Input Range, enter the location of the time series data, and for output range enter a convenient place on the worksheet. After selecting the OK button, Excel returns the complex coefficients in the selected output column.

The power in each frequency bin is proportional to the square of the magnitude of the complex coefficient normalized with (divided by) the square of the size of the time series, $p = \frac{|z|^2}{N^2}$. The absolute value of a complex number $|z|$ is accomplished with the function IMABS(z). Applying the IMABS (z) function to the complex coefficients z and dividing by N produces the magnitude of the Fourier coefficients, where $magnitude = \frac{|z|}{N}$. The power at each frequency (except for zero frequency) is the square of the magnitude.

To select Inverse Fourier Analysis, click on Data in the Excel menu bar, and select Data Analysis. In Data Analysis, select Fourier Analysis, and a simple dialog box appears. Make sure that the “Inverse” box is checked (selected).

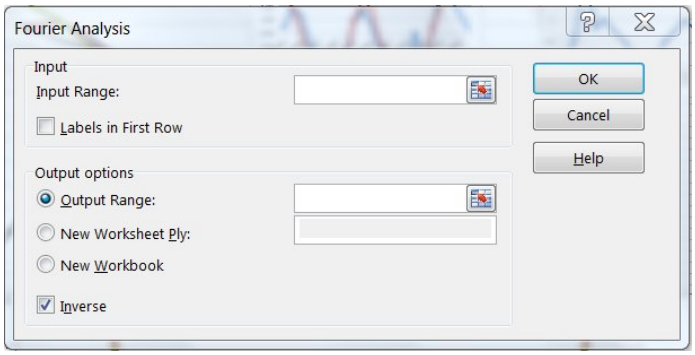


FIGURE 6-11 EXCEL DATA ANALYSIS INVERSE FOURIER

For Input Range, enter the location of the complex coefficient, and for output range enter a convenient place on the worksheet. After selecting the OK button, Excel returns the Inverse Fourier Transform in the selected output column. The results of the inverse transform are real numbers; however, Excel formats them as text (that its, as the real part of an imaginary number). The *IMREAL* function is used to convert these imaginary numbers into values which can be used.

The multiplication of two complex numbers is accomplished with the function *IMPRODUCT(z1, z2)*. This is used when applying a Gaussian filter to the complex coefficient in order to isolate a specific frequency.

6.5. Experimental Definition in MATLAB

A common use of Fourier transforms is to find the frequency components of a signal buried in a noisy time domain signal, such as is the case with share price data.

The DFT takes a discrete signal in the time domain and transforms that signal into the discrete frequency domain. The DFT is only defined in the region between 0 and the sampling frequency f_s. When the range [0, f_s] is examined, it can be seen that there is an even symmetry around the centre point, 0.5 f_s, which is the Nyquist frequency. This symmetry adds redundant information.

The DFT of a vector *x* of length *n* is another vector y of length *n*:

$$y_{k+1} = \sum_{n=0}^{N-1} \omega_N^{-k} \; x_{n+1} \tag{6-11}$$

where ω is a complex nth root of unity:

$$\omega = e^{-2\pi j/N} \tag{6-12}$$

This notation uses *j* for the imaginary unit. Data in the vector *x* are assumed to be separated by a constant interval in time or space, $dt = \frac{1}{f_s}$, where f_s is the sampling frequency. The DFT *y* is complex-valued. The absolute value of *y* at index *p+1* measures the amount of the frequency present in the data.

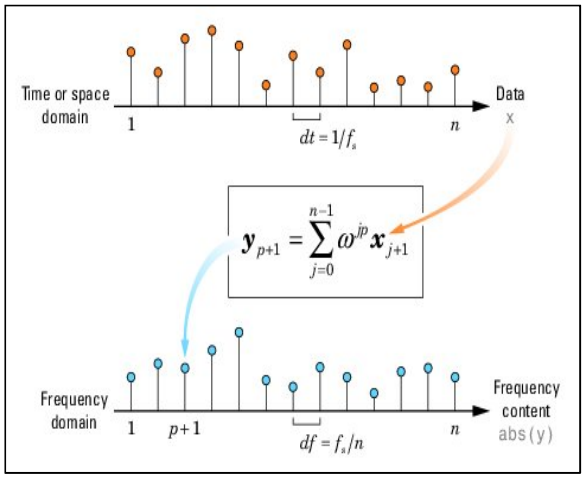


FIGURE 6-12 TIME TO FREQUENCY DOMAIN TRANSFORMATION MODEL

The first element of *y*, corresponding to zero frequency, is the sum of the data in *x*. This data component is often removed from y so that it does not obscure the positive frequency content of the data.

When using FFT algorithms, a distinction is made between window length and transform length. The window length is the length of the input data vector. The transform length is the length of the output, which is the computed DFT. An FFT algorithm pads or chops the input to achieve the desired transform length.

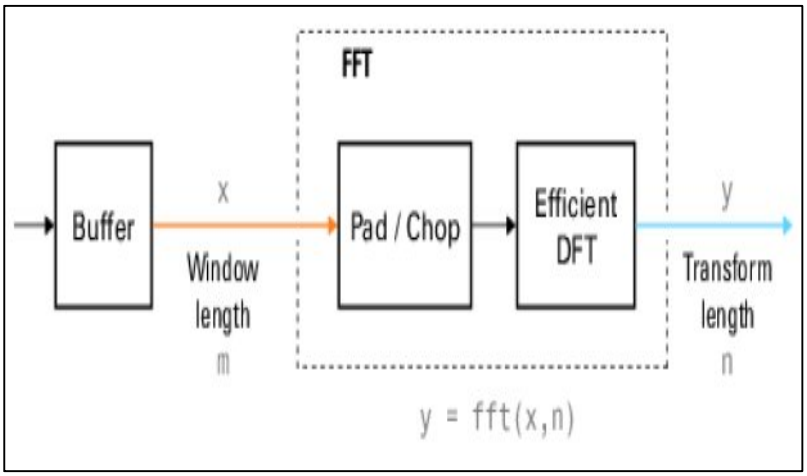


FIGURE 6-13 FFT ALGORITHM DIAGRAM

The FFT is a faster version of the Discrete Fourier Transform (DFT).

The MATLAB *fft* function returns the DFT *y* of an input vector *x* using a fast Fourier transform algorithm:

y = fft(x); (6-13)

In this call to *fft*, the window length *m = length(x)* and the transform length *n = length(y)* are the same.

The transform length is specified by an optional second argument:

y = fft(x,n); (6-14)

If the length of *x* is less than *n*, *x* is padded with trailing zeroes to increase its length to n before computing the DFT. If the length of *x* is greater than n, only the first n elements of x are used to compute the transform.

The basic spectral analysis using FFT allows the component frequencies in data to be efficiently estimated from a discrete set of values sampled at a fixed rate. Relevant quantities in a spectral analysis are listed in Table 6-1.

TABLE 6-1 RELEVANT SPECTRAL ANALYSIS QUANTITIES

Quantity	Description
x	Sampled data
m = length(x)	Window length (number of samples)
fs	Samples/unit time
dt = 1/fs	Time increment per sample
t = (0:m-1)/fs	Time range for data
y = fft(x,n)	Discrete Fourier transform (DFT)
abs(y)	Amplitude of the DFT
(abs(y).^2)/n	Power of the DFT
fs/n	Frequency increment
f = (0:n-1)*(fs/n)	Frequency range
fs/2	Nyquist frequency

Consider the following 'Close' price data in Appendix C: Table 25 and with a graph as shown in Figure 6-14.

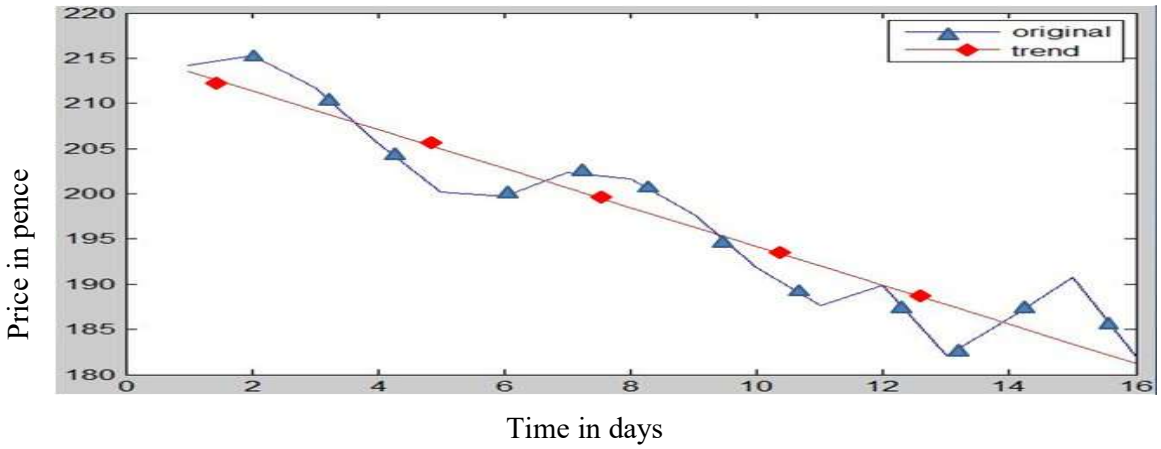


FIGURE 6-14 ORIGINAL AND TREND LINE SHARES TIME SERIES

The DFT, *dft* of *y* ('Close' price) is computed using *fft* with *dft=fft(y).*The power of the DFT *y* and its *Power* are then computed *Power=abs(dft)* as shown in Appendix C: Table 24 and the graph as shown in Figure 6-15.

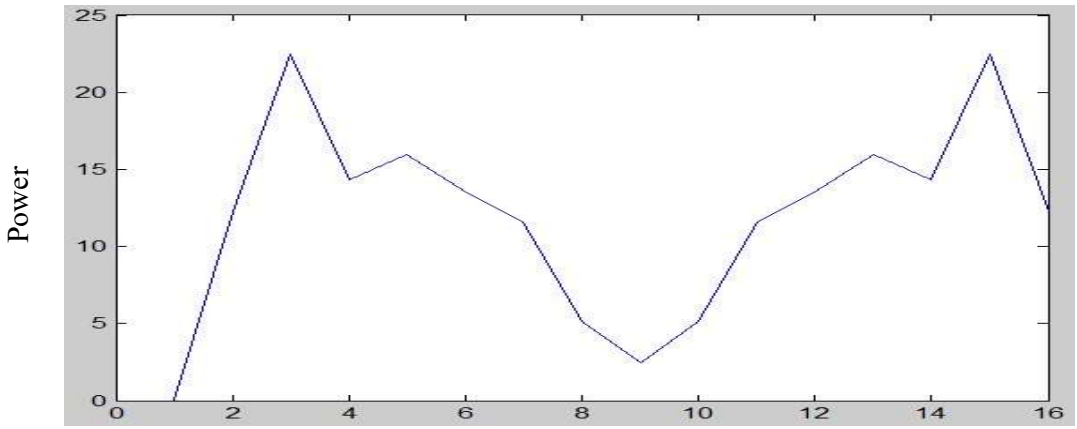


FIGURE 6-15PDF PLOT OF THE SHARE TIME SERIES

6.6. Forecasting Exploration

For a time series that is known to have seasonal or daily patterns, Fourier analysis could be used for forecasting. After running a discrete Fourier transformation *fft* on time series data and obtaining coefficients, these could be used to make predictions.

The FFT assumes that all data it receives constitute one period and then, if the coefficient data are simply expanded appropriately, it also regenerates the continuation of the time series, and so using *ifft* these values can be used for prediction. Simply put, *fft* is run for t=0, 1, 2, ... 10 then *ifft* is used on the coefficients to regenerate the time series for t = 11, 12, … 20.

Generally, the process involves finding "patterns" in the time series, finding the dominant frequency components in the observed time series, taking the Fourier transform and preserving the most important, for example the largest coefficients, and eliminating the rest. Preserving the most important coefficients has a "smoothing", "blurring", or "de-noising" effect on the signal as well. Then, to expand time series Fourier coefficients *Y* to twice the size of the most important coefficients, *Z* are repeated in every second element in the expanded *Y;* to be exact, *Y(1,1)= Z(1,1), Y(1,2) = 0, Y(1,3)= Z(1,2)*. Then a following *ifft*(*Y)* regenerates the expanded time series *y*. The new series will be twice as long as the original. Note that the amplitude has to be multiplied by 2 if x is expanded twice. The algorithm is illustrated with code below.

```
window_expand=2*window;

X = fft(x) % DFT original signal x

Y = zeros (1, window_expand) % expanded

% an algorithm for finding important frequencies in X, to be defined

% Z [important frequencies] = X [important frequencies] % to be defined

Z=X; % original frequencies (until the "important" frequencies are defined)
```

```
Y(1:2:end) = Z % every second element in Y is equal to next Z (or the original X)
y=2*real(ifft(Y)); % it has to be multiplied by 2 because of twice period
figure; hold on;
plot(x);
plot(y,'-r'); %
```

Alternatively, adding up all the harmonics *k* corresponding to the coefficient indices *y = sum(y(k))* will essentially expand the original time series, where *y(k)= Amplitude(k)cos(Period(k)+Phase(k))*, and *Amplitude(k)*, *Period(k)* and *Phase(k)* are the *Amplitude(k)*, period and the phase of the k_{th} harmonic respectively. The algorithm is illustrated with code below,

```
y=zeros(1, window_expand);
for i=1:max_frequency,
   period=i;
   phase=angle(Z(period+1));
   amplitude=abs(Z(period+1))/window*2;
   d1=amplitude*cos(period*((0:window_expand-1)/window*2*pi)+phase);
   y=y+d1;
end
plot(y,'-k'); %
```

The main disadvantage of Fourier extrapolation is that it merely repeats the original time series with the period *N*, where *N* is the length of the original time series as shown in Figure 6-16 which repeats the graph 1-15 to 16-31 days.

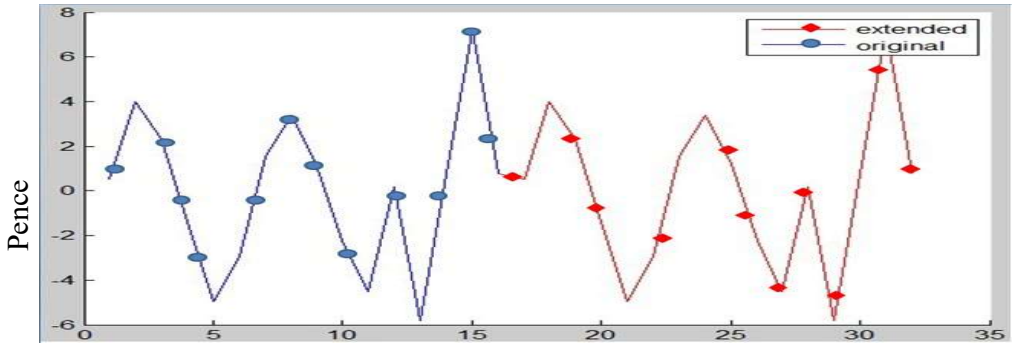


FIGURE 6-16 DFT REPETITION FORECASTING EXAMPLE

The visual example in Figure 6-16 demonstrates the weakness of the method. The frequency domain, by its nature, produces fixed cycles in the time domain. The red line (from day 17 to day 32) in the extrapolation above is simply a copy of the beginning segment of the blue (observed) line (from day 1 to day 16), although de-noised slightly. Therefore, to perform any meaningful short-term prediction over *h* time units in the future, where *h*<*the number* of historical observations, only the most significant high frequency coefficients should be used in the prediction. A "high" frequency threshold can be arbitrarily defined in relation to *h*.

To further clarify this, the extrapolated hump at time ~ day32 in Figure 6-16 is just a copy of the blue hump at ~ day15. If the historical period had instead been initiated right before the hump at 15, then the very first predicted units would have that copy of the hump, which seems pointless and arbitrary. Thus, by either eliminating or down-weighting the high frequency components, the arbitrariness induced by the starting point in the historical data can be reduced.

Usually, modelling assumes that the future will behave as the model suggests. Models depend on parameters, which are estimated using present or past observations, and so "predicting" is actually merely fitting a model. Usually in a signal, there are some frequencies that have significantly higher amplitudes than others, and so selecting these frequencies allows the periodic nature of the time series to be identified.

6.7. DFT and ANN for Regression Dataset

Using a dataset of about ten time series 8 or 16 days long, the DFT is calculated for each of the time series and is the target of the neural network. Time series values are the features. The output DFT of the most recent (the last) training model is used for forecasting. In the example in Table 6-2, $sample_5$ is with the most recent data, and sox_{12} is the latest data and the neural network model output $modelDFT_5$ would be used for forecasting.

TABLE 6-2 NEURAL NETWORK FOR REGRESSION

	$attr_1$	$attr_2$	$attr_3$	$attr_4$	$attr_5$	$attr_6$	$attr_7$	$attr_8$	Target	Model
$sample_1$	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	DFT_1	$modelDFT_1$
$sample_2$	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	DFT_2	$modelDFT_2$
$sample_3$	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	DFT_3	$modelDFT_3$
$sample_4$	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	DFT_4	$modelDFT_4$
$sample_5$	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	DFT_5	$modelDFT_5$

6.8. Standard Error of the Mean and Coefficient of Determination

The standard error of the mean (SEM) describes the certainty of the mean in the underlying population based on its sample. The SEM is the theoretical standard deviation of the sample-mean's estimate of a population mean. The SEM is more informative when converted into a confidence interval. With a confidence interval, assuming normality, there is an X% chance that the underlying population mean falls within certain limits. The limits can be calculated for any certainty level. A 95% confidence interval means that there is a 95% chance that the underlying population mean falls within that certain range of values. To calculate it, the SEM is simply scaled by the appropriate quantile from the normal distribution. For example, 95% of the data will fall within 1.96 standard deviations of a normal distribution. So the 95% confidence limits are as follows:

sem=std(data)/sqrt(length(data)); % standard error of the mean (6-15)

*sem = sem * 1.96 % 95% confidence interval* (6-16)

Generally, the coefficient of determination (R-square) measures the variability of the estimation errors against the variability of the original values:

$$r2 = 1 - \frac{\sum(y-f)^2}{\sum(y-\bar{y})^2} \quad (6\text{-}17)$$

6.9. Fast and Inverse Fourier Transforms

6.9.1 fft - Fast Fourier Transform

Y = fft(x) (6-18)

Y = fft(X,n) (6-19)

The functions *Y = fft(x)* and *x = ifft(Y)* implement the transform and inverse transform pair which for vectors of length *N is* given by:

$$X(k) = \sum_{j=1}^{N} x(j)\omega_N^{(j-1)(k-1)} \quad (6\text{-}20)$$

$$x(j) = (1/N)\sum_{k=1}^{N} X(k)\omega_N^{-(j-1)(k-1)} \quad (6\text{-}21)$$

where$\omega_N = e^{(-2\pi i)/N}$

Y = fft(x) returns the discrete Fourier transform (DFT) of vector *x*, computed with a fast Fourier transform (FFT) algorithm.

Y = fft(X,n) returns the *n*-point DFT. *fft(X)* is equivalent to *fft(X,n),* where *n* is the size of *X*. If the length of *X* is less than *n*, *X* is padded with trailing zeroes to length *n*. If the length of *X* is greater than *n*, the sequence *X* is truncated.

6.9.2 *ifft* - Inverse Fast Fourier Transform

y = ifft(X) (6-22)

y = ifft(X,n) (6-23)

y = ifft(X) returns the inverse discrete Fourier transform (DFT) of vector *X*, computed with a fast Fourier transform (FFT) algorithm.

It should be noted that the sinusoid's frequency is k/N cycles per sample.

It can be concluded that the DFT fully describes the discrete-time Fourier transform (DTFT) of an N-periodic sequence, which comprises only discrete frequency components. Further improvements of the discrete Fourier transform approach would involve using wavelets.

6.10. DFT Investigations

To investigate further the method of forecasting with the DFT, sixteen training sample time series are used to forecast the subsequent sixteen share prices from the full dataset. The model with three cyclical trends is shown in Figure 6-17. Experiments are conducted only with sixteen sample time series, and not the long term full dataset, so as to focus on the purpose of this thesis which is short-term forecasting.

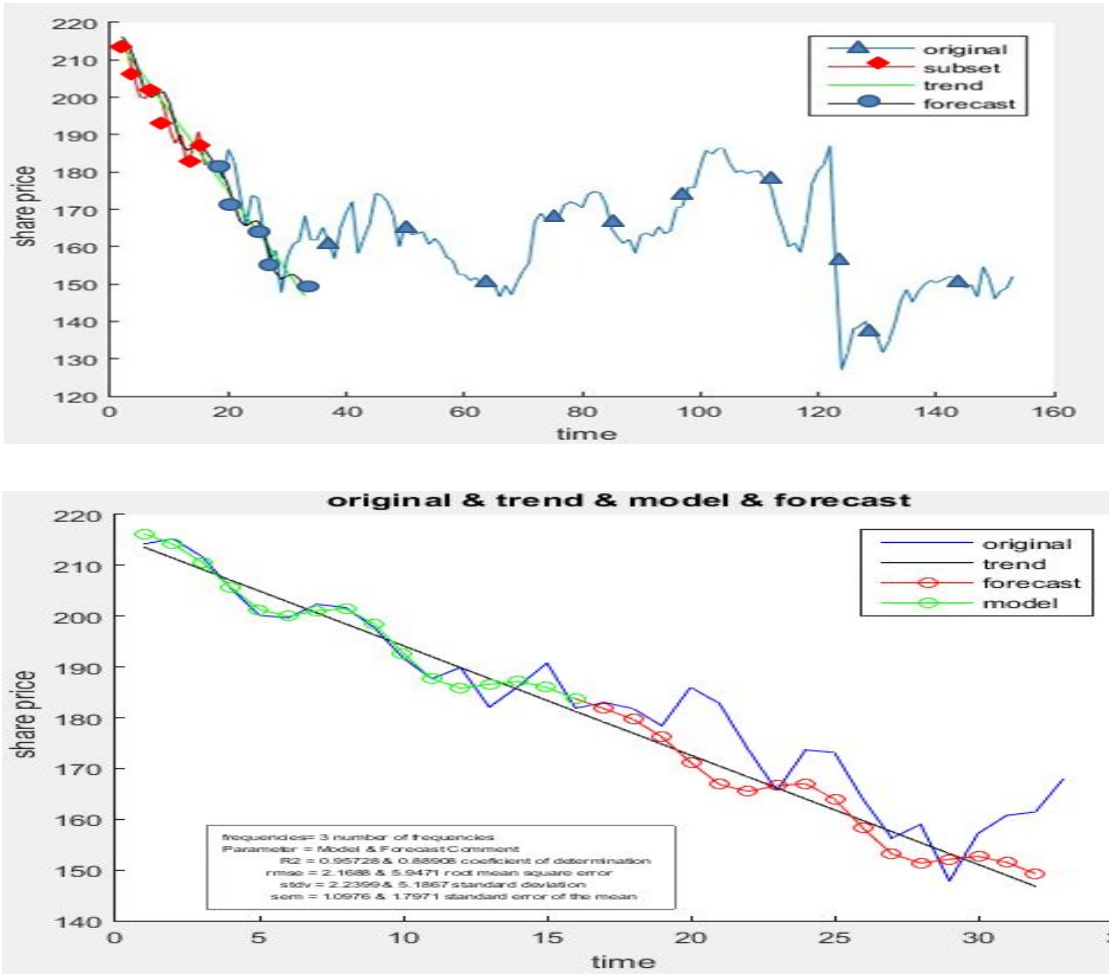


FIGURE 6-17 FORECAST WITH THREE HARMONICS MODEL

The parameters of the model are as follows:

R2 = 0.95728 and 0.88908 coefficient of determination,

rmse = 2.1688 and 5.9471 root mean square error,

stdv = 2.2399 and 5.1867 standard deviation,

sem = 1.0976 and 1.7971 standard error of the mean.

6.10.1 Model with Trend and Zero Frequency Component

$$x_m^{(0)} = A_0 \tag{6-24}$$

Trend $y=mx+b$ (6-25)

where m = -2.15, b = 213.57, *SQRT(err^2)* = 13.53, R^2 = 0.90

TABLE 6-3 TREND WITH ZERO FREQUENCY HARMONICS COMPONENT

Date	Frequency	n	y	appr	trend	cntrd	err^2
1/4/2016	0	0	214.25	213.57	213.5702	0.680	0.46
1/5/2016	0.0625	1	215.25	211.42	211.4154	3.835	14.70
1/6/2016	0.125	2	211.75	209.26	209.2607	2.489	6.20
1/7/2016	0.1875	3	205.55	207.11	207.1059	-1.556	2.42
1/8/2016	0.25	4	200.15	204.95	204.9511	-4.801	23.05
1/11/2016	0.3125	5	199.7	202.80	202.7963	-3.096	9.59
1/12/2016	0.375	6	202.35	200.64	200.6415	1.708	2.92
1/13/2016	0.4375	7	201.7	198.49	198.4868	3.213	10.32
1/14/2016	0.5	8	197.7	196.33	196.332	1.368	1.87
1/15/2016	0.5625	9	191.8	194.18	194.1772	-2.377	5.65
1/18/2016	0.625	10	187.65	192.02	192.0224	-4.372	19.12
1/19/2016	0.6875	11	189.9	189.87	189.8676	0.032	0.00
1/20/2016	0.75	12	182.05	187.71	187.7129	-5.663	32.07
1/21/2016	0.8125	13	186.15	185.56	185.5581	0.592	0.35
1/22/2016	0.875	14	190.75	183.40	183.4033	7.347	53.97
1/25/2016	0.9375	15	181.85	181.25	181.2485	0.601	0.36

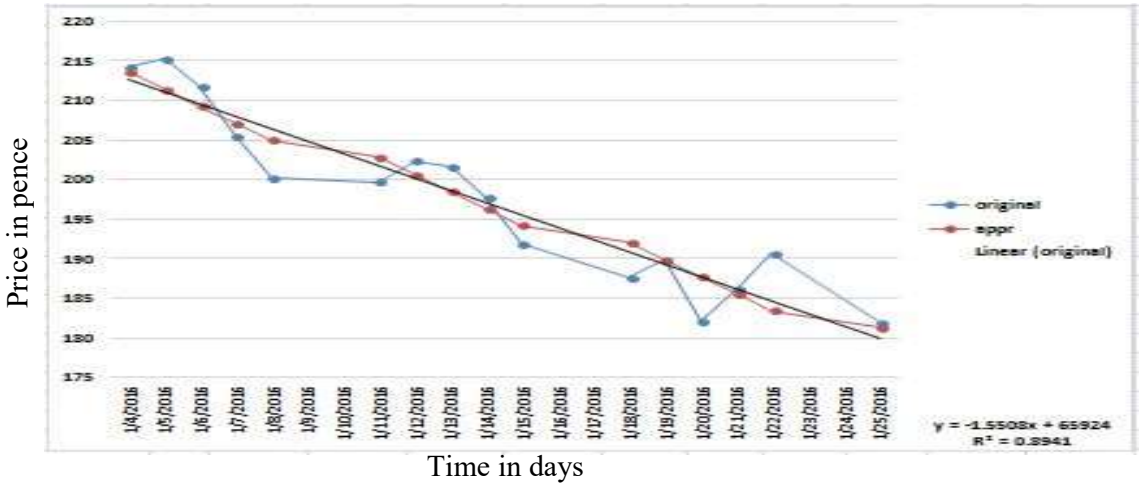


FIGURE 6-18 TREND WITH NO TREND LINE HARMONICS

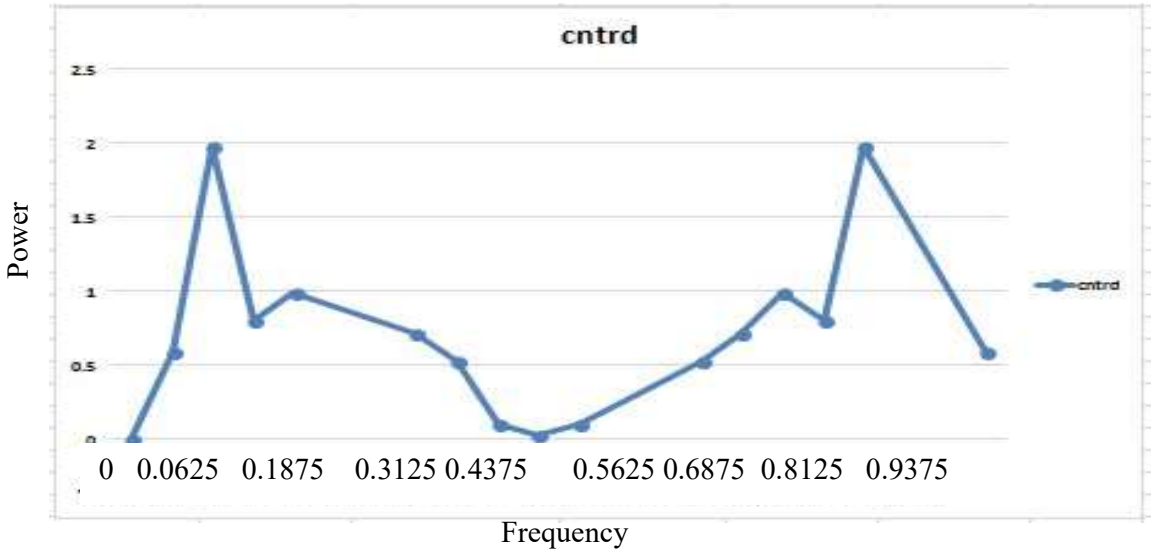


FIGURE 6-19 PDF TREND WITH NO FREQUENCY COMPONENT

6.10.2 Model with Trend and One Frequency Component

Using the first frequency component (k=1):

$$x_m^{(0)} = A_0 \tag{6-26}$$

$$x_m^{(1)} = A_0 + A_1 \cos(\phi_1 + \omega \quad) = x_m^{(0)} + A_1 \cos(\phi_1 + \omega m) \tag{6-27}$$

where *A_t-s*= 3.74, *T_t-s*= 6.49, *F_t-s* = 0.66, *SQRT(err^2)*= 8.32, R^2= 0.96.

TABLE 6-4 TREND WITH ONE FREQUENCY HARMONIC COMPONENT

Date	Frequency	n	y	appr	trend	cntrd	err^2	F1	frqSUM	err(cntr)^2
1/4/2016	0	0	214.25	215.81	213.5702	0.680	2.44	2.24	2.24	2.44
1/5/2016	0.0625	1	215.25	215.16	211.4154	3.835	0.01	3.74	3.74	0.01
1/6/2016	0.125	2	211.75	211.26	209.2607	2.489	0.24	2.00	2.00	0.24
1/7/2016	0.1875	3	205.55	205.63	207.1059	-1.556	0.01	-1.48	-1.48	0.01
1/8/2016	0.25	4	200.15	201.28	204.9511	-4.801	1.27	-3.67	-3.67	1.27
1/11/2016	0.3125	5	199.7	200.12	202.7963	-3.096	0.17	-2.68	-2.68	0.17
1/12/2016	0.375	6	202.35	201.28	200.6415	1.708	1.15	0.64	0.64	1.15
1/13/2016	0.4375	7	201.7	201.89	198.4868	3.213	0.04	3.40	3.40	0.04
1/14/2016	0.5	8	197.7	199.55	196.332	1.368	3.42	3.22	3.22	3.42
1/15/2016	0.5625	9	191.8	194.42	194.1772	-2.377	6.86	0.24	0.24	6.86
1/18/2016	0.625	10	187.65	189.08	192.0224	-4.372	2.05	-2.94	-2.94	2.05
1/19/2016	0.6875	11	189.9	186.29	189.8676	0.032	13.01	-3.57	-3.57	13.01
1/20/2016	0.75	12	182.05	186.61	187.7129	-5.663	20.75	-1.11	-1.11	20.75
1/21/2016	0.8125	13	186.15	187.88	185.5581	0.592	2.99	2.32	2.32	2.99
1/22/2016	0.875	14	190.75	187.14	183.4033	7.347	13.04	3.74	3.74	13.04
1/25/2016	0.9375	15	181.85	183.16	181.2485	0.601	1.72	1.91	1.91	1.72

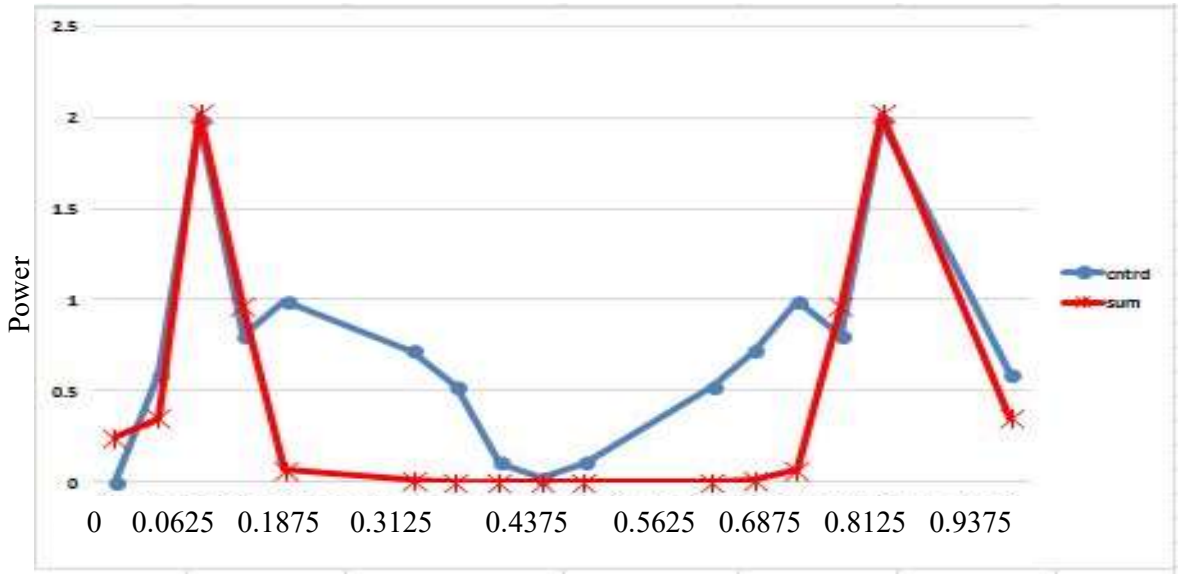


FIGURE 6-20 PDF TREND WITH ONE FREQUENCY HARMONIC COMPONENT

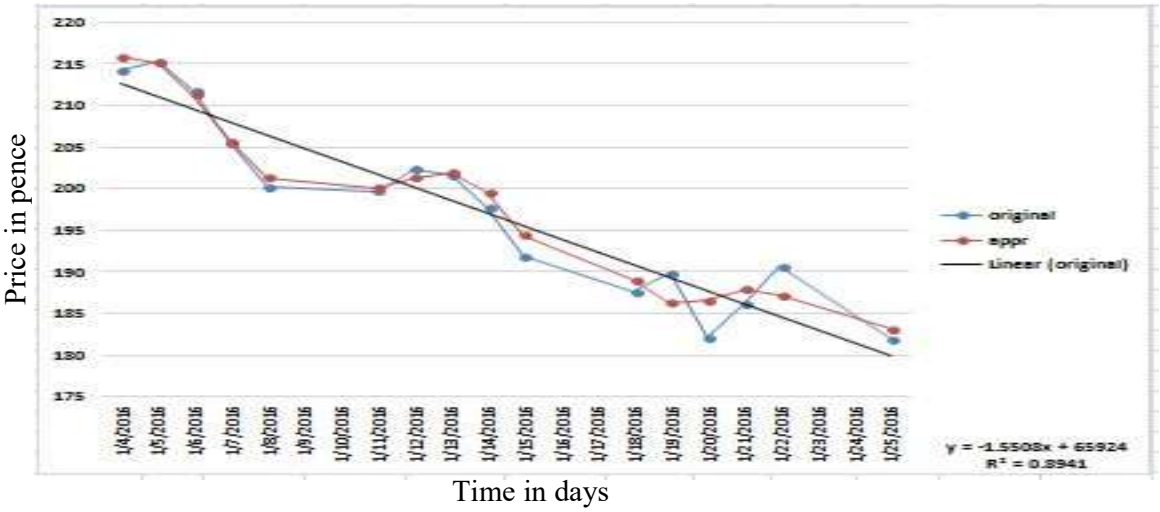


FIGURE 6-21 TREND WITH ONE FREQUENCY HARMONIC COMPONENT

6.10.3 Model with Trend and Two Frequency Components

Using the first and second frequency components (k=2):

$$x_m^{(0)} = A_0 \tag{6-28}$$

$$x_m^{(1)} = x_m^{(0)} + A_1 \cos(\phi_1 + \omega m) \tag{6-29}$$

$$x_m^{(2)} = x_m^{(1)} + A_2 \cos(\phi_2 + 2\omega m \tag{6-30}$$

where *A_t-s* = 3.72 1.95, *T_t-s* = 6.51 3.99, *F_t-s* = 0.84 -1.33, *SQRT(err^2)* = 6.39, R^2 = 0.98.

TABLE 6-5 TREND WITH TWO FREQUENCY HARMONIC COMPONENTS

Date	Freqnc	n	y	appr	trend	cntrd	err^2	F1	F2	frqSUM
1/4/2016	0	0	214.25	214.58	213.5702	0.680	0.11	2.71	-1.70	1.01
1/5/2016	0.0625	1	215.25	214.10	211.4154	3.835	1.32	3.64	-0.96	2.69
1/6/2016	0.125	2	211.75	212.41	209.2607	2.489	0.44	1.44	1.71	3.15
1/7/2016	0.1875	3	205.55	206.06	207.1059	-1.556	0.26	-2.00	0.95	-1.05
1/8/2016	0.25	4	200.15	199.52	204.9511	-4.801	0.40	-3.72	-1.71	-5.43
1/11/2016	0.3125	5	199.7	199.61	202.7963	-3.096	0.01	-2.24	-0.94	-3.18
1/12/2016	0.375	6	202.35	203.52	200.6415	1.708	1.38	1.17	1.71	2.88
1/13/2016	0.4375	7	201.7	202.99	198.4868	3.213	1.67	3.57	0.94	4.51
1/14/2016	0.5	8	197.7	197.52	196.332	1.368	0.03	2.90	-1.72	1.18
1/15/2016	0.5625	9	191.8	192.98	194.1772	-2.377	1.40	-0.27	-0.93	-1.19
1/18/2016	0.625	10	187.65	190.54	192.0224	-4.372	8.35	-3.20	1.72	-1.48
1/19/2016	0.6875	11	189.9	187.40	189.8676	0.032	6.23	-3.38	0.92	-2.46
1/20/2016	0.75	12	182.05	185.34	187.7129	-5.663	10.80	-0.65	-1.73	-2.38
1/21/2016	0.8125	13	186.15	187.29	185.5581	0.592	1.29	2.64	-0.91	1.73
1/22/2016	0.875	14	190.75	188.79	183.4033	7.347	3.82	3.66	1.73	5.39
1/25/2016	0.9375	15	181.85	183.68	181.2485	0.601	3.36	1.53	0.91	2.43

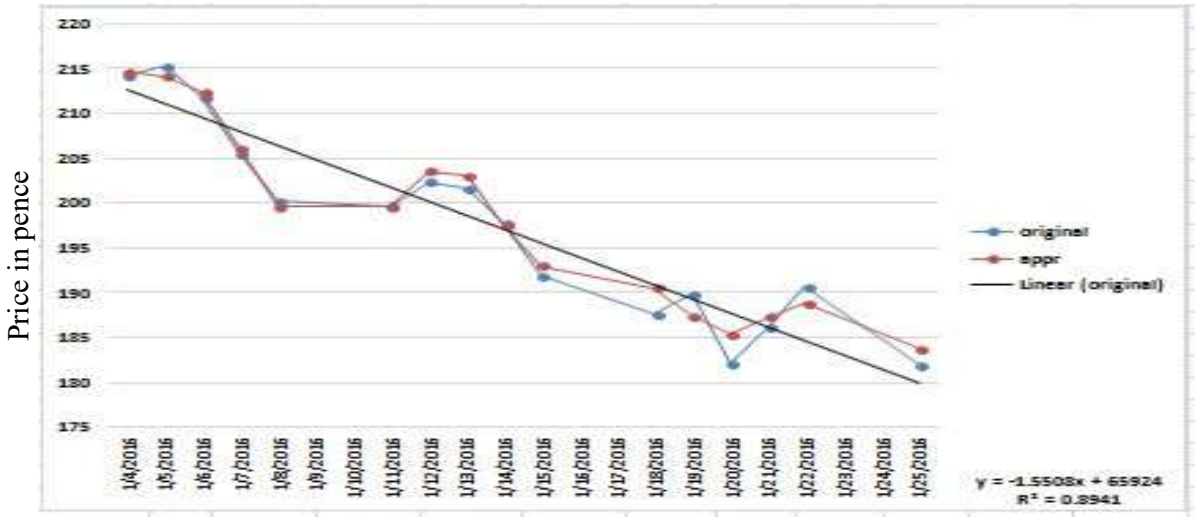


FIGURE 6-22 TREND WITH TWO FREQUENCY HARMONIC COMPONENTS

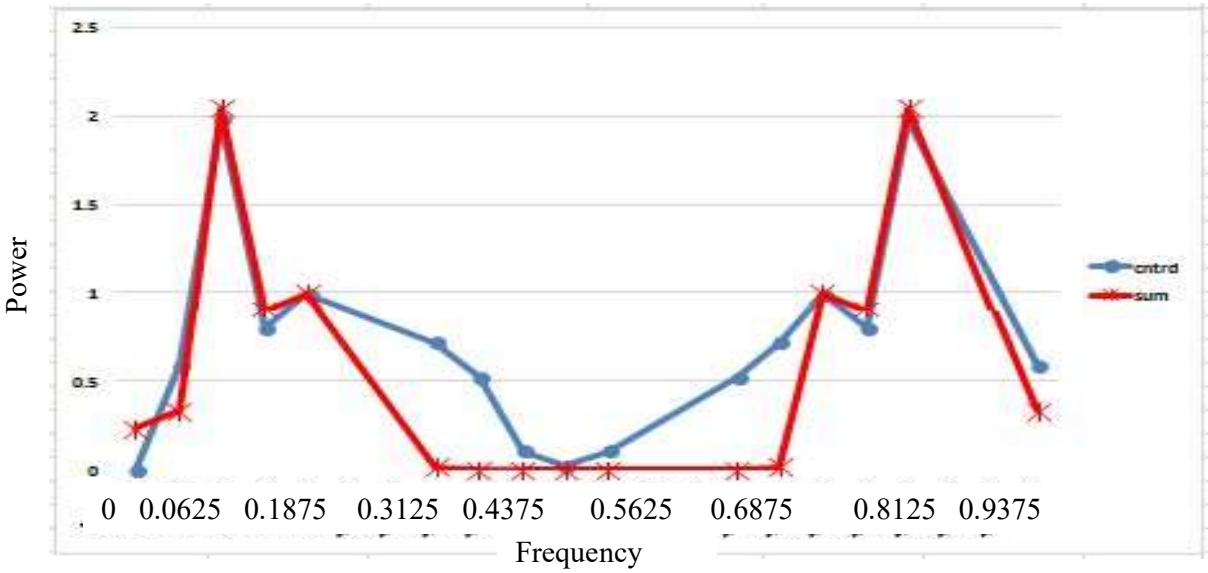


FIGURE 6-23 PDF TREND WITH TWO FREQUENCY HARMONICS COMPONENT

6.10.4 Model with Trend and Three Frequency Components

Using the first three frequency components (k=3):

$$x_m^{(0)} = A_0 \tag{6-31}$$

$$x_m^{(1)} = x_m^{(0)} + A_1 \cos(\phi_1 + \omega m) \tag{6-32}$$

$$x_m^{(2)} = x_m^{(1)} + A_2 \cos(\phi_2 + 2\omega m) \tag{6-33}$$

$$x_m^{(3)} = x_m^{(2)} + A_3 \cos(\phi_3 + 3\omega m \tag{6-34}$$

where*A_t-s* = 3.70 1.93 1.43, *T_t-s* = 6.60 4.02 2.56, *F_t-s* = 0.94 -1.33 -0.21, *SQRT(err^2)* = 4.99, R^2 = 0.99

TABLE 6-6 TREND WITH THREE FREQUENCY HARMONICS COMPONENTS

Date	Frequency	n	y	apprx	trend	cntrd	err^2	F1	F2	F3	frqSUM
1/4/2016	0	0	214.25	214.06	213.5702	0.680	0.04	2.88	-1.69	-0.70	0.49
1/5/2016	0.0625	1	215.25	215.35	211.4154	3.835	0.01	3.56	-0.95	1.33	3.94
1/6/2016	0.125	2	211.75	210.83	209.2607	2.489	0.84	1.26	1.67	-1.36	1.57
1/7/2016	0.1875	3	205.55	206.75	207.1059	-1.556	1.44	-2.11	0.98	0.78	-0.35
1/8/2016	0.25	4	200.15	199.74	204.9511	-4.801	0.16	-3.70	-1.66	0.15	-5.21
1/11/2016	0.3125	5	199.7	198.60	202.7963	-3.096	1.22	-2.18	-1.00	-1.01	-4.20
1/12/2016	0.375	6	202.35	204.87	200.6415	1.708	6.36	1.17	1.64	1.42	4.23
1/13/2016	0.4375	7	201.7	201.86	198.4868	3.213	0.03	3.54	1.02	-1.19	3.37
1/14/2016	0.5	8	197.7	198.06	196.332	1.368	0.13	2.94	-1.63	0.42	1.73
1/15/2016	0.5625	9	191.8	193.53	194.1772	-2.377	2.99	-0.13	-1.05	0.53	-0.65
1/18/2016	0.625	10	187.65	189.30	192.0224	-4.372	2.73	-3.09	1.62	-1.25	-2.72
1/19/2016	0.6875	11	189.9	188.89	189.8676	0.032	1.02	-3.45	1.07	1.40	-0.98
1/20/2016	0.75	12	182.05	184.27	187.7129	-5.663	4.91	-0.92	-1.60	-0.93	-3.45
1/21/2016	0.8125	13	186.15	186.89	185.5581	0.592	0.55	2.39	-1.09	0.04	1.34
1/22/2016	0.875	14	190.75	189.54	183.4033	7.347	1.45	3.69	1.58	0.87	6.14
1/25/2016	0.9375	15	181.85	182.87	181.2485	0.601	1.03	1.89	1.12	-1.39	1.62

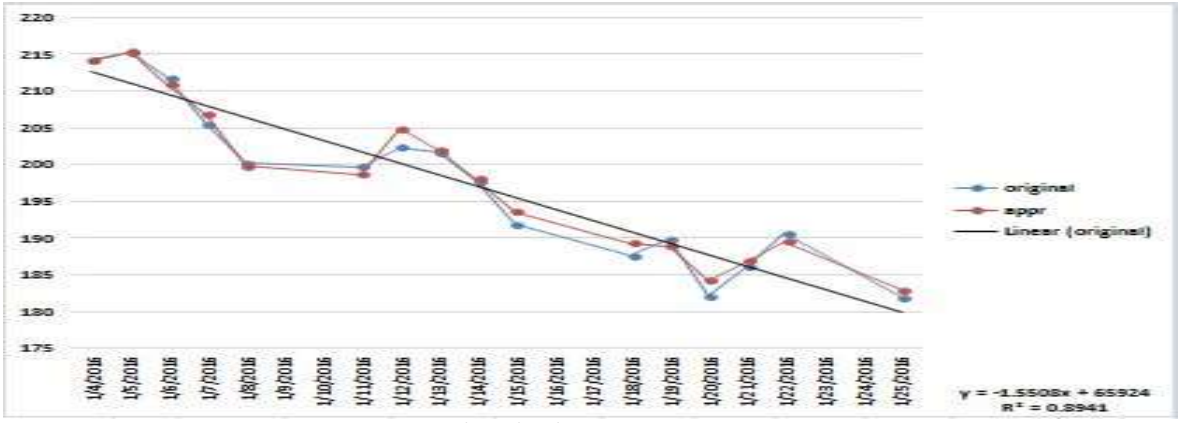


FIGURE 6-24 TREND WITH THREE FREQUENCY HARMONIC COMPONENTS

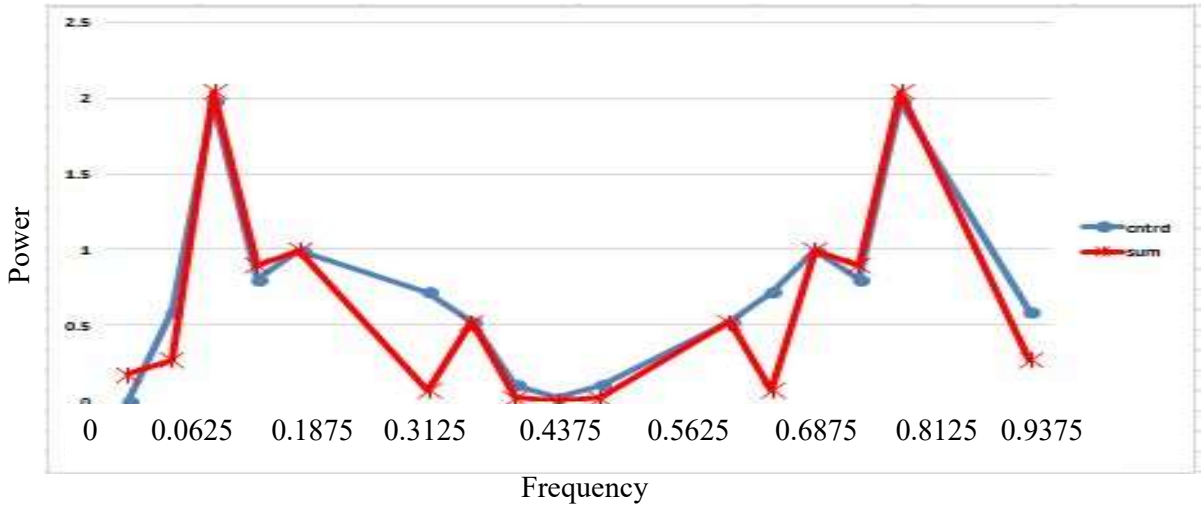


FIGURE 6-25 PDF TREND WITH THREE FREQUENCY HARMONIC COMPONENTS

6.10.5 Model with Trend and Frequency Components in MATLAB

De-trending of the time series is done with a linear trend as follows:

$$y=mx+b \tag{6-35}$$

where m = -2.15, b = 213.57

An example of the use of the first frequency components k= 3 is:

$$x_m^{(0)} = A_0 \tag{6-36}$$

$$x_m^{(1)} = x_m^{(0)} + A_1\cos(\phi_1 + \omega m) \tag{6-37}$$

$$x_m^{(2)} = x_m^{(1)} + A_2\cos(\phi_2 + 2\omega m) \tag{6-38}$$

$$x_m^{(3)} = x_m^{(2)} + A_3\cos(\phi_3 + 3\omega m) \tag{6-39}$$

TABLE 6-7 BARCLAYS SHARE PRICES TIME SERIES FROM 2016-01-04 TO 2016-01-25

Date	Frequency	n	y	trend	centred
1/4/2016	0	0	214.25	213.5702	0.680
1/5/2016	0.0625	1	215.25	211.4154	3.835
1/6/2016	0.125	2	211.75	209.2607	2.489
1/7/2016	0.1875	3	205.55	207.1059	-1.556
1/8/2016	0.25	4	200.15	204.9511	-4.801
1/11/2016	0.3125	5	199.7	202.7963	-3.096
1/12/2016	0.375	6	202.35	200.6415	1.708
1/13/2016	0.4375	7	201.7	198.4868	3.213
1/14/2016	0.5	8	197.7	196.332	1.368
1/15/2016	0.5625	9	191.8	194.1772	-2.377
1/18/2016	0.625	10	187.65	192.0224	-4.372
1/19/2016	0.6875	11	189.9	189.8676	0.032
1/20/2016	0.75	12	182.05	187.7129	-5.663
1/21/2016	0.8125	13	186.15	185.5581	0.592
1/22/2016	0.875	14	190.75	183.4033	7.347
1/25/2016	0.9375	15	181.85	181.2485	0.601

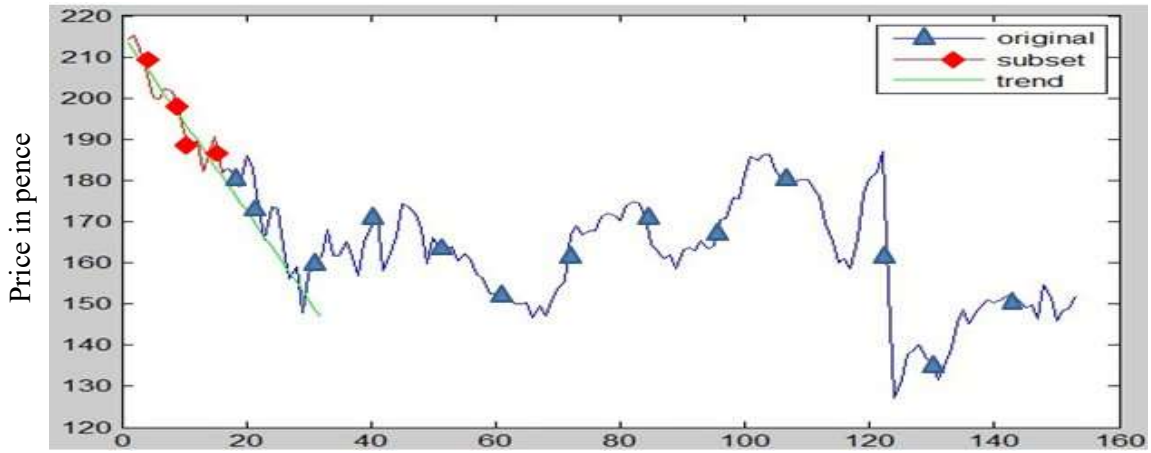


FIGURE 6-26 BARC.L SHARE PRICES TIME SERIES FROM 16-01-04 TO 16-01-25

6.11. Time Domain

An investigation of the effect of the number of harmonics included in the model is shown in Figure 6-27 to Figure 6-30.The results show a closer fit between the model and the target, although with no significant difference, with the additional of more than three harmonics.

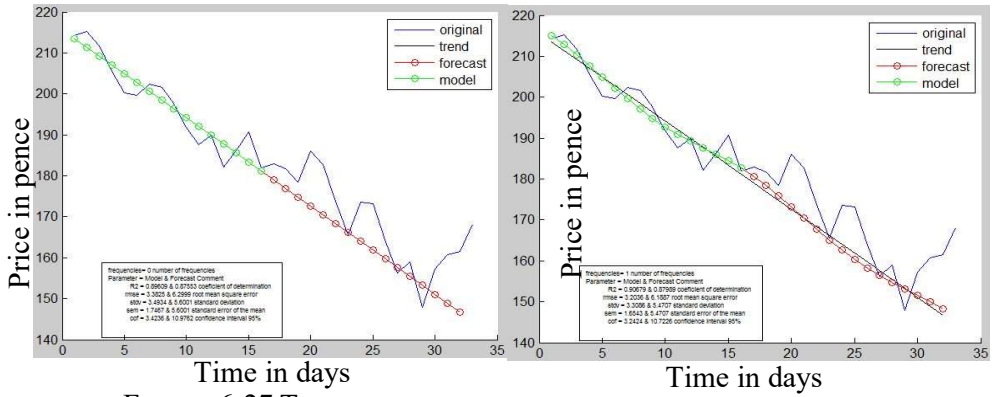


FIGURE 6-27 TRENDS WITH NO AND ONE HARMONIC COMPONENT

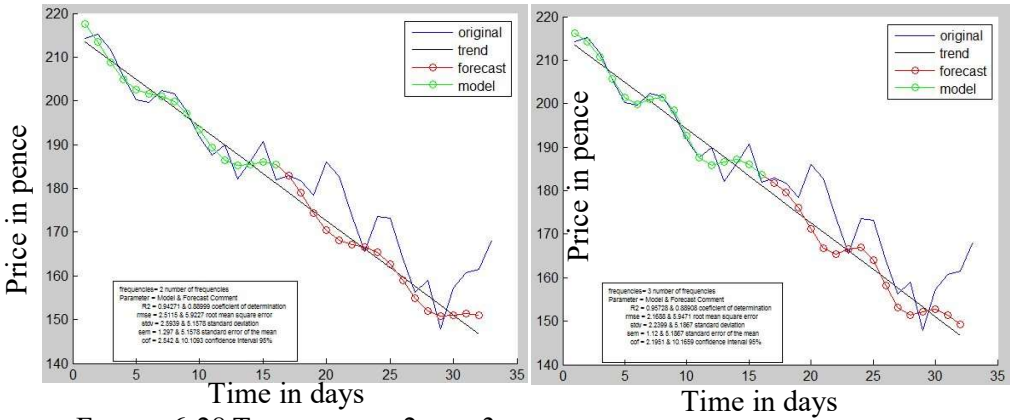


FIGURE 6-28 TRENDS WITH 2 AND 3 FREQUENCY HARMONICS COMPONENTS

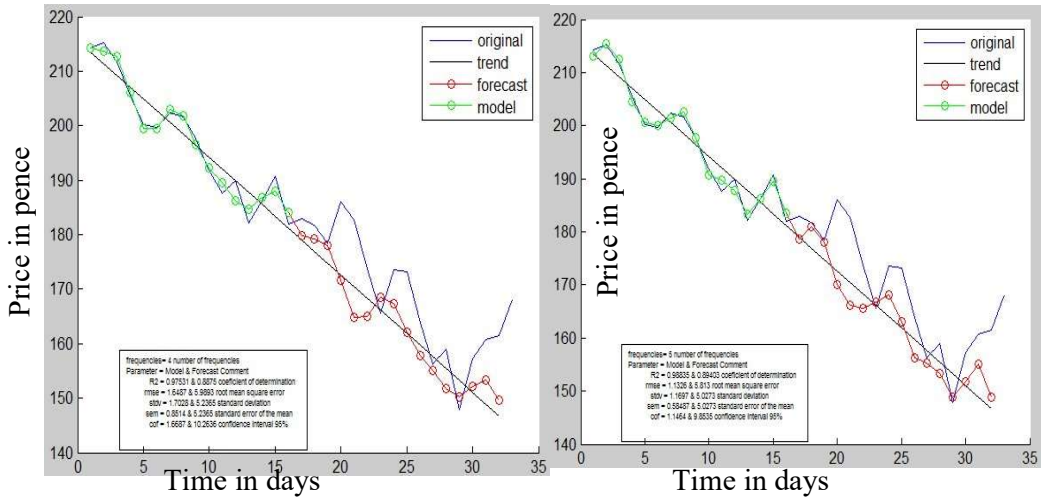


FIGURE 6-29 TRENDS WITH 4 AND 5 FREQUENCY HARMONICS COMPONENTS

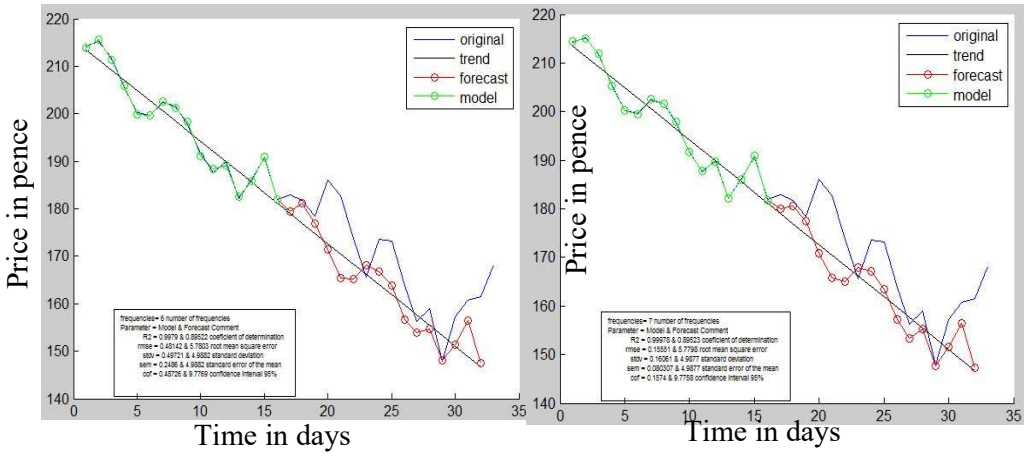


FIGURE 6-30 TRENDS WITH 6 AND 7 FREQUENCY HARMONICS COMPONENTS

6.12. Power Distribution Frequencies

The investigation of the effect of the number of approximation frequencies starts at the position at day 1 with a time series of 16 samples long.

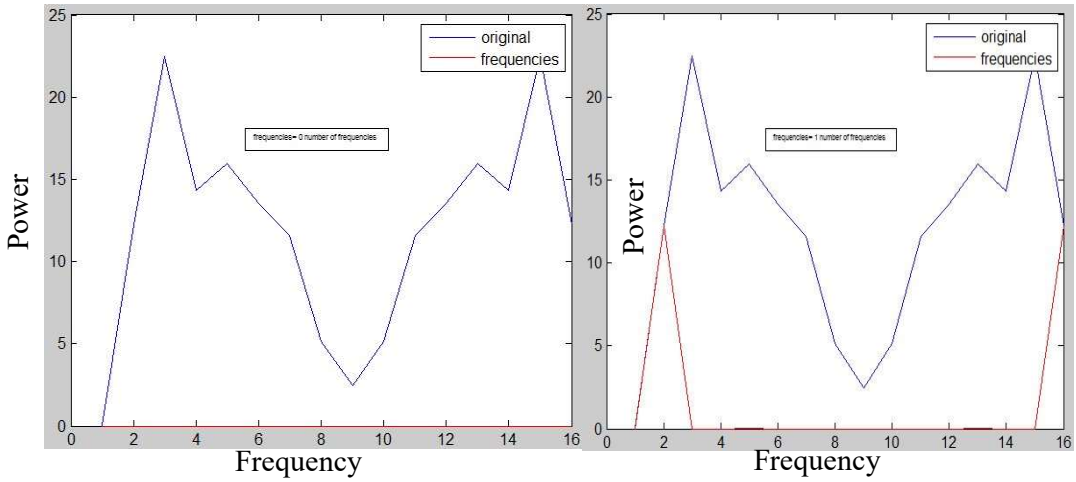


FIGURE 6-31 PDF WITH NO AND ONE FREQUENCY HARMONIC COMPONENT

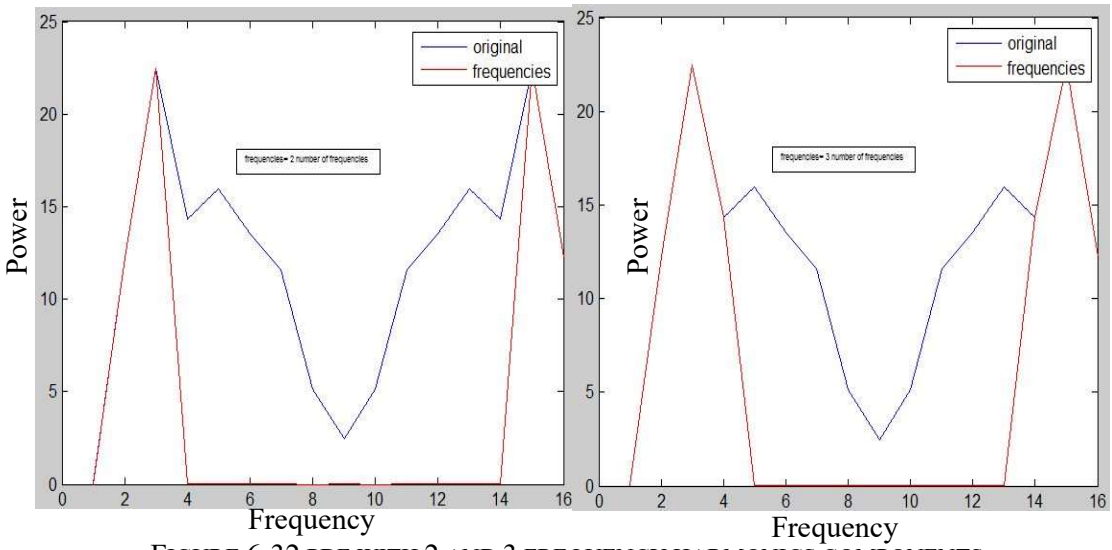


FIGURE 6-32 PDF WITH 2 AND 3 FREQUENCY HARMONICS COMPONENTS

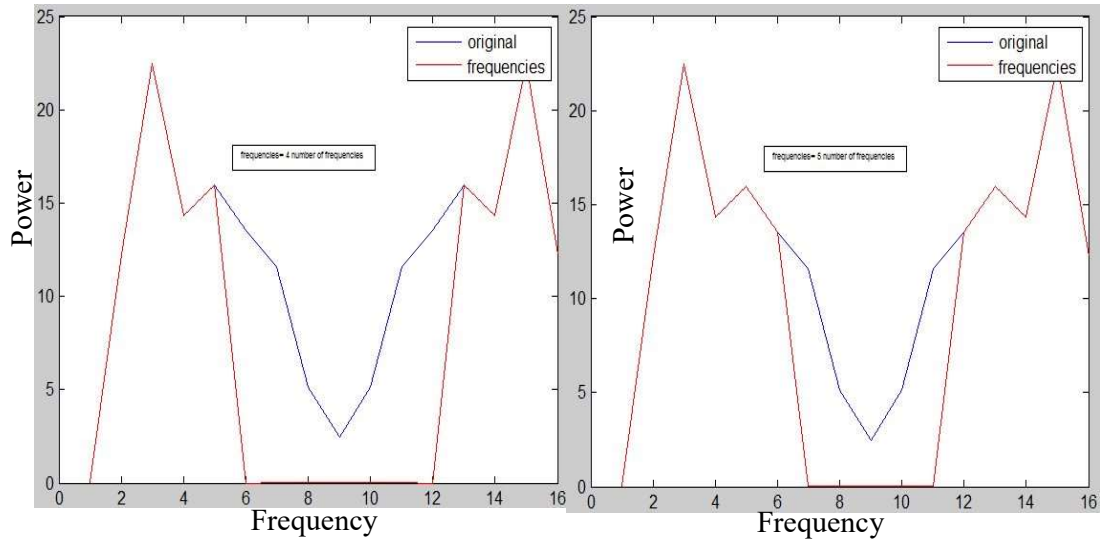


FIGURE 6-33 PDF WITH 4 AND 5 FREQUENCY HARMONICS COMPONENTS

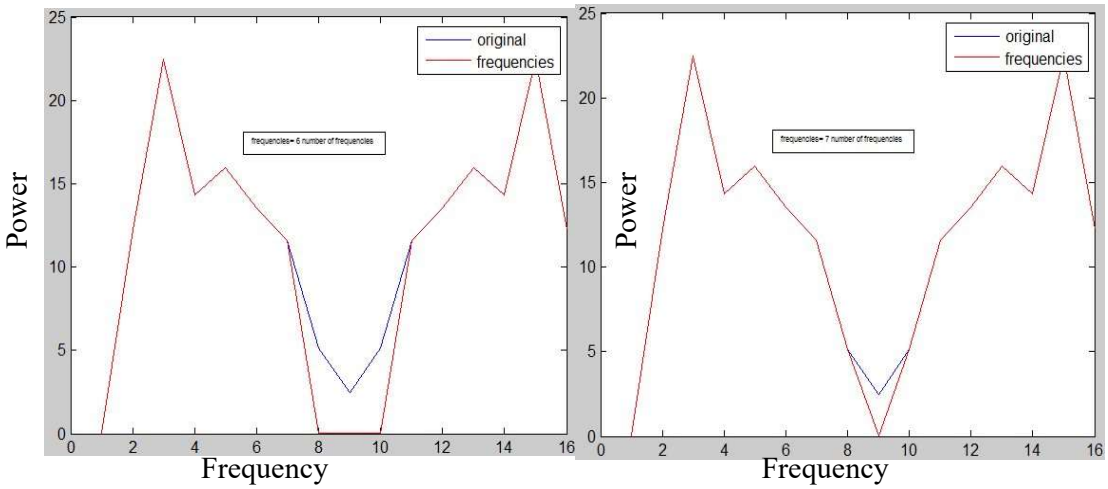


FIGURE 6-34 PDF WITH 6 AND 7 FREQUENCY HARMONIC COMPONENT

6.12.1 Model Performance per Frequency from 0 to 8

TABLE 6-8 MODEL PERFORMANCE PER FREQUENCY FROM 0 TO 8

Frequencies	0	1	2	3	5	6	7	8
R2_model	0.89609	0.90679	0.94271	0.95728	0.98835	0.9979	0.99978	0.99978
rmse_model	3.3825	3.2036	2.5115	2.1688	1.1326	0.48142	0.15551	0.15551
stdv_model	3.4934	3.3086	2.5939	2.2399	1.1697	0.49721	0.16061	0.16061
sem_model	1.7118	1.6212	1.271	1.0976	0.57318	0.24363	0.0787	0.0787

TABLE 6-9 FORECAST PERFORMANCE PER FREQUENCY FROM 0 TO 8

Frequencies	0	1	2	3	5	6	7	8
R2_forecast	0.87553	0.87989	0.88999	0.88908	0.89403	0.89522	0.89523	0.89668
rmse_forecast	6.2999	6.1887	5.9227	5.9471	5.813	5.7803	5.7798	5.7398
stdv_forecast	5.6001	5.4707	5.1578	5.1867	5.0273	4.9882	4.9877	4.9398
sem_forecast	1.9403	1.8955	1.7871	1.7971	1.7419	1.7283	1.7281	1.7115

6.12.2 Model Performance with Varying Starting Date

A daily investigation into whether or not the harmonics at different consecutive days vary with window size of 16 and 2 harmonics is shown in Table 6-10 to Table 6-14

TABLE 6-10 MODEL PERFORMANCE WITH DIFFERENT STARTING DATE AND TWO FREQUENCIES

Position/Date	1	2	3	5	6	7	8	9	10
R2_model	0.94271	0.942	0.924	0.862	0.869	0.844	0.781	0.746	0.729
rmse_model	2.5115	2.4268	2.529	2.884	2.703	3.077	3.9415	3.864	3.678
stdv_model	2.5939	2.5064	2.612	2.978	2.792	3.178	4.070	3.991	3.799
sem_model	2.542	2.4563	2.560	2.919	2.736	3.115	3.989	3.911	3.723

TABLE 6-11 FORECAST PERFORMANCE WITH DIFFERENT STARTING DATE AND 2 FREQUENCIES

Position/Date	1	2	3	6	7	8	9	10
R2_forecast	0.88999	0.88595	0.90792	0.85394	0.86639	0.8104	0.7114	0.56217
rmse_forecast	5.9227	5.788	4.972	5.6241	5.2844	5.9519	6.831	7.8647
stdv_forecast	5.1578	5.1818	4.7808	5.4373	5.2892	5.951	6.9235	7.9896
sem_forecast	1.787	1.795	1.656	1.8839	1.832	2.061	2.398	2.768

TABLE 6-12 FORECAST PERFORMANCE WITH PERIOD 16

Period1	16							
Position/Date	1	2	3	6	7	8	9	10
Amplitude1	1.5353	1.915	1.8085	3.4161	3.0968	2.5319	1.7948	1.7069
Phase1	-0.018623	0.10291	-0.05114	0.11293	0.62914	1.5819	2.2815	3.1266

TABLE 6-13 FORECAST PERFORMANCE WITH PERIOD 8

Period2	8							
Position/Date	1	2	3	5	7	8	9	10
Amplitude2	2.8124	2.9585	2.4278	1.7869	1.4224	0.1959	0.83631	1.2349
Phase2	0.55791	1.2178	2.0417	-2.0949	-0.23716	-1.9307	-1.5891	-1.125

TABLE 6-14 FORECAST PERFORMANCE WITH PERIOD 4

Period3	4							
Position/Date	1	2	3	6	7	8	9	10
Amplitude3	1.7911	1.7198	1.8521	1.6149	1.9693	3.3818	3.7404	3.584
Phase2	-2.352	-0.98566	-0.01777	2.364	-2.6409	-1.7282	-0.68093	0.38325

The introduction of the frequency components reduces the error from 13.53 to 4.99 and improves the fitness from 0.90 to 0.99. Generally, wider coverage of the time series spectrum improves performance. Furthermore, including more harmonics improves the model's fitness, as shown in Table 6-14 and Figure 6-35.

Model with trend and combination of harmonics:

- +no harmonics SQRT*(err^2)* = 13.53, *R2* = 0.90
- +one harmonics *SQRT(err^2)* = 8.32, *R2* = 0.96
- +two harmonics *SQRT(err^2)* = 6.39, *R2* = 0.98
- +three harmonics *SQRT(err^2)* = 4.99, *R2* = 0.99

TABLE 6-15 TREND - FREQUENCIES HARMONIC COMPONENT SQRT AND R2

	trend	+one	+two	+three frequencies
SQRT	13.53	8.32	6.39	4.99
R2	0.9	0.96	0.98	0.99

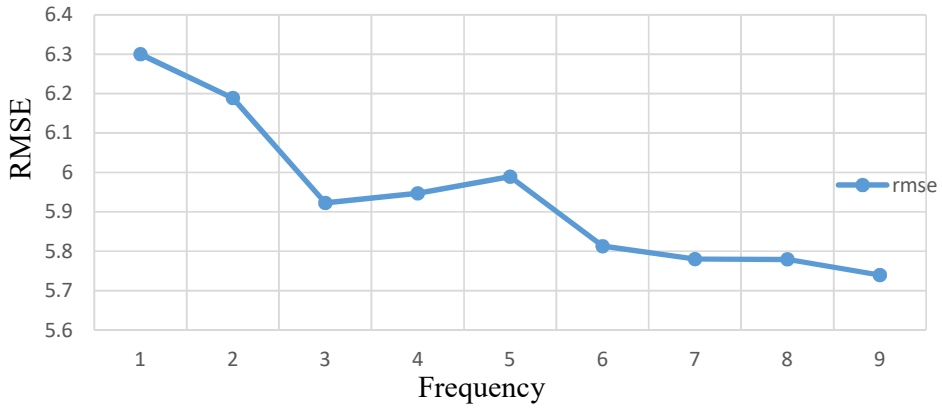


FIGURE 6-35 RMSE VS. FREQUENCIES WITH 16 DAYS WINDOW

The performance (RMSE) deteriorates significantly with predictions over seven days, and this confirms the hypothesis that shorter predictions in daily trading are the recommended option.

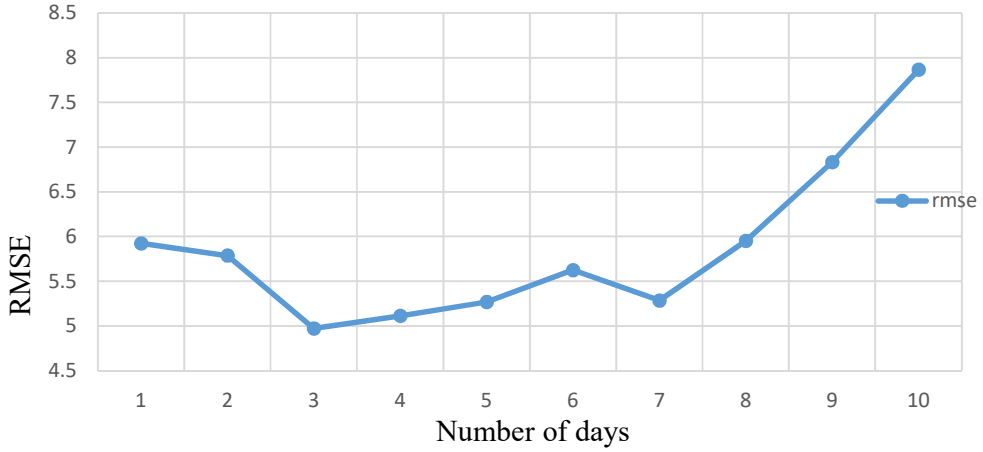


FIGURE 6-36 RMSE VS. NUMBER OF DAYS WITH 16 DAYS WINDOW

Harmonics parameters of both amplitude and phase vary with time (days),as shown in Figure 6-37 and Figure 6-38, although still in reasonably close ranges to allow some approximation.

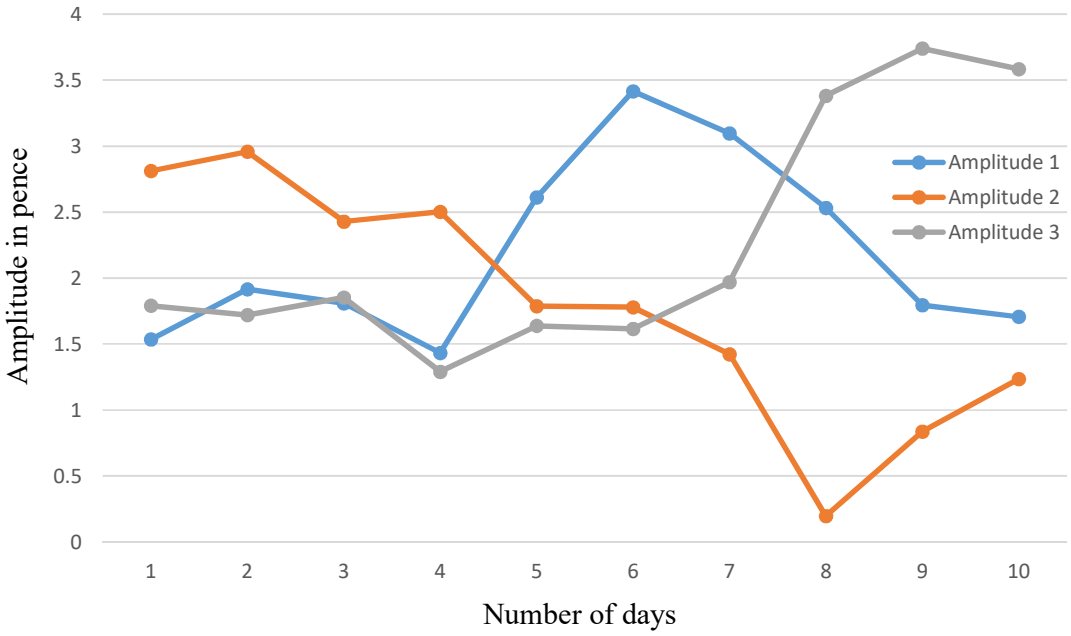


FIGURE 6-37 AMPLITUDES VS. NUMBER OF DAYS WITH 16 DAYS WINDOW

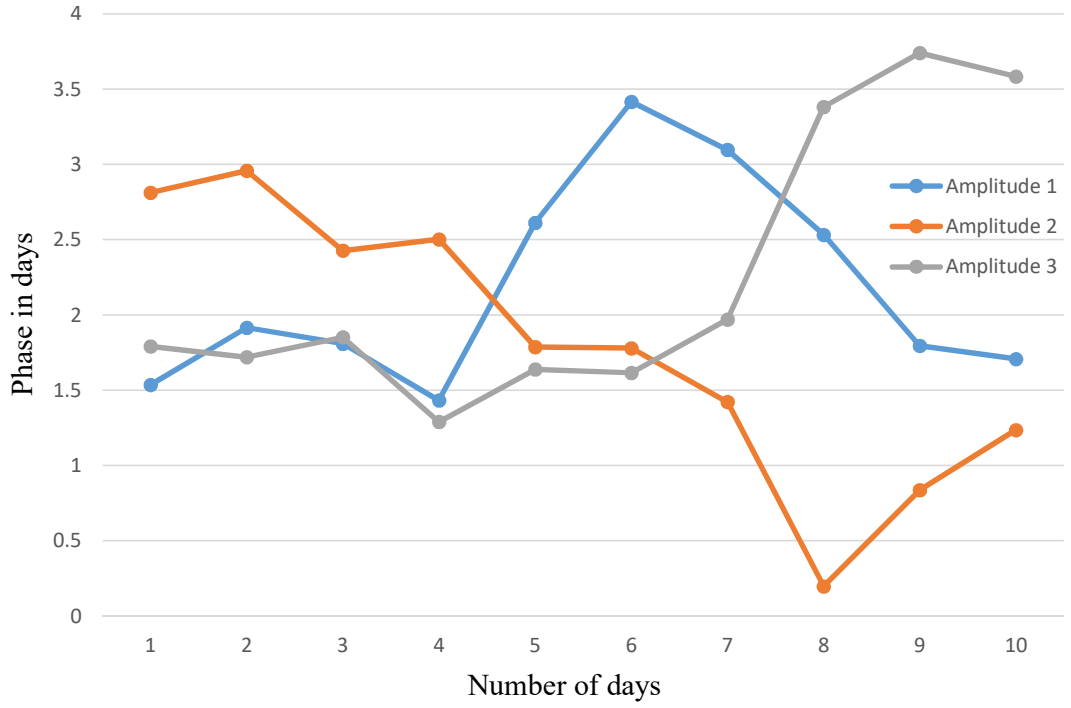


FIGURE 6-38 PHASES VS. DAYS WITH 16 DAYS WINDOW

The window size effect is that smaller windows tend to produce less RMSE error, as shown in Figure 6-39 and Figure 6-40.

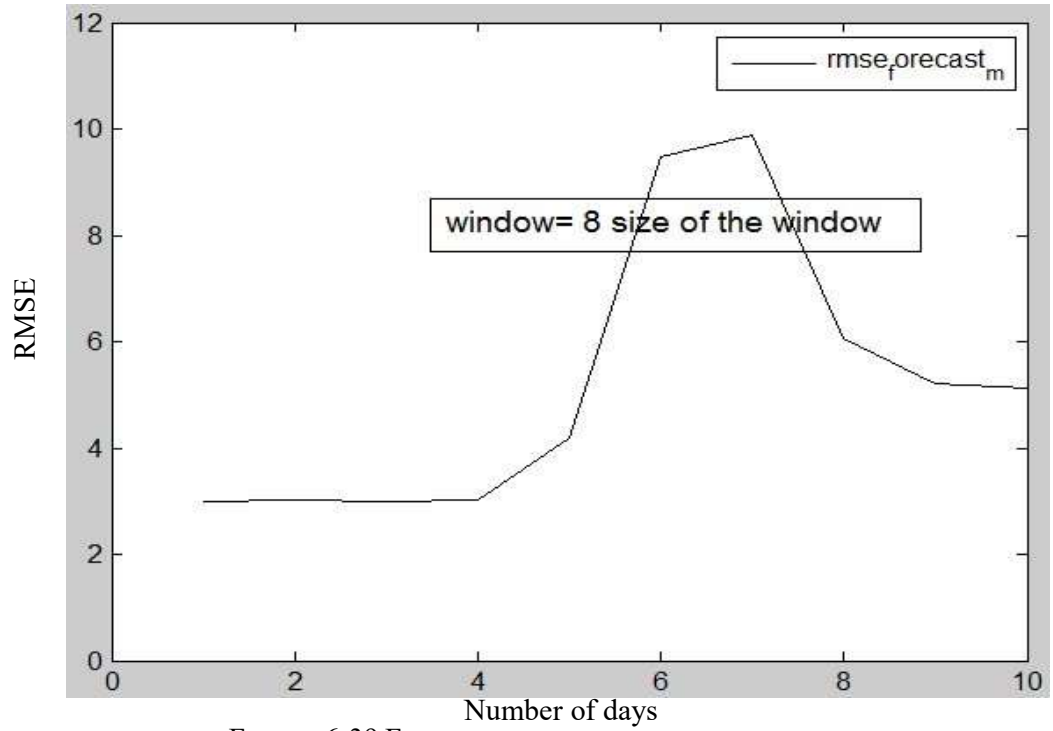


FIGURE 6-39 FORECAST RMSE VS. WINDOW SIZE EIGHT

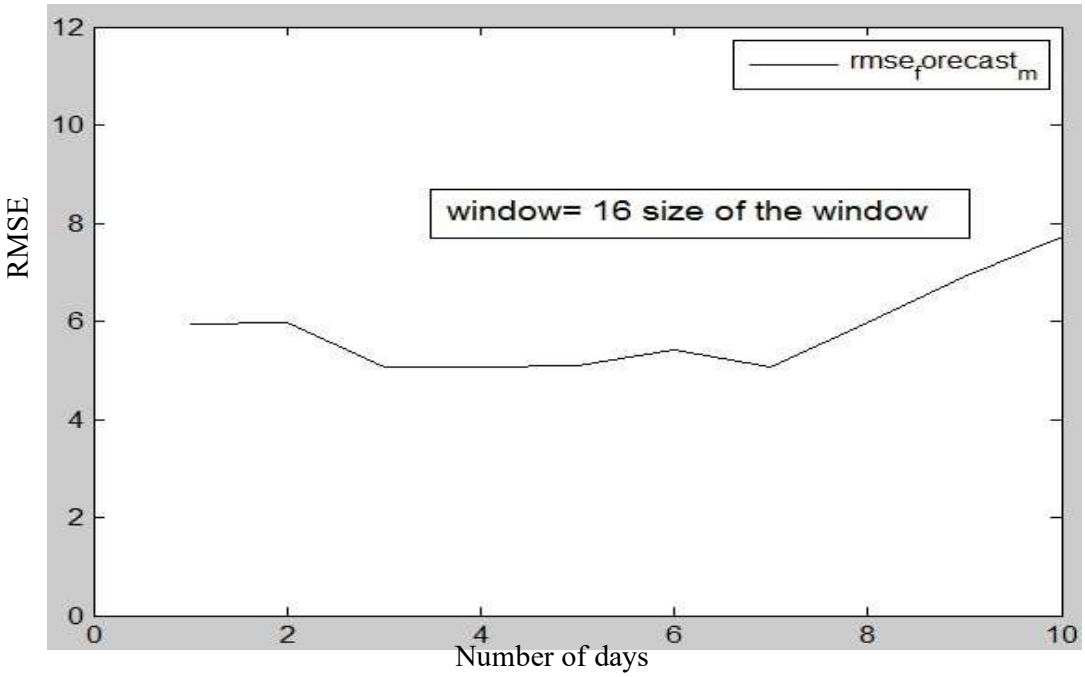


FIGURE 6-40 FORECAST RMSE VS. WINDOW SIZE SIXTEEN

6.13. Summary

Discrete Fourier Transform (DFT) time series analysis and decomposition proved to provide significantly better performance, robustness and generalization than analysis in the time domain.

The results are obtained for a time series of three trading weeks of data investigating the effect of various harmonics characteristics:

- Number of harmonics components in the model. The introduction of the frequency components reduces the error and improves the fitness (likeliness) of the model. Generally, wider coverage of the time series spectrum improves performance.
- The amplitudes and phases of the harmonics vary, although in a reasonable range allowing some reasonable average model approximations.
- The window size effect, where smaller windows tend to produce less prediction error.

The model's harmonics set could be improved with the major contributing harmonics instead of the first harmonics (low frequencies) in the DFT. Selection of the first harmonic is still justified by the power distribution, and generally, the major harmonics are at the beginning of the frequencies. These results are for share prices over a relatively quiet period without large disturbances. If there are such fluctuations, a different approach should be considered. Usually, however, significant changes and jumps in prices are rare.

Chapter 7. Discrete Wavelet Transform

7.1. Overview

Similar to the discrete Further transform, this chapter extends the financial time series investigation using the discrete wavelet transformation .

The Discrete Wavelet Transform (DWT) has gained widespread acceptance in signal and time series processing. Because of their inherent multi-resolution nature, wavelet-coding schemes are especially suitable for applications where scalability and tolerable signal degradation are important. Wavelet transforms have been successfully applied to financial time series because of their powerful feature extraction capability (Hsieh, et al., 2011). A wavelet transform simultaneously analyses the time and frequency domains.

This chapter covers the following areas:

- Mother wavelets examples
- MATLAB wavelets
- Forecasting with different wavelets
- Comparison of wavelet forecast performance
- Comparison of neural networks with wavelets
- Wavelet performance

7.2. Introduction

Wavelet transforms decompose a signal into a set of "basis" functions called wavelets. Wavelets are obtained from a single prototype wavelet $\psi_{a,b}(t)$, called a mother wavelet, by dilation and shifting:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a}) \quad (7\text{-}1)$$

where "a" is the scaling parameter and "b" is the shifting parameter. These parameters enable the transform to give a space-frequency localization of the signal.

7.3. Mother Wavelets Examples

Examples of mother wavelets are shown in Figure 7-1.

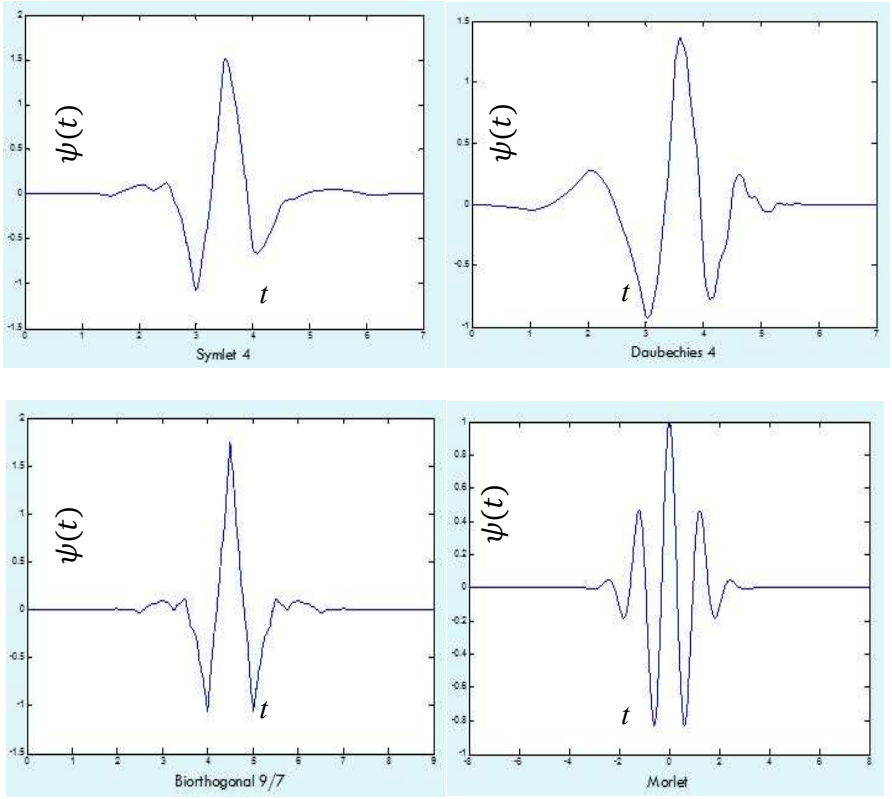


FIGURE 7-1 MOTHER WAVELET EXAMPLES

In wavelet analysis a mathematical model is built that allows the decomposition of a given signal into many frequency bands (Mallat, 1989). The given input data is decomposed into low frequency approximation coefficients (ACs) via a low-pass (LP) filter and detail coefficients (DC) of high frequency via a high-pass (HP) filter. Approximation coefficients characterize the coarse structure of the data and identify long-term trends, while detail coefficients capture ruptures and discontinuities. Generally, these two types of coefficient allow a better time series presentation than the original data. Wavelet analysis allows important hidden information and significant temporal features of the original time series data to be extracted (Chandar, et al., 2016).

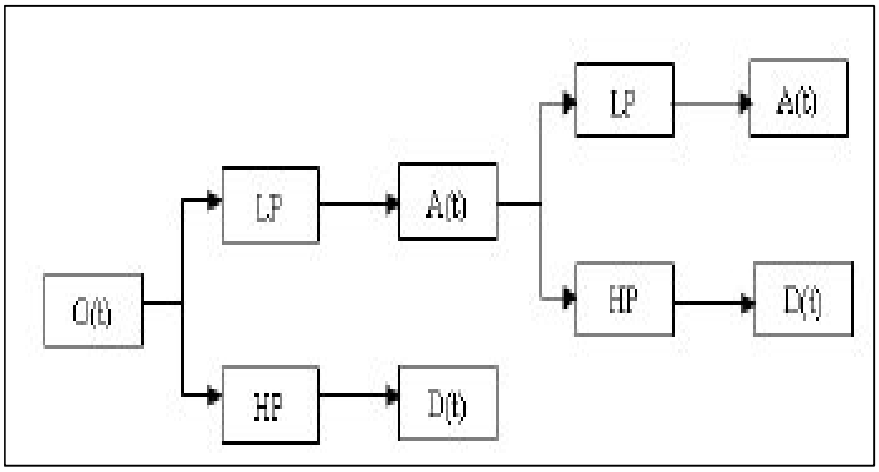


FIGURE 7-2 WAVELET DECOMPOSITION MODEL

In financial time series forecasting, different types of wavelets can be used, such as Daubechie's, Haar, Morlet or Mexican Hat wavelets.

The main properties of the DWT are as follows:

- Completeness, as it is invertible.
- De-correlation, since its coefficients are not correlated.
- Energy compaction, where the energy of the time series is mostly grouped in low frequencies in the DWT domain.
- Adjustability, because there is not just a single wavelet, and many wavelets can be designed to fit individual applications.
- Time-frequency localization, allowing a more accurate local description of signal characteristics.
- Robustness, as the DWT coefficients can be robust against many time series processing operations.

7.4. Forecasting with Different Wavelets

Forecasting has been conducted using different wavelets for a sixteen sample dataset from the day 1 starting point in the full time series, and the performance evaluated, including the coefficient of determination (*R2),* root mean square error *(rmse),* standard deviation *(stdv)* and standard error of the mean *(sem)*.

7.4.1 Forecast with Daubechie's "db4" Wavelet

wavelet = db4 name of the wavelet

R2 = 0.96961 and 0.90046 coefficient of determination

rmse = 1.8293 and 5.6339 root mean square error

stdv = 1.8892 and 4.8396 standard deviation

sem = 0.92573 and 1.6768 standard error of the mean

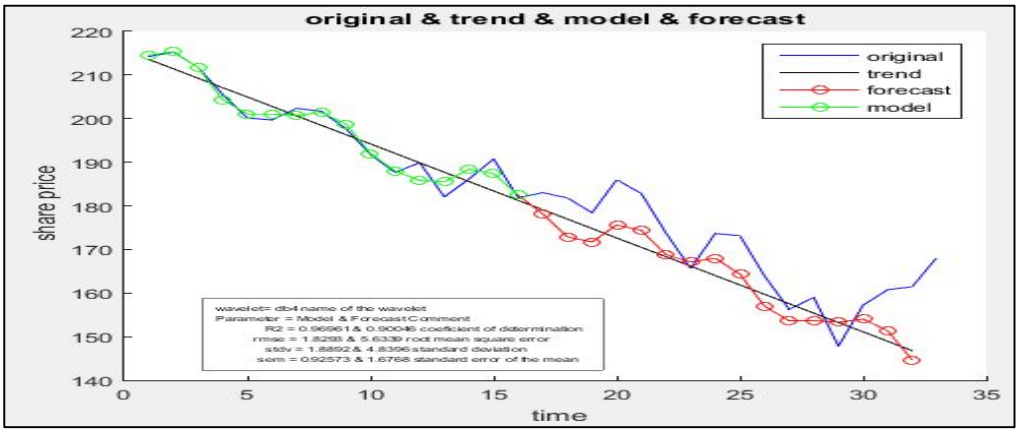


FIGURE 7-3 DAUBECHIE'S DB4 WAVELET FORECAST 160 AND 35 DAYS DATASET

7.4.2 Forecast with Symlets "sym4" Wavelet

wavelet = sym4 name of the wavelet

R2 = 0.89189 and 0.90362 coefficient of determination

rmse = 3.4501 and 5.5436 root mean square error

stdv = 3.4934 and 4.9449 standard deviation

sem = 1.7118 and 1.7133 standard error of the mean

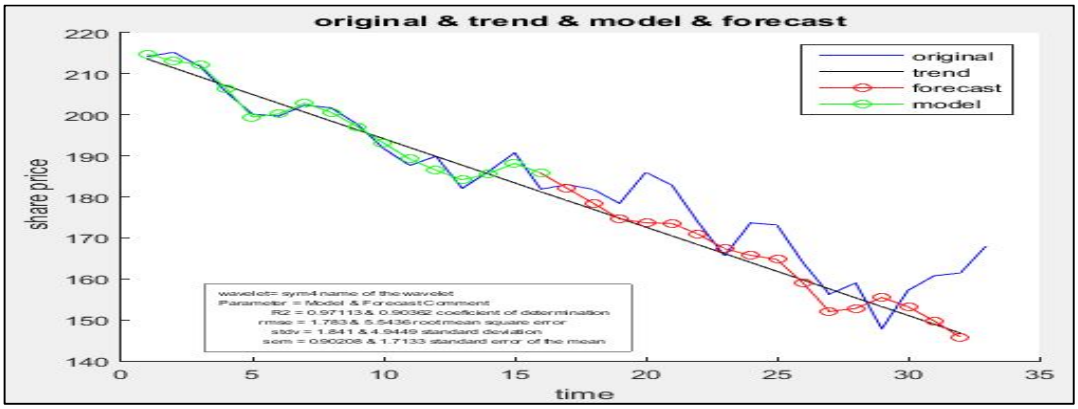


FIGURE 7-4 SYMLETS SYM4 WAVELET FORECAST 160 AND 35 DAYS

7.4.3 Forecast with Biorthogonal “bior4.4” Wavelet

wavelet = bior4.4 name of the wavelet

R2 = 0.97341 and 0.89619 coefficient of determination

rmse = 1.7112 and 5.7534 root mean square error

stdv = 1.7658 and 5.1443 standard deviation

sem = 0.86525 and 1.7824 standard error of the mean

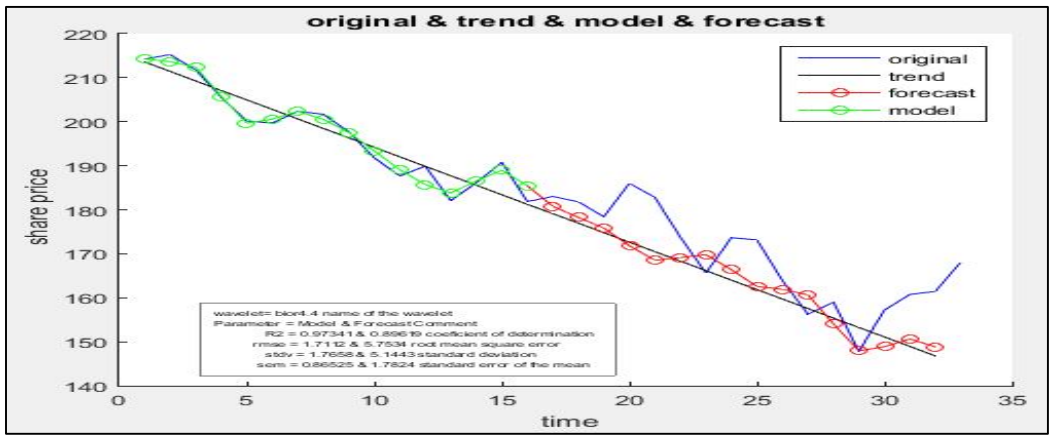


FIGURE 7-5 FORECAST WITH BIORTHOGONAL BIOR4 .4 WAVELET

7.4.4 Forecast with Reverse Biorthogonal “rbio2.2” Wavelet

wavelet = rbio4.4 name of the wavelet

R2 = 0.9746 and 0.89556 coefficient of determination

rmse = 1.6722 and 5.7709 root mean square error

stdv = 1.7268 and 5.1747 standard deviation

sem = 0.84615 and 1.793 standard error of the mean

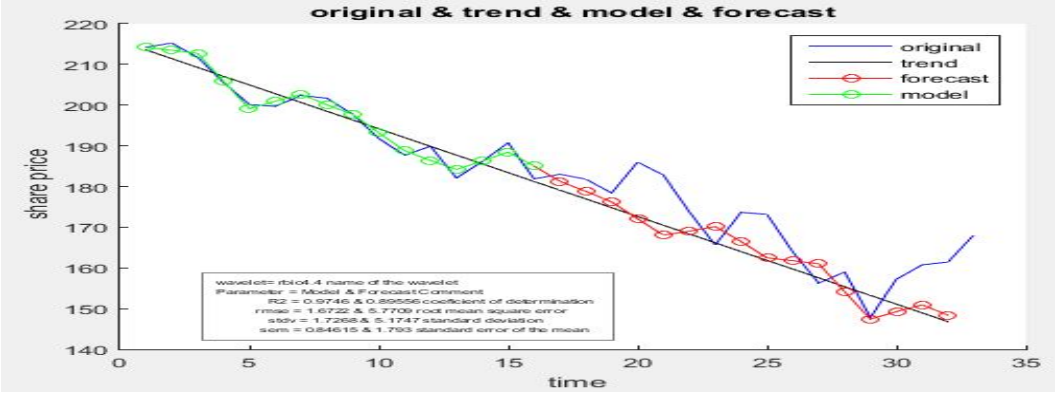


FIGURE 7-6 FORECAST WITH REVERSE BIORTHOGONAL RBIO2.2 WAVELET

7.5. Comparison of Wavelet Forecast Performance

A comparison of the wavelets forecasting performance is shown in Table 7-1.

TABLE 7-1 COMPARISON OF DIFFERENT WAVELET FORECASTS

	DFT	db4	Sym4	bior2.2	rbio2.2
R2	0.88908	0.90046	0.90362	0.89619	0.89556
rmse	5.9471	5.6339	5.5436	5.7534	5.7709
stdv	5.1867	4.8396	4.9449	5.1443	5.1747
sem	1.7971	1.6768	1.7133	1.7824	1.793

The best performance is with the Symlets "sym4" wavelet and the worst is with the DFT. Figure 7-7 shows the Symlets wavelet forecasting graph. There is a good likeliness fit R2=0.9362 between the target and the model.

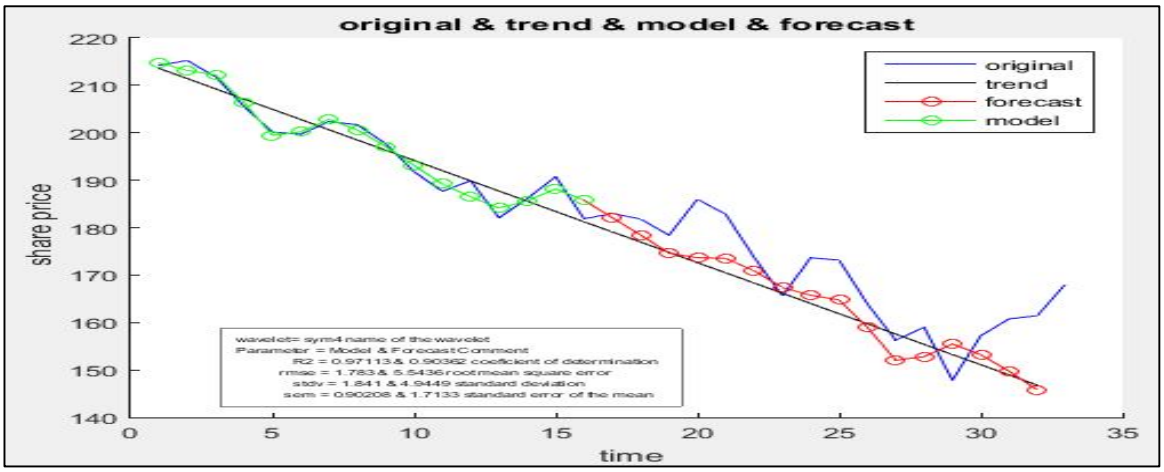


FIGURE 7-7 BEST PERFORMANCE FORECAST WITH SYM4 WAVELET

The Locally Stationary Wavelet (LSW) may be better than Fourier extrapolation and is commonly used in predicting time series.

7.6. Comparison of Neural Networks with Wavelets

Chandar proposed a hybrid model combining the Discrete Wavelet Transform (DWT) and an Artificial Neural Network (ANN) to forecast future stock prices (Chandar, et al., 2016). The historical data series, for example of closing prices, are decomposed via DWT and the coefficients form the feature vector (A and optionally D) of the ANN, for example using a

back propagation neural network (BPNN). The proposed model was compared to the ANN model with the original data feature vector, although the structure of the feature vector and the forecast target are not clearly described. Others authors have used different ANN and DWT combinations and approaches, such as a Recurrent Self-Organizing Map (RSOM) neural network with multiple kernel regression succeeding lower forecasting RMSE (Huang, et al., 2010); the DWT and Recurrent Neural Network (RNN) (Hsieh, et al., 2011); the DFT with kernel Partial Least Square (PLS) regression, support vector machines and GARCH models outperforming traditional NN (Huang, et al., 2011); an ANN at many DFT decomposed levels (Wang, et al., 2011); the DWT with Support Vector Machines (SVMs) with different kernels (Lahmiri, 2013); and the DWT with BPNN compared to the time domain model (Lahmiri, 2014).

7.6.1 Conceptual Model

Generally, a time series of historical share prices, such as closing prices, is decomposed with the DWT and the feature vector of the approximation (A) and sometimes detail (D) coefficients is fed into the ANN. Three-layer feed-forward back propagation neural networks (BPNNs) as training algorithms are capable of approximating any continuous function with the desired accuracy (Devadoss, et al., 2013) and are common in financial forecasting.

The usual practice is that 75% of the dataset is used for training and 25% for testing. Finally, the performance of the model is measured with statistics such as coefficient of variation (CoV), mean absolute deviation (MAD), root mean square error(RMSE), and mean absolute error (MAE).

From the empirical results, the assertion is that the models with DWT input outperform those with original historical time data. The proposed model can be combined further with other machine learning algorithms such as Particle Swarm Optimization (PSO) to improve the accuracy of prediction.

7.6.2 Experimental Set-up

An ANN with DWT has been empirically investigated for better forecast performance in the banking UK sector for the period 2016-01-01 to 2016-08-05. Historic closing prices were downloaded from uk.finance.yahoo.com for Barclays, HSBC, RBS Lloyds and Virgin banks. The tuning of the Back Propagation Neural Network (BPNN) varied the random seed from 0-100 with a threshold of 10% of the shares range. The performance indicators used are square root power of two error (*sqre*) and maximum error (*maxe*) in percentage of the share price range. The experiments are conducted for the closing price in the time domain (without using the discrete wavelet transformation DWT) and in the discrete wavelet domain using the Daubechies-db4, Biorthogonal-bior4.4, Symlets-sym4 and Reverse Biorthogonal-rbio2.2 algorithms. The time series window is 16 days. There are two datasets, one for training of ninety days from the starting date and one for testing of twenty days from day 110.

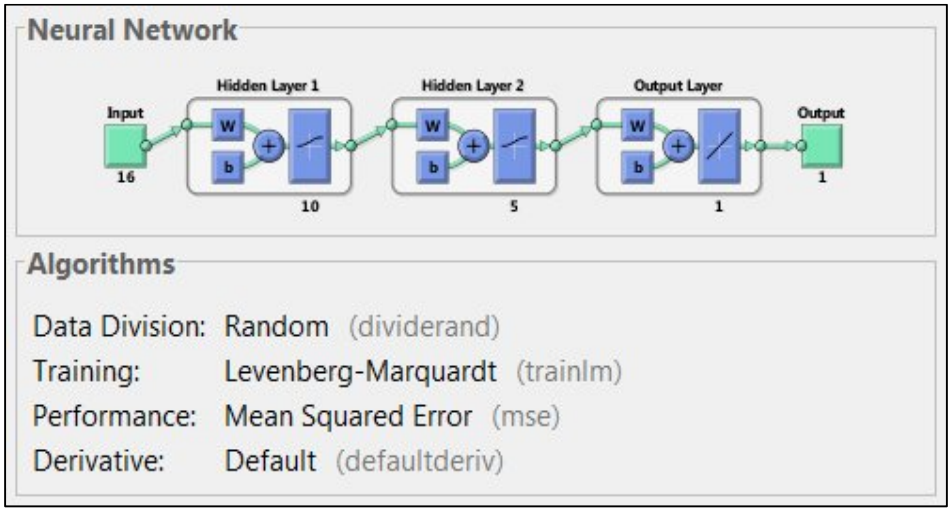


FIGURE 7-8 NEURAL NETWORK ARCHITECTURE WITH WAVELETS

7.6.3 Neural Networks with Wavelets Experiments

Experiments are conducted with different wavelets, db4, sym4, bio4.4 and rbio4.4, for different share prices, HSBC, Barclays, RBS, Virgin and Lloyds. The random generator seed is given as well to facilitate the eventual reproduction of the experiments.

TABLE 7-2 FORECASTING HSBC NEURAL NETWORKS WITH WAVELETS

HSBA.L	no	db4	sym4	bior4.4	rbio4.4
sqre	6.8646	6.2146	4.216	6.4765	6.502
maxe	12.4141	14.9278	10.647	16.4901	13.7189
seed	12	32	92	76	32

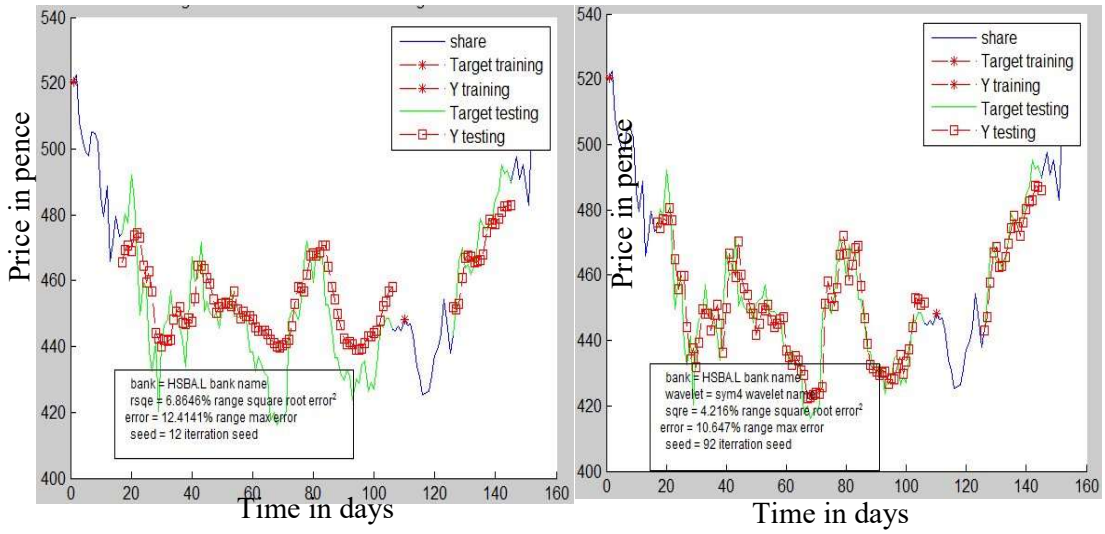


FIGURE 7-9 FORECAST HSBC ANN WITHOUT AND WITH SYM4

TABLE 7-3 FORECASTING BARCLAYS NEURAL NETWORKS AND WAVELETS

BARC.L	no	db4	sym4	bior4.4	rbio2.2
sqre	5.0829	5.853	7.5832	7.8834	6.1368
maxe	9.7337	10.7108	15.011	15.3157	12.4065
seed	54	53	82	58	27

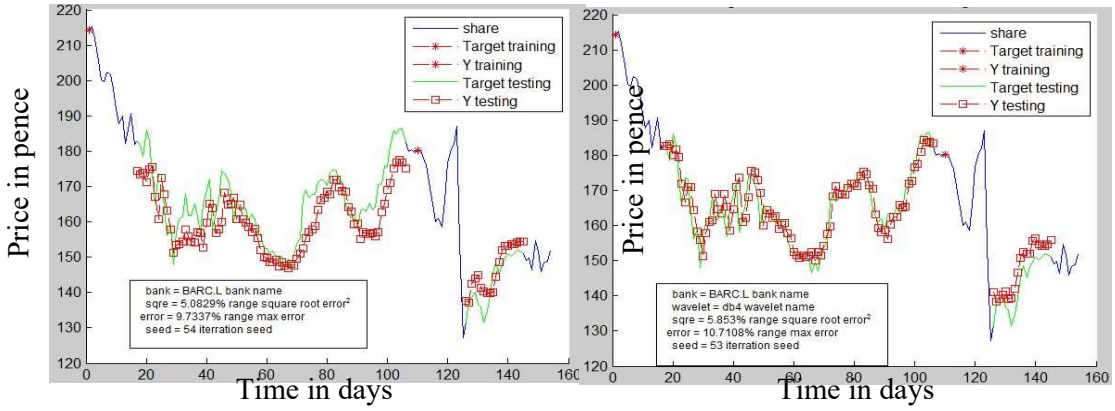


FIGURE 7-10 FORECAST BARC.L ANN WITHOUT AND WITH DB4

TABLE 7-4 FORECAST RBS NEURAL NETWORKS WITH BIOR4 WAVELETS

RBS.L	no	db4	sym4	bior4.4	rbio4.4
sqre	15.4566	15.2195	15.4342	11.4388	17.4986
maxe	29.9803	30.6334	30.3305	21.0302	29.7185
seed	98	29	82	22	67

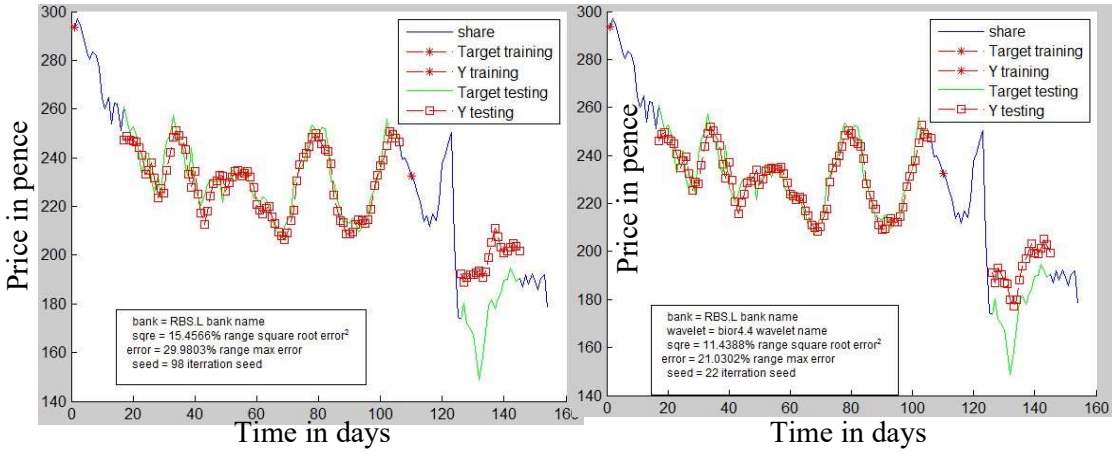


FIGURE 7-11 FORECAST RBS ANN WITHOUT AND WITH BIOR4

TABLE 7-5 FORECAST VIRGIN NEURAL NETWORK DB4 WAVELETS

VM.L	no	db4	sym4	bior4.4	rbio4.4
sqre	19.4306	18.9692	22.4121	26.2529	27.2544
maxe	27.1149	36.3175	45.5554	55.8468	42.2174
seed	25	19	29	12	85

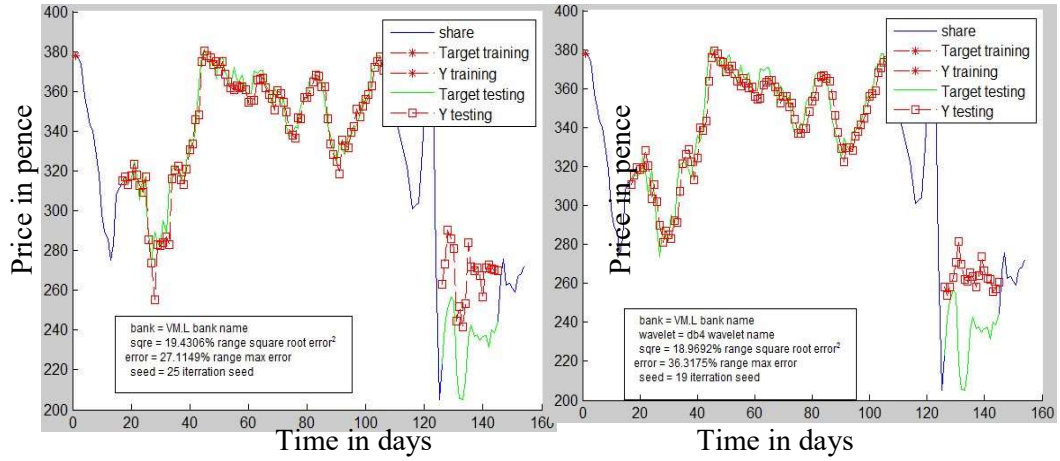


FIGURE 7-12 FORECAST VIRGIN ANN WITHOUT AND WITH DB4

TABLE 7-6 FORECAST LLOYDS NEURAL NETWORKS WITH DB4 WAVELETS

LLOY.L	no	db4	sym4	bior4.4	rbio4.4
sqre	11.019	5.9454	19.3016	14.7169	19.2868
maxe	16.6179	15.8725	31.2209	31.4723	38.82
seed	34	88	24	48	48

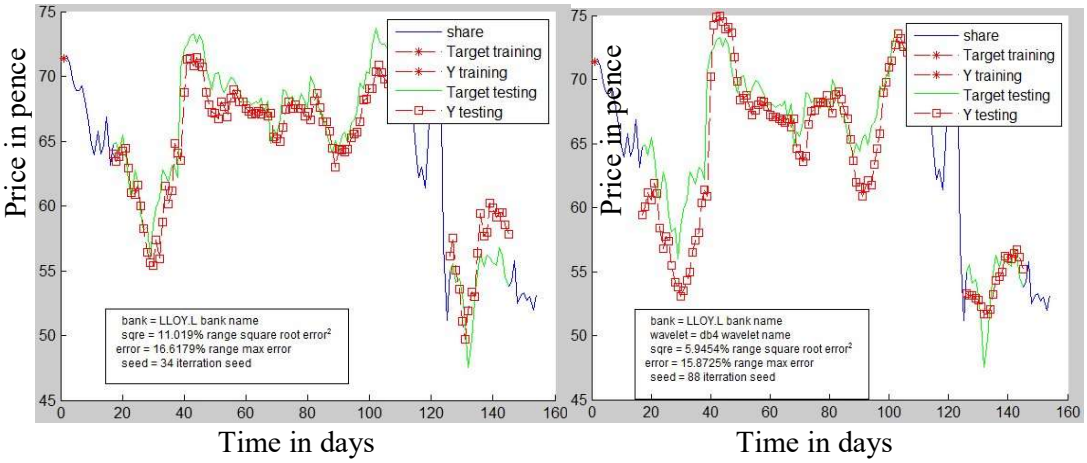


FIGURE 7-13 FORECAST LLOYDS ANN WITHOUT AND WITH DB4

The best performance achieved for RBS is with the Biorthogona wavelet, for HSBC it is with the Symlets wavelet and for Virgin, Lloyds and Barclays it is with Daubechie's wavelet. The models deal reasonably well with the drop in price around sample 120 except for the Virgin model which has poor performance bias following the drop.

7.7. Summary

Wavelet transform (DWT) time series analysis and decomposition has proven to provide significantly better performance, robustness and generalization than analysis in the time domain. Generally, the discrete wavelet transform improves forecasting performance, but the comparison with different wavelets is indecisive because there is no common choice of representative wavelet for the banking sector.. The broader range of wavelets could be explored for a better fit, but overall it seems that Daubechie's is suitable for the sector in the first instance. The model is very sensitive to the neural network tuning.

The data for the banking sector were obtained from the following websites

- Barclays PLC (BARC.L)

https://uk.finance.yahoo.com/q?s=+BARC.L&ql=1

- HSBC Holdings plc (HSBA.L)

https://uk.finance.yahoo.com/q?s=HSBA.L&ql=1

- The Royal Bank of Scotland Group plc (RBS.L)

https://uk.finance.yahoo.com/q?s=RBS.L&ql=0

- Virgin Money Holdings (UK) plc (VM.L)

https://uk.finance.yahoo.com/q?s=VM.L&ql=0

- LLOY.L Lloyds

https://uk.finance.yahoo.com/q/hp?s=LLOY.L

The following types of wavelets have been used:,

- Biorthogonal “bior4.4” wavelet
- Symlets “sym4” wavelet
- Daubechies “db4” wavelet
- Reverse Biorthogonal “rbio2.2” wavelet

Chapter 8. Particle Swarm Optimization

8.1. Overview

This chapter introduces the particle swarm optimization (PSO) paradigm and through the control theory techniques, models and analyzes its stability. A further proportional, derivative and integral (PID) extension of the basic algorithm is proposed. Integration with neural networks and a methodology of application is proposed.

Particle Swarm Optimization (PSO) is a stochastic optimization technique originally formulated by Edward and Kennedy in 1995. The algorithm is inspired by and based on the behaviour of swarms, such as groups of birds or fish. Despite the relative simplicity of individuals, swarm systems display complex behaviour. They are made up of numerous individuals and tend to be flexible and robust. Swarm intelligence thus provides a framework for the design and implementation of systems made up of many agents that are capable of cooperation for the solution of complex problems.

This chapter covers the following:

- Mathematical analysis of the PSO algorithm
- Inertia and consolidated convergence canonical form
- Analysis in continuous time Laplace transform
- Analysis in the discrete domain
- Randomization investigations
- Boundary conditions
- PSO with exponentially varying inertia
- Chaotic adaptive PSO using logistics and Gauss map
- Particle swarm optimization PID extension
- PSO and neural networks

8.2. Introduction

The PSO idea has expanded to become a common heuristic optimization algorithm with many interpretations of its concepts, issues, and applications. One common feature of heuristic approaches is that they use probabilistic rules to find the global optimal solution and may prove to be very effective in solving problems without modifying the shape of their cost curves. Olson (2011) has presented information on particle swarm optimisation in a comparative study of different approaches in theory and practice. Subjects considered include using mono-objective or multi-objective particle swarm optimisation for the tuning of process control laws, convergence issues in particle swarm optimisation, and a study on topology problems using enhanced particle swarm optimisation. Yang, et al. (2013) has reviewed the latest developments in theory and applications concerning swarm intelligence and bio-inspired computation and provided an overview of some of the most widely used bio-inspired algorithms, especially those based on swarm intelligence (SI) such as the cuckoo search, firefly algorithm, and particle swarm optimization. The essence of the algorithms and their connections to self-organization are also analyzed. Furthermore, the main challenging issues associated with these metaheuristic algorithms, for example the tuning of algorithm dependent parameters, randomization techniques and convergence, are considered along with significant applications and case studies such as structural optimization and improvement using memory-based gradients. Various opportunities and challenges regarding dimensionality and convergence are also discussed. Despite its simplicity, the large numbers of variations of the algorithm give a wide variety of choices. This variety makes it challenging to determine which version can be the most appropriate for a particular problem. Bratton et al. (2007) addressed the need for an updated definition and suggested extensions of the original algorithm that could improve performance. This represents the evolution of the original algorithm designed to take advantage of subsequent generally applicable improvements such as in topology, the number of particles, and

inertia. It can be used as a baseline test for the performance of improvements to the technique.

The advantage of PSO is that it can be used in cases with non-differentiable transfer functions and when no gradient information is available. The disadvantages are that performance is not necessarily competitive for some problems, and the representation of weights is difficult and they have to be carefully selected or developed. Furthermore, the potential advantages of the swarm intelligence approach are as follows (Dehuri, 2011):

- collective robustness, where the failure of individual components does not significantly hinder performance;
- individual simplicity, in that cooperative behaviour makes it possible to reduce the complexity of the individuals; and
- scalability, since the control mechanisms used are not dependent on the number of agents in the swarm.

The system starts with a population of random solutions and searches for optima by updating generations. The particles in the swarm are defined with their corresponding parameters. All particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flight of the particles. The particles float through the solution space by following the current personal and global optimum particles. The PSO algorithm begins with a random population, and random values of parameters. Each particle moves around in the cost solution space. The particles update the change in their position parameters, which are termed velocity (*vel*) and position (*par*) by referring to the local (*pBest*) and global best common minima (*gBest*) of the cost function.

The pseudo-code of the procedure is as follows:

Initialize the swarm particles

While maximum iterations or minimum error criteria is not attained

For each particle

Randomly change particle position (vel)

Calculate fitness value

Check and update the personal best pBest (p)

Check and update the global best gBest (g)

Calculate particle velocity

Update particle position

End

End

After finding the two best values, the particle updates its velocity and positions using the following equation:

$$v_{t+1} = w * v_t + c_1 * r_1 * (p - x_t) + c_2 * r_2 * (g - x_t) \quad (8\text{-}1)$$

$x_{t+1} = x_t + v_{t+1}$, updates particle position (8-2)

where,

v is velocity

x is a parameter

r_1, r_1 are independent uniform random numbers

c_1 is the cognitive parameter

c_2 is the social parameter

p is the local best

g is the global best

w is inertia

The PSO algorithm updates the velocity vector for each particle and then adds that velocity to the particle position or values. Velocity updates are influenced by both the best global solution associated with the lowest cost ever found by a particle and the best local solution associated with the lowest cost in the present population (see Figure 8-1).

If the best local solution has a cost less than the cost of the current global solution, then the best local solution replaces the best global solution. The particle velocity is a derivative of

position. The constant c_1 is the cognitive parameter, and the constant c_2 is the social parameter. The advantages of PSO are that it is easy to implement and there are few parameters to adjust (Haupt, et al., 2004).

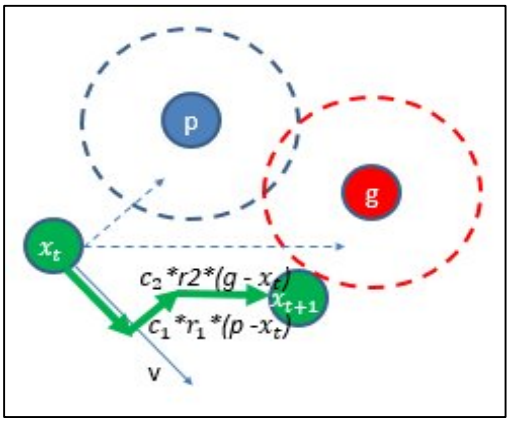


FIGURE 8-1 PSO SEARCH MODEL

The PSO is able to resolve cost functions with many local minima. Figure shows the initial random swarm set moving in the parameter space. Particle swarming becomes evident as the generations pass to reach the global minima.

The largest group of particles ends up close to the global minimum and the next largest group is near to the next lowest minimum. A few other particles are roaming the cost surface at some distance away from the two groups (see Figure 8-2).

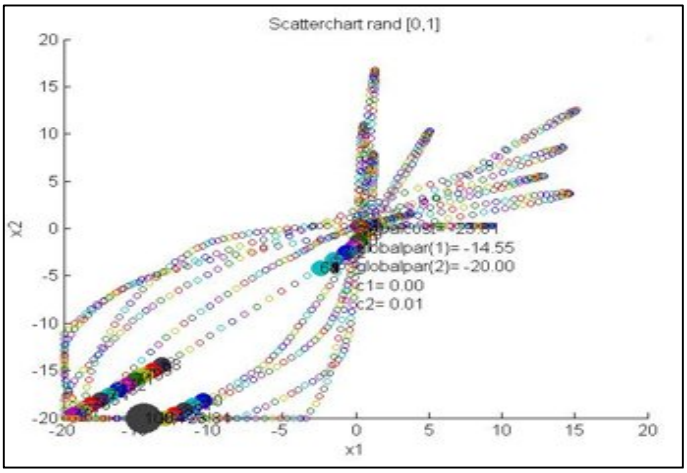


FIGURE 8-2 PSO SEARCH PATH

Figure 8-3 shows plots of the local and global best values as well as the population average as a function of generation. The chaotic swarming process is illustrated by

following the path of one of the particles until it reaches the global minimum. In this implementation, the particles frequently bounce off the boundaries.

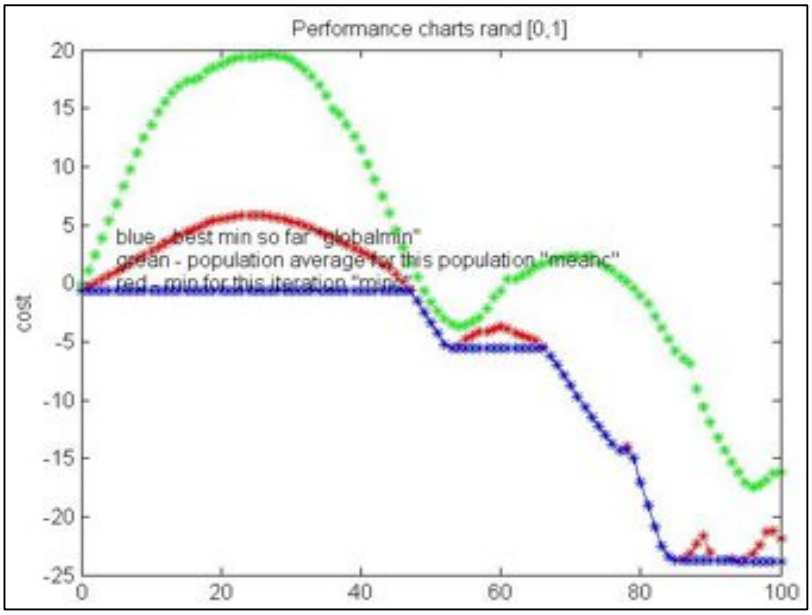


FIGURE 8-3 PSO PERFORMANCE CHART BEST MIN, AVERAGE AND CURRENT MIN

Das and Dehuri (2011) presented a survey of PSO for single- and multi-objective problems. A considerable number of algorithms have been and are being proposed for these problems based on either tuning or introducing various PSO parameters. The authors identified some application areas where PSO has clear advantages over other meta-heuristic approaches for solving single and multi-objective optimization problems.

8.3. Mathematical Analysis of the PSO Algorithm

As stated previously, the general PSO algorithm in the scalar case can be written as:

$$v_{t+1} = w * v_t + c_1 * r_1 * (p_t - x_t) + c_2 * r_2 * (g_t - x_t) \quad (8\text{-}3)$$

$$x_{t+1} = x_t + v_{t+1} \quad (8\text{-}4)$$

This can be seen as a discretization of the following continuous system:

$$\frac{dx(t)}{dt} = v(t) \quad (8\text{-}5)$$

$$\frac{dv(t)}{dt} = wv(t) + \varphi_1 r_1\big(p - x(t)\big) + \varphi_2 r_2(g - x(t)) \quad (8\text{-}6)$$

where equations are discretised using the Euler method:

$$\frac{dx(t)}{dt} = \frac{x(t+1)-x(t)}{T} \tag{8-7}$$

$$\frac{dv(t)}{dt} = \frac{v(t+1)-v(t)}{T} \tag{8-8}$$

where T is the sampling interval.

$$v_{t+1} = (1 + wT)v_t + T\varphi_1 r_1(p_t - x_t) + T\varphi_2 r_2(g_t - x_t) \tag{8-9}$$

$$v_{t+1} = (1 + wT)v_t + u_t \tag{8-10}$$

where $u_t = T\varphi_1 r_1(p_t - x_t) + T\varphi_2 r_2(g_t - x_t)$ (8-11)

$$x_{t+1} = x_t + Tv_{t+1} \tag{8-12}$$

where $(1 + Tw) = w, \quad T\varphi_1 = c_1 \quad$ and $\quad T\varphi_2 = c_2$

$$u_t = k_1(p_t - x_t) + k_0(g_t - x_t) \tag{8-13}$$

where $k_1 = c_1 r_1$ and $k_0 = c_0 r_2$

The latter is the equation of a closed-loop feedback system with a proportional control term, as illustrated in Figure 8-4.

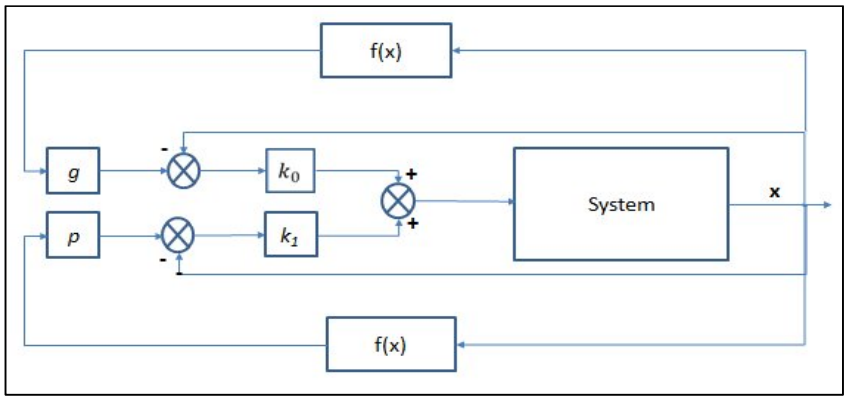


FIGURE 8-4 CLOSED LOOP FEEDBACK SYSTEM PROPORTIONAL CONTROL

Furthermore, transformations for x_t and p_t signals are:

$$u_t = \varphi_1 * r_1 * (p_t - x_t) + \varphi_2 * r_2 * (g_t - \boldsymbol{p_t} + \boldsymbol{p_t} - x_t) \tag{8-14}$$

$$u_t = (\varphi_1 r_1 + \varphi_2 r_2) * (p_t - x_t) + \varphi_2 r_2(g_t - p_t) \tag{8-15}$$

$$u_t = k_P(p_t - x_t) + k_0(g_t - p_t) \tag{8-16}$$

$$k_P = (\varphi_1 r_1 + \varphi_2 r_2) \; and k_0 = \varphi_2 r_2 \tag{8-17}$$

This shows proportional control with a feed-forward term, as shown in Figure 8-5.

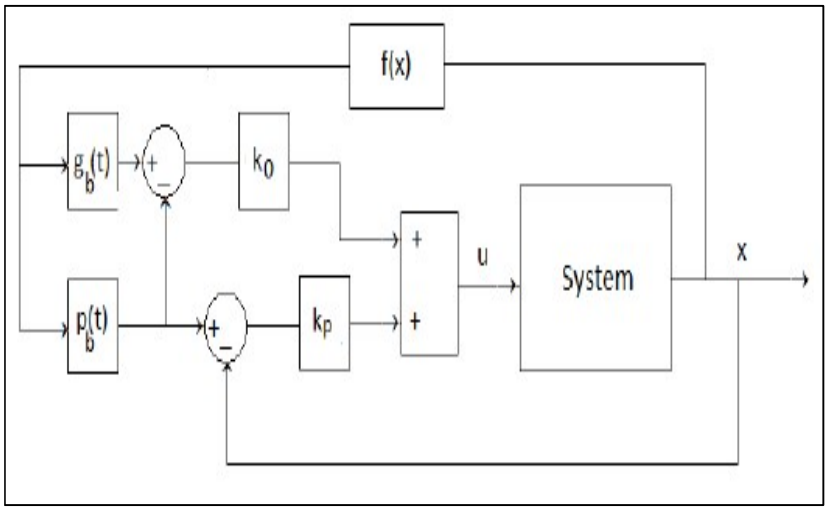


FIGURE 8-5 PROPORTIONAL CONTROL WITH FEED-FORWARD TERM

8.4. Inertia and Consolidated Convergence Canonical Form

Bradford et al. (2011) have discussed convergence issues caused by the PSO structure, as a population-based swarm moving together to the optimal point. They present heuristic methods to address these issues in current practice and provide some optimization solutions, and investigate how much convergence is required, when to stop by redefining the search space, the local and multiple optima landscape and methods for resuming exploration after an optimal success. Furthermore, topics such as the control of population velocity and exploration are addressed. They also suggest some guidance for convergence if the problem landscape is unknown, where different algorithms are tried first to determine if the landscape is multimodal or unimodal. If the diversity is too great, then the search space should be tightened. Campana et al. (2010) discussed a class of unconstrained optimization problems where evaluation is costly and the exact algorithm is too large to compute. They consider the evolutionary algorithm (Kennedy, et al., 1995) and introduced some global convergent modifications following previous research (Lucidi, et al., 2002) and convergence conditions that are useful for developing and analysing new derivative-free algorithms with guaranteed global convergence. The sequences of stationary limit points for the objective function, with suitable ranges of parameters, are identified to avoid particle trajectory divergence. Under mild assumptions, at least a sub-sequence of the

iterates produced by the modified PSO method, combining PSO with two derivative free algorithms, converges to a stationary point which is possibly a minimum point.

The particle velocities are commonly clamped at a maximum velocity. If this is not done the system is prone to enter an unstable state wherein the random weighting values cause velocities and thus particle positions to accelerate rapidly (Bratton, et al., 2007). Shi et al. (1998) proposed an inertia parameter setting to try to improve convergence by reducing the velocity towards the end of the search, assuming that the solution is close to the optimum. The inertia for each iteration *iter* is a fraction of the maximum iterations *maxiter:*

$$w = \frac{maxiter - it}{maxiter} \tag{8-18}$$

This would be a fraction of one and would slow down the search. Choosing a value of inertia greater than one could cause non-convergence, and hence requires a check for convergence.

With the control theory analysis, the core criteria for convergence can be identified. The equation for velocity is:

$$v_{t+1} = w * v_t + c_1 * r_1 * (p_t - x_t) + c_2 * r_2 * (g_t - x_t) \tag{8-19}$$

$$x_{t+1} = x_t + v_{t+1} \tag{8-20}$$

This can be transformed into a canonical form:

$$v_{t+1} = w * v_t + c_1 * r_1 * p_t - c_1 * r_1 * x_t + c_2 * r_2 * g_t - c_2 * r_2 * x_t \tag{8-21}$$

$$v_{t+1} = w * v_t + c_1 * r_1 * p_t + c_2 * r_2 * g_t - (c_1 * r_1 + c_2 * r_2) * x_t \tag{8-22}$$

$$v_{t+1} = w * v_t + (c_1 * r_1 + c_2 * r_2) * \left[\frac{c_1 * r_1 * p_t + c_2 * r_2 * g_t}{(c_1 * r_1 + c_2 * r_2)} - x_t\right] \tag{8-23}$$

Further substitutions are then made:

$$\varphi = (c_1 * r_1 + c_2 * r_2) \tag{8-24}$$

$$q_t = \frac{c_1 * r_1 * p_t + c_2 * r_2 * g_t}{\varphi} \tag{8-25}$$

So, the canonical form is now:

$$v_{t+1} = w * v_t + \varphi * (q_t - x_t) \tag{8-26}$$

$$x_{t+1} = x_t + v_{t+1} \tag{8-27}$$

The usual values are $\varphi = 4.1$ and swarm size = 20 (Clerc, et al., 2002)

Furthermore, replacing $e = q - x$:

$$v_{t+1} = w * v_t + \varphi * e_t \tag{8-28}$$

$$e_{t+1} = -w * v_t + (1 - \varphi) * e_t \tag{8-29}$$

The matrix form is then:

$$\begin{bmatrix} v_{t+1} \\ e_{t+1} \end{bmatrix} = \begin{bmatrix} w & \varphi \\ -w & 1-\varphi \end{bmatrix} \begin{bmatrix} v_t \\ e_t \end{bmatrix} = C \begin{bmatrix} v_t \\ e_t \end{bmatrix} \tag{8-30}$$

This is a dynamic system, whose behaviour is determined by the eigenvalues of the matrix C. A condition of convergence is where the eigenvalues are two combined complex numbers of modulus less than 1 or two real numbers with absolute values less than 1. They are solutions to the following equation:

$$\begin{vmatrix} w - \lambda & \varphi \\ -w & 1 - \varphi - \lambda \end{vmatrix} = \lambda^2 + (\varphi - w - 1) * \lambda + w = 0 \tag{8-31}$$

with the discriminant:

$$\Delta = (\varphi - w - 1)^2 - 4 * w \tag{8-32}$$

Convergence means that the particle tends towards a stable position with velocity tending towards zero although there are no guarantees that this is the optimum (Clerc, et al., 2002). Furthermore, balancing of the global and local searches, known as constriction, is proposed. Similar to the inertia weight method, this method introduces a new parameter χ, known as the constriction factor. The value ofχ is derived from the existing constants in the velocity update equation:

$$\chi = \frac{2}{|-\varphi - \sqrt{\varphi^2 - 4\varphi}|}, \ \varphi = c_1 + c_2 \tag{8-33}$$

It was found that, when $\varphi < 4$, the swarm would slowly "spiral" toward and around the best solution found in the search space with no guarantee of convergence, while for $\varphi > 4$ convergence would be quick and guaranteed. Most implementations of constricted particle swarms use equal values for both parameters. Using the constant $\varphi = 4.1$ to ensure convergence, the values of $\chi \approx 0.72984$ and $c1 = c2 = 2.05$. This constriction factor is applied to the entire velocity update equation:

$$v_{i+1} = \chi(v_{i+1} + c_1 r_1(p_i - x_i) + c_2 r_2(g_i - x_i)) \quad (8\text{-}34)$$

Swarm behaviour is eventually limited to a small area of the feasible search space containing the best known solution. A comparison study of constriction and inertia showed that the former is in fact a special case of inertia in which the values for the parameters have been determined analytically (Eberhart, et al., 2010). The parameter values above are used in most cases due to their stability (Clerc, et al., 2002).

8.5. Analysis in Continuous Time Laplace Transform

The open-loop system is described by the following equations:

$$\frac{dv}{dt} = wv + u \quad (8\text{-}35)$$

$$\frac{dx^2}{dt} = w\frac{dx}{dt} + u \quad (8\text{-}36)$$

$$u_t = k_P(p_t - x_t) + k_0(g_t - p_t) \quad (8\text{-}37)$$

The open-loop Laplace transform transfer function of the system is:

$$s^2X(s) = wsX(s) + U(s) \quad (8\text{-}38)$$

$$H(s) = \frac{X(s)}{U(s)} = \frac{1}{s^2 - ws} = \frac{1}{s(s-w)} \quad (8\text{-}39)$$

$$X(s) = H(s)U(s) \quad (8\text{-}40)$$

$$U(s) = k_P\big(P(s) - X(s)\big) + k_0(G(s) - P(s)) \quad (8\text{-}41)$$

The close system transfer function, replacing U(s), is:

$$X(s) = H(s)(k_P\big(P(s) - X(s)\big) + k_0(G(s) - P(s))) \quad (8\text{-}42)$$

$$X(s) = k_P H(s)P(s) - k_P H(s)X(s) + k_0 H(s)G(s) - k_0 H(s)P(s) \quad (8\text{-}43)$$

$$X(s) = \Gamma(s)P(s) + \Omega(s)G(s) \tag{8-44}$$

$$\Gamma(s) = \frac{(k_P - k_0)H(s)}{1 + k_P H(s)} = \frac{(k_P - k_0)}{s^2 - ws + k_P} \tag{8-45}$$

$$\Omega(s) = \frac{k_0 H(s)}{1 + k_P H(s)} = \frac{k_0}{s^2 - ws + k_P} \tag{8-46}$$

For a system to be stable, it is sufficient that its transfer function has no poles on the right semi-plane:

$$s_{1,2} = \frac{w^{+/-}\sqrt{w^2 - 4k_P}}{2} \tag{8-47}$$

To achieve this, $w \leq 0$, and for $w = 0$ it is a stable harmonic.

The simulation results prove that, for $w < 0$, the solution is stable.

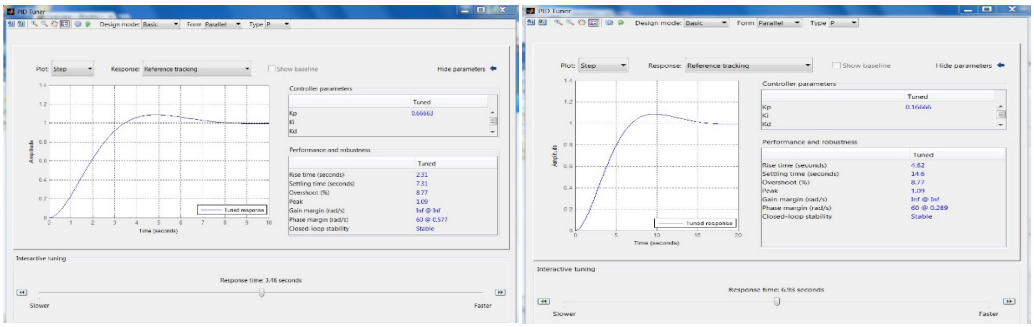


FIGURE 8-6 STEP FUNCTION RESPONSE FOR INERTIA *W*=-1 AND *W*=-0.5

For the case of $w = 0$, the simulation is a harmonic oscillation.

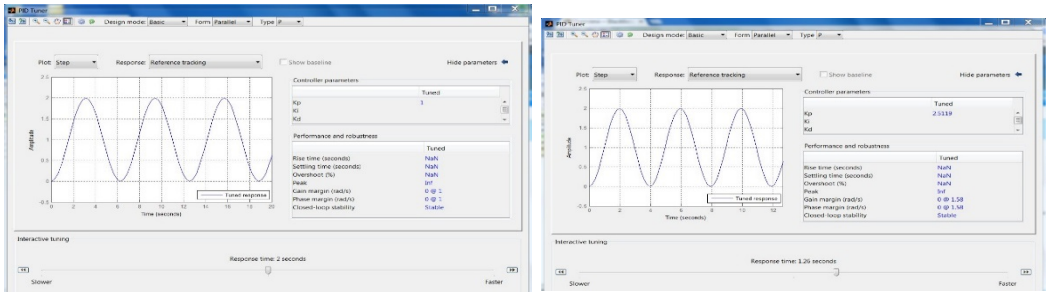


FIGURE 8-7 STEP FUNCTION RESPONSE (HARMONIC) FOR INERTIA *W*=0 AND *KP*=1 AND 2.5

The simulation failed to find an initial stabilizing controller for the case of $w > 0$. For the common practice inertia value of $w = 1$and four different values of *Kp* the step function response is unstable, as shown in Figure 8-8.

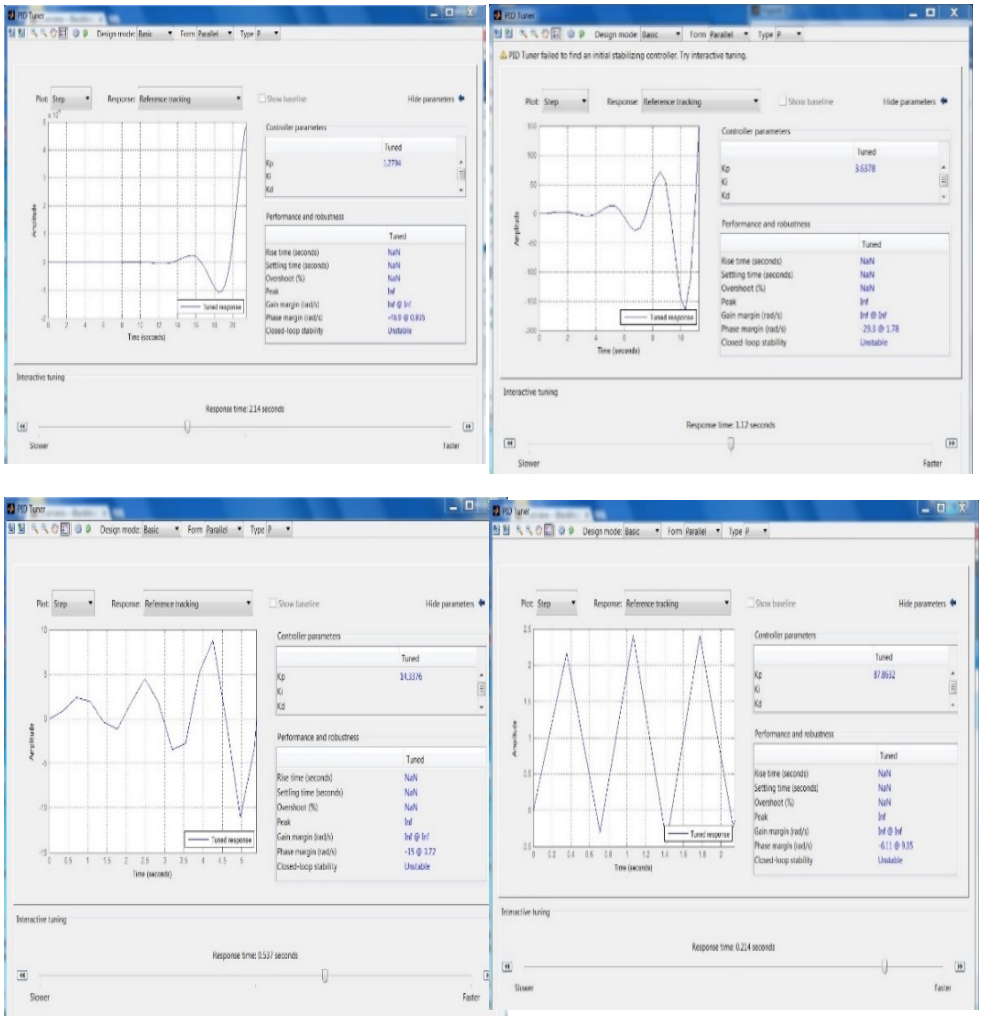


FIGURE 8-8 STEP FUNCTION RESPONSE INERTIA W=1 AND FOUR DIFFERENT VALUES OF KP

8.6. Analysis in the Discrete Domain

The PSO algorithm is generally given in the discrete domain. Consequently, the analysis in the discrete domain is more relevant in terms of practical implementation. For the $k+1$ iteration:

$$x_{k+1} = x_k + v_{k+1} \tag{8-48}$$

$$v_{k+1} = wv_k + u_k \tag{8-49}$$

$$u_k = k_p(p_k - x_k) + k_o(g_k - p_k) \tag{8-50}$$

The z-transform is:

$$zX(z) = X(z) + zV(z) \tag{8-51}$$

$$zV(z) = wV(z) + U(z) \tag{8-52}$$

$$X(z) = H(z)U(z) = \frac{z}{(z-w)(z-1)}U(z) \tag{8-53}$$

$H(z) = \frac{z}{(z-w)(z-1)}$ is the z-transform of the open system (8-54)

$$U(z) = k_P\big(P(z) - X(z)\big) + k_0\big(G(z) - P(z)\big) \tag{8-55}$$

$$X(z) = \frac{(k_P+k_0)H(z)}{1+k_PH(z)}P(z) + \frac{k_0H(z)}{1+k_PH(z)}G(z) \tag{8-56}$$

$$X(z) = \Gamma(z)P(z) + \Omega(z)G(z) \tag{8-57}$$

$$\Gamma(z) = \frac{(k_P+k_0)H(z)}{1+k_PH(z)} = \frac{(k_P-k_0)z}{z^2+(k_P-w-1)z+w} \tag{8-58}$$

$$\Omega(z) = \frac{k_0H(z)}{1+k_PH(z)} = \frac{k_0z}{z^2+(k_P-w-1)z+w} \tag{8-59}$$

The poles of the closed loop system are:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a} \tag{8-60}$$

$$z_{1,2} = \frac{-(k_P-w-1) \pm \sqrt{(k_P-w-1)^2-4w}}{2} \tag{8-61}$$

For stability, if w=1 and poles are in the unit circle, the conditions are as follows:

$$z_1 = \frac{-(k_P-2)+\sqrt{(k_P-2)^2-4}}{2},\ -1 < z_1 < 1 \tag{8-62}$$

$$z_2 = \frac{-(k_P-2)-\sqrt{(k_P-2)^2-4}}{2},\ -1 < z_2 < 1 \tag{8-63}$$

8.7. Test Functions

In order to determine how well an optimization algorithm works, a variety of test functions have been used as a check. There are 16 test functions, as shown in Appendix E: Figure 1, Appendix E: Figure 2, Appendix E: Figure 3 and Appendix E: Figure 4. In each case a general form of the function is given, and its value plotted in one or two dimensions, where the global optimum is given in one or two dimensions. Some of the functions are generalizable to *N* dimensions. Some of the research into the development of test functions has been published elsewhere (Haupt, et al., 2004).

8.8. Randomization Investigations

On analysing bio-inspired algorithms in more detail, the type of randomness that a particular algorithm is employing can be identified. For example, the simplest and yet often very efficient method is to introduce a random starting point for a deterministic algorithm. The well-known hill-climbing method with random re-start is a good example. It attempts to maximize (or minimize) a target function, where the parameter is a vector of continuous and/or discrete values. At each iteration, the hill-climbing method will adjust a single parameter and determine if the change improves the value of the target function. This differs from gradient descent methods, which adjust all of the values of parameters at each iteration according to the gradient of the hill. With hill climbing, any change that improves the target function is accepted, and the process continues until no change can be found which will improve the value of the target function. Then the parameters are said to be "locally optimal". This simple strategy is both efficient, in most cases, and easy to implement.

A more elaborate way to introduce randomness into an algorithm is to use randomness inside its different components, and various probability distributions such as uniform, Gaussian, and Levy distributions can be used for randomization (Talbi, et al., 2009; Yang et al, 2008; Yang, et al., 2010). In essence, randomization is an efficient component of global search algorithms.

Obviously, which is the best way to provide sufficient randomness without slowing down the convergence of an algorithm remains an open question,. In fact, the development of meta-heuristic algorithms is a popular research topic, with new algorithms appearing almost yearly, and many new techniques being explored (Yang, et al., 2008; Yang, et al., 2010).

The common practice is to use the *rand* generator with populations of continuous values in the range [0, 1]. This is a limitation from the random search point of view, eventually

limiting the search proximities. Scaling the generator output in the range [-1, 1] gives promising results for both convergence and proximity optimal value search, as shown in Figure 8-9. The search in the test case converges four times faster. Furthermore, there is more spread in the search, which eventually will help with the global optimum search. For these reasons, it is recommended to use the [-1, 1] range, as shown in Figure 8-10.

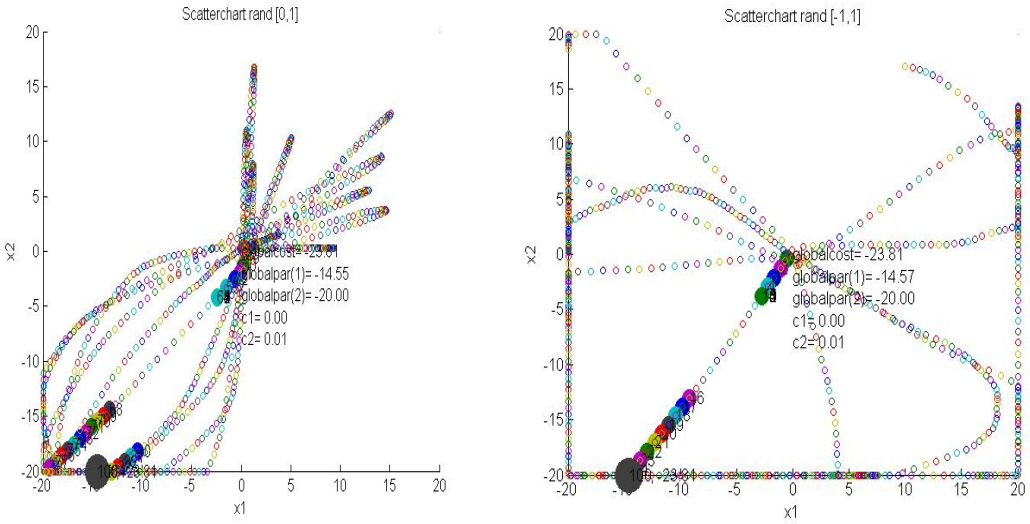


FIGURE 8-9 SEARCH DOMAIN WITH *RAND* [0,1] AND *RAND*[-1,1]

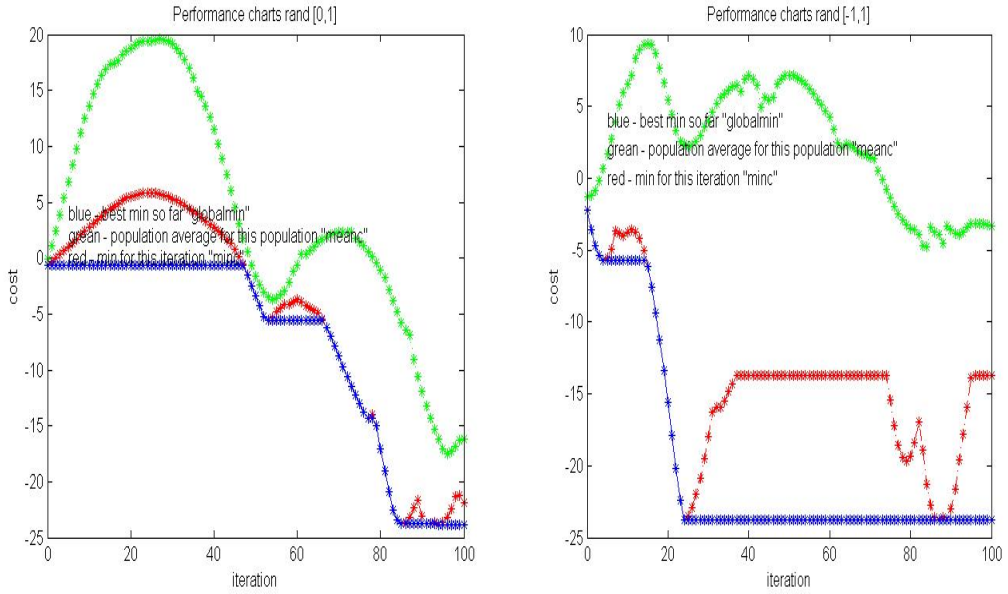


FIGURE 8-10 CONVERGENCE DOMAIN RAND [0, 1] RAND [-1, 1]

8.9. Boundary Conditions

To avoid searching outside of the solution space of interest, various boundary conditions can be used to limit the parameters. Performance varies considerably depending on the location of the global optimum in the solution space as well. Whether or not the boundary particles are relocated inside the allowable solution space, the approaches used may be restricted or unrestricted. Furthermore, there could be different hybrid boundary conditions. The performance of the boundary conditions can be evaluated based on both mathematical benchmark functions and a real-world financial shares time series to evaluate the efficiency and convergence of the algorithm. The general, current understanding is that unrestricted boundary conditions are expected to be more efficient when the global optimum is inside the boundary of the solution space, and the damping boundary condition is more robust and consistent when the global optimum is close to the boundary (Xu, et al., 2007). Thiem et al. (2011) investigated the effect of boundary conditions on algorithm performance. They described three different possible boundary conditions as introduced by Robinson et al. (2004): invisible, absorbing and reflecting boundaries. They compared the performance of different variants for a set of continuous benchmark functions, concentrating on the mean value and as small as possible a best-so-far value. Their conclusions on the influence of heuristic and boundary conditions are for the Levy, Rastrigin and Rosenbrock function. The invisible boundary is used as a base-line for the experiments. The absorbing boundary sporadically and ambiguously improves performance, so decisions about its use should be taken in connection with the particular application. The reflecting boundary is most likely to lead to bad convergence due to re-setting the velocity of the particle. Furthermore, an initial velocity restriction of 10% and maximum velocity of one-eighth of the search space domain range and a minimum velocity of 0.01 give a minor improvement in performance. These results have generally been confirmed in financial time series forecasting.

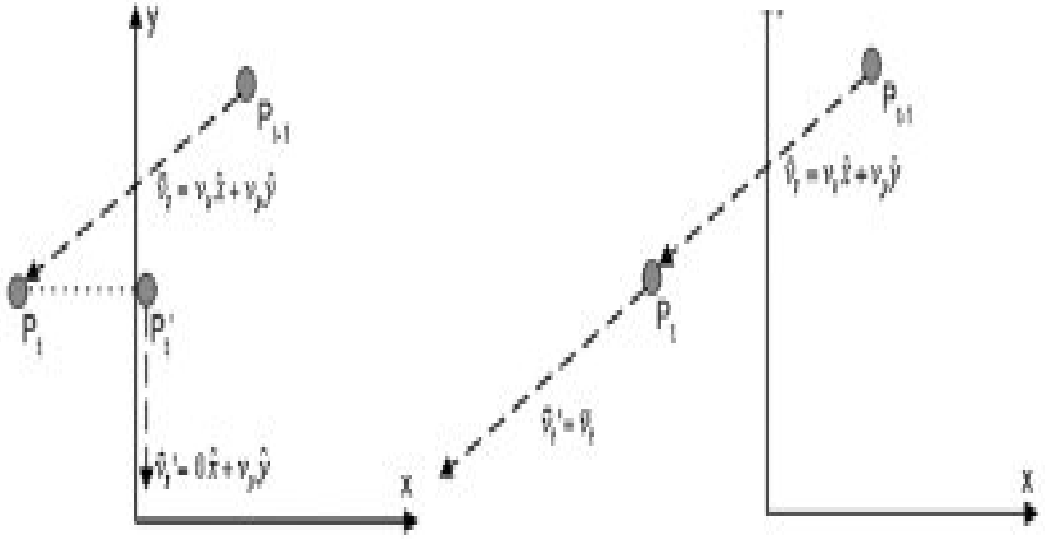


FIGURE 8-11 RESTRICTED AND UNRESTRICTED SEARCH (XU, ET AL., 2007)

The search process is illustrated in Figure 8-11. On the left, the restricted search is shown and on the right unrestricted search. In restricted or absorbing search, when a particle goes outside the allowable solution space in one of the dimensions, it is limited to the boundary of the solution space in that dimension, and the velocity component in that dimension is zeroed. It seems as if the energy of the particle trying to escape the solution space is absorbed by a soft wall so that the particle is stuck to it, and that particle will eventually be pulled back by its memory of best locations only.

Restricted search results are as shown in Figure 8-12:

iter	*global best par*	*global cost*
[30]	[x1=-14.7918, x2=-20.0000]	[-23.7849]

Figure 8-13 shows the performance of the unrestricted search algorithm. A particle is allowed to stay outside the solution space; however, the fitness evaluation of that position is skipped and a bad fitness value is assigned to it. Therefore, the attraction of personal and global best locations will counteract the particle's momentum and eventually drag it back inside the solution space.

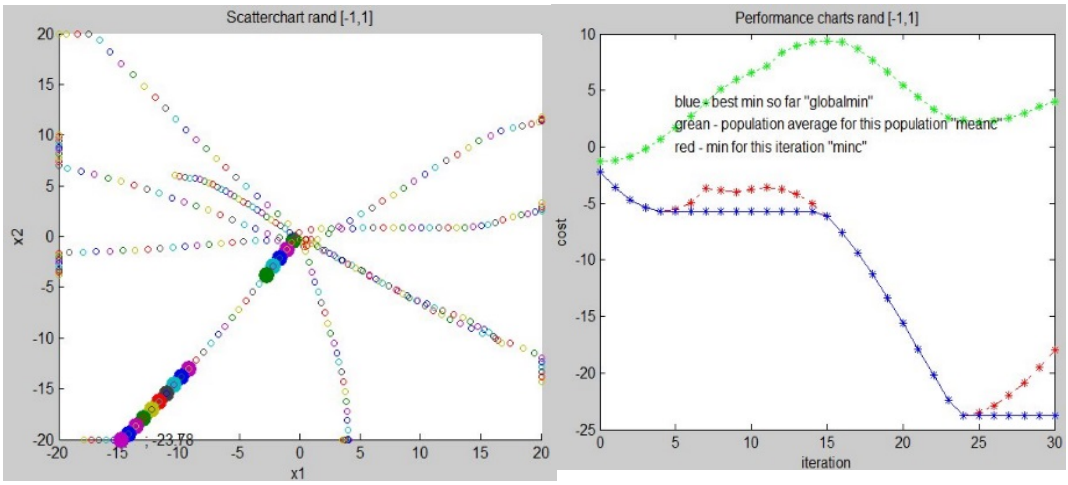


FIGURE 8-12 RESTRICTED SEARCH ALGORITHM PERFORMANCE

Unrestricted search results are as shown in Figure 8-13:

Iter *globalpar* *globalcost*

[30] [*x1*=-31.1643, *x2*=10.4176] [-40.9512]

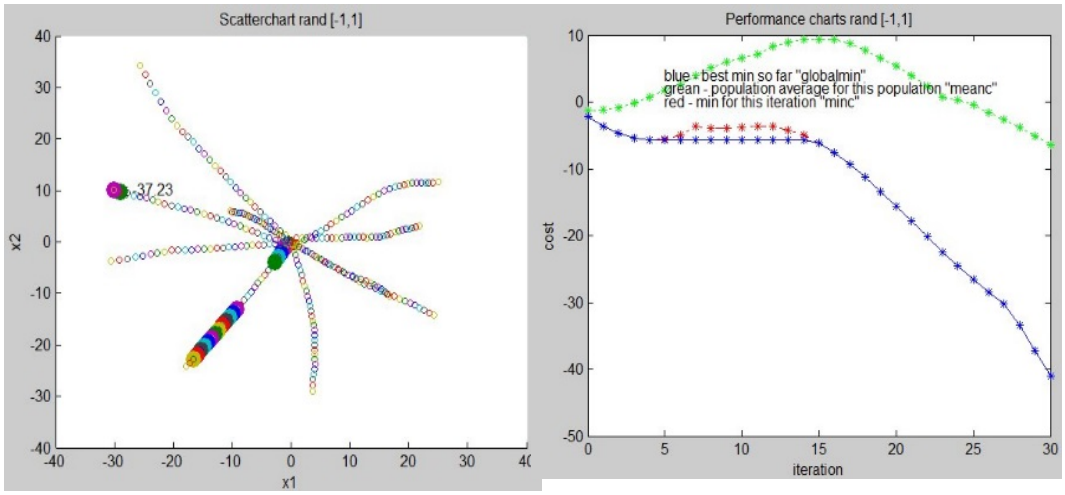


FIGURE 8-13 UNRESTRICTED SEARCH ALGORITHM PERFORMANCE

8.10. PSO with Exponentially Varying Inertia

Among the population-based algorithms, PSO and its variants have been popular heuristic algorithms and have great potential for solving highly complex non-linear optimization problems. Compared to other optimization methods, PSO exhibits fast convergence characteristics. Wang et al. (1992) proposed a decomposition approach to solve non-linear scheduling problems using expert systems with constraints. Jayabarathi et al. (2000) used evolutionary programming, and Manoharan et al. (2005) proposed an evolutionary

programming approach for multiple options. Recently, Lingfeng et al. (2009) proposed an enhanced PSO approach with local search. Manisha et al. (2011) proposed differential evolution enhanced with time-varying mutation with reserve constraints. One common feature of all these methods is that they use probabilistic rules to find the global optimal solution and may prove to be very effective algorithms in solving problems without modifying the shape of their cost curves.

However, the performance of the traditional PSO depends greatly on its parameters and it often suffers from the phenomenon of premature convergence, as well as lacking mechanisms to deal with various constraints. In order to address these drawbacks, an exponentially varying inertia weight factor (EVIWF) algorithm is introduced into the PSO algorithm to improve its convergence and performance (Rani, et al., 2014). This choice is encouraged by the overall good performance of this method as reported by several researchers (Ting, et al., 2012).

8.10.1 Mathematical Formulation

The EVIWF method combines an exponentially varying inertia weight factor *w* with traditional PSO to perform global exploration. The first part of the following equation represents the influence of previous velocity, which provides the necessary momentum for particles to roam across the search space:

$$v_i^{iter} = w v_i^{iter} + c_1 r_1 \left(P_i - x_i^{iter}\right) + c_2 r_2 (G_i - x_i^{iter}) \quad (8\text{-}64)$$

where *v* is the velocity, *x* is parameter,r_1, $r_1$are independent uniform random numbers, c_1 is the cognitive parameter, c_2 is the social parameter, *P* is the local best, *G* is the global best and *w* is inertia.

The inertia weight w is the modulus that controls the impact of the previous velocity on the current velocity. So, the balance between exploration and exploitation in PSO is dictated by the value of *w*. Thus, proper control of the inertia weight is very important in finding the optimal solution accurately and efficiently (Al-Sumait, et al., 2008). A larger inertia

weight causes a tendency towards global exploration, whereas a smaller inertia weight leads toward the fine-tuning of the current search area. Shi et al (1999) significantly improved the performance of PSO with a linearly varying inertia weight over the generations, varying from 0.9 at the beginning of the search to 0.4 at the end. To achieve a trade-off between exploration and exploitation, w varies exponentially in response to the objective values of the particles. In particular, the EVIWF is determined as follows:

$$w^{iter} = w_{max} - (w_{max} - w_{min})(1 - K) \tag{8-65}$$

$$K = e^{-a_0 * \frac{iter}{iter_{max}}} \tag{8-66}$$

where w_{max} and w_{min} are maximum and minimum inertia weight factors respectively, and a_0 is the convergence factor.

8.10.2 Implementation of the Proposed Algorithm

The proposed algorithm is implemented as follows:

Step 1: Specify the lower and upper bounds of the parameter limits.

Step 2: Generate particles randomly between the maximum and minimum.

Step 3: Calculate w^{iter} with the EVIWF formula defined above.

Step 4: Generate particle velocity in the range $[v_i^{min}, v_i^{max}]$.

$$v_i^{max} = \frac{x_i^{min} - x_i^{max}}{R} \tag{8-67}$$

$$v_i^{min} = -v_i^{max} \tag{8-68}$$

where R is percentage change for the population or 'step size'.

Step 5: Update the velocity for each particle

Step 6: After updating the velocity, an individual velocity may violate its velocity maximum or minimum constraints. This violation is corrected as follows:

$$v_i^{iter+1} > v_i^{max}, then \ {}_i^{iter+} = v_i^{max} \tag{8-69}$$

$$v_i^{iter+1} < v_i^{min}, then \ {}_i^{iter+1} = v_i^{min} \tag{8-70}$$

Step 7: Update the position.

Step 8: Check the position and adjust to its maximum and minimum limits.

Step 9: Evaluate the fitness of each individual according to the cost function.

Step 10: If the evaluation of the value of the individual is better than the previous, the current value is to be set to *P* best. If the best is better than the global best, it becomes the value set to be the *G* best.

Step 11: If the criteria to stop are met, then go to step 12, otherwise go to step 3.

Step 12: The individual that generates the latest *G* best is the optimal final solution.

8.10.3 Conclusions

The effectiveness of the proposed algorithm has been compared with the results obtained from conventional PSO and verified with conventional test bench functions, multi-area economic dispatch and an banking sector financial share prices time series regression model. Experiments and simulation results show that the PSO-EVIWF achieves high-quality solutions and smooth convergence characteristics and it can be considered an alternative method for solving optimization and forecasting problems.

8.11. Chaotic Adaptive PSO Using Logistics and Gauss Map

Many modern stochastic search algorithms such as evolutionary programming methods, Particle Swarm Optimisation (PSO), genetic algorithms and firefly algorithms may prove to be very effective optimisation techniques (Happ, 1997). One common feature of all these methods is that they use probabilistic rules to update the particle positions and velocity in the solution space. In the past decade, the cubic cost function has captured the attention of several researchers. Kumaran et al (2001) solved the cost function problem by using a genetic algorithm. Adhinarayanan et al (2006) proposed a non-iterative logic based algorithm, and Al-Sumait et al. (2008) have implemented a pattern search algorithm. Among the above population-based algorithms, PSO is one of the modern heuristic algorithms and has great potential to solve highly complex non-linear optimisation problems. Compared to other optimisation methods, PSO has fast convergence

characteristics. However, the performance of the traditional PSO greatly depends on its parameters and it often suffers from the phenomenon of premature convergence as well as lacking a mechanism to deal with various constraints in various problems. In order to address the above-mentioned drawbacks, the CAPSO algorithm is proposed, which incorporates a chaotic local search (CLS) operator and adaptive inertia weight factor (AIWF) to find an optimal solution and avoid premature convergence (Rani, et al., 2014). The basic strategy is to combine PSO with AIWF and CLS, in which PSO with AIWF is applied to perform global exploration and CLS is used to find the optimal solution (exploitation). Logistics and Gauss mapping techniques are used in performing CLS. This choice is encouraged by the good overall performance of this method reported by several researchers (Chuanwen, et al., 2011). Moreover, the method has not been used before in the context of solving financial share prices forecasting problems.

8.11.1 Mathematical Formulation

For the Adaptive Inertia Weight Factor (AIWF), the first part of the following equation for v_i^{iter+1} represents the influence of previous velocity:

$$v_i^{iter+} = w v_i^{iter} + c_1 r_1 \left(P_{best} - x_i^{iter}\right) + c_2 r_2 (G_{best} - x_i^{iter}) \quad (8\text{-}71)$$

This provides the necessary momentum for particles to roam across the search space. The inertia weight *w* is the modulus that controls the impact of the previous velocity on the current velocity. So, the balance between exploration and exploitation in PSO is dictated by the value of *w*. Thus, proper control of the inertia weight is very important to find the optimum solution accurately and efficiently (Cai, et al., 2007).

It is clear that a larger inertia weight pursues towards global exploration, whereas a smaller inertia weight pursues toward the fine-tuning of the current search area. Niknam (2010) made a significant improvement in the performance of PSO with a linearly varying inertia weight over the generations from 0.9 at the beginning of the search to 0.4 at the end. To achieve a trade-off between exploration and exploitation, w^{iter} is varied adaptively in

response to the objective values of the particles. In particular, the AIWF is determined as follows:

$$w^{iter} = w_{max} - (w_{max} - w_{min})(1 - K) \tag{8-72}$$

$$K = e^{-(G_{best} + \frac{iter}{iter_{max}})} \tag{8-73}$$

where, w_{max} and w_{min} are maximum and minimum inertia weight factors respectively.

In Chaotic Local Search (CLS), in order to enrich the searching behaviour and to avoid being trapped in local optima, chaotic dynamics are incorporated into the PSO with AIWF. Many types of chaotic mapping exist with different dimensions, such as logistics, Gaussian, tent and quadratic mapping. Here, the well-known logistic and Gaussian equations are considered.

The logistic equation in the CAPSO-logistics method exhibits sensitive dependence on initial conditions, and is introduced in the process of chaotic local search as defined by the following equation:

$$cx_i^{iter+1} = \sigma cx_i^{iter}(1 - cx_i^{iter}) \tag{8-74}$$

Obviously, cx_i^{iter} is distributed in the interval [0, 1]. Although this equation is deterministic, it exhibits chaotic dynamics when $\sigma = 4$.

The CAPSO-Gauss method is used to analyse the influence of the chaotic mapping technique on the convergence of the algorithm. To make the comparison meaningful, Gaussian mapping is used as defined by the following equation:

$$cx_i^{iter+1} = e^{-\alpha cx_i^{iter^2}} + \beta \tag{8-75}$$

Here, cx_i denotes the i^{th} chaotic variable, *iter* represents the iteration number, σ is the control parameter and α and β are real parameters. Normally the value of α ranges from 4 to 6.5 and that of β from -1 to 1.

8.11.2 Implementation Procedure for CLS

CLS is implemented as follows:

Step 1: Set *iter* =0 and mapping the decision variables $x_i^{iter}, i = 1,2,3,..n$ among the intervals $\left(x_i^{min}, x_i^{max}\right), i = 1,2,3,..n$ to the chaotic variables cx_i^{iter+1}, located in the interval [0, 1] using the following equation:

$$cx_i^{iter} = \frac{x_i^{iter} - x_i^{min}}{x_i^{max} - x_i^{min}} \tag{8-76}$$

Step 2: Determine the chaotic variable cx_i^{iter+} for the next iteration using the Logistics or Gaussian equation according to cx_i^{iter}.

Step 3: Convert chaotic cx_i^{ite} variables into the decision variable using the following equation:

$$x_i^{iter+1} = x_i^{min} + cx_i^{iter+} \ (x_i^{max} - x_i^{min}) \tag{8-77}$$

Step 4: Evaluate the new solution with decision variables x_i^{iter+} for $i = 1,2,3, \dots, n$.

Step 5: If the new solution is better than x_i^{iter+1} then the predicted maximum iteration is reached, and output the new solution as the result of the CLS; otherwise let $iter = iter + 1$ and go to Step 2. (8-78)

A chaotic local search operator is introduced in the proposed algorithm to avoid premature convergence. The basic strategy of the proposed algorithm is to combine PSO with an adaptive inertia weight factor and chaotic local search. Logistics and Gaussian mapping techniques are used in performing chaotic local search and the results are compared. The applicability and high feasibility of the proposed method is validated on a standard function test bench and in financial share price forecasting. The numerical results show that the proposed method can obtain quality solutions for an optimal cost function and shows excellent convergence characteristics. Hence, the proposed algorithm is competitive with other algorithms in terms of its overall performance.

8.11.3 Conclusions

The applicability and high feasibility of the proposed method is validated with a cubic cost function, in forecasting with a financial time series, and with test bench functions. The numerical simulation results show that the proposed algorithm is capable of giving higher quality solutions and shows excellent convergence characteristics. Hence, the proposed algorithm is competitive with other algorithms in terms of its overall performance.

8.12. Particle Swarm Optimization PID Extension

The traditional PSO algorithm is a closed-loop second order system with a proportional controller term calculating the velocity gain/change of the parameters using the proportional "error" difference between the best local and best global values as shown in Figure 8-14. It limits the search and, in principle, reaches a stable solution which is offset from the eventual optima. A natural extension of PSO consists of implementing a proportional, integral and derivative (PID) controller. Integration of the accumulated previous errors and error derivatives could improve performance, and these can be utilized with the integral and derivative gain parts of the PID controller.

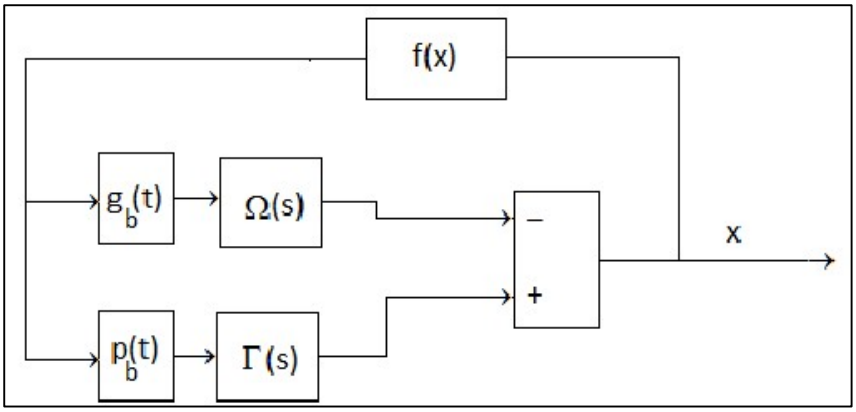


FIGURE 8-14 PSO PID MODEL

It is therefore not surprising that a numbers of research studies have attempted to improve the performance of the algorithm as well as to find possible extensions. Most research related to the improvement of PSO is generally concerned with the choice of parameters, rapidity of convergence and stability of the extended algorithm. Indeed Van den Bergh et

al (2006) described various methods of choosing parameters and presented a convergence stability analysis for each particle. Kadirkamanathan et al. (2006) provided sufficient conditions for asymptotic stability using tools developed from control theory; namely, passivity and a Lyapunov-based approach. However, these conditions are conservative in the sense that violation of the latter does not necessarily imply instability.

Here, a mathematical control theory analysis of the PSO algorithm is used from a system point of view in both a continuous-time and discrete-time setting to present a stability analysis of the PSO algorithm (Busawon, et al., 2016). It is shown that the evolution of a particle is governed by a closed-loop system subjected to a proportional control law together with a feed-forward control term. This observation leads the proposal of a further extension of the PSO algorithm by employing other control laws; namely, a PID (proportional, integral and derivative) controller and methods to choose the parameters of the algorithm. The analysis is carried out in the scalar case only in order to simplify the demonstration. Methods for choosing the parameters of the proposed extended algorithm, such as those presented by the Routh-Hurwitz theorem (Routh, 1877, Hurwitz, 1895), are discussed. Finally, the results obtained are applied and compared to those of traditional PSO via a typical financial share price time series forecast.

During the main loop of the algorithm, the velocities and positions of the particles are iteratively updated until a stopping criterion is met. The updating rules are given as follows:

$$x_i(n+1) = x_i(n) + v_i(n+1) \tag{8-79}$$

$$v_i(n+1) = wv_i(n) + c_1r_1(p_i(n) - x_i(n)) + c_2r_2(g_i(n) - x_i(n)) \tag{8-80}$$

where w is a parameter called the inertia weight, c_1 and c_2 are two parameters called acceleration coefficients, and r_1 and r_2 are two N-dimensional diagonal matrices in which the entries in the main diagonal are random numbers uniformly distributed in the interval [0, 1]. At each iteration, these matrices are regenerated. The vector $p_i(n)$ is referred to as

the local best. The vector $g_i(n)$ is referred to as the neighbourhood best or global best, and this is the best position ever found by any particle in the neighbourhood of particle x_i. If the values of w, c_1 and c_2 are properly chosen, it is guaranteed that the particles' velocities will not increase to infinity. It should be noted that, to date, no clear procedure or method has been proposed to choose such values appropriately.

The system can be rewritten as:

$$x_i(n+1) = x_i(n) + v_i(n+1) \tag{8-81}$$

$$v_i(n+1) = wv_i(n) + u_i(n+1) \tag{8-82}$$

with

$$u_i(n) = c_1 r_1(p_i(n) - x_i(n)) + c_2 r_2(g_i(n) - x_i(n)) \tag{8-83}$$

One can easily show that:

$$u_i = k_0(g_i(n) - p_i(n)) + k_P(p_i(n) - x_i(n)) \tag{8-84}$$

where

$$k_P = c_1 r_1 + c_2 r_2, \text{ and } k_0 = c_2 r_2 \tag{8-85}$$

The function $u_i(n)$ can be seen as a control input to the system. Therefore, from a control point of view, the evolution of a particle, x_i, is governed by the above closed loop system where $u_i(n)$ is viewed as the feedback control. The control $u_i(n)$ is nothing more than a proportional control component together with a feed-forward term, $k_0(g_i(n) - p_i(n))$ with reference input $p_i(n)$. Consequently, it is possible to extend the above PSO algorithm by employing other control terms in the algorithm; in effect, extending the traditional PSO algorithm by replacing the proportional term $k_P(p_i(n) - x_i(n)$ in $u_i(n)$ by a PID controller. This allows approaches to be proposed to choose the parameters of the algorithm.

The PID controller is used in a closed-loop unity feedback system. The error denotes the difference between the current particle parameter difference and the local best error, which is sent to the PSO algorithm; that is, the PID controller. The velocity of the control signal

from the controller to the regression system is equal to the proportional gain (K_P) times the magnitude of the error plus the integral gain (K_I) times the integral of the error plus the derivative gain (K_D) times the derivative of the error:

$$u = K_P * e + K_I * \int e * dt + K_D * \frac{de}{dt} \tag{8-86}$$

8.12.1 Mathematical Analysis

A mathematical analysis of the PSO algorithm is conducted from a system point of view in both a continuous-time and discrete-time setting in order to provide a stability analysis of the PSO and to justify the choice of the parameters currently employed for the algorithm. The PID Laplace transfer function is:

$$PID(s) = K_P + \frac{K_I}{s} + K_D = \frac{K_D * s^2 + K_p * s + K_I}{s} \tag{8-87}$$

The algorithm could be extended, including in $u(t)$, with the integral and derivative terms:

$$u(t) = k_0(p_t - g_t) + k_P(p_t - x_t) + k_I \int_0^t (p_t - x_t)dt + k_D \frac{d}{dt}(p_t - x_t) \tag{8-88}$$

$$U(s) = k_0\big(P(s) - G(s)\big) + \Big(k_p + k_I \frac{1}{s} + k_D s\Big)\big(P(s) - X(s)\big) \tag{8-89}$$

$$X(s) = H(s)U(s) \tag{8-90}$$

$$H(s) = \frac{1}{s(s-w)} \tag{8-91}$$

$$X(s) = \frac{1}{s(s-w)} * (k_0\big(G(s) - P(s)\big) + \Big(k_p + k_I \frac{1}{s} + k_D s\Big)\big(P(s) - X(s)\big) \tag{8-92}$$

$$X(s) = \frac{k_D s^2 + (k_p - k_0)s + k_I}{(s^3 + (k_D - w)s^2 + k_p s + k_I)} P(s) + \frac{s k_0}{(s^3 + (k_D - w)s^2 + k_p s + k_I)} G(s) \tag{8-93}$$

$$X(s) = \Gamma(s)P(s) + \Omega(s)G(s) \tag{8-94}$$

$$\Gamma(s) = \frac{k_D s^2 + (k_p - k_0)s + k_I}{(s^3 + (k_D - w)s^2 + k_p s + k_I)} \tag{8-95}$$

$$\Omega(s) = \frac{s k_0}{(s^3 + (k_D - w)s^2 + k_p s + k_I)} \tag{8-96}$$

In the case of the P controller already derived:

$$X(s) = \frac{k_P - k_0}{s^2 - ws + k_P} P(s) \tag{8-97}$$

In the case of the PI controller:

$$X(s) = \frac{(k_p - k_0)s + k_I}{(s^3 - ws^2 + k_p s + k_I)} P(s) \quad (8\text{-}98)$$

According to the Routh-Hurwitz criterion, the above transfer function is stable if $k_I > 0$, $k_D > w$ and $k_P > \frac{k_I}{k_D - w}$.

Assuming that $r_i = 1$ then:

$$k_0 = c_0 r_0 = c_0 \quad (8\text{-}99)$$

$$k_1 = [k_P + k_I + k_D] = c_1 r_1 + c_2 r_2 + c_3 r_3 = c_1 + c_2 + c_3 \quad (8\text{-}100)$$

$$k_2 = [k_P + 2k_D] = c_1 r_1 + c_3 r_3 = c_1 + c_3 \quad (8\text{-}101)$$

$$k_3 = k_D = c_3 r_3 = c_3 \quad (8\text{-}102)$$

$$c_1 > \frac{c_2}{c_3 - w}, c_2 > 0 \; and c_3 > w \quad (8\text{-}103)$$

A more precise method of choosing the parameters would be to use the Kharitonov theorem (Kharitonov, 1996) on an interval polynomial. Consider the characteristic polynomial:

$$p(s) = s^3 + (k_D - w)s^2 + k_P s + k_I \quad (8\text{-}104)$$

$$p(s) = a_3 s^3 + a_2 s^2 + a_1 s + a_0 \quad (8\text{-}105)$$

The stability of $p(s)$ is defined as follows:

$$p_1(s) = (c_3 - w)s^2 + s^3 \quad (8\text{-}106)$$

$$p_2(s) = c_2 + c_1 s + s^2 \quad (8\text{-}107)$$

$$p_3(s) = c_1 s + (c_3 - w)s^2 + s^3 \quad (8\text{-}108)$$

$$p_4(s) = c_2 + s^3 \quad (8\text{-}109)$$

Therefore$c_2 > 0, c_3 > w and c_1 > \frac{c_2}{\varepsilon} > 0$, and this agrees with the results of Routh-Hurwitz.

Another direct method to find the parameters c_i is to fix the poles of the characteristic polynomial and then to find the values of c_i by identification.

Simulations confirming the analytical findings are shown below.

For $w < 0$ with the PI controllers, the simulation solution is stable. For $w = -0.5$ the result is shown in Figure 8-15.

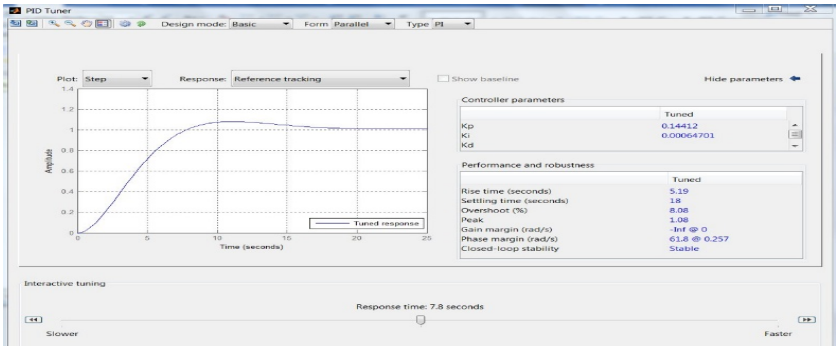


FIGURE 8-15 SIMULATION OF PI CONTROLLER AND $W = -0.5$

For $w = 1$ with the PI controller, the simulation failed to find a stable solution and is shown in Figure 8-16.

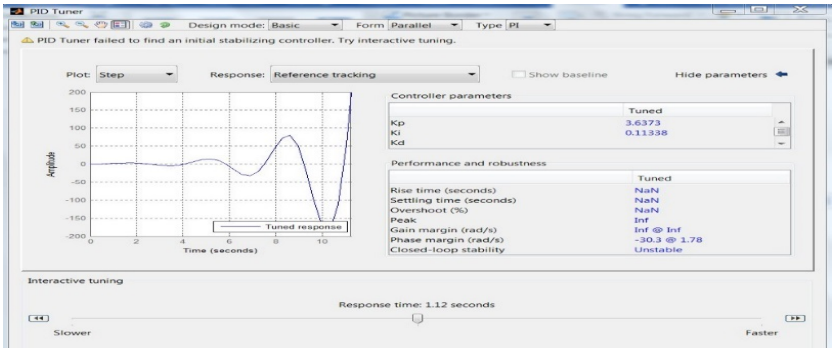


FIGURE 8-16 SIMULATION OF PI CONTROLLER AND W=1

With the addition of derivative components in the closed-loop second system controller (PID) and for $w = 1$, the simulation found a stable system solution as shown in Figure 8-17 which confirms the Routh-Hurwitz criterion and the Kharitonov theorem stable system solution conditions.

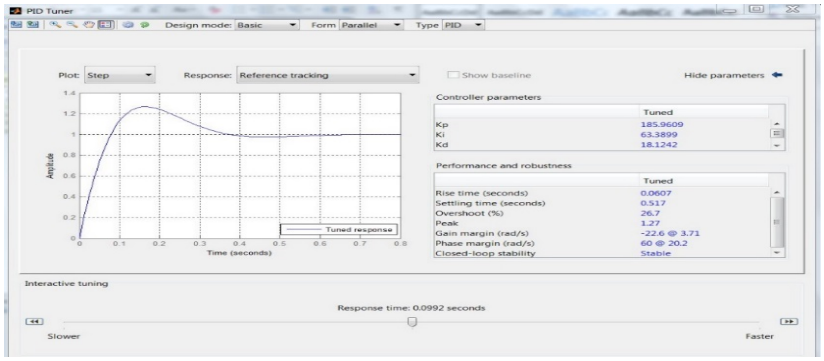


FIGURE 8-17 SIMULATION PID CONTROLLER AND $W = 1$

With the PID controller and for $w = 0$, the simulation found a stable system solution as shown in Figure 8-18.

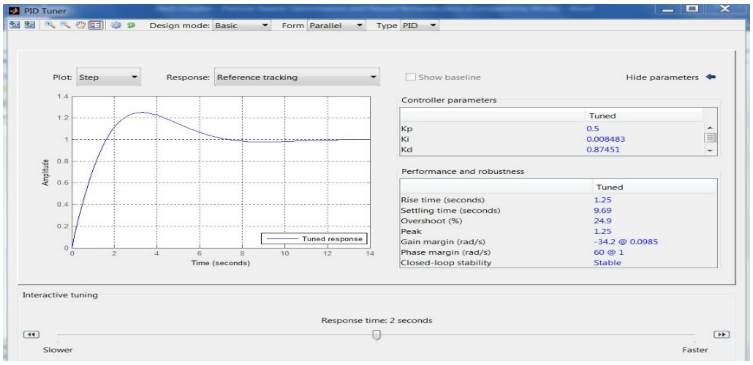


FIGURE 8-18 SIMULATION AND $W = 0.0$

With the PID controller and for $w = -0.5$, the simulation found a stable system solution as shown in Figure 8-19.

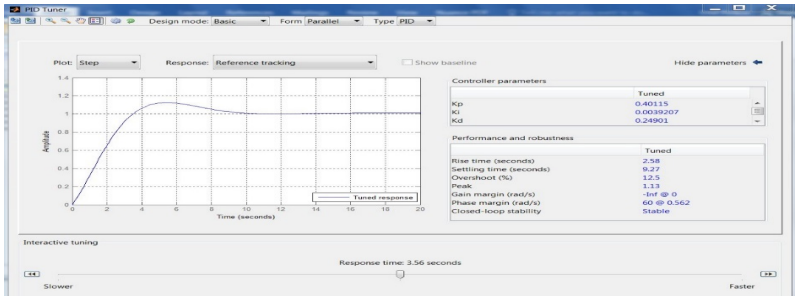


FIGURE 8-19 SIMULATION PID CONTROLLER AND $W = -0.5$

To have a stable system for the P and PI controllers, the inertia w should be < 0.0. Generally, finding a stable system solution and excellent system response characteristics for the common case of $w = 1.0$ is straightforward, and an example of the performance and robustness of PID for PSO is shown in Figure 8-20.

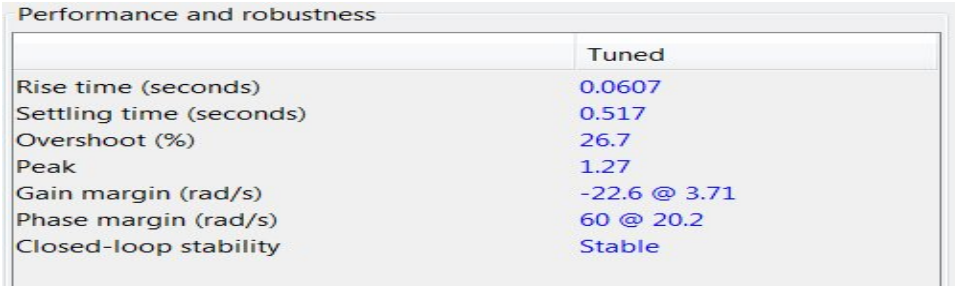


FIGURE 8-20 PERFORMANCE AND ROBUSTNESS OF PID FOR PSO AND $W < 0.0$

8.12.2 Tuning of the PID Controller

Dynamic system behaviour and convergence is examined next and is defined with the system step input signal response. An initial set of the parameters of the PID controller used for the PSO algorithm is derived using the Ziegler-Nichols method (Zhong, 2006).

The PID controller can be tuned without an extensive background in control theory. This is not the case with many other modern controllers that are much more complex but often provide only marginal improvements. In fact, most PID controllers are tuned per individual case. However, the lengthy calculations for an initial guess at PID parameters can often be bypassed by using a few tuning rules. This is especially useful when the system is unknown. The four most important major characteristics of the closed-loop step response are as follows:

1. Rise time: the time it takes for the system output y to rise beyond 90% of the desired level for the first time.
2. Overshoot: how much the peak level is higher than the steady state, normalized against the steady state.
3. Settling time: the time it takes for the system to converge to its steady state.
4. Steady-state error (SSE): the difference between the steady-state output and the desired output.

The effects of increasing each of the PID controller parameters K_P, $K_I$and K_D are summarized below in Table 8-1.

TABLE 8-1 PID CONTROLLER PARAMETERS TUNING GUIDANCE

Response	Rise time	Overshoot	Settling time	S-S error
K_P	Decrease	Increase	-	Decrease
K_I	Decrease	Increase	Increase	Eliminate
K_D	-	Decrease	Decrease	-

Typical steps in designing a PID controller are as follows:

1. Determine what characteristics of the system need to be improved.
2. Tune K_P to reduce the rise time.
3. Tune K_I to reduce the overshoot and settling time.
4. Tune K_D to eliminate steady-state error.

The initial parameters can be found by using the rules proposed by Ziegler and Nichols for determining the values of K_P,K_I, and K_D, based on the transient step response of the system. The Ziegler-Nichols method applies to systems whose unit-step responses resemble an S-shaped curve with no overshoot as shown in Figure 8-21.

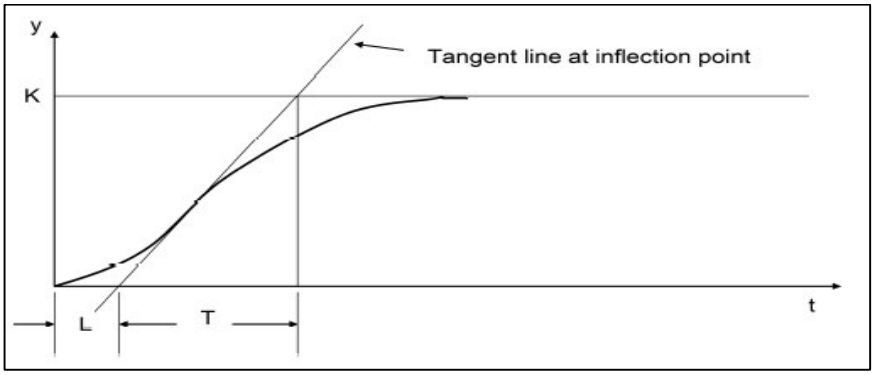


FIGURE 8-21 ZIEGLER-NICHOLS PID TUNING RULES (ZHONG, 2006)

The S-shaped reaction curve can be characterized by two constants, the delay time *L* and time constant *T*, which are determined by drawing a tangent line at the inflection point of the curve and finding the intersections of the tangent line with the time axis and the steady-state level line. A response with an overshoot of about 25% and a good settling time can be used for the initial controller settings K_P, K_I and K_D as presented below in Table 8-2.

TABLE 8-2 INITIAL PID CONTROLLER SETTINGS (ZHONG, 2006)

Controller	K_P	K_I	K_D
P	$\frac{T}{L}$	0	0
PI	$0.9 * \frac{T}{L}$	$0.27 * \frac{T}{L^2}$	0
PID	$1.2 * \frac{T}{L}$	$0.6 * \frac{T}{L^2}$	$0.6 * T$

8.12.3 Discrete Domain PID

The PSO algorithm is generally given in the discrete domain. Consequently, the analysis in the discrete domain is more relevant in terms of practical implementation. For the $k+1$ iteration:

$$x_{k+1} = x_k + v_{k+1} \tag{8-110}$$

$$v_{k+1} = wv_k + u_k \tag{8-111}$$

$$u_k = k_p(p_k - x_k) + k_o(g_k - p_k) \tag{8-112}$$

The z-transform is given by:

$$zX(z) = X(z) + zV(z) \tag{8-113}$$

$$zV(z) = wV(z) + U(z) \tag{8-114}$$

$$X(z) = H(z)U(z) = \frac{z}{(z-w)(z-1)}U(z) \tag{8-115}$$

The z-transform of the open system is:

$$H(z) = \frac{z}{(z-w)(z-1)} \tag{8-116}$$

$$U(z) = k_P\big(P(z) - X(z)\big) + k_0\big(G(z) - P(z)\big) \tag{8-117}$$

$$X(z) = \frac{(k_P+k_0)H(z)}{1+k_PH(z)}P(z) + \frac{k_0H(z)}{1+k_PH(z)}G(z) \tag{8-118}$$

$$X(z) = \Gamma(z)P(z) + \Omega(z)G(z) \tag{8-119}$$

$$\Gamma(z) = \frac{(k_P+k_0)H(z)}{1+k_PH(z)} = \frac{(k_P-k_0)z}{z^2+(k_P-w-1)z+w} \tag{8-120}$$

$$\Omega(z) = \frac{k_0H(z)}{1+k_PH(z)} = \frac{k_0z}{z^2+(k_P-w-1)z+w} \tag{8-121}$$

The poles of the closed loop system are:

$$x_{1,2} = \frac{-b^{+}/_{-}\sqrt{b^2-4ac}}{2a} \tag{8-122}$$

$$z_{1,2} = \frac{-(k_P-w-1)^{+}/_{-}\sqrt{(k_P-w-1)^2-4w}}{2} \tag{8-123}$$

If w=1 and poles are to be in the unit circle for stability then:

$$z_1 = \frac{-(k_P-2)+\sqrt{(k_P-2)^2-4}}{2}, -1 < z_1 < 1 \quad (8\text{-}124)$$

$$z_2 = \frac{-(k_P-2)-\sqrt{(k_P-2)^2-4}}{2}, -1 < z_2 < 1 \quad (8\text{-}125)$$

The PSO algorithm could be extended by including the integral and derivative terms in $u(t)$, the discrete time domain (Basdogan, 2016), and the discrete time derivative is given by:

$$y_t = f_t - f_{t-1} \quad (8\text{-}126)$$

The discrete time integral (simple rectangular method) is given by:

$$\int_{t-1}^{t} f(t)dt = y_t \approx y_{t-1} + f_t \quad (8\text{-}127)$$

The discrete transfer function z-transforms are derived as follows:

- discrete TF of derivative:

$$y_t = f_t - f_{t-1} \quad (8\text{-}128)$$

$$Y[z] = F[z] - z^{-1} * F[z] \quad (8\text{-}129)$$

$$\frac{Y[z]}{F[z]} = \frac{z-1}{z} \quad (8\text{-}130)$$

- discrete TF of integral (simple rectangular method):

$$y_t = y_{t-1} + f_t \quad (8\text{-}131)$$

$$Y[z](1 - z^{-1}) = F[z] \quad (8\text{-}132)$$

$$\frac{Y[z]}{F[z]} = \frac{z}{z-1} \quad (8\text{-}133)$$

The discrete transfer function of the PID is then given by:

$$\frac{V[z]}{E[z]} = K_P + K_I \frac{z}{z-1} + K_D \frac{z-1}{z} \quad (8\text{-}134)$$

$$\frac{V[z]}{E[z]} = \frac{K_P(z^2 - z) + K_I z^2 + K_D(z^2 - 2z + 1)}{z^2 - z}$$

$$\frac{V[z]}{E[z]} = \frac{(K_P + K_I + K_D)z^2 - (K_P + 2K_D)z + K_D}{z^2 - z}$$

$$\frac{V[z]}{E[z]} = \frac{(K_P + K_I + K_D) - (K_P + 2K_D)z^{-1} + K_D z^{-2}}{1 - z^{-1}}$$

$$V[z] = z^{-1}V[z] + aE[z] + bz^{-1}E[z] + cz^{-2}E[z] \quad (8\text{-}135)$$

Finally:

$$v_t = v_{t-1} + a * e_t + b * e_{t-1} + c * e_{t-2} \quad (8\text{-}136)$$

$$a = (K_p + K_I + K_D) \quad (8\text{-}137)$$

$$b = -(K_P + 2 * K_D) \quad (8\text{-}138)$$

$$c = K_D \quad (8\text{-}139)$$

In the traditional PSO algorithm, $k_I = 0$ and $k_D = 0$, thus:

$$k_1 = [k_P + k_I + k_D] = c_1 r_1 + c_2 r_2 + c_3 r_3 = c_1 r_1 \quad (8\text{-}140)$$

$$k_2 = [k_P + 2k_D] = c_1 r_1 + c_3 r_3 = c_1 r_1 \quad (8\text{-}141)$$

$$k_3 = k_D = c_3 r_3 = 0 \quad (8\text{-}142)$$

$$p(z) = z^3 + (c_1 r_1 - w - 2)z^2 + (2w + c_1 r_1 + 1)z - w \quad (8\text{-}143)$$

In many studies (Shi, et al., 1998, Trelea, 2003, Engelbrecht, 2007) the values of c_1= 1.496 and w = 0.729 can be chosen. For these values of c_1 and w there are two poles inside the unit circle of the complex plane and one on the unit circle for the worst case scenario r_1= r_2= 1. This justifies such a choice of parameters.

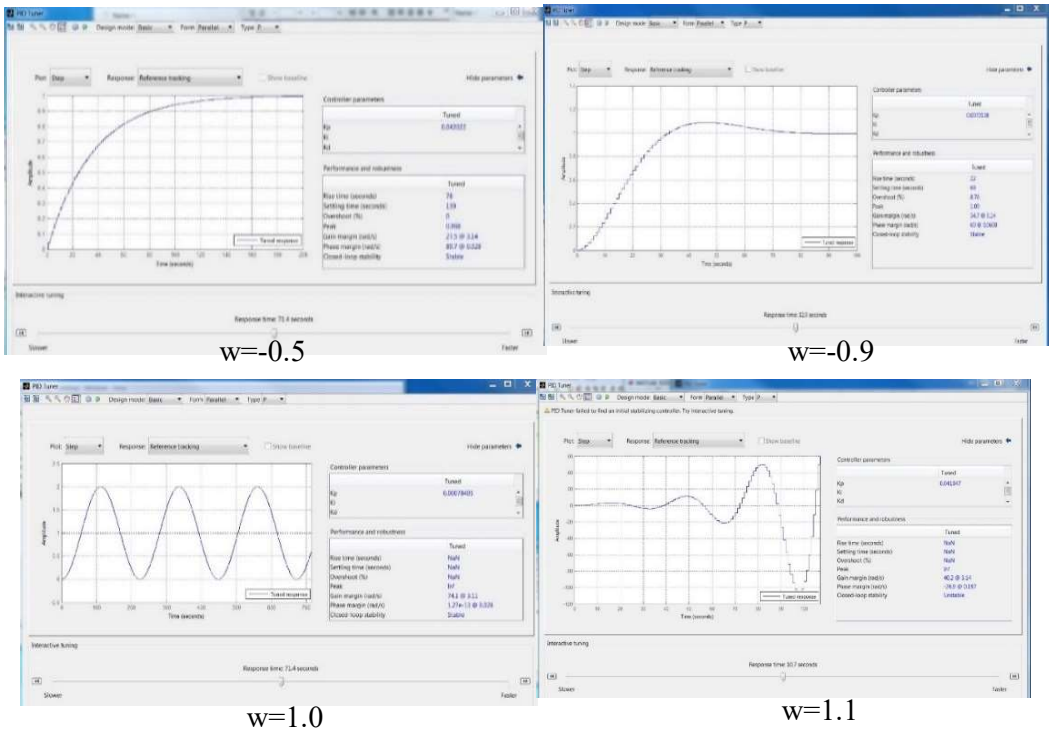
8.12.4 Simulations

Matlab simulation code:

```
% clear all; clc; close all;

% G = z/((z-w)(z-1))= z /((z^2-(w+1)z+w))

w=1;

G=tf([1 0], [1 -(w+1) w],1);

%%

w=1;
```

```
Kp=0.048752;

Ki=0.00023405;

Kd=0.24724;

C=pid(Kp,Ki,Kd,0,1);

H=1;

T=feedback(GC,H);

step(T)
```

P controller simulation results are shown in Figure 8-22.



w=-0.5 w=-0.9

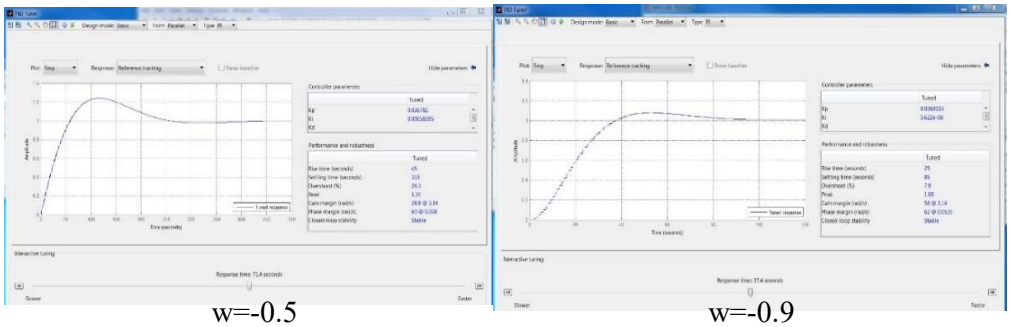
w=1.0 w=1.1

FIGURE 8-22 DISCRETE PSO WITH P CONTROLLER AND w = [-0.5, -0.9, 1.0, 1.1]

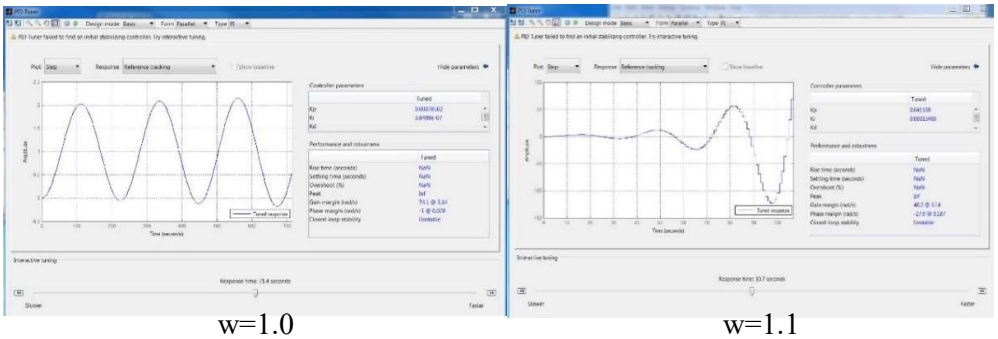
PID tool *pidtool* code:

```
% G = z/((z-w)(z-1))= z /((z^2-(w+1)z+w))

w=1.0;

G=tf([1 0], [1 -(w+1) w],1);

pidtool(G,'pid')
```

PI controller simulation results are shown in Figure 8-23.



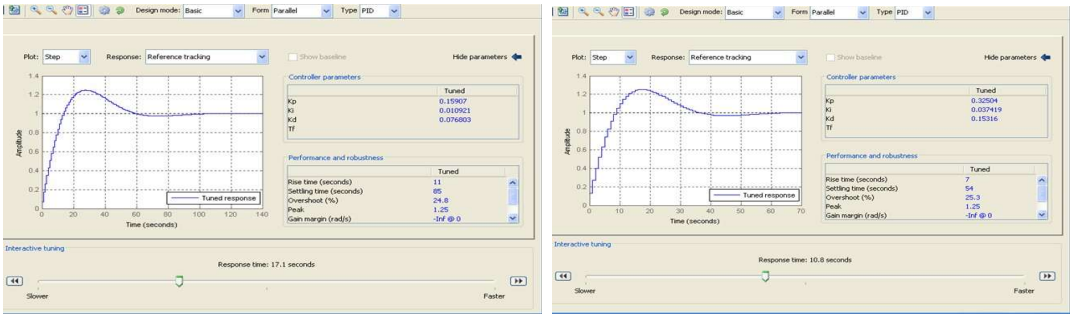
w=-0.5 w=-0.9



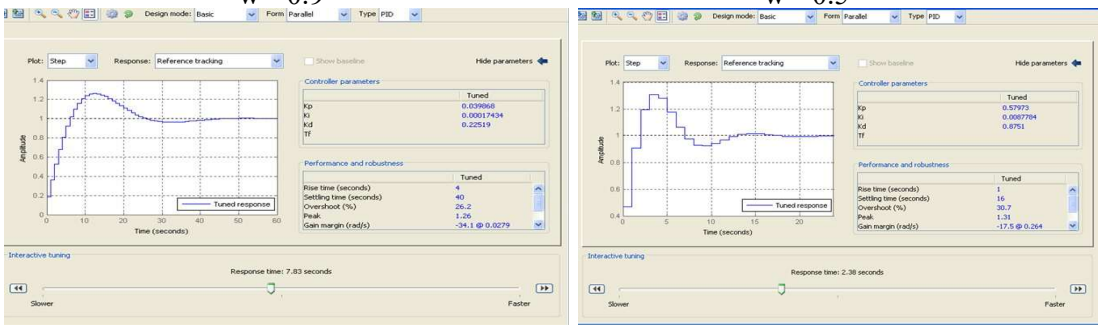
w=1.0 w=1.1

FIGURE 8-23 DISCRETE PI CONTROLLER AND *W* = [-0.5, -0.9, 1.0, 1.1]

PID controller simulation results are shown in Figure 8-24.



w=-0.9 w=-0.5



w=1.0 w=1.1

FIGURE 8-24 DISCRETE PID CONTROLLER AND W = [-0.9, -0.5,1.0, 1.1]

The traditional PSO algorithm with the common value of w=1 is stable harmonically. Values of w<1 are stable but slow; for example, for w=-0.9 the response time is 32.9 sec. The PI algorithm generally has similar performance to that of the P algorithm. The PID algorithm is stable with various values of *w* with very good response times; for example,

for w=1 the response time is 7.83 sec with very good performance and robustness, as shown in Figure 8-25 and Table 8-3.

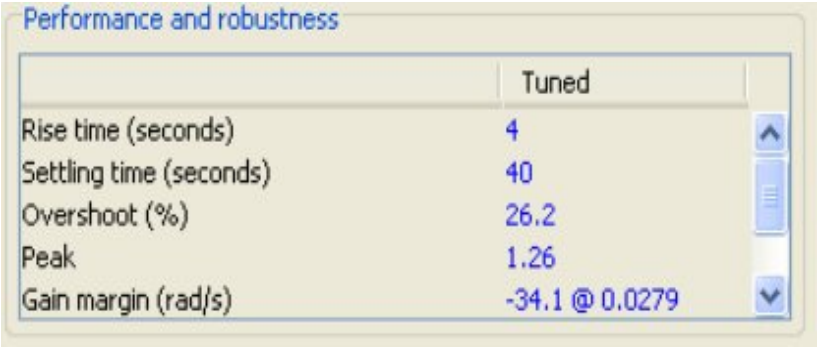


FIGURE 8-25 PID EXTENSION ALGORITHM PERFORMANCE AND ROBUSTNESS

TABLE 8-3 PID PERFORMANCE AND ROBUSTNESS

w	Response [s]	Rise[s]	Stable [s]	Overshoot[%]
-0.9	17.1	11.0	85.0	24.8
-0.5	10.08	7.0	54.0	25.3
1.0	7.83	4.0	40.0	26.2
1.1	2.38	1	16.0	30.7

8.12.5 Application to Financial Time Series

Tests have been conducted with banking sector financial time series data to validate the feasibility and effectiveness of the proposed algorithm. The results represent the average of 30 trials. The various parameters of the proposed method are set as follows: $c_0 = c_1 = 2$; $c_2 = -2c_3 = 1, w = 0.4$. For this choice of parameters, one can check that the roots of the characteristic equation $p(z)$ all lie within the unit circle of the complex plane. The number of iterations used was 300. Clearly, the proposed method converged to a better solution, while the execution time was much lower.

8.12.6 Conclusions

The proposed extension of the PSO algorithm from a control system point of view has shown that, in the scalar case, each sub-system of the traditional PSO can be viewed as a closed-loop second order system controlled using a proportional controller. Consequently, an extension of the PSO was made by including a PID controller in the sub-system.

Various methods to choose the underlying tuning parameters were discussed. It was shown that the performance of the proposed method is better than that of traditional PSO using an PSO-PID with financial share price time series data. The performance of the proposed extended algorithms has been tested for other benchmark functions.

8.13. Hybrid Model with PSO and Neural Networks

Research on the integration of Artificial Neural Networks (ANNs) and Swarm Intelligence (SI) is promoting unified development in computational models for machine learning. Recently swarm intelligence methods like PSO have been successfully applied for training feed-forward and recurrent ANNs (Wang, et al., 2008, Kiranyaz, et al., 2009). Dehuri et al , 2011) have reviewed recent theoretical and applied research on swarm intelligence (SI) and artificial neural networks. The authors discuss recent developments in swarm intelligence and neural networks, focusing on particle swarm optimization (PSO), ant colony optimization (ACO) and bee colony optimization (BCO) techniques. Artificial neural networks (ANNs) have the ability to learn and adapt although it is very difficult to obtain an optimal neural network architecture with, for example, the optimal numbers of hidden layers and hidden neurons in each hidden layer. They argue that the use of swarm intelligence will allow more robust and rapid solutions to be found. Furthermore, the potential of higher order neural networks and their possible integration with swarm intelligence is also discussed. Omkar and Senthilnath (2011) explore data mining utilizing neural network and swarm intelligence algorithms. These techniques have an advantage over conventional statistical techniques because they require no prior knowledge of the distribution of data. The extraction of information from a dataset is possible in the form of weights and rules using various neural network and swarm intelligence techniques. Performance analysis is based on per class and overall accuracy. Furthermore, the computational complexity of neural network and swarm intelligence techniques are evaluated. Swarm intelligence meta-heuristic methods are mainly characterized by their distributiveness, flexibility, capacity of interaction among simple agents, and robustness.

They have been successfully applied to single-objective optimization problems and have great potential to cope with multi-objective optimization problems, as addressed by Leguizamon and Coello (2011). Here, a taxonomy of the types of swarm intelligence (i.e. ant colony optimization) for multi-objective optimization is proposed that would be helpful in multi-objective optimization problems. The wavelet transform (WT) is one of the most frequently used signal processing techniques for the extraction of information in different frequency sub-bands from non-stationary signals such as time series. WT-derived features are used to identify the nature of the signals. Panigrahi et al. (2011) used the multi-layer feed -forward neural network (FNN) with both back-propagation and evolved with an integrated Adaptive Particle Swarm Optimization (APSO) neural network classifier and the assessment of classification performance compares both back-propagation and APSO-based learning. Misra et al. (2011) presented a classifier model using a Polynomial Neural Network (PNN), which is a flexible neural network architecture. The number of layers in the PNN is not fixed in advance but is developed on the fly. Each node of the PNN realizes a polynomial mapping between input and output variables. An artificial neural network (ANN) is incorporated with PNN in order to enhance performance. A comparison of the performance of gradient descent and PSO training frameworks was conducted on benchmark databases in different domains. The simulation results of the hybrid ANN-PNN with PSO showed that its performance is much better in comparison to the conventional and gradient descent PNN models. Majhi et al. (2011) developed an efficient adaptive prediction model using the recently developed Differential Evolution (DE) technique. The forecasting model employs adaptive linear combiner architecture, neural network alike, and a DE-based learning rule to predict seasonally adjusted (SA) and non-seasonally adjusted (NSA) sales data for short and long ranges. The prediction performance of the proposed model was assessed through simulation and using real life data. For comparison purposes, the corresponding results were also obtained based on a genetic algorithm (GA), bacterial foraging optimization (BFO) and PSO-based forecasting models. It was observed

that the new DE forecasting model offered the fastest training, best prediction and the lowest least mean squares error after training compared to the three other evolutionary computing-based models.

8.13.1 Evolutionary Neural Networks Algorithms

Evolutionary algorithms (EAs) (Back, et al., 1993) such as PSO have been used in studies of optimization problem-solving, such as the optimal design of artificial neural networks (ANNs). As they are heuristic and stochastic based on populations made up of individuals with specified behaviour, they are robust and efficient at exploring an entire optimization space. There have been successful efforts to evolve the weights, structures, and learning parameters of ANNs. Yao and Liu (1997) proposed an evolutionary ANN approach called EPNet by using an evolutionary programming (EP) algorithm. Weights and structure are evolved simultaneously by using partial training, the mutation of weights and the addition or removal of connections or nodes. EPNet encourages smaller networks, as removals are attempted before additions, and a behavioural link is maintained between parents and offspring through partial training and node splitting. Castillo et al. (2000) proposed a method that attempts to search for the initial weights and hidden-layer size of multilayer perceptrons (MLPs). The application of the G-Prop algorithm to several real-world and benchmark problems showed that the MLPs which evolved are smaller and achieve better generalization than other perceptron training algorithms. Palmes et al. (2005) proposed a mutation-based genetic ANN (MGNN) algorithm. The MGNN can evolve the structure and weights of ANNs simultaneously. It implements a stopping criterion where occurrences of over-fitness are monitored through sliding-windows to avoid premature learning and over-learning. PSO is considered to be capable of reducing slow convergence speeds in training, but it is easy to get stuck in a local minimum with the BP algorithm of feed-forward ANNs because it does not require a gradient and differentiable information. Salerno (1997) used PSO to evolve the weights and bias of neurons in ANNs used to solve XOR problem and for parsing natural language. The results demonstrated that a PSO-based

ANN has better training performance and a faster convergence rate, as well as better predictive ability than a BP-based ANN. Juang (2001) proposed a hybrid of GA and PSO (HGAPSO) for training recurrent networks. The HGAPSO used PSO to enhance the elites generated by GA in order to produce higher quality individuals. The performance of HGAPSO was compared to both GA and PSO in recurrent network design problems and its superiority was demonstrated. Da and Ge (2005) proposed an improved PSO-based ANN with simulated annealing (SA) technique, and the results showed better training and generalization performance than a PSO-based ANN. A hybrid of ACO and BP algorithms was proposed by Shi and Li (2009) to evolve NNs. The ACO-BP algorithm firstly uses the ACO algorithm to search for the near-optimal solution and then adopts the BP algorithm to find the accurate solution. It avoids being trapped in local optima and can rapidly find the accurate solution to accelerate its speed of evolution. Through the selection of neural network parameters, the solution expressed is as an adjustment of the updated neural network parameters. When all the particles choose the same parameters or algorithm to a predetermined number of cycles, then the algorithm is terminated. This approach has been utilized with PSO and extended further with an additional post-PSO application in the investigation in the present study. Karaboga et al. (2009) has introduced another technique for training ANNs, proposing an Artificial Bee Colony (ABC) algorithm, which has good exploration and exploitation capabilities in searching for the optimal weight set for training neural networks.

8.13.2 Numerical Experiments

Approaches using PSO to replace the back-propagation learning algorithm in ANNs have been proposed in recent years. It has been shown that PSO is a promising method for use in training ANNs. It is faster and gets better results in most cases. Furthermore, hybrid models with PSO and BPNN combinations are very promising. The selection of a fitness function for a regression problem such as time series financial share prices forecasting is the minimum square root of the model fitness error.

A hybrid example combining a Back Propagation Neural Network (BPNN) with PSO is investigated here (see Figure 8-26). The problem to be solved is the forecasting of a financial share price time series regression fitting function in a banking sector dataset. Sixteen time series datasets of fifteen consecutive closing price values (over three trading weeks) are the attributes (features) with a forecast target being the sixteenth day closing price. A three-layer neural network is used to conduct the regression.

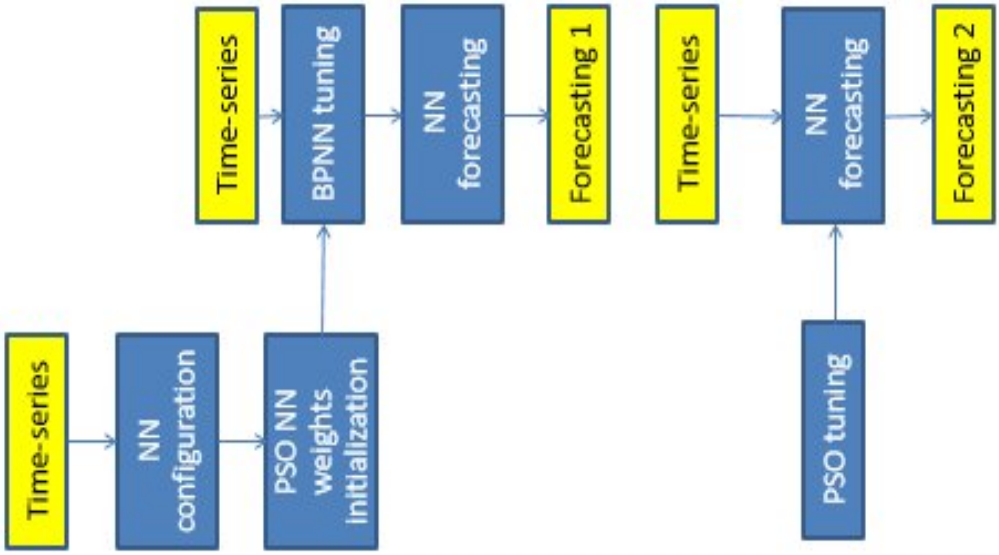


FIGURE 8-26 HYBRID BPNN AND PSO MODEL

There are sixteen inputs and one output. The number of hidden neurons can be changed. If the hidden layer has ten neurons defined with a group of weights, there will be 181 weights, and so the particle is defined with 181 real numbers.

The range of weights is set to [0, 1], absorbing boundaries. The algorithm is as follows:

```
overlimit=par<=1;

underlimit=par>=0;

par=par.*overlimit+not(overlimit);

par=par.*underlimit;
```

The fitness function for the regression problem is the minimum square root of the sum of squared error between the forecast output of the regression model and the target next day

closing price for the particle. First the PSO is used to calculate the initial neural network weights. Next the time series are fed to the back propagation neural network. Finally PSO is applied to train the ANN to get the lowest fitting error possible.

The experimental share price time series dataset is given in Appendix C: Table 26. Running PSO for a population of 10 with 300 iterations with a neural network of one hidden layer with 10 nodes, and 300 epochs gives very steady and promising results of training on the banking sector data as shown in Table 8-4. The results for BARC.L are shown in Figure 8-27, for HSBA.L are shown in Figure 8-28 and for LLOY.L are shown in Figure 8-29 showing extraordinary target and model likeliness overleaping graphs.

TABLE 8-4 BPNN AND PSO FORECAST PERFORMANCE

bank= BARC.L ANN= 0.0015246 ANN&PSO= 0.00021352 improvement= 7.1402

bank= LLOY.L ANN= 0.0010452 ANN&PSO = 0.00011553 improvement= 9.0471

bank= RBS.L ANN = 0.0070449 ANN&PSO = 0.0012869 improvement= 5.4744

bank= HSBA.L ANN= 0.0096447 ANN&PSO = 0.0017702 improvement= 5.4483

bank= VM. L ANN = 0.016416 ANN&PSO = 0.0035544 improvement= 4.6186

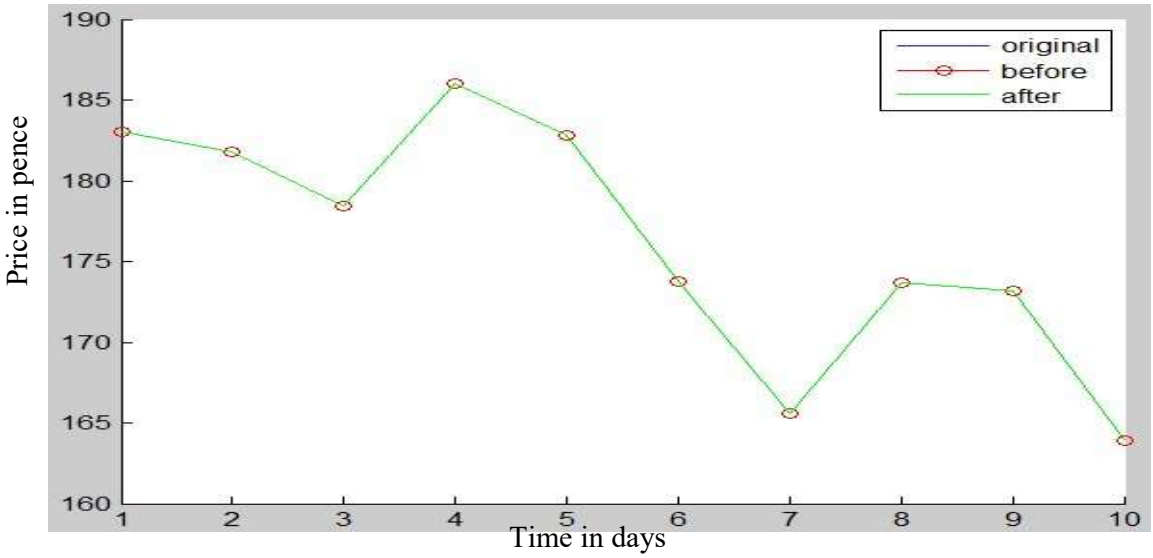


FIGURE 8-27 HYBRID BPNN AND PSO BARC.L FORECAST

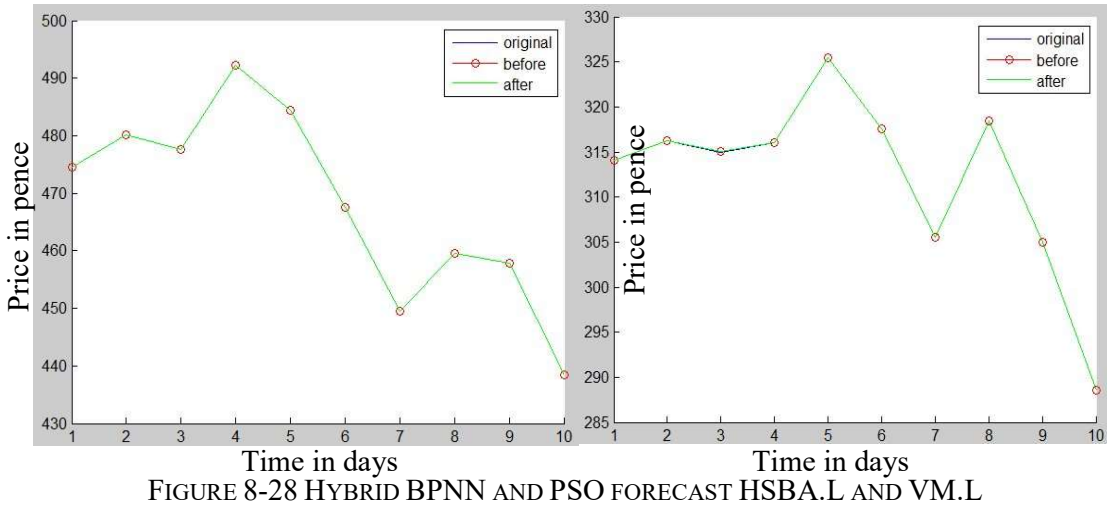


FIGURE 8-28 HYBRID BPNN AND PSO FORECAST HSBA.L AND VM.L

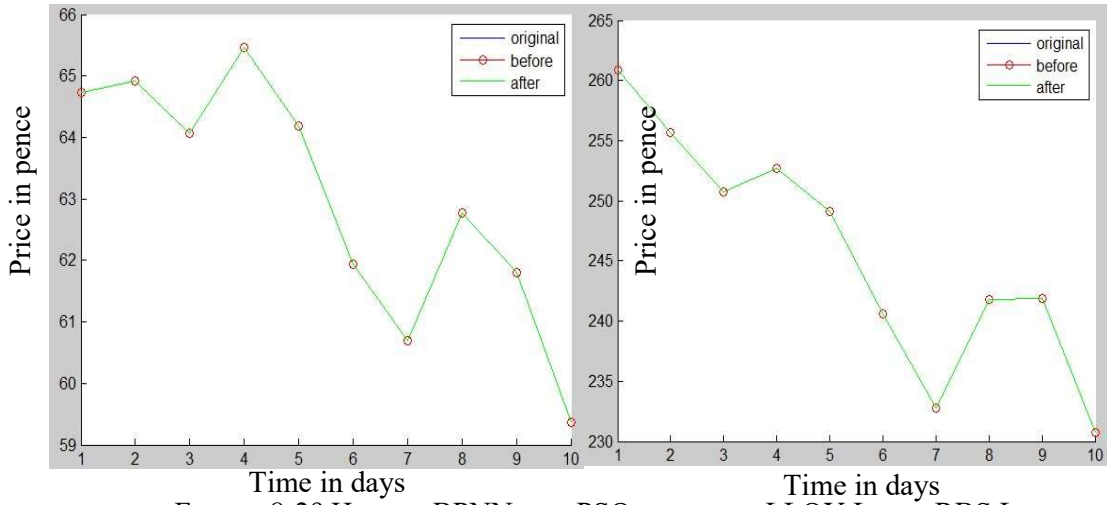


FIGURE 8-29 HYBRID BPNN AND PSO FORECAST LLOY.L AND RBS.L

8.13.3 PSO and Neural Network Conclusions

Recently there have been significant research efforts to apply evolutionary computation techniques such as PSO for the purposes of developing one or more aspects of artificial neural networks. Various approaches using PSO to replace the back-propagation learning algorithm in ANNs have been proposed in recent years. These have shown that PSO is a promising method in training ANNs. It is faster and gets better results in most cases. Further hybrid models with combinations of PSO and BPNN are very promising. These techniques have been applied to three main attributes of neural networks: network connection weights, the network topology and learning algorithms. Here the work was conducted on the network weight parameters.

The code, integrating the PSO and NN weights techniques is apparently very technically complex. Both the NN and PSO are very sensitive to initial state randomness (Xiaohui, 2013). There are many parameters in PSO that need to be adjusted to get better results in different trials, but the main ones are the number of hidden layers and the range of the weights (Xiaohui, 2013). The typical number of particles is 20 – 40, but for most problems such as time series forecasting 10 particles is large enough to get good results. The dimensionality of particles is determined by the problem to be optimized, and in the case of neural networks the number of weight parameters massively increases. In the experiment, there are over 500, comprising the number of NN node weights times the number of time series samples. The range of particles (NN weights) is also determined by the problem to be optimized.

Other performance and convergence algorithm-dependent parameters are:

- Maximum change possible for one particle during one iteration.
- Learning factors where usually c1 is equals to c2 and ranges from [0, 4].
- Stop condition maximum number of iterations the PSO execute.
- Minimum square root error requirement; for example, for ANN training in share price time series it is 1% of the price range.

8.14. Summary

The advantage of PSO is that it can be used in cases with non-differentiable transfer functions and when no gradient information is available. The disadvantages are that its performance is not necessarily competitive for some problems and the representation of the weights is difficult and they have to be carefully selected or developed.

There are two key steps when applying PSO to optimization problems: the representation of the solution, and the fitness function. One of the advantages of PSO is that it takes real numbers as particles. Then a standard procedure can be used to find the optimum. Searching is an iterative process, and the criteria for finishing the search are that the

maximum iteration number is reached or the minimum error condition is satisfied. There can be global and local versions. The global version considers only the global best, and is faster but might converge to a local optimum. The local version considers both the local and global best, and is slower but avoids local optima. Usually the global version gives an approximate quick result and the local version is used for refining the search.

However, many parameters need to be tuned, such as the dimensionality of particles, which is determined by the problem to be optimized. In the case of neural networks it is the number of weights and the inertia weight with PSO is a significant factor in convergence. PSO still has some limitations in mathematical analysis in terms of convergence.

It has been shown that, in the scalar case, each subsystem of the traditional PSO can be viewed as a closed-loop second order system controlled using a proportional controller. A mathematical analysis of the PSO algorithm from a systemic point of view along with stability analysis was performed in continuous and discrete time domains to determine the choice of parameters. Consequently, a possible proportional, integral and differential (PID) algorithm extension was recommended including a PID controller in the subsystem. With the PID controller the simulation easily found a stable system solution.

Conceptually, neural networks and PSO are different, ANN has a memory and is non-linear while PSO can search for the global optimum and is linear. The interpretation of PSO is natural to understand , while ANN is a black box, very difficult with complex multilayer architectures. A hybrid example combining a neural network with PSO was used to investigate the forecasting of a financial shares time series regression fitting function with the banking sector dataset.

These approaches have been applied and tested by a new (unique) application methodology developed for hybrid systems where neural network nodes are considered as particles and layered swarm architecture and neural network optimization are integrated. The code required to integrate the techniques is very technically complex. It was found that both

algorithms are very sensitive to initial state randomness. In the end, the efficiency of an algorithm may depend on its algorithm-dependent parameters, and the optimal parameter setting of the PSO algorithm is itself an optimization problem.

Chapter 9. CAPM and Risk Analysis

9.1. Overview

This chapter explains the classical capital asset pricing model (CAPM) and the related risk concept.

The rate of return or change is expected to reflect and compensate for taking on risk. There are two types of risks associated with a share: general market risk, which is systematic; and specific share risk which is unsystematic. Systematic risks are those that cannot be diversified away, for example because of interest rates and recessions. Specific risk represents the component of a share's return that is not correlated with general market changes.

This chapter covers the following subjects:

- CAPM model
- Beta as a measure of the risk
- Beta regression calculation
- Return on investment (ROI)

9.2. CAPM Model

The capital asset pricing model (CAPM) is a widely used financial theory in which a linear relationship is established between the return required on an investment and risk. The model is based on the relationship between an asset's beta, the risk-free rate (typically the Treasury bill rate) and the equity risk premium, which is the expected return on the market minus the risk-free rate.

The derivation of the CAPM requires the following assumptions to be made:

1. The returns from two assets are correlated with each other only because of their correlation with the return from the market. This is equivalent to assuming that there is only one factor driving return.

2. Investors focus on returns over a single period and that period is the same for all investors.
3. All investors make the same estimates of expected returns, standard deviations of returns, and correlations between returns.

The CAPM model can be written as:

$$\bar{r}_S = r_f + \beta_S(\bar{r}_m - r_f) \quad (9\text{-}1)$$

where $\bar{r}_S$ is the share return, r_f is the risk-free rate, typically of a 10-year government bond yield, $\bar{r}_m$ is the expected market return as a whole such as for a well-diversified stock index like the S&P 500 or FTSE-100, 250, $(\bar{r}_m - r_f)$ is the equity market premium for taking risk, and β_S is the beta value of the share.

9.3. Beta Measure of Risk

The beta is a measure of a share's risk and its relative volatility. It compares the fluctuation of a particular share with that of the market as a whole. If a share price follows the market, its beta is 1. A share with a beta of 1.5 would rise by 15% if the market rose by 10%, and fall by 15% if the market fell by 10%. The value of beta indicates the amount of compensation for the associated additional risk. Investors hold securities with betas greater than 1 while the market is rising, and securities with betas of less than 1 when the market is falling.

Generally, beta is a fundamental trade-off between minimizing risk and maximizing return. Beta is a historical measure of a stock's volatility and past beta figures or historical volatility does not necessarily guarantee future beta values or volatility.

Various types of beta are as follows:

- Negative beta. A beta less than 0, which would indicate an inverse relationship to the market, is possible but highly unlikely. Gold and gold stocks should have negative betas because they tend to do better when the stock market declines.

- Beta of 0. Basically, cash has a beta of 0. In other words, regardless of which way the market moves, the value of cash remains unchanged (given no inflation).
- Beta between 0 and 1. Companies with lower than that of the market have a beta of less than 1 but more than 0. Many utilities fall in this range.
- Beta of 1. A beta of 1 represents the volatility of the given index used to represent the overall market, against which other stocks and their betas are measured. The S&P 500 is such an index. If a stock has a beta of one, it will move in the same amount and direction as the index. So, an index fund that mirrors the S&P 500 will have a beta close to 1.
- Beta greater than 1. This denotes a volatility that is greater than the broad-based index. Many technology companies on the NASDAQ have a beta higher than 1. For the most part, stocks of well-known companies rarely have a beta higher than 4.

The value of beta is found by statistical analysis of individual, daily share price returns in comparison with the market's daily returns over the same period. The beta (β) of an asset is a measure of the sensitivity of its returns to returns from the market. It can be estimated from historical data as the slope obtained when the excess return on the asset over the risk-free rate is regressed against the excess return on the market over the risk-free rate. When $\beta=0$, an asset's returns are not sensitive to returns from the market. In this case, it has no systematic risk and the equation shows that its expected return is the risk-free rate.

A beta's reliability can be calculated using the coefficient of determination, or the r-squared, to determine how well it measures risk. The range of this statistic is zero to one, and the closer it is to one, the more reliable the beta is.

The index used to calculate the beta should be that of the stock market where the share is traded, such as in America the S&P 500 Index and in London the FTSE 100 (FTSE), 250(FTMC) or 350(FTLC).

There are two ways to calculate the value of beta using either a regression technique or the capital asset pricing model (CAPM). Regression, as used by investment practitioners, allows for a better explanation of returns pertaining to the market and takes interest rates as well as market returns into account. The CAPM is more academic and gives a theoretical explanation of the overall return of an asset.

9.4. Beta Regression Calculation

The closing price columns in order from newest to oldest for the index and share should be copied into a new spreadsheet. To obtain the correct format for calculation, these prices must be converted into return percentages for both the index and the stock price. To do this, the price from yesterday is subtracted from the price from today and the answer is divided by yesterday's price. The result is the percentage change.

Beta β is calculated on returns and not on prices, where "x" values are the market values and "y" values are the stock values:

$$\beta = Cov_{xy}/Var_x \tag{9-2}$$

For the calculation of beta using the CAPM model:

$$\bar{r}_S = r_f + \beta_S(\bar{r}_m - r_f) \tag{9-3}$$

So that:

$$\beta_S = \frac{\bar{r}_S - r_f}{\bar{r}_m - r_f} \tag{9-4}$$

where $\bar{r}_S$ is the share return, r_f is the risk free rate typically of a 10-year government bond yield, $\bar{r}_m$ is the expected market return as a whole such as from a well-diversified stock index such as the S&P 500 or FTSE-100 or 250,$(\bar{r}_m - r_f)$ is the equity market premium for taking risk and, and β_S is the beta value of the share.

On the Yahoo!! Finance website beta values can be found by entering a company name, then clicking on Key Statistics and looking under Stock Price History. The beta that is

calculated on Yahoo! compares the activity of the stock over the last five years and then compares it to the S&P 500.

For a period of 15 days, BARC.L and FTSE-2500 are similar with a slight linear similarity trend and BARC.L second order polynomial similarity R2 beta. The similarity between the actual BARC.L price and the price calculated using the CAPM is relatively good at R2=0.45. Beta is calculated for each point (date) and further modelling could improve the forecasting capability. Due to the very small risk-free rate, the unsystematic risk is the share risk with a linear beta dependency as shown in Figure 9-1.

9.5. Return on Investment (ROI)

The return on investment (ROI) is calculated as follows:

$$ROI_S = \frac{(\bar{r}_S + P_S)}{P_S} \tag{9-5}$$

where $\bar{r}_S$ is the share return, P_S is the share price, δ_S is the rate of change (ROC), and beta calculations are completed as shown in Table 9-1 and Table 9-2.

TABLE 9-1 BETA ROI CALCULATIONS PARAMETERS

S(BARC.L)	BARC share price
FTSE 250	FTSE-250 index
BoE	Bank of England (BoE) 10-y security
Rf(BoE)	BoE rate of change
Rs (BARC)	BARC.L rate of change
Rm (FTSE)	Market FTSE-250 rate of change
R2(Rs,Rm)	Similarity BARC and FTSE rate of change
Rs(CAPM)	CAPM rate of change
S(CAPM)	CAPM calculated share price BARC
R2(BARC,CAPM)	Similarity between actual and CAPM prices

TABLE 9-2 BETA ROI CALCULATION RESULTS BARC.L

Date	BARC.L	FTSE 250	BoE	Rf(BoE)	Rs (BARC)	Rm (FTSE)	Beta	Rs(CAPM)	CAPM
28/07/2016	146.5	17252.3	0.838	-0.071	-2.170	-0.079	1.27	-0.0808	149.63
27/07/2016	149.75	17265.9	0.902	-0.112	0.503	1.153	1.28	1.5052	151.24
26/07/2016	149	17069.1	1.016	0.041	-1.325	-0.128	1.34	-0.1859	150.72
25/07/2016	151	17091	0.976	-0.002	-0.527	0.633	1.46	0.9239	153.20
22/07/2016	151.8	16983.5	0.978	-0.018	-0.066	-0.375	1.08	-0.4016	151.29
21/07/2016	151.9	17047.4	0.996	0.005	0.629	0.167	1.10	0.1844	151.23
20/07/2016	150.95	17018.9	0.991	0.005	0.366	0.666	1.08	0.7218	151.49
19/07/2016	150.4	16906.3	0.986	0.008	-0.364	0.229	1.02	0.2334	151.30
18/07/2016	150.95	16867.7	0.978	0.008	0.768	0.839	1.09	0.9133	151.17
15/07/2016	149.8	16727.3	0.970	0.017	1.216	-0.362	1.03	-0.3715	147.45
14/07/2016	148	16788	0.954	0.023	1.999	0.221	1.70	0.3599	145.62
13/07/2016	145.1	16751	0.933	-0.069	-2.322	-0.334	1.86	-0.5608	147.72
12/07/2016	148.55	16807.1	1.001	0.077	2.166	0.603	1.86	1.0549	146.93
11/07/2016	145.4	16706.4	0.930	-0.008	4.417	3.267	1.86	6.0697	147.70

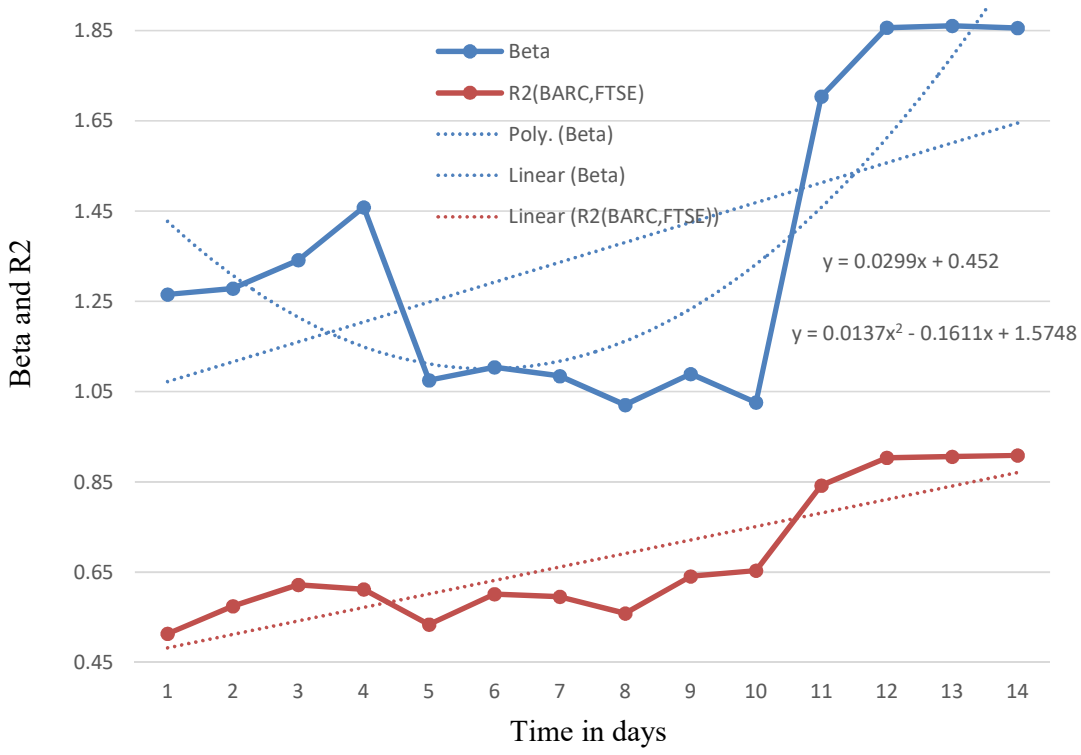


FIGURE 9-1 BETA AND R2 RESULTS BARC.L AND FTSE

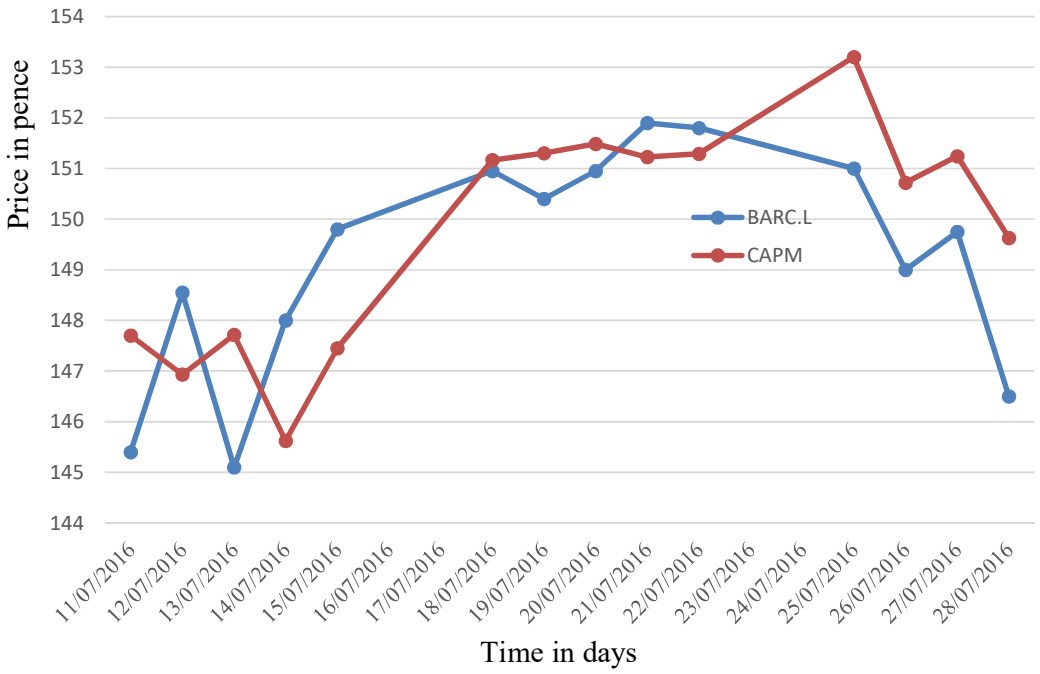
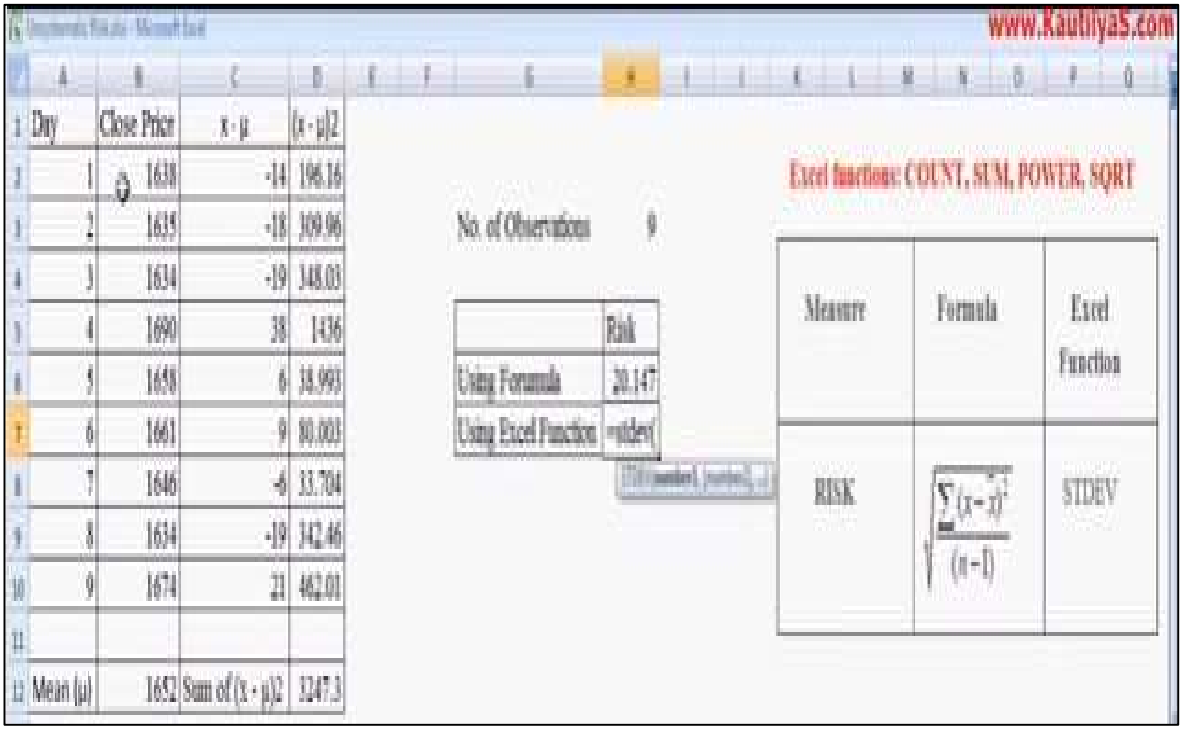


FIGURE 9-2 CAPM AND BARC.L SHARE

Unsystematic or total risk for a company or industry is the specific risk that is inherent in each investment. In Excel this is calculated using the standard deviation STDEV function and is shown in Table 9-3

.H7=STDEV(B2:B10) (9-6)

TABLE 9-3 UNSYSTEMATIC RISK CALCULATIONS APPROACH



The systematic and non-systematic associated risk calculations are as follows:

$$Risk\_total = Risk\_Systematic + Risk\_Non\text{-}systematic$$

$$Risk\_Unsystematic = Beta(Risk\_nmarket - Risk\_free)$$

$$Risk\_total = STDEV(Rs)$$

$$Risk\_Systematic = STDEV(Rf)$$

$$Risk\_market = STDEV(Rm)$$

STDEV(Rf) = 0.0486

STDEV(Rs) = 1.7828

STDEV(Rm) = 0.9377

Ten year bond yield historic data is shown in Appendix C: Table 27.

(Source: http://uk.investing.com/rates-bonds/uk-10-year-bond-yield-historical-data)

The CAPM relates the risk of investment to the expected return. Black et al. (1972) confirmed a linear relationship between the financial returns of stock portfolios and their betas values in a study of the price movements of stocks on the New York Stock Exchange between 1931 and 1965 (see Figure 9-3).

FIGURE 9-3 CLASSIC TEST CAPM AVERAGE MONTHLY RETURNS VS. BETA

Eugene Fama and Kenneth French (1993) looked at share returns on the New York Stock Exchange, the American Stock Exchange and NASDAQ between 1963 and 1990, and found that differences in betas over that lengthy period did not explain the performance of different stocks. The linear relationship between beta and individual stock returns also breaks down over shorter periods.

9.6. Summary

Theoretically, risk can be eventually removed to a certain degree by buying more different shares, although even a combination of all different shares in the stock market cannot eliminate all systematic risk.

No matter how much investments are diversified, it is impossible to get rid of all risk. There is always the possibility that a stock will lose some or all of its value, although by making the right choices market volatility makes it possible to make a profit.

With beta as a measure of volatility, shares can be chosen that meet expected criteria for risk. Low-risk shares have low betas such as utility stocks and treasury bills. High-risk shares, which eventually may yield more profit, have higher betas.

Chapter 10. Conclusions and Future Work

10.1. Overview

This chapter outlines the study's major conclusions and future work.

The primary goal of the present research is to model short-term daily trading in FTSE 100 shares to forecast with certain levels of confidence and associated risk. The hypothesis tested is that financial shares time series contain significant non-linearity and that ANN, either separately or in conjunction with PSO, could be utilised effectively.

Investigation of the periodicity and trend lines in short- and long-term trading and models using an ANN with the discrete Fourier transform (DFT) and discrete Wavelet transform (DWT) model features performed significantly better than analysis in the time domain. A mathematical analysis of the PSO algorithm from a systemic point of view along with stability analysis was performed to determine the choice of parameters, and a possible proportional, integral and differential (PID) algorithm extension was recommended.

The evaluation of statistical confidence for the models gave good results, which is encouraging for further experimentation considering model cross-validation for generalisation with an independent dataset to show how accurately the predictive models will perform in practice.

This chapter covers the following:

- Conclusions
- Future work

10.2. Conclusions

The conclusions of the research are positive, with good statistical confidence encouraging further experimentation.

The validation of the ANN model for share price investigations has found that non-linear models are likely to be a better choice than traditional linear regression for short-term

trading, and furthermore the bi-linear model outperforms the ANN. The experiments have been conducted with a single layer in order to compare clearly the ANN with linear regression models. It is expected that a multi-layer ANN would improve the results further. However, the interpretation of the results with the ANN model is more difficult and comparison with the linear model is less straightforward.

A Particle Swarm Optimisation with an Exponentially Varying Inertia Weight Factor (EVIWF) algorithm has been proposed, considering constraints. The effectiveness and applicability of the proposed algorithm has been tested and the results are compared with those in the literature. It is observed from the comparison that the proposed PSO-EVIWF has the ability to converge to a good quality solution without any sudden oscillations, and it is thus proven to be a better alternative method.

A chaotic adaptive particle swarm optimisation algorithm is proposed. A chaotic local search operator is introduced in the proposed algorithm to avoid premature convergence. The basic strategy of the proposed algorithm is to combine PSO with an adaptive inertia weight factor and chaotic local search. Logistics and a Gauss mapping technique are used in performing the chaotic local search and the results are compared. Numerical results show that the proposed method can obtain quality solutions for optimal cost and that it shows excellent convergence characteristics. Hence, the proposed algorithm is competitive with other algorithms in terms of its overall performance.

An extension of the PSO algorithm from a system control point of view was also proposed. It has been shown that, in the scalar case, each subsystem of the traditional PSO can be viewed as a closed-loop second order system controlled using a proportional controller. Consequently, an extension of the PSO was made by including a PID controller in the subsystem. Various methods as to how to choose the underlying tuning parameters were presented. It was shown that the performance of the proposed method is better than that of

traditional PSO. In future work, the performance of the proposed extended algorithms will be tested for other benchmark functions and applications.

Discrete Fourier Transform and Wavelet Transform time series analysis and decomposition for the features of the ANN forecasting model proved to perform significantly better than analysis in the time domain. Some periodicity was apparent in short-term and long-term trading.

Decomposition of the share time series revealed trends, harmonics and seasonality in the banking share sector, which helps with the generalization of the model. The ANN network has been applied at all decomposition stages and demonstrated very good robustness and prediction performance.

A hybrid example combining a Back Propagation Neural Network (BPNN) with PSO was used to investigate the forecasting of a financial shares time series regression fitting function with the banking sector dataset. Sixteen records of time series sets of fifteen consecutive closing price values for three trading weeks are the attributes (features), with the forecast target being the sixteenth day closing price. A three-layer neural network was used to perform the regression. The code required to integrate the techniques is very technically complex. Both NN and PSO are very sensitive to initial state randomness. There are many parameters in PSO that need to be adjusted but the main ones are the number of hidden layers and the range of the weights needed to get better results in different trials. The typical number of particles is 20 – 40, but for a time series ten particles is large enough to get good results. The dimensionality of particles is determined by the problem to be optimized, where in the case of neural networks there is an explosion in the number of weight parameters to over 500 in the experiments, comprising the number of NN nodes times the number of the time series samples. The range of particles (NN weights) is also determined by the problem to be optimized. The maximum change possible for one particle during an iteration is typically given when the learning factor *c1* is

equal to *c2* and ranges from [0, 4] and their sum for convergence reasons is about 4. For the stop condition, the maximum number of iterations the PSO executed and the minimum error requirement was found for ANN training in share time series to be 1% of the price range. The code used to integrate the techniques is again very technically complex. Both NN and PSO are very sensitive to initial state randomness. For parameter tuning, the efficiency of an algorithm may depend on its algorithm-dependent parameters, and the optimal parameter setting of the PSO algorithm is itself an optimization problem, especially with an NN.

10.3. Future Work

This work could be advanced in general by using either engineering simulation or mathematical abstract theoretical approaches, although the right balance is required to keep the analysis appropriate. Likely approaches could be to use mathematical models to improve the convergence of the simulation or for validation of the models developed. The understanding at this point is to measure the associated risk, which would probably be further explored with cautious statistical (Bayesian) methods. Furthermore, further generalization could be achieved with experimental cross-validation, the inclusion of noise or other methods used for the model's parameters with assumptions made concerning the input signal for data fitting and prediction or estimation.

Promising future investigations would consider the fitness (Kaastra, et al., 1995) of self-organizing neural networks, the Hopfield network, bidirectional associative memory or evolutionary computation genetic algorithms with concern for convergence ambiguity (Qi, et al., 2001). Further possible investigations would involve combining inverse finite-element modelling using the Galerkin residual integral method and multi-layer perceptron (MLP) neural network models (Ondimu, et al., 2007). Also time-varying weights in neural networks could be used in a semi-non-parametric technique to make approximations by means of the Galerkin method (Barucci, et al., 2010, Barucci, et al., 1996) and a

Galerkin/neural-network-based design initially applying the Galerkin method to the model to derive an ordinary differential equation (ODE) system with unknown non linearity subsequently parameterized by a multilayer neural network (MNN) with one hidden layer (Wu, et al., 2008), whose relevance to the hypothesis could be checked.

10.4. Summary

The major conclusions are:

- The existing literature suggests that neural networks have not been extensively utilised in the field of the modelling of short-term financial stock market shares.
- The systemic (control theory) analysis and control algorithm extension can considerably improve convergence and the system's stability and robustness.
- The combined application of artificial neural networks and particle swarm optimization modelling and acceleration approaches can improve quality, robustness, convergence and performance.
- Deep machine learning with digital Fourier and wavelets transforms has produced superior results in generalization, performance and robustness compared to time domain analysis.

A comparison of the performance of the methods with the coefficient of determination or likeliness $(R)^2$ and root mean square error (RMS) for BARC.L shows evolution of performance with the gradual application and combination of concepts and techniques

- Analytical Stochastic Model RMS~16.07, R^2 ~0.75
- ANN time domain RMS~5.42, R^2 ~0.85
- PSO-PID (convergence quality) RMS~5.6, R^2 ~0.82
- ANN with DFT & DWT RMS~5.1, R^2 ~0.91
- Hybrid ANN and PSO RMS~1, R^2 ~0.95

The most technically challenging contributions are as follows:

- Experiments and simulations, there is a data explosion in the hybrid model algorithms tuning over 500 variables in a relatively simple architecture/structure
- The randomness in the algorithm calculations, the initial point and tuning of neural network initial weights and PSO particle's position with social and learning factors make the evaluation and comparison of the results challenging.

The contributions which most represent breakthroughs:

- The deep learning with particle swarm optimization, neural network and digital Fourier and wavelet transformations in the hybrid model is absolute sophistication dealing with non-linearity (ANN), global optima (PSO) and robustness (digital domain)
- The real-time real-data sector model insight features and timescale classification criteria, justifying fourteen days data analysis (trading vs. investment)
- PSO control theory analysis and proportional derivative and integral extension following the application of structured and knowledgeable tuning (existing guidance and techniques for stability, convergence and response well developed in control theory

Trading strategy references:

- The short term trading sector has more volatility and hence more eventual profit / loss risk, although it depends on the momentary state of the market. Eventually fundamental market research could help to understand the state of the market.
- Market analysis of two weeks historic data is suggested to be the most general period for daily trading.
- Trading strategy depends on the risk that one is prepared to take, with the portfolio helping to hedge the risk.

Future work could be advanced in general in following respects:

- Using advanced mathematical analytical models to improve the convergence of the simulation or for validation of the models developed. The understanding at this point is to measure the associated risk, which would probably be further explored with cautious statistical (Bayesian) methods.
- Generalization of the models could be achieved with experimental cross-validation, the inclusion of noise or other methods used for the model's parameters selection with assumptions made concerning the input signal for data fitting and prediction or estimation.

References

Abido, M.A., 2006. Multi objective evolutionary algorithms for electric power dispatch problem. *IEEE transactions on evolutionary computation*, *10*(3), pp.315-329.

Adhinarayanan, T. and Sydulu, M., 2006. August. Fast and effective algorithm for economic dispatch of cubic fuel cost based thermal units. In *First international conference on industrial and information systems* (pp. 156-160). IEEE.

Aminian, F., Suarez, E.D., Aminian, M. and Walz, D.T., 2006. Forecasting economic data with neural networks. *Computational Economics*, *28*(1), pp.71-88.

Amoli, N.A., Jalid, S., Shayanfar, H.A. and Barzinpour, F., 2012. September. Solving economic dispatch problem with cubic fuel cost function by firefly algorithm. In *8th International Conference on "Technical and Physical Problems of Power Engineering" (ICTPE) Fredrikstad, Norway* (pp. 5-7).

Amman, H.M., Rustem, B. and Whinston, A.B. eds., 2013. *Computational approaches to economic problems* (Vol. 6). Springer Science & Business Media.

Al-Sumait, J.S., Sykulski, J.K. and Al-Othman, A.K., 2008. Solution of different types of economic load dispatch problems using a pattern search method. *Electric Power Components and Systems*, *36*(3), pp.250-265.

Al Wadia, M.T.I.S. and Tahir Ismail, M., 2011. Selecting wavelet transforms model in forecasting financial time series data based on ARIMA model. *Applied Mathematical Sciences*, *5*(7), pp.315-326.

Atsalakis, G.S. and Valavanis, K.P., 2009. Surveying stock market forecasting techniques–Part II: Soft computing methods. *Expert Systems with Applications*, *36*(3), pp.5932-5941.

Bäck, T. and Schwefel, H.P., 1993. An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, *1*(1), pp.1-23.

Baestaens, D. E., Van Den Bergh, W.M and Wood, D., 1994. *Neural network solutions for trading in financial market*. Pitman Publishing

Bank of England. 2016. *Yields, British Government Securities, 10 years daily*. [online] www.bankofengland.co.uk Available at: http://www.bankofengland.co.uk. [Accessed 1 August 2016].

Barucci, E., Landi, L. and Cherubini, U., 1996. Computational methods in finance: Option pricing. *IEEE Computational Science & Engineering*, *3*(1), pp.66-80.

Barucci, E., Landi, L. and Cherubini, U., 1996. Computational methods in finance: Option pricing. *IEEE Computational Science & Engineering*, *3*(1), pp.66-80.

Barucci, E., Cherubini, U. and Landi, L., 1997. Neural networks for contingent claim pricing via the Galerkin method. In *Computational Approaches to Economic Problems* (pp. 127-141). Springer US.

Basdogan, C., 2016. *Discrete PID*. [online] http://portal.ku.edu.tr. Available at: http://portal.ku.edu.tr/~cbasdogan/Courses/Robotics/projects/Discrete_PID.pdf. [Accessed 1 November 2016].

BBC News. 2016. *FTSE 100 Index 15 min delay*. [online] http://www.bbc.co.uk. Available at: http://www.bbc.co.uk/news/business/market_data/stockmarket/3/. [Accessed 3 November 2016].

Bennell, J. and Sutcliffe, C., 2004. Black–Scholes versus artificial neural networks in pricing FTSE 100 options. *Intelligent Systems in Accounting, Finance and Management*, *12*(4), pp.243-260.

Van den Bergh, F. and Engelbrecht, A.P., 2006. A study of particle swarm optimization particle trajectories. *Information sciences*, *176*(8), pp.937-971.

Black, F., Jensen, M.C. and Scholes, M.S., 1972. *The Capital Asset Pricing Model: Some empirical tests.* financial economists

Black, F., Scholes, M.S., 1973. The pricing of Options and corporate liabilities. *Journal of Political Economy*. 81 (3): 637–654

Borgelt, C. and Kruse, R., 2002. *Graphical models: Methods for data analysis and mining*. John Wiley & Sons.

Borse, G.J., 1996. *Numerical methods with MATLAB: A resource for scientists and engineers* (1st ed.). International Thomson Publishing.

Box, G.E., and *Jenkins,* G.M., 1970. *Time series analysis: forecasting and control.* San Francisco: Holden-Day.

Box, G.E., Jenkins, G.M., Reinsel, G.C. and Ljung, G.M., 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

Bourg, D.M., 2006. *Excel scientific and engineering cookbook*. O'Reilly Media, Inc.

Brockwell, P.J. and Davis, R.A., 1987. *Time series: theory and methods*. Springer-Verlag.

Brockwell, P.J. and Davis, R.A., 2013. *Time series: theory and methods*. Springer Science & Business Media.

Bradford Jr., D. and Hung, C.C., 2011. Convergence issues in particle swarm optimization, in Olson, A.E. (ed.) *Particle Swarm Optimization: Theory, Techniques and Applications*, Nova Science Publishers, Inc.

Bratton, D. and Kennedy, J., 2007, April. Defining a standard for particle swarm optimization. In *2007 IEEE swarm intelligence symposium* (pp. 120-127). IEEE.

Busawon, K., Rani, R., Turkedjiev, E., and Binns, R. (in prep) Extension of particle swarm optimisation algorithm: application to economic dispatch. In *Transaction on Evolutionary Computation*

Cai, J., Ma, X., Li, L. and Haipeng, P., 2007. Chaotic particle swarm optimization for economic dispatch considering the generator constraints. *Energy Conversion and Management*, *48*(2), pp.645-653.

Campana, E.F., Fasano, G. and Peri, D., 2006. Globally convergent modifications of particle swarm optimization for unconstrained optimization. *Particle Swarm Optimization: Theory, Techniques and Applications. Advances in Engineering Mechanics*, pp.97-118.

Capinski, M. and Zastawniak T., 2011. *Mathematics for finance*. Springer.

Castillo, P.A., Merelo, J.J., Prieto, A., Rivas, V. and Romero, G., 2000. G-Prop: Global optimization of multilayer perceptions using GAs. *Neurocomputing*, *35*(1), pp.149-163.

CBOE. 2016. *VXD Volatility Index*. [online] www.cboe.com/micro/vxd. Available at: at: http://www.cboe.com/micro/vxd/. [Accessed 3 August 2016].

Chandar, S.K., Sumathi, M. and Sivanandam, S.N., 2015. Forecasting of foreign currency exchange rate using neural network. *International Journal of Engineering and Technology*, 7(1), 99-108.

Chuanwen, J. and Bompard, E., 2005. A self-adaptive chaotic particle swarm algorithm for short term hydroelectric system scheduling in deregulated environment. *Energy Conversion and Management*, *46*(17), pp.2689-2696.

Chou, P.C. and Hong, S.B., 2012. Particle swarm optimization algorithms for mini-benchmark problems. In *2012 International Conference on Machine Learning and Cybernetics* (Vol. 5, pp. 1656-1661). IEEE.

Clarke, J., Jandik, T. and Mandelker, G., 2001. The efficient markets hypothesis. *Expert financial planning: Advice from industry leaders*, pp.126-141.

Clerc, M. and Kennedy, J., 2002. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation*, *6*(1), pp.58-73.

Cooley, J.W. and Tukey, J.W., 1965. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, *19*(90), pp.297-301.

Da, Y. and Xiurun, G., 2005. An improved PSO-based ANN with simulated annealing technique. *Neurocomputing*, *63*, pp.527-533.

Das, S., Abraham, A. and Konar, A., 2008. Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. In *Advances of Computational Intelligence in Industrial Systems* (pp. 1-38). Springer Berlin Heidelberg.

Das, M. and Dehuri, S., 2011. Some studies on particle swarm optimization for single and multi-objective problems, In Dehuri, S., Ghosh, S. and Cho, S.-B. (ed.) *Integration of Swarm Intelligence and Artificial Neural Network.* World Scientific Pub.

Day, A.D., 2001. *Mastering financial modelling. A practitioner's guide to applied corporate finance.* FT Prentice Hall London.

Deboeck, G., 1992. Pre-processing and evaluation of neural nets for trading stocks. *Advanced Technology for Developers*.

Deboeck, G.J., 1994. *Trading on the edge: neural, genetic, and fuzzy systems for chaotic financial markets*, Wiley, 1994.

Debevec, P.T., 2016. *Excel FFT Instructions*. [online] www.stem2.org Available at: http://www.stem2.org/je/Excel_FFT_Instructions.pdf. [Accessed 12 August 2016].

Dehuri, S., Ghosh, S. and Cho, S.B., 2011. *Integration of swarm intelligence and artificial neural network*. World Scientific.

Devadoss, A.V. and Ligori, T.A.A., 2013. Stock prediction using artificial neural networks. *International Journal of Data Mining Techniques and Applications*, *2*, pp.283-291.

Duhamel, P. and Vetterli, M., 1990. Fast Fourier transforms: a tutorial review and a state of the art. *Signal processing*, *19*(4), pp.259-299.

Eberhart, R.C. and Shi, Y., 1998, March. Comparison between genetic algorithms and particle swarm optimization. In *International Conference on Evolutionary Programming* (pp. 611-616). Springer Berlin Heidelberg.

Eberhart, R.C. and Shi, Y., 2000. Comparing inertia weights and constriction factors in particle swarm optimization. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on* (Vol. 1, pp. 84-88). IEEE.

Egeli, B., Ozturan, M. and Badur, B., 2003. Stock market prediction using artificial neural networks. In *Proceedings of the 3rd International Conference on Business*, June 18-21, Hawaii, pp 1-8.

Engelbrecht, A.P., 2007. *Computational intelligence: an introduction*. John Wiley & Sons.

Ericsson, K., Estep, D., Hansbo, P. and Johnson, C., 1996. *Computational deferential equations*. Cambridge University Press.

Fama, E.F. and French, K.R., 1993. Common risk factors in the returns on stocks and bonds. *Journal of financial economics*, *33*(1), pp.3-56.

Lexicon.ft.com. (2017). *Common Stock Definition from Financial Times Lexicon*. [online] Available at: http://lexicon.ft.com/Term?term=common-stock [Accessed 16 Apr. 2017].

Funahashi, K.I., 1989. On the approximate realization of continuous mappings by neural networks. *Neural networks*, *2*(3), pp.183-192.

Gaing, Z.L., 2003. Particle swarm optimization to solving the economic dispatch considering the generator constraints. *IEEE transactions on power systems*, *18*(3), pp.1187-1195.

Gaing, Z.L., 2004. A particle swarm optimization approach for optimum design of PID controller in AVR system. *IEEE transactions on energy conversion*, *19*(2), pp.384-391.

Gilks, W.R., Richardson, S. and Spiegelhalter, D.J., 1996. Introducing Markov chain Monte Carlo. *Markov chain Monte Carlo in practice*, *1*, p.19

Godoi, A., 2009. Valuation models and Simon's bounded rationality. *Revista De Economia Política*, 29(3), 54-70.

Happ, H.H., 1977. Optimal power dispatchꟻA comprehensive survey. *IEEE Transactions on Power Apparatus and Systems*, *96*(3), pp.841-854

Hamilton, J.D., 1994. *Time series analysis.* Princeton University Press.

Harper, D., 2012. *Find The Right Fit with Probability Distributions*. [online] Available at: http://www.investopedia.com/articles/06/probabilitydistribution.asp#12921468310312&close. [Accessed 14 July 2015].

Haugh, M., 2012. *The Monte Carlo Framework, Examples from Finance and Generating Correlated Random Variables*. [online] www.columbia.edu Available at: http://www.columbia.edu/~mh2078/MCS04/MCS_framework_FEegs.pdf. [Accessed 29 July 2015].

Haupt, R.L. and Haupt, S.E., 2004. *Practical Genetic Algorithms*, Second Edition. John Wiley and Sons, Inc.

Helmick, S.D. and Shoults, R.R., 1985. A practical approach to an interim multi-area economic dispatch using limited computer resources. *IEEE Power Engineering Review*, (6), pp.46-47.

Hornik, K., Stinchcombe, M. and White, H., 1989. Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), pp.359-366.

Hsieh, T.J., Hsiao, H.F. and Yeh, W.C., 2011. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Applied soft computing*, *11*(2), pp.2510-2525.

Huang, S.C. and Wu, T.K., 2010. Integrating recurrent SOM with wavelet-based kernel partial least square regressions for financial forecasting. *Expert Systems with Applications*, *37*(8), pp.5698-5705.

Huang, S.C., 2011. Forecasting stock indices with wavelet domain kernel partial least square regressions. *Applied Soft Computing*, *11*(8), pp.5433-5443

Hull, J.C., 2008. *Options, futures and other derivatives*, 7th ed., Prentice Hall.

Hull, J., 2014. *Fundamentals of futures and options markets*, eBook. English, Published Harlow, Essex: Pearson.

Hurwitz, A., 1895. *"Ueber die Bedingungen, unter welchen eine Gleichung nur Wurzeln mit negativen reellen Theilen besitzt". Math. Ann.* ***46*** *(2): 273–284.* (English translation "On the conditions under which an equation has only roots with negative real parts" by Bergmann, H.G. in *Selected Papers on Mathematical Trends in Control Theory* Bellman, R. and Kalaba, R. Eds. New York: Dover, 1964 pp. 70–82.)

Investing Staff. 2016. *UK 10-Year Bond Yield*. [online] uk.investing.com Available at: http://uk.investing.com/rates-bonds/uk-10-year-bond-yield-historical-data. [Accessed 31 July 2016].

Investopedia Staff. 2016. *Beta: Gauging Price Fluctuations*. [online] Available at: http://www.investopedia.com/articles/01/102401.asp. [Accessed 31 July 2016].

Jani, Dr Kamlesh. *Managerial economics*. Place of publication not identified: Lulu Com, 2012. Print.

Jayabarathi, G., Sadasivam, V., Ramachandran, T., 2000. Evolutionary programming-based multiarea economic dispatch with tie line constraints. *Electric Machines &Power Systems*, *28*(12), pp.1165-1176

Jensen, M.C., Black, F. and Scholes, M.S., 1972. *The capital asset pricing model: Some empirical tests*.

Jeyakumar, D.N., Jayabarathi, T. and Raghunathan, T., 2006. Particle swarm optimization for various types of economic dispatch problems. *International Journal of Electrical Power & Energy Systems*, *28*(1), pp.36-42.

Justel, A.; Peña, D.; Zamar, R., 1997. A multivariate Kolmogorov–Smirnov test of goodness of fit. *Statistics & Probability Letters*. **35** (3): 251–259.

Johnson, N F, Jefferies, P and Hui P M (2003). *Financial Market Complexity*. Oxford University Press.

JStock Staff. 2010. *JStock*. [online] jstock.sourceforge.net. Available at: http://jstock.sourceforge.net. [Accessed 5 July 2016].

Juang, C.F., 2004. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *34*(2), pp.997-1006

Kaastra, I. and Boyd, M., 1996. Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, *10*(3), pp.215-236

Kadirkamanathan, V., Selvarajah, K. and Fleming, P.J., 2006. Stability analysis of the particle dynamics in particle swarm optimizer. *IEEE Transactions on Evolutionary Computation*, *10*(3), pp.245-255.

Karaboga, D. and Akay, B., 2009. A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, *31*(1-4), pp.61-85.

Karayiannis, N. and Venetsanopoulos, A.N., 2013. *Artificial neural networks: learning algorithms, performance evaluation, and applications* (Vol. 209). Springer Science & Business Media.

Kautillyas. 2016. *How to calculate Unsystematic Risk or total risk using Excel*. [online] Available at: https://www.youtube.com/watch?v=kWW6aMHWXPQ&feature=youtu.be. [Accessed 12 July 2016].

Kautilyas., 2016. *How to calculate CAPM using Excel* . [online] www.youtube.com Available at: https://www.youtube.com/watch?v=YO8YdCX_e1g&feature=youtu.be. [Accessed 31 August 2016].

Kautilyas. 2016. *Systematic Risk of a stock*. [online] www.youtube.com Available at: https://www.youtube.com/watch?v=4_lbdCpvaD4. [Accessed 31 July 2016].

Kennedy, J., and Eberhart, R.C., 1995. Particle swarm optimization. *Proc. IEEE Int. Conf. on Neural Networks*, IV. Piscataway, NJ: IEEE Service Center, pp. 1942–1948.

Kharitonov, V.L., 1996. Robust stability of nested polynomial families. *Automatica*, *32*(3), pp.365-367.

Kiranyaz, S., Ince, T., Yildirim, A. and Gabbouj, M., 2009. Evolutionary artificial neural networks by multi-dimensional particle swarm optimization. *Neural Networks*, *22*(10), pp.1448-1462.

Klingenberg, L. 2016. *Frequency Domain Using Excel*. [online] Available at: http://physics.oregonstate.edu/~grahamat/COURSES/ph421/lec/ExcelFFT.pdf. [Accessed 12 August 2016].

Kodogiannis, V. and Lolis, A., 2002. Forecasting financial time series using neural network and fuzzy system-based techniques. *Neural computing & applications*, *11*(2), pp.90-102.

Kothari, D.P., 2012. Power system optimisation Invited lecture. *IEEE International conference proceedings, CISP'12.* 18-21, 2012, Symposium, 2007, pp. 120-127.

Kroese, D.P., Taimre, T., Botev, Z.I. and Rubinstein, R.Y., 2007. *Solutions manual to accompany simulation and the Monte Carlo Method* (pp. 1-188). Wiley-Interscience.

Kumaran, G. and Mouly, V.S.R.K., 2001. Using evolutionary computation to solve the economic load dispatch problem. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on* (Vol. 1, pp. 296-301). IEEE.

Lahmiri S., 2014. Wavelet low-and high-frequency components as features for predicting stock prices with back propagation neural networks. *Journal of King Saud University – Computer and Information Sciences*. 2014; 26(2):218–27.

Lahmiri S., 2013. Forecasting direction of the S & P 500 movement using wavelet transform and support vector machines. *International Journal of Strategic Decision Sciences*. 2013; 4(1):78–88.

Lapidus, L. and Pinder, G.F., 2011. *Numerical solution of partial differential equations in science and engineering*. John Wiley & Sons.

Lasdon, L.S., Fox, R.L. and Ratner, M.W., 1974. Non-linear optimization using the generalized reduced gradient method. *Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle*, *8*(3), pp.73-103

Lawrence, R., 1997. Using neural networks to forecast stock market prices. *University of Manitoba.*

Lee, A.C., Lee, J.C. and Lee, C.F., 2009. *Financial analysis, planning & forecasting: Theory and application*. World Scientific.

Leguizamón, G. and Coello, C.A.C., 2011. Multi-objective ant colony optimization: A taxonomy and review of approaches. *Integration of Swarm Intelligence and Artificial Neural Network*, pp.67-94.

LINEST.Support.office.com., 2017. *LINEST function - Office Support*. [online] Available at: https://support.office.com/en-us/article/LINEST-function-84d7d0d9-6e50-4101-977a-fa7abf772b6d [Accessed 17 Apr. 2017].

Liu, B., Wang, L., Jin, Y.H., Tang, F. and Huang, D.X., 2005. Improved particle swarm optimization combined with chaos. *Chaos, Solitons & Fractals*, *25*(5), pp.1261-1271.

Ljung, G.M., Box, G.E.P., 1978. On a measure of a lack of fit in time series models. *Biometrika* 65: 297–303.

Lucidi, S. and Sciandrone, M., 2002. On the global convergence of derivative-free methods for unconstrained optimization. *SIAM Journal on Optimization*, *13*(1), pp.97-116.

Luminouslogic. 2016. *How to get a free real time stock quote from Google finance*. [online] Available at: http://luminouslogic.com/how-to-get-a-free-real-time-stock-quote-from-google-finance-into-matlab.htm. [Accessed 12 August 2016].

Majhi, R., Majhi, B. and Panda, G., 2011. Efficient Prediction of Retail Sales Using Differential Evolution Based Adaptive Model, *Integration of Swarm Intelligence and Artificial Neural Network, series in Machine Perception and Artificial Intelligence*, p.213. World Scientific Pub Co Pte.

Mallat, S.G., 1989. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, *11*(7), pp.674-693.

Manoharan, P.S., Kannan, P.S. and Ramanathan, V., 2009. A Novel EP approach for Multi-area economic dispatch with multiple fuel options. *Turkish Journal of Electrical Engineering & Computer Sciences*, *17*(1), pp.1-19

Mathworks, 2016. *Fast Fourier transform*. [online] Available at: http://uk.mathworks.com/help/matlab/math/fast-fourier-transform-fft.html. [Accessed 16 August 2016].

Mathworks, 2016. *Discrete Fourier transform.* [online] Available at: http://uk.mathworks.com/help/matlab/math/discrete-fourier-transform-dft.html. [Accessed 15 August 2016

Mathworks. 2016. *R-square: The coefficient of determination.* [online] Available at: https://uk.mathworks.com/matlabcentral/fileexchange/34492-r-square--the-coefficient-of-determination. [Accessed 15 August 2016].

Mathworks. 2016. *Wavelet analysis of financial data*. [online] Available at: http://uk.mathworks.com/help/wavelet/examples/wavelet-analysis-of-financial-data.html. [Accessed 15 August 2016].

Mathworks. 2016. *Calculating the standard error of the mean*. [ONLINE] Available at: http://www.matlab-cookbook.com/recipes/0100_Statistics/010_sem.html. [Accessed 15 August 2016].

Mathworks. 2016. *Hands-on time series analysis with analysis with Matlab*. [ONLINE] Available at: http://alumni.cs.ucr.edu/~mvlachos/ICDM06/tutorial_icdm06.pdf. [Accessed 15 August 2016].

McCutcheon J.J. and Scott W.F., 1986. *An introduction to the mathematics of finance*, Butterworth-Heinemann.

Michalewicz, Z., 1992. *Genetic algorithms + data structures = evolution programs*. New York: Springer-Verlag.

Minsky, M. L. and Papert, S. A., 1990. *Perceptrons,* Expanded Edition. s.l. : The MIT Press.

Misra, B.B., Dash, P.K. and Panda, G., 2011. Efficient Classifier Design with Hybrid Polynomial Neural Network. *Integration of Swarm Intelligence and Artificial Neural Network, series in Machine Perception and Artificial Intelligence*, *78*, pp.179-212. World Scientific Pub Co Pte.

Moré, J.J., 1978. The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical analysis* (pp. 105-116). Springer Berlin Heidelberg.

Negnevitsky, M., 2002. *Artificial intelligence A guide to intelligent systems*. Addison Wesley.

Niknam, T., 2010. A new fuzzy adaptive hybrid particle swarm optimization algorithm for non-linear, non-smooth and non-convex economic dispatch problem. *Applied Energy*, *87*(1), pp.327-339

Nyquist, H., 1928. Certain topics in telegraph transmission theory., *Trans. AIEE*, Vol. 47, pp. 617-644. Reprint as classic paper in: *Proc. IEEE*, Vol. 90, No. 2, Feb 2002.

Olsson, A., 2011. *Particle swarm optimization: theory, techniques, and application*s. Hauppauge, N.Y.: Nova Science.

Omkar, S.N. and J Senthilnath, J., 2011. Neural Network and Swarm Intelligence for Data Mining, *Integration of Swarm Intelligence and Artificial Neural Network, series in Machine Perception and Artificial Intelligence*, p23. World Scientific Pub Co Pte.

Ondimu, S. N. and Murase, H., 2007. *Combining Galerkin methods and neural network analysis to inversely determine thermal conductivity of living green roof material*s. Elsevier, ScienceDirect.

Ouyang, Z. and Shahidehpour, S.M., 1991, May. Heuristic multi-area unit commitment with economic dispatch. In *IEE Proceedings C-Generation, Transmission and Distribution* (Vol. 138, No. 3, pp. 242-252). IET.

Palmes, P.P., Hayasaka, T. and Usui, S., 2005. Mutation-based genetic neural network. *IEEE Transactions on Neural Networks*, *16*(3), pp.587-600

Panigrahi, B.K., Pandi, V.R. and Das, S., 2008. Adaptive particle swarm optimization approach for static and dynamic economic load dispatch. *Energy conversion and management*, *49*(6), pp.1407-1415.

Panigrahi, B.K., Mohapatra, A., Ray, P. and Das, S., 2011. Automated Power Quality Disturbance Classification Using Evolvable Neural Network*, Integration of Swarm Intelligence and Artificial Neural Network, series in Machine Perception and Artificial Intelligence*, p121. World Scientific Pub Co Pte.

Pankratz, A., 2009. *Forecasting with univariate Box-Jenkins models: Concepts and cases* (Vol. 224). John Wiley & Sons

Pardo, R., 1992. *Design, testing, and optimization of trading systems* (Vol. 2). John Wiley & Sons

Qi, M. and Zhang, G.P., 2001. An investigation of model selection criteria for neural network time series forecasting. *European Journal of Operational Research*, *132*(3), pp.666-680.

Quintana, V.H., Lopez, R., Romano, R. and Valadez, V., 1981. Constrained economic dispatch of multi-area systems using the Dantzig-Wolfe decomposition principle. *IEEE Transactions on Power Apparatus and Systems*, (4), pp.2127-2137.

Rani, C., Petkov, E., Busawon, K. and Farrag, M., 2014, November. Chaotic adaptive particle swarm optimisation using logistics and Gauss map for solving cubic cost economic dispatch problem. In *Environmental Friendly Energies and Applications (EFEA), 2014 3rd International Symposium on* (pp. 1-5). IEEE.

Rani, C., Petkov, E., Busawon, K. and Farrag, M., 2014, November. Particle swarm optimization with exponentially varying inertia weight factor for solving multi area economic dispatch. In *Environmental Friendly Energies and Applications (EFEA), 2014 3rd International Symposium on* (pp. 402-407). IEEE.

Robinson, J. and Rahmat-Samii, Y., 2004. Particle swarm optimization in electromagnetics. *IEEE transactions on antennas and propagation*, *52*(2), pp.397-407

Roiger, R. J. and Geatz, M. W., 2003. *Data mining a tutorial–based primer*. Addison Wesley.

Rosenblatt, F., 1960. Perceptron simulation experiments. *Proceedings of the IRE*, *48*(3), pp.301-309

Ross S.M., 1999. *An introduction to mathematical finance*, Cambridge University Press.

Rotundo, G., 2004. Neural networks for large financial crashes forecast. *Physica A: Statistical Mechanics and its Applications*, *344*(1), pp.77-80.

Routh, E. J., 1877. *A Treatise on the Stability of a Given State of Motion: Particularly Steady Motion.* Macmillan.

Salerno. J., 1997. Using the particle swarm optimization technique to train a recurrent neural model, In *Proceedings of Ninth International Conference on Tools with Artificial Intelligence (ICTAP97)*, IEEE Press.

Schwefel, H., 1995. *Evolution and optimum seeking*. New York: Wiley.

Shannon, C.E., 1949. Communication in the presence of noise. *Proceedings of the IRE*, *37*(1), pp.10-21.

Sharma, M., Pandit, M. and Srivastava, L., 2011. Reserve constrained multi-area economic dispatch employing differential evolution with time-varying mutation. *International journal of electrical power & energy systems*, *33*(3), pp.753-766

Sharpe, W.F., 1994. The sharpe ratio. *The journal of portfolio management*, *21*(1), pp.49-58.

Shoults, R.R., Chang, S.K., Helmick, S. and Grady, W.M., 1980. A practical approach to unit commitment, economic dispatch and savings allocation for multiple-area pool operation with import/export constraints. *IEEE Transactions on Power Apparatus and Systems*, (2), pp.625-635.

Shi, Y. and Eberhart, R.C., 1998, March. Parameter selection in particle swarm optimization. In *International Conference on Evolutionary Programming* (pp. 591-600). Springer Berlin Heidelberg.

Shi, Y. and Eberhart, R.C., 1999. Empirical study of particle swarm optimization. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on* (Vol. 3). IEEE.

Shi, H. and Li, W., 2009, December. Artificial neural networks with ant colony optimization for assessing performance of residential buildings. In *2009 International Conference on Future BioMedical Information Engineering (FBIE)* (pp. 379-382). IEEE.

Simon, H.A., 1982. *Models of bounded rationality: Empirically grounded economic reason* (Vol. 3). MIT press.

Siraj, Fadzilah, and Mansour Ali Abdoulha. *Mining Enrollment Data Using Descriptive and Predictive Approaches*. N.p.: INTECH Open Access Publisher, 2011. Print.

Staff MACD, I., 2017. *MACD Technical Indicator*. [online] Investopedia. Available at: http://www.investopedia.com/terms/m/macd.asp [Accessed 17 Apr. 2017].

Streiffert D., 1995. Multi-area economic dispatch with tie line constraints. *IEEE Trans Power Systems,* 10(4):1946-51.

Spiderfinancial. 2016. *Spectral Analysis* . [online] www.spiderfinancial.com Available at: http://www.spiderfinancial.com/support/documentation/numxl/reference-manual/spectral-analysis. [Accessed 4 August 2016]

Stackoverflow., 2016. *Using-Fourier-analysis-for-time-series-prediction* . [online] Available at: http://stackoverflow.com/questions/4479463/using-fourier-analysis-for-time-series-prediction. [Accessed 12 August 2016].

Staff, I. (2017). *Shares*. [online] Investopedia. Available at: http://www.investopedia.com/terms/s/shares.asp [Accessed 16 Apr. 2017].

Talbi, E.G., 2009. *Metaheuristics: from design to implementation* (Vol. 74). John Wiley & Sons

Tarantola, A., 2005. *Inverse problem theory and methods for model parameter estimation*. siam.

Thiem, S. and J. Lassig, J. 2011. Comparative Study of Different Approaches to Particle Swarm Optimization in *Theory and Practice, Particle Swarm Optimization: Theory, Techniques and Applications Department of Mathematics and Statistics*, pp. 127-167, The Open University, Milton Keynes, United Kingdom

Ting, T.O, Shi, Y., Cheng, S., and Lee, S.., 2012. Exponential inertia weight for particle swarm optimisation. *Advances in Swarm Intelligence*. Vol. 7331, 83-90.

Tong, H., 2012. *Threshold models in non-linear time series analysis* (Vol. 21). Springer Science & Business Media.

Trelea, I.C., 2003. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters*, *85*(6), pp.317-325.

Tsay, R.S., 2005. *Analysis of financial time series* (Vol. 543). John Wiley & Sons.

Turkedjiev, E., Busawon, K. and Angelova, M., 2013. Validation of artificial neural network model for share price. In: *UKSim2013: 15th International Conference on Modelling and Simulation*, Cambridge, UK.

Wang, C. and Shahidehpour, S.M., 1992. A decomposition approach to non-linear multi-area generation scheduling with tie-line constraints using expert systems. *IEEE Transactions on power systems*, *7*(4), pp.1409-1418.

Wang, L., Singh, C., 2009. Reserve constrained multi area environmental/economic dispatch based on particle swarm optimisation with local search. *Engineering Applications of Artificial Intelligence* 22, 298-307.

Wang, J.Z., Wang, J.J., Zhang, Z.G. and Guo, S.P., 2011. Forecasting stock indices with back propagation neural network. *Expert Systems with Applications*, *38*(11), pp.14346-14355.

Wang J, Wang J, Fang W, Niu H., 2016. Financial time series prediction using Elman recurrent random neural networks. in *Computational Intelligence and Neuroscience*. 2016(12):1-14.

Weisstein, E. W. 2016. *Galerkin Method*. [online] mathworld.wolfram.com Available at: http://mathworld.wolfram.com/GalerkinMethod.html. [Accessed 4 August 2016].

Whittaker, J. M., 1935. *Interpolatory function theory. Cambridge* : Cambridge Univ. Press.

Whitley, D., R. Beveridge, C. Graves, and K. Mathias, K., 1995. Test driving three 1995 genetic algorithms: new test functions and geometric matching. *Journal of Heuristics.* 1:77–104.

Whitley, D., Rana, S., Dzubera, J. and Mathias, K.E., 1996. Evaluating evolutionary algorithms. *Artificial intelligence*, *85*(1), pp.245-276.

Wilmott, P., Dewynne, J. and Howison, S., 1993. *Option pricing – mathematical models and computation*, Oxford Finance Press.

Wilmott, P., Howison, S. and Dewynne, J., 1996. *The mathematics of financial derivatives*. A Student Introduction. Cambridge University Press.

Wilmott, P., 1998. *Derivatives – the theory and practice of financial engineering* (University Edition), John Wiley.

Witten, I.H. and Frank, E., 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Wikipedia, 2016. *FTSE 100 Index*. [online] en.wikipedia.org Available at: http://en.wikipedia.org/wiki/FTSE_100_Index. [Accessed 9 August 2016].

Wikipedia . 2016. *Confidence interval*. [online] en.wikipedia.org Available at: http://en.wikipedia.org/wiki/Confidence_level. [Accessed 23 August 2016].

Wikipedia . 2016. *Artificial Neural Network*. [online] en.wikipedia.org Available at: http://en.wikipedia.org/wiki/Artificial_neural_network. [Accessed 17 August 2016].

Wikipedia . 2016. *Perceptron*. [online] en.wikipedia.org Available at: http://en.wikipedia.org/wiki/Perceptron. [Accessed 2 August 2016].

Wu, H.N. and Li, H.X., 2008. A Galerkin neural-network-based design of guaranteed cost control for non-linear distributed parameter systems. *IEEE Transactions on Neural Networks*, *19*(5), pp.795-807.

Xia, X. and Elaiw, A.M., 2010. Optimal dynamic economic dispatch of generation: a review. *Electric Power Systems Research*, *80*(8), pp.975-986.

Xiaohui, H., 2013. *PSO Tutorial*. [online] www.swarmintelligence.org Available at: http://www.swarmintelligence.org/tutorials.php. [Accessed 22 June 2016].

Xu, S. and Rahmat-Samii, Y., 2007. Boundary conditions in particle swarm optimization revisited. *IEEE Transactions on Antennas and Propagation*, *55*(3), pp.760-765.

Xu, R., Venayagamoorthy, G.K. and Wunsch, D.C., 2007. Modeling of gene regulatory networks with hybrid differential evolution and particle swarm optimization. *Neural Networks*, *20*(8), pp.917-927.

Yalcinoz, T. and Short, M.J., 1998. Neural networks approach for solving economic dispatch problem with transmission capacity constraints. *IEEE Transactions on Power Systems*, *13*(2), pp.307-313.

Yang, X.S., 2010. *Nature-inspired metaheuristic algorithms*. Luniver press.

Yang, X.S., 2010. *Engineering optimization: an introduction with metaheuristic applications*. John Wiley & Sons.

Yang, X.S., Cui, Z., Xiao, R., Gandomi, A.H. and Karamanoglu, M. eds., 2013. *Swarm intelligence and bio-inspired computation: theory and applications*. Newnes.

Yao, X. and Liu, Y., 1997. A new evolutionary system for evolving artificial neural networks. *IEEE transactions on neural networks*, *8*(3), pp.694-713.

Yoshida, H., Kawata, K., Fukuyama, Y., Takayama, S. and Nakanishi, Y., 2000. A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *IEEE Transactions on power systems*, *15*(4), pp.1232-1239.

Yu, J., Wang, S. and Xi, L., 2008. Evolving artificial neural networks using an improved PSO and DPSO. *Neurocomputing*, *71*(4), pp.1054-1060.

Zekic, M., 1998, September. Neural network applications in stock market predictions-a methodology analysis. In *proceedings of the 9th International Conference on Information and Intelligent Systems* (Vol. 98, pp. 255-263).

Zhong, J. 2006. *PID Controller Tuning: A Short Tutorial Mechanical Engineering*. [online] Available at: http://saba.kntu.ac.ir/eecd/pcl/download/PIDtutorial.pdf. [Accessed 27 July 2016].

Zucchi, K.C. (2017). *Lognormal and Normal Distribution*. [online] Investopedia. Available at: http://www.investopedia.com/articles/investing/102014/lognormal-and-normal-distribution.asp [Accessed 16 Apr. 2017].

Appendix A: List of Figures

Appendix B: List of Tables

Appendix C: Datasets

APPENDIX C: TABLE 1 BARC.L WORKING DATASET FROM 27/08/2012 TO 21/09/2012

Date	Open	High	Low	Close
8/27/2012	187.2	187.2	187.2	187.2
8/28/2012	186.55	191.52	186.1	188.95
8/29/2012	188.7	188.7	184.85	186.35
8/30/2012	184.3	185.66	182.44	183.5
8/31/2012	182	186.4	180.4	183.25
9/3/2012	182.3	185.55	181.6	184.3
9/4/2012	183.45	185.44	180.35	181.25
9/5/2012	180.6	183.83	178.8	181.95
9/6/2012	182.1	194.13	180.25	193.05
9/7/2012	197.3	207.25	194.05	206.4
9/10/2012	206.05	210.75	205.23	207.75
9/11/2012	206.15	214.56	205	213.5
9/12/2012	214.9	219.8	214.15	217
9/13/2012	217	219.38	215	217.95
9/14/2012	225.55	237	214.9	229.05
9/17/2012	227.4	230.34	221.63	228
9/18/2012	226.35	226.95	218.15	225.4
9/19/2012	225.15	227.15	219	225.15
9/20/2012	224	224.9	218.1	222.05
9/21/2012	223.5	235.55	219.6	223.75

APPENDIX C: TABLE 2 BARC.L CLOSING PRICE DATASET 27/08/2012 - 21/09/2012

Date	T-target	T-1	T-2	T-3	T-4	T-5
8/27/2012	187.2	187.2	191	194.15	197.05	190.9
8/28/2012	188.95	187.2	187.2	191	194.15	197.05
8/29/2012	186.35	188.95	187.2	187.2	191	194.15
8/30/2012	183.5	186.35	188.95	187.2	187.2	191
8/31/2012	183.25	183.5	186.35	188.95	187.2	187.2
9/3/2012	184.3	183.25	183.5	186.35	188.95	187.2
9/4/2012	181.25	184.3	183.25	183.5	186.35	188.95
9/5/2012	181.95	181.25	184.3	183.25	183.5	186.35
9/6/2012	193.05	181.95	181.25	184.3	183.25	183.5
9/7/2012	206.4	193.05	181.95	181.25	184.3	183.25
9/10/2012	207.75	206.4	193.05	181.95	181.25	184.3
9/11/2012	213.5	207.75	206.4	193.05	181.95	181.25
9/12/2012	217	213.5	207.75	206.4	193.05	181.95
9/13/2012	217.95	217	213.5	207.75	206.4	193.05
9/14/2012	229.05	217.95	217	213.5	207.75	206.4
9/17/2012	228	229.05	217.95	217	213.5	207.75
9/18/2012	225.4	228	229.05	217.95	217	213.5
9/19/2012	225.15	225.4	228	229.05	217.95	217
9/20/2012	222.05	225.15	225.4	228	229.05	217.95
9/21/2012	223.75	222.05	225.15	225.4	228	229.05

APPENDIX C: TABLE 3 BARC.L ARMA DATASET

Date	Target	Close-1	Close-2	Close-3	Close-4	Close-5	rand	rand-1	rand-2	ARMA(5,3)	abs(err)
8/27/2012	187.20	187.20	191.00	194.15	197.05	190.90	-0.64	-0.06	-0.90	184.82	2.38
8/28/2012	188.95	187.20	187.20	191.00	194.15	197.05	0.45	-0.64	-0.06	188.11	0.84
8/29/2012	186.35	188.95	187.20	187.20	191.00	194.15	0.26	0.45	-0.64	190.14	3.79
8/30/2012	183.50	186.35	188.95	187.20	187.20	191.00	0.08	0.26	0.45	188.06	4.56
8/31/2012	183.25	183.50	186.35	188.95	187.20	187.20	0.25	0.08	0.26	185.86	2.61
9/3/2012	184.30	183.25	183.50	186.35	188.95	187.20	-0.47	0.25	0.08	184.34	0.04
9/4/2012	181.25	184.30	183.25	183.50	186.35	188.95	-0.25	-0.47	0.25	185.37	4.12
9/5/2012	181.95	181.25	184.30	183.25	183.50	186.35	-0.91	-0.25	-0.47	180.57	1.38
9/6/2012	193.05	181.95	181.25	184.30	183.25	183.50	0.27	-0.91	-0.25	183.97	9.08
9/7/2012	206.40	193.05	181.95	181.25	184.30	183.25	0.98	0.27	-0.91	197.52	8.88
9/10/2012	207.75	206.40	193.05	181.95	181.25	184.30	-0.46	0.98	0.27	211.78	4.03
9/11/2012	213.50	207.75	206.40	193.05	181.95	181.25	0.93	-0.46	0.98	212.27	1.23
9/12/2012	217.00	213.50	207.75	206.40	193.05	181.95	0.76	0.93	-0.46	220.17	3.17
9/13/2012	217.95	217.00	213.50	207.75	206.40	193.05	0.65	0.76	0.93	219.41	1.46
9/14/2012	229.05	217.95	217.00	213.50	207.75	206.40	0.12	0.65	0.76	220.83	8.22
9/17/2012	228.00	229.05	217.95	217.00	213.50	207.75	-0.61	0.12	0.65	232.68	4.68
9/18/2012	225.40	228.00	229.05	217.95	217.00	213.50	0.25	-0.61	0.12	225.16	0.24
9/19/2012	225.15	225.40	228.00	229.05	217.95	217.00	-0.89	0.25	-0.61	225.70	0.55
9/20/2012	222.05	225.15	225.40	228.00	229.05	217.95	0.48	-0.89	0.25	223.09	1.04
9/21/2012	223.75	222.05	225.15	225.40	228.00	229.05	0.53	0.48	-0.89	219.37	4.38

APPENDIX C: TABLE 4 BARC.L SHARE PRICE IN PENCE FOR ONE MONTH JANUARY 2013

	Date	Open	High	Low	Close	^Close
1	1/1/2013	262.4	262.4	262.4	262.4	0.15792575
2	1/2/2013	272.85	277.08	269.6	275.6	0.3134944
3	1/3/2013	274.65	278.89	272.83	276	0.3182086
4	1/4/2013	275.15	278.56	273.5	276.7	0.32645846
5	1/7/2013	281	288.46	276.04	287.2	0.45020625
6	1/8/2013	285.6	295.48	282.5	287.2	0.45020625
7	1/9/2013	289	298.15	288.8	294.75	0.5391868
8	1/10/2013	293.8	298.79	291.65	294.6	0.53741897
9	1/11/2013	295.9	301.75	295.54	299.65	0.59693577
10	1/14/2013	299.55	301.3	297.9	298.9	0.58809664
11	1/15/2013	298	298.69	292	295.4	0.54684738
12	1/16/2013	293.95	296	287.25	293.4	0.52327637
13	1/17/2013	291.95	298.97	289.05	296.05	0.55450796
14	1/18/2013	296.15	299.9	294.93	297	0.56570418
15	1/21/2013	298.7	299.37	294.32	297.4	0.57041839
16	1/22/2013	298	300.05	293.94	296.05	0.55450796
17	1/23/2013	296.2	299.9	295.75	296	0.55391868
18	1/24/2013	295	300.05	294.25	300	0.6010607
19	1/25/2013	299.15	303.9	298.05	300.7	0.60931055
20	1/28/2013	300.5	307.66	297.47	305.85	0.67000589
21	1/29/2013	305	306.65	297.1	300.9	0.61166765

APPENDIX C: TABLE 5 BARC.L PROBABILITY FREQUENCY

Bin	Frequency	Probability	Normdist
249	1	0.0035	0.0026
249.8485	2	0.0070	0.0028
250.697	1	0.0035	0.0031
251.5455	4	0.0141	0.0034
252.394	1	0.0035	0.0037
253.2425	2	0.0070	0.0040
254.091	1	0.0035	0.0043
254.9395	1	0.0035	0.0046
255.788	3	0.0106	0.0050
256.6365	1	0.0035	0.0053
257.485	4	0.0141	0.0057
258.3335	1	0.0035	0.0061
259.182	1	0.0035	0.0066
260.0305	2	0.0070	0.0070
260.879	1	0.0035	0.0074
261.7275	0	0.0000	0.0079
262.576	3	0.0106	0.0084
263.4245	2	0.0070	0.0089
264.273	2	0.0070	0.0094

APPENDIX C: TABLE 6 BARC.L SHARE PRICES 13/01/2014 - 31/01/2014

Date	Open	High	Low	Close	^Close
1/13/2014	286.5	294.25	286.5	291.7	0.50324101
1/14/2014	287.6	293.15	286.15	291.75	0.50383029
1/15/2014	293.3	298.03	292.35	296.5	0.55981143
1/16/2014	297.4	298.13	287.4	290.45	0.48850913
1/17/2014	291.4	293	286.5	288.6	0.46670595
1/20/2014	284.5	286.38	282.2	282.8	0.39835003
1/21/2014	283	285.62	279.93	280.6	0.37242192
1/22/2014	281.6	283.8	277.49	278.2	0.34413671
1/23/2014	278	284.55	277.21	278.9	0.35238656
1/24/2014	279.3	280.89	270.84	272.25	0.27401296
1/27/2014	272	275.73	268.04	269.35	0.239835
1/28/2014	271.95	274.64	270.08	273.3	0.28638774
1/29/2014	281.95	284.8	268.8	274.95	0.30583382
1/30/2014	275	275.9	271.6	275.05	0.30701237
1/31/2014	274.35	274.4	267.55	272.5	0.27695934

APPENDIX C: TABLE 7 BARC.L SHARE PRICE 23/12/2013 - 10/01/2014

Date	Open	High	Low	Close	^Close
12/23/2013	259.35	265.07	259.35	264.35	0.18090748
12/24/2013	268.55	268.55	262.92	265.45	0.19387154
12/25/2013	265.45	265.45	265.45	265.45	0.19387154
12/26/2013	265.45	265.45	265.45	265.45	0.19387154
12/27/2013	268.9	271.12	268.02	269.7	0.24395993
12/30/2013	272.5	273.86	269.55	271.1	0.26045963
12/31/2013	271.55	274.3	271.05	271.95	0.27047731
1/1/2014	271.95	271.95	271.95	271.95	0.27047731
1/2/2014	273	274.65	268.35	271.05	0.25987036
1/3/2014	271.25	273.91	270.4	272.85	0.28108427
1/6/2014	271.55	278.67	271.2	277.5	0.33588686
1/7/2014	276.95	282.78	275	280.95	0.37654685
1/8/2014	282.35	285.9	281.25	283.7	0.40895698
1/9/2014	283.9	289.71	282.48	284.4	0.41720684
1/10/2014	286	287.14	282.2	283.6	0.40777843

APPENDIX C: TABLE 8 BARC.L FOR 15-DAYS FROM 09/01/2014 AND 13/01/2014

Date	Price	Price	Date
1/13/2014	291.7	284.4	1/9/2014
1/14/2014	291.75	283.6	1/10/2014
1/15/2014	296.5	291.7	1/13/2014
1/16/2014	290.45	291.75	1/14/2014
1/17/2014	288.6	296.5	1/15/2014
1/20/2014	282.8	290.45	1/16/2014
1/21/2014	280.6	288.6	1/17/2014
1/22/2014	278.2	282.8	1/20/2014
1/23/2014	278.9	280.6	1/21/2014
1/24/2014	272.25	278.2	1/22/2014
1/27/2014	269.35	278.9	1/23/2014
1/28/2014	273.3	272.25	1/24/2014
1/29/2014	274.95	269.35	1/27/2014
1/30/2014	275.05	273.3	1/28/2014
1/31/2014	272.5	274.95	1/29/2014

APPENDIX C: TABLE 9 BARC.L FOR 15-DAYS FROM 20/12/2013 AND 13/01/2014

Date	Price	Price	Date
1/13/2014	291.7	259.7	12/20/2013
1/14/2014	291.75	264.35	12/23/2013
1/15/2014	296.5	265.45	12/24/2013
1/16/2014	290.45	265.45	12/25/2013
1/17/2014	288.6	265.45	12/26/2013
1/20/2014	282.8	269.7	12/27/2013
1/21/2014	280.6	271.1	12/30/2013
1/22/2014	278.2	271.95	12/31/2013
1/23/2014	278.9	271.95	1/1/2014
1/24/2014	272.25	271.05	1/2/2014
1/27/2014	269.35	272.85	1/3/2014
1/28/2014	273.3	277.5	1/6/2014
1/29/2014	274.95	280.95	1/7/2014
1/30/2014	275.05	283.7	1/8/2014
1/31/2014	272.5	284.4	1/9/2014

APPENDIX C: TABLE 10 BARC.L PRICE 01/01/2013 - 14/01/2013

	Date	Open	High	Low	Close
1	1/1/2013	262.4	262.4	262.4	262.4
2	1/2/2013	272.85	277.08	269.6	275.6
3	1/3/2013	274.65	278.89	272.83	276
4	1/4/2013	275.15	278.56	273.5	276.7
5	1/7/2013	281	288.46	276.04	287.2
6	1/8/2013	285.6	295.48	282.5	287.2
7	1/9/2013	289	298.15	288.8	294.75
8	1/10/2013	293.8	298.79	291.65	294.6
9	1/11/2013	295.9	301.75	295.54	299.65
10	1/14/2013	299.55	301.3	297.9	298.9

APPENDIX C: TABLE 11 BARC.L DATASETS SHARE PRICES FOR DIFFERENT OVERLAP STARTING DATE

	13-Jan	10-Jan	9-Jan	8-Jan	26-Dec	24-Dec	20-Dec	18-Nov
1	291.7	283.6	284.4	283.7	265.45	265.45	259.7	251.4
2	291.75	291.7	283.6	284.4	269.7	265.45	264.35	250.8
3	296.5	291.75	291.7	283.6	271.1	265.45	265.45	252.6
4	290.45	296.5	291.75	291.7	271.95	269.7	265.45	257.1
5	288.6	290.45	296.5	291.75	271.95	271.1	265.45	256.95
6	282.8	288.6	290.45	296.5	271.05	271.95	269.7	259.1
7	280.6	282.8	288.6	290.45	272.85	271.95	271.1	260.8
8	278.2	280.6	282.8	288.6	277.5	271.05	271.95	262.5
9	278.9	278.2	280.6	282.8	280.95	272.85	271.95	265.6
10	272.25	278.9	278.2	280.6	283.7	277.5	271.05	271.7
11	269.35	272.25	278.9	278.2	284.4	280.95	272.85	270.25
12	273.3	269.35	272.25	278.9	283.6	283.7	277.5	266.2
13	274.95	273.3	269.35	272.25	291.7	284.4	280.95	262.8
14	275.05	274.95	273.3	269.35	291.75	283.6	283.7	262.35
15	272.5	275.05	274.95	273.3	296.5	291.7	284.4	265.65

APPENDIX C: TABLE 12 RANDOM SELECTED CORRELATION DATASET

	13-Jan	17-Dec	15-Nov	22-Oct	5-Aug	27-Sep
	Series	Series	Series	Series	Series	Series
	1	2	3	4	5	6
1	291.7	251.3	249.45	272.6	285.5	265.8
2	291.75	252.05	251.4	268.2	284.5	265.5
3	296.5	257.45	250.8	266.45	282.15	269.8
4	290.45	259.7	252.6	267.9	286.95	272.55
5	288.6	264.35	257.1	263.25	287.1	273
6	282.8	265.45	256.95	266.05	285	271.35
7	280.6	265.45	259.1	268.45	283.65	272.55
8	278.2	265.45	260.8	263.6	285.45	268.2
9	278.9	269.7	262.5	256.3	284.3	267.8
10	272.25	271.1	265.6	255.15	288.05	274.5
11	269.35	271.95	271.7	249	288.7	278
12	273.3	271.95	270.25	254.65	286.75	276.85
13	274.95	271.05	266.2	252.7	283	279.85
14	275.05	272.85	262.8	255.3	284.7	283.65
15	272.5	277.5	262.35	257.65	286.85	278.3

APPENDIX C: TABLE 13 TREND LINE ANN FIRST AND SECOND LAYERS

t	*S*	ANN-1	*S1*	SQRT(e^2)	ANN-2	*S2*	SQRT(e^2)
1	291.7	0.54	294.60	2.90	0.54	294.60	2.90
2	291.75	0.51	292.49	0.74	0.51	292.49	0.74
3	296.5	0.49	290.36	6.14	0.49	290.36	6.14
4	290.45	0.46	288.25	2.20	0.46	288.25	2.20
5	288.6	0.44	286.15	2.45	0.44	286.18	2.42
6	282.8	0.41	284.08	1.28	0.41	284.15	1.35
7	280.6	0.39	282.04	1.44	0.39	282.19	1.59
8	278.2	0.37	280.04	1.84	0.37	280.31	2.11
9	278.9	0.34	278.10	0.80	0.35	278.52	0.38
10	272.25	0.32	276.22	3.97	0.33	276.84	4.59
11	269.35	0.30	274.41	5.06	0.31	275.26	5.91
12	273.3	0.28	272.66	0.64	0.29	273.80	0.50
13	274.95	0.26	271.00	3.95	0.28	272.44	2.51
14	275.05	0.24	269.41	5.64	0.26	271.19	3.86
15	272.5	0.22	267.90	4.60	0.25	270.04	2.46
			sum|e|	43.65		*sum|e|*	39.63
Mean	281.1267		R^2	0.832361		R^2	0.832393

APPENDIX C: TABLE 14 CENTRED LINEAR TREND AND POLY-2 TIME SERIES

	Date	S	Linear	Centred	Poly-2	Centred
1	1/13/2014	291.7	293.4146	-1.7146	296.6652	-4.9652
2	1/14/2014	291.75	291.6592	0.0908	293.515	-1.765
3	1/15/2014	296.5	289.9038	6.5962	290.5794	5.9206
4	1/16/2014	290.45	288.1484	2.3016	287.8584	2.5916
5	1/17/2014	288.6	286.393	2.207	285.352	3.248
6	1/20/2014	282.8	284.6376	-1.8376	283.0602	-0.2602
7	1/21/2014	280.6	282.8822	-2.2822	280.983	-0.383
8	1/22/2014	278.2	281.1268	-2.9268	279.1204	-0.9204
9	1/23/2014	278.9	279.3714	-0.4714	277.4724	1.4276
10	1/24/2014	272.25	277.616	-5.366	276.039	-3.789
11	1/27/2014	269.35	275.8606	-6.5106	274.8202	-5.4702
12	1/28/2014	273.3	274.1052	-0.8052	273.816	-0.516
13	1/29/2014	274.95	272.3498	2.6002	273.0264	1.9236
14	1/30/2014	275.05	270.5944	4.4556	272.4514	2.5986
15	1/31/2014	272.5	268.839	3.661	272.091	0.409

APPENDIX C: TABLE 15 CENTRED LONG TERM SHARE PRICES TIME SERIES

	Date	*Close*	*Linear*	*Centered*
1	1/1/2013	262.4	310.0884	-47.6884
2	1/2/2013	275.6	309.9368	-34.3368
3	1/3/2013	276	309.7852	-33.7852
.....	.....	.....	.....	.....
281	1/28/2014	273.3	267.6404	5.6596
282	1/29/2014	274.95	267.4888	7.4612
283	1/30/2014	275.05	267.3372	7.7128
284	1/31/2014	272.5	267.1856	5.3144

APPENDIX C: TABLE 16 CENTRED SHORT-TERM SHARE PRICES TIME SERIES

	Date	Scntr	Linear	Centred
0	1/10/2014	14.1404	23.073	-8.9326
1	1/13/2014	22.392	21.7245	0.6675
2	1/14/2014	22.5936	20.376	2.2176
3	1/15/2014	27.4952	19.0275	8.4677
4	1/16/2014	21.5968	17.679	3.9178
5	1/17/2014	19.8984	16.3305	3.5679
6	1/20/2014	14.25	14.982	-0.732
7	1/21/2014	12.2016	13.6335	-1.4319
8	1/22/2014	9.9532	12.285	-2.3318
9	1/23/2014	10.8048	10.9365	-0.1317
10	1/24/2014	4.3064	9.588	-5.2816
11	1/27/2014	1.558	8.2395	-6.6815
12	1/28/2014	5.6596	6.891	-1.2314
13	1/29/2014	7.4612	5.5425	1.9187
14	1/30/2014	7.7128	4.194	3.5188
15	1/31/2014	5.3144	2.8455	2.4689

APPENDIX C: TABLE 17 CYCLIC COMPONENT WITH THREE HARMONIC COMPONENTS

Date	S	Long	Short	Centred	F1	F2	F3	CYCL	COMP	ERR
1/10/2014	283.6	269.4596	23.07	-8.93	-2.16	-5.20	-2.00	-9.36	283.17	0.43
1/13/2014	291.7	269.308	21.72	0.67	2.16	-6.00	2.00	-1.84	289.20	2.50
1/14/2014	291.75	269.1564	20.38	2.22	5.66	-5.20	4.00	4.47	294.00	-2.25
1/15/2014	296.5	269.0048	19.03	8.47	7.00	-3.00	2.00	6.00	294.03	2.47
1/16/2014	290.45	268.8532	17.68	3.92	5.66	0.00	-2.00	3.66	290.20	0.25
....	....	....	....	....	....	....	....	....	....	....
1/27/2014	269.35	267.792	8.24	-6.68	2.16	-3.00	-4.00	-4.84	271.19	-1.84
1/28/2014	273.3	267.6404	6.89	-1.23	5.66	-5.20	-2.00	-1.53	273.00	0.30
1/29/2014	274.95	267.4888	5.54	1.92	7.00	-6.00	2.00	3.00	276.03	-1.08
1/30/2014	275.05	267.3372	4.19	3.52	5.66	-5.20	4.00	4.47	276.00	-0.95
1/31/2014	272.5	267.1856	2.85	2.47	2.16	-3.00	2.00	1.16	271.19	1.31

APPENDIX C: TABLE 18 SHARE PRICES DATASET WITH TRAINING AND TESTING SUBSETS

Time	Price	Train	Test
1	262.4		
2	275.6		
3	276		
263	271.05		
264	272.85		
265	277.5	284.9258	
267	283.7	284.0129	
268	284.4	283.4979	
269	283.6	282.952	
270	291.7	282.3842	282
271	291.75	281.8051	281
272	296.5	281.2259	281
273	290.45	280.6582	280
274	288.6	280.1123	280
275	282.8	279.5972	279
276	280.6	279.1198	279
277	278.2	278.6844	278
278	278.9	278.2932	278
279	272.25		277
280	269.35		277
281	273.3		277
282	274.95		277
283	275.05		276
284	272.5		

APPENDIX C: TABLE 19 TRAINING AND TESTING DATASET WITH MLPN

Time	Price	Train	Test
265	277.5	283.03	
266	280.95	283.40	
267	283.7	283.70	
268	284.4	284.44	
269	283.6	286.66	
270	291.7	291.09	291.0
271	291.75	294.11	294.0
272	296.5	294.29	294.0
273	290.45	290.95	290.0
274	288.6	288.50	288.0
275	282.8	282.82	282.0
276	280.6	280.58	280.0
277	278.2	278.21	278.0
278	278.9	274.763	274.0
279	272.25	272.253	272.0
280	269.35		270.0
281	273.3		270.0
282	274.95		270.0
283	275.05		270.0
284	272.5		270.0

APPENDIX C: TABLE 20 MULTIPLE ATTRIBUTES TRAINING AND TESTING DATASET

	Date	Close	Training	Testing
1	01/01/2013	262.4		
2	02/01/2013	275.6		
...	....	....		
253	19/12/2013	257.45		
254	20/12/2013	259.7		
255	23/12/2013	264.35	264.35	
256	24/12/2013	265.45	265.45	
258	26/12/2013	265.45	265.45	
259	27/12/2013	269.7	269.7	
260	30/12/2013	271.1	271.1	271.1
261	31/12/2013	271.95	271.95	271.95
262	01/01/2014	271.95	271.95	271.95
263	02/01/2014	271.05	271.05	271.05
264	03/01/2014	272.85	272.85	272.85
265	06/01/2014	277.5	277.5	277.5
266	07/01/2014	280.95	280.95	280.95
277	22/01/2014	278.2	278.2	278.2
278	23/01/2014	278.9	278.9	278.9
279	24/01/2014	272.25	272.25	272.25
280	27/01/2014	269.35		269.35
281	28/01/2014	273.3		273.3
282	29/01/2014	274.95		274.95
283	30/01/2014	275.05		275.05
284	31/01/2014	272.5		272.5

APPENDIX C: TABLE 21 MULTIPLE ATTRIBUTES TRANSFORMED TRAINING DATASET

Att1	Att2	Att3	Att4	Att7	Att8	Att9	Att10	Att11	Att12	target	Y	Day
P(t-12)	P(t-11)	P(t-10)	P(t-9)	P(t-6)	P(t-5)	P(t-4)	P(t-3)	P(t-2)	P(t-1)	P(t)	P(t)	
264.4	265.5	265.5	265.5	272	272	271.1	272.9	277.5	281	283.7	282.5	267
265.5	265.5	265.5	269.7	272	271.1	272.9	277.5	281	283.7	284.4	280.7	268
265.5	265.5	269.7	271.1	271.1	272.9	277.5	281	283.7	284.4	283.6	283.6	269
265.5	269.7	271.1	272	272.9	277.5	281	283.7	284.4	283.6	291.7	288.9	…
269.7	271.1	272	272	277.5	281	283.7	284.4	283.6	291.7	291.8	291.7	…
271.1	272	272	271.1	281	283.7	284.4	283.6	291.7	291.8	296.5	296.5	272
272	272	271.1	272.9	283.7	284.4	283.6	291.7	291.8	296.5	290.5	290.4	…
272	271.1	272.9	277.5	284.4	283.6	291.7	291.8	296.5	290.5	288.6	288.6	…
271.1	272.9	277.5	281	283.6	291.7	291.8	296.5	290.5	288.6	282.8	282.8	…
272.9	277.5	281	283.7	291.7	291.8	296.5	290.5	288.6	282.8	280.6	280.6	…
277.5	281	283.7	284.4	291.8	296.5	290.5	288.6	282.8	280.6	278.2	278.2	277
281	283.7	284.4	283.6	296.5	290.5	288.6	282.8	280.6	278.2	278.9	276.79	278

APPENDIX C: TABLE 22 MULTIPLE ATTRIBUTES TRANSFORMED TESTING DATASET

Att1	Att2	Att3	Att4	Att7	Att8	Att9	Att10	Att11	Att12	target	Y	Day
P(t-12)	P(t-11)	P(t-10)	P(t-9)	P(t-6)	P(t-5)	P(t-4)	P(t-3)	P(t-2)	P(t-1)	P(t)	P(t)	
271.1	272	272	271.1	281	283.7	284.4	283.6	291.7	291.8	296.5	296.2	272
272	272	271.1	272.9	283.7	284.4	283.6	291.7	291.8	296.5	290.5	290.1	273
272	271.1	272.9	277.5	284.4	283.6	291.7	291.8	296.5	290.5	288.6	288.3	274
271.1	272.9	277.5	281	283.6	291.7	291.8	296.5	290.5	288.6	282.8	282.6	…
272.9	277.5	281	283.7	291.7	291.8	296.5	290.5	288.6	282.8	280.6	280.5	…
277.5	281	283.7	284.4	291.8	296.5	290.5	288.6	282.8	280.6	278.2	278.1	277
281	283.7	284.4	283.6	296.5	290.5	288.6	282.8	280.6	278.2	278.9	276.1	…
283.7	284.4	283.6	291.7	290.5	288.6	282.8	280.6	278.2	278.9	272.3	276.1	…
284.4	283.6	291.7	291.8	288.6	282.8	280.6	278.2	278.9	272.3	269.4	276.1	…
283.6	291.7	291.8	296.5	282.8	280.6	278.2	278.9	272.3	269.4	273.3	276.1	281
291.7	291.8	296.5	290.5	280.6	278.2	278.9	272.3	269.4	273.3	275	276.1	…
291.8	296.5	290.5	288.6	278.2	278.9	272.3	269.4	273.3	275	275.1	277	283

APPENDIX C: TABLE 23 FREQUENCY ANALYSIS SHARES TRADING DATASET

Date	Frequency t/N	n	Frequency trading	Y
04/01/2016	0	0	0.000	214.25
05/01/2016	0.0625	1	0.004	215.25
06/01/2016	0.125	2	0.008	211.75
08/01/2016	0.25	4	0.016	200.15
11/01/2016	0.3125	5	0.020	199.7
13/01/2016	0.4375	7	0.028	201.7
14/01/2016	-0.5	8	0.032	197.7
15/01/2016	-0.4375	9	0.036	191.8
19/01/2016	-0.3125	11	0.044	189.9
20/01/2016	-0.25	12	0.048	182.05
21/01/2016	-0.1875	13	0.052	186.15
22/01/2016	-0.125	14	0.056	190.75
25/01/2016	-0.0625	15	0.060	181.85

APPENDIX C: TABLE 24 DFT, INVERSE DFT AND REAL VALUES OF 16 SAMPLES TIME SERIES

Frequency	Y	centred	DFT(Y)	power(Y)	invDFT(Y)	Real
0.000	214.25	0.68	0	0.000	0.679	0.68
0.063	215.25	3.83	12.280-0.228i	0.589	3.834	3.83
0.125	211.75	2.49	19.087+11.911i	1.977	2.489	2.49
0.250	200.15	-4.80	-15.588+3.338i	0.993	-4.801	-4.80
0.313	199.7	-3.10	-8.964-10.167i	0.718	-3.096	-3.10
0.375	202.35	1.71	5.935-9.964i	0.526	1.708	1.71
0.438	201.7	3.21	4.020+3.225i	0.104	3.213	3.21
-0.438	191.8	-2.38	4.020-3.225i	0.104	-2.377	-2.38
-0.375	187.65	-4.37	5.935+9.964i	0.526	-4.372	-4.37
-0.313	189.9	0.03	-8.964+10.167i	0.718	3.235E-02	0.03
-0.188	186.15	0.59	-10.089+10.174i	0.802	0.591	0.59
-0.125	190.75	7.35	19.087-11.911i	1.977	7.346	7.35
-0.063	181.85	0.60	12.280+0.2289i	0.589	0.601	0.60

APPENDIX C: TABLE 25 CLOSE PRICE DATASET

Date	Open	High	Low	Close	Volume	Adj Close
1/4/2016	216.35	217	212.65	214.25	28332200	206.72
1/5/2016	214.85	217.6	212.05	215.25	20472900	207.685
1/6/2016	214.9	214.9	209.85	211.75	26042700	204.308
1/7/2016	206.6	208.85	201.6	205.55	41970400	198.325
1/8/2016	206	208.65	200.15	200.15	42733200	193.115
1/11/2016	200	203.35	199.4	199.7	45116200	192.681
1/12/2016	201	204.95	199.77	202.35	31436200	195.238
1/13/2016	204.35	205.8	200.336	201.7	41349200	194.611
1/14/2016	198.25	200.15	192.52	197.7	57104700	190.751
1/15/2016	196.45	198.85	190.97	191.8	49774600	185.059
1/18/2016	191.3	198	185.85	187.65	37826600	181.055
1/19/2016	190	193.6	187.925	189.9	35139300	183.225
1/20/2016	185.9	186	179.902	182.05	52318600	175.651
1/21/2016	182	186.6	179.987	186.15	65099400	179.607
1/22/2016	190.95	192.95	188.2	190.75	50097300	184.046
1/25/2016	191.95	192.25	181.35	181.85	48650600	175.458

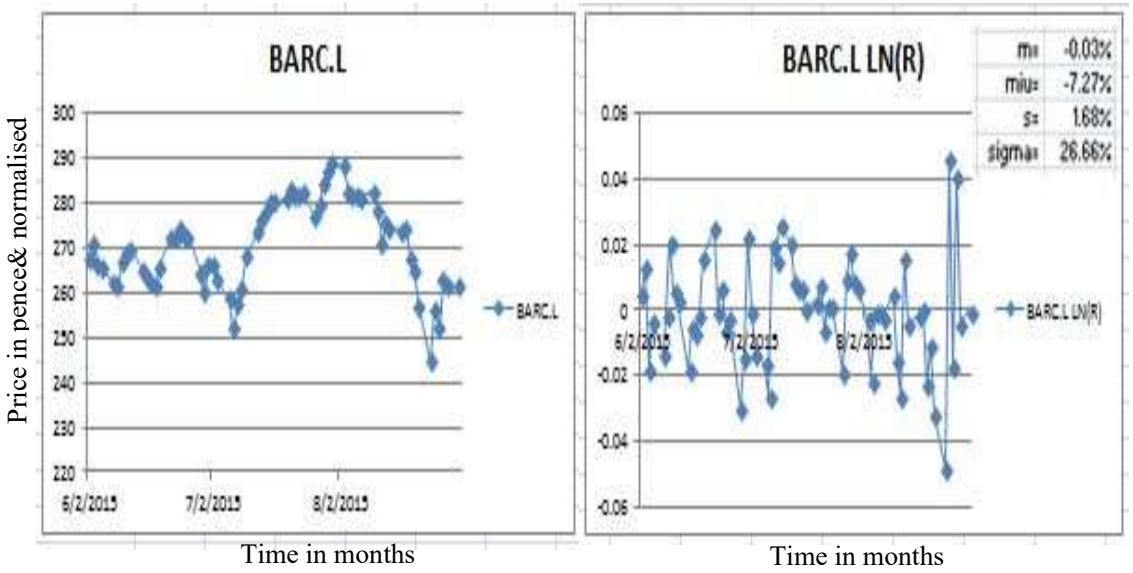
APPENDIX C: TABLE 26 NEURAL NETWORK AND PSO EXPERIMENTS DATASET

Date	Open	High	Low	Close	Volume	Adj.Close
1/4/2016	216.35	217	212.65	214.25	28332200	208.51
1/5/2016	214.85	217.6	212.05	215.25	20472900	209.483
1/6/2016	214.9	214.9	209.85	211.75	26042700	206.077
1/7/2016	206.6	208.85	201.6	205.55	41970400	200.043
1/8/2016	206	208.65	200.15	200.15	42733200	194.788
1/11/2016	200	203.35	199.4	199.7	45116200	194.35
1/12/2016	201	204.95	199.77	202.35	31436200	196.929
1/13/2016	204.35	205.8	200.336	201.7	41349200	196.296
1/14/2016	198.25	200.15	192.52	197.7	57104700	192.404
1/15/2016	196.45	198.85	190.97	191.8	49774600	186.662
1/18/2016	191.3	198	185.85	187.65	37826600	182.623
1/19/2016	190	193.6	187.925	189.9	35139300	184.813
1/20/2016	185.9	186	179.902	182.05	52318600	177.173
1/21/2016	182	186.6	179.987	186.15	65099400	181.163
1/22/2016	190.95	192.95	188.2	190.75	50097300	185.64
1/25/2016	191.95	192.25	181.35	181.85	48650600	176.978

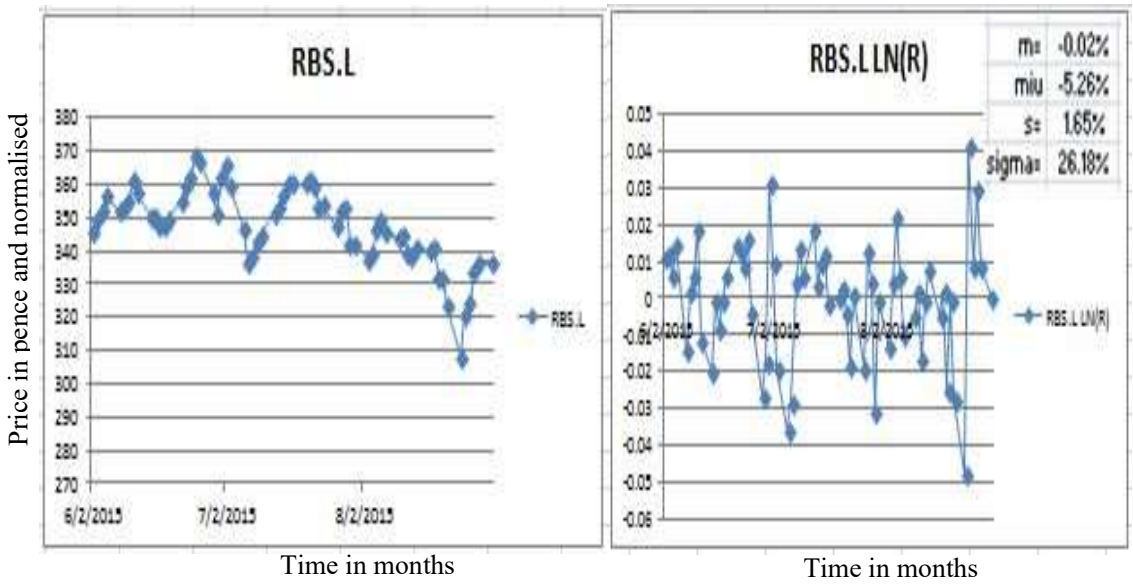
APPENDIX C: TABLE 27 UK 10 YEAR BOND YIELD HISTORICAL DATA

Date	Price	Open	High	Low	Change %
Jul 29, 2016	0.686	0.756	0.758	0.686	-3.92%
Jul 28, 2016	0.714	0.721	0.741	0.698	-3.38%
Jul 27, 2016	0.739	0.826	0.826	0.733	-10.21%
Jul 26, 2016	0.823	0.776	0.839	0.764	1.48%
Jul 25, 2016	0.811	0.824	0.840	0.806	1.50%
Jul 22, 2016	0.799	0.821	0.844	0.797	-4.31%
Jul 21, 2016	0.835	0.853	0.885	0.830	-0.12%
Jul 20, 2016	0.836	0.803	0.853	0.791	4.50%
Jul 19, 2016	0.800	0.783	0.821	0.771	-3.15%

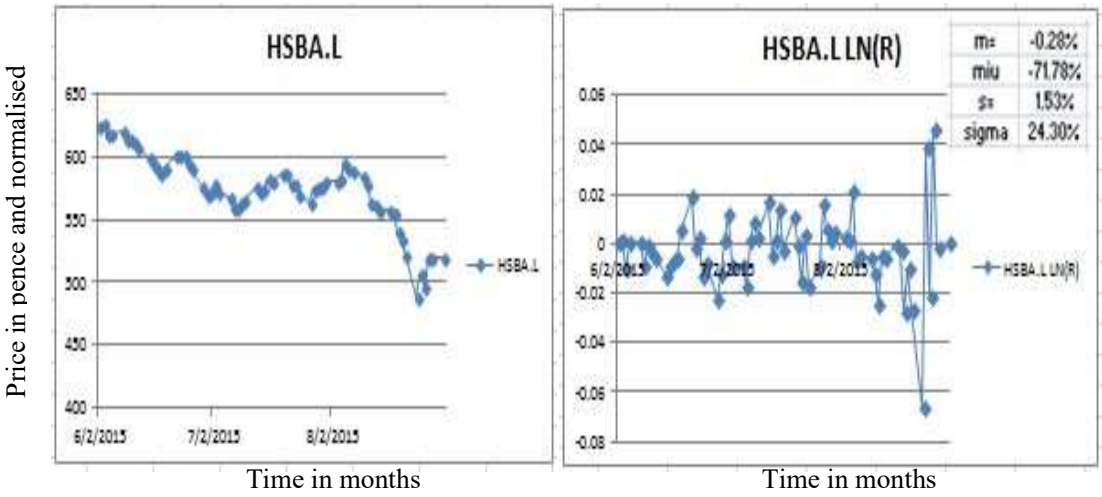
Share prices are shown for selected retail and financial sector companies.



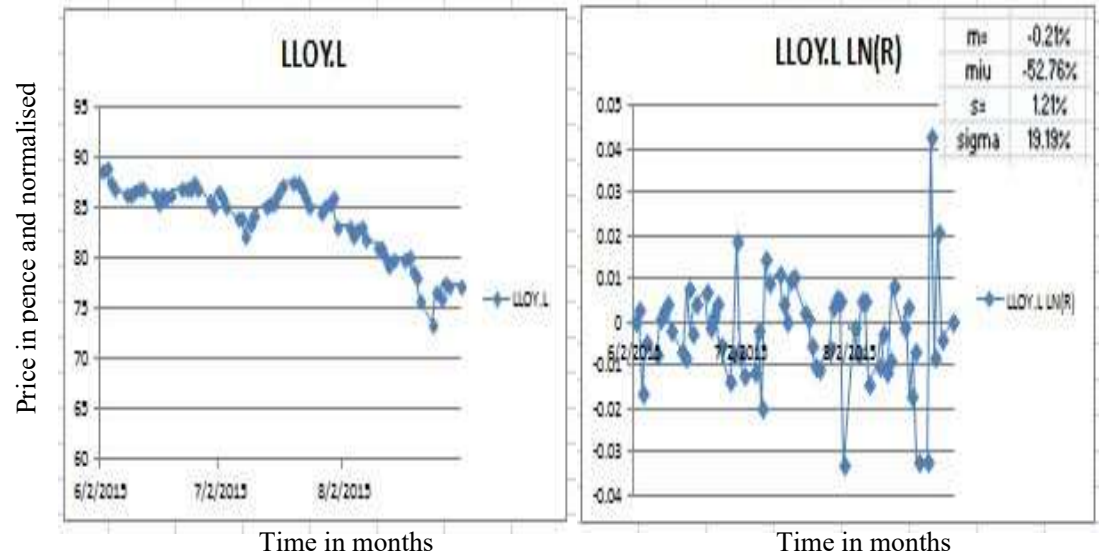
APPENDIX C: FIGURE 1 STOCHASTIC MODEL DATASET BARC.L



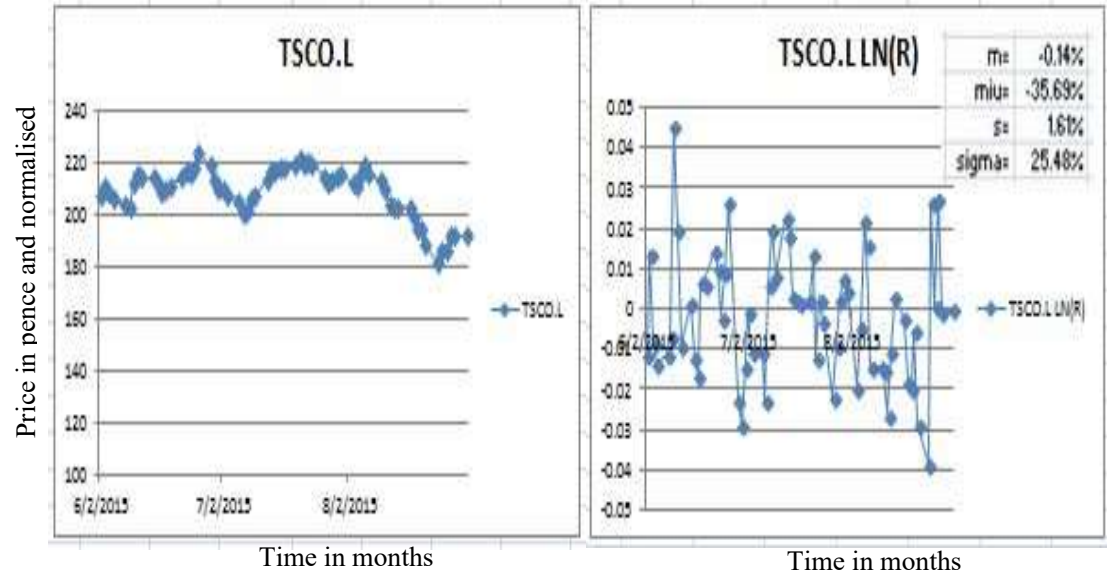
APPENDIX C: FIGURE 2 STOCHASTIC MODEL DATASET RBS.L



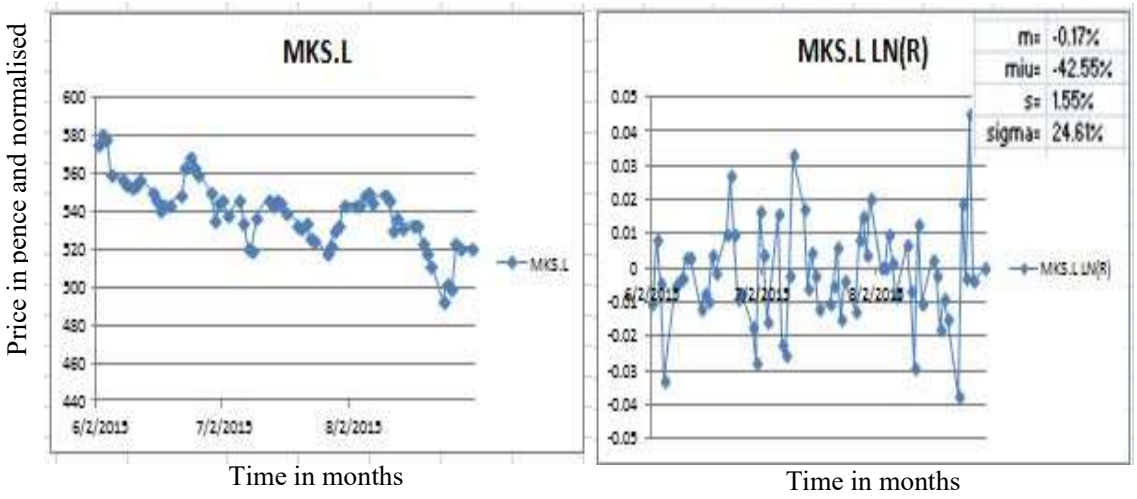
APPENDIX C: FIGURE 3 STOCHASTIC MODEL DATASET HSBA.L



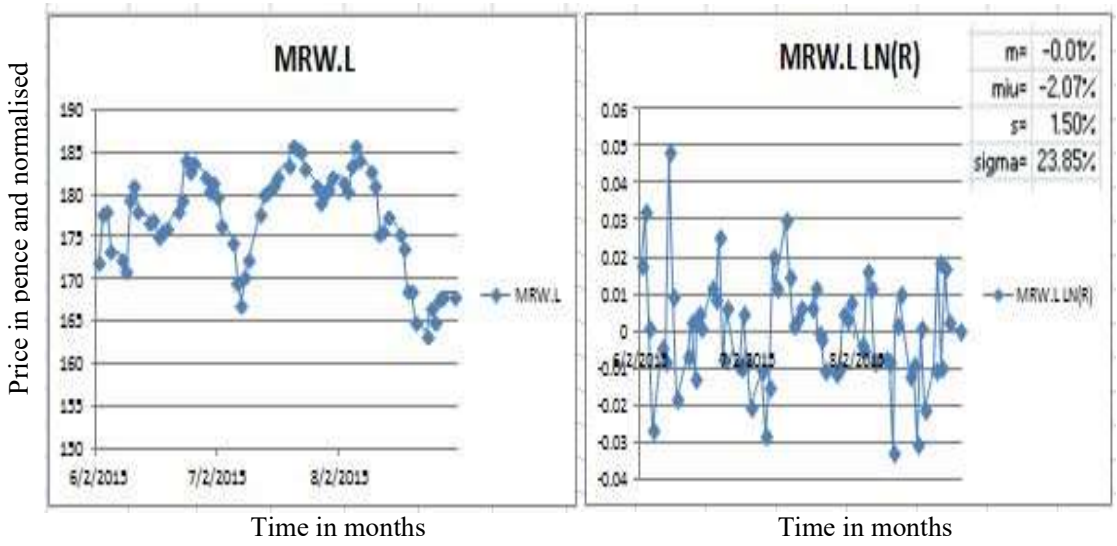
APPENDIX C: FIGURE 4 STOCHASTIC MODEL DATASET LLOY.L



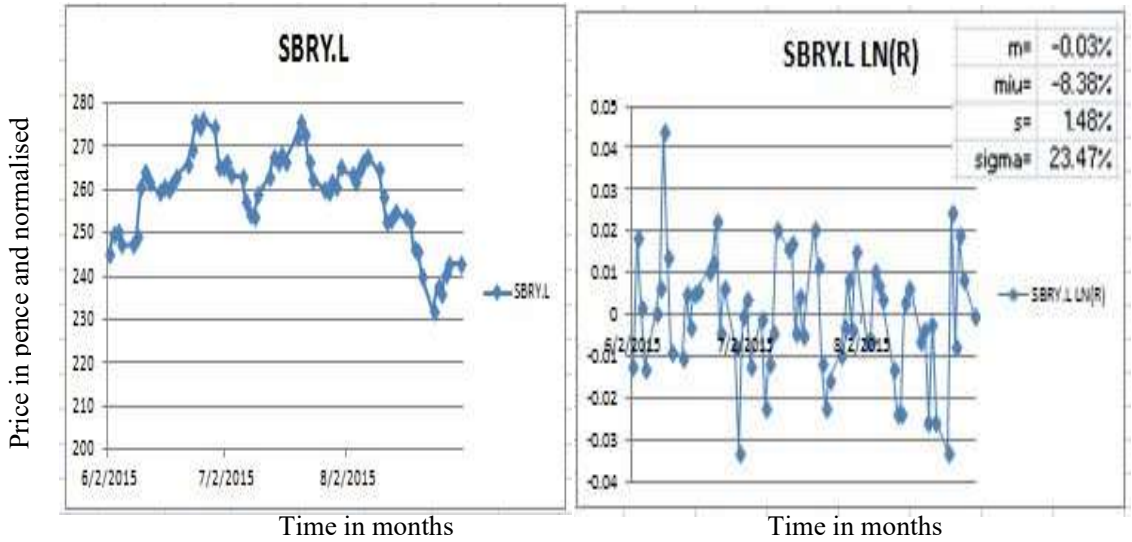
APPENDIX C: FIGURE 5 STOCHASTIC MODEL DATASET TSCO.L



APPENDIX C: FIGURE 6 STOCHASTIC MODEL DATASET MKS.L

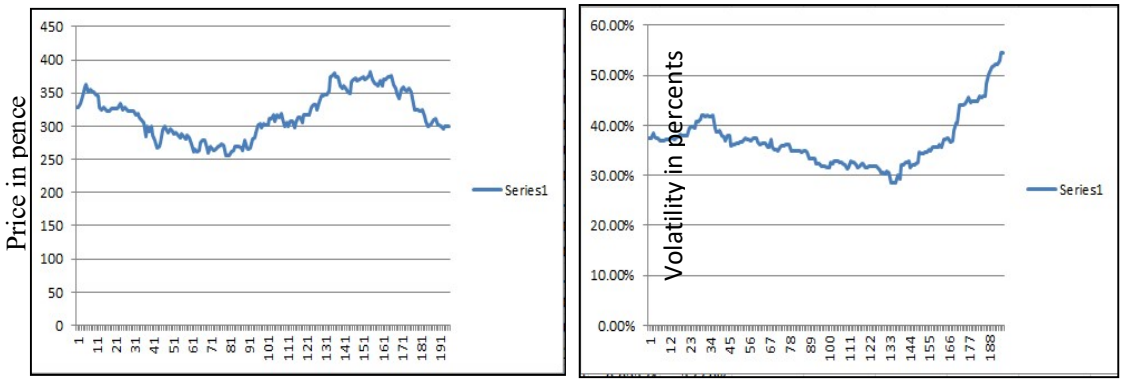


APPENDIX C: FIGURE 7 STOCHASTIC MODEL DATASET MRW.L



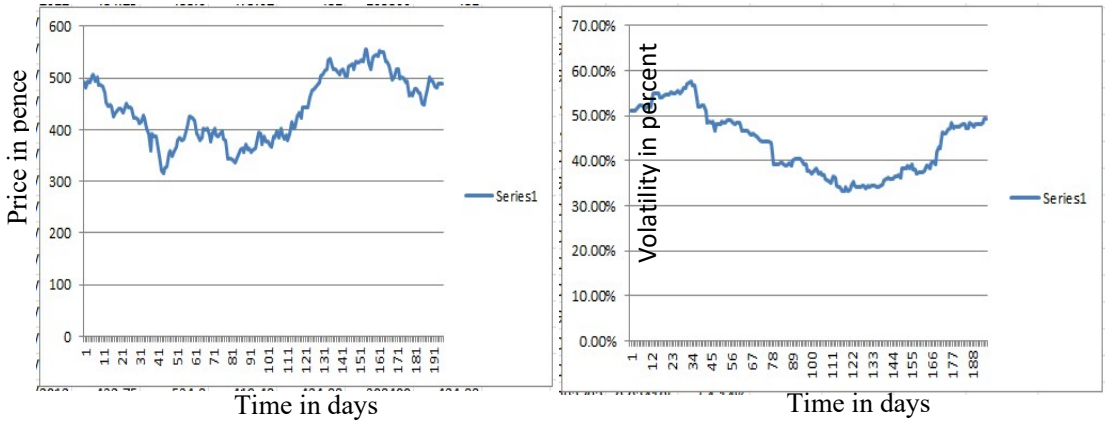
APPENDIX C: FIGURE 8 STOCHASTIC MODEL DATASET SBRY.L

Share chart 66 days (three months) Volatility 66 days (three months)



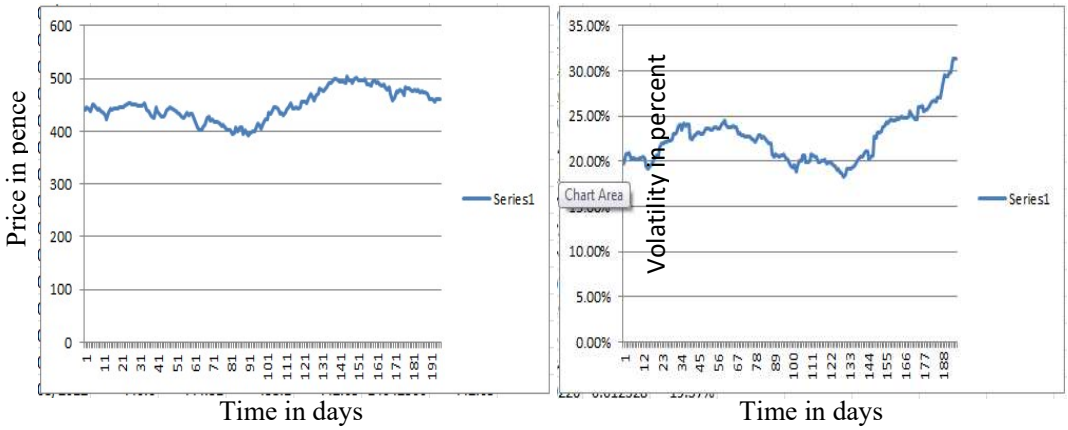
APPENDIX C: FIGURE 9 AVIVA CHART AND VOLATILITY 66 DAYS

Share chart 66 days (three months) Volatility 66 days (three months)



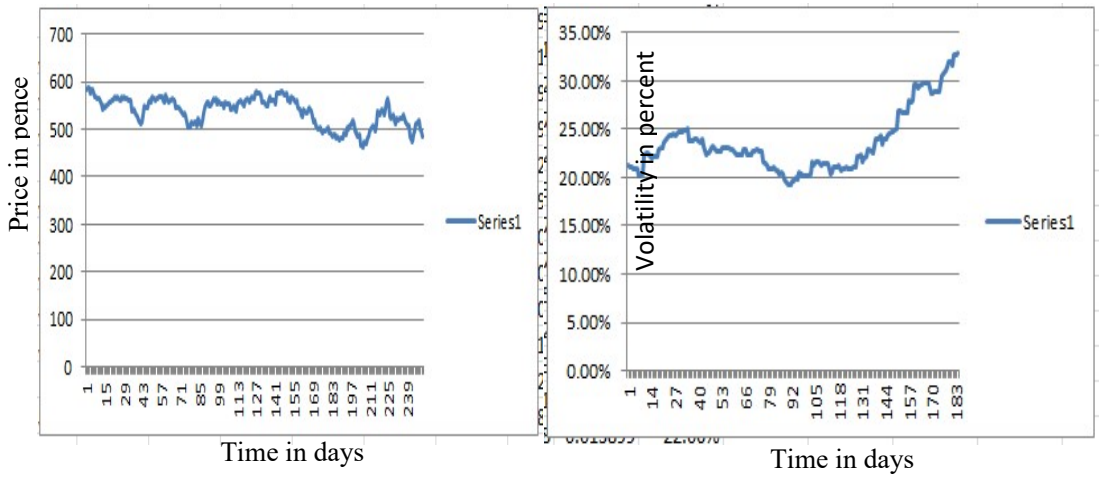
APPENDIX C: FIGURE 10 SANTANDER CHART AND VOLATILITY 66 DAYS

Share chart 66 days (three months) Volatility 66 days (three months)



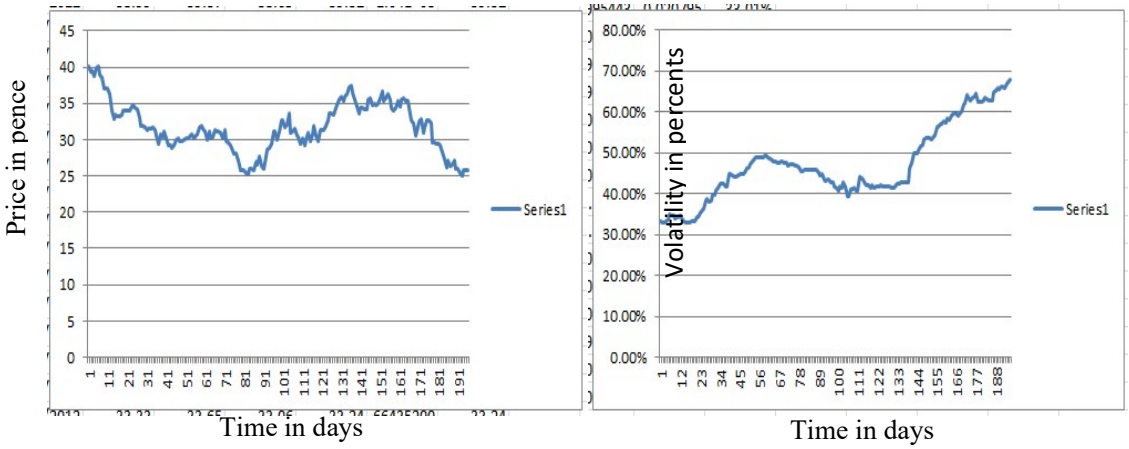
APPENDIX C: FIGURE 11 BP CHART AND VOLATILITY 66 DAYS

Share chart 66 days (three months) Volatility 66 days (three months)



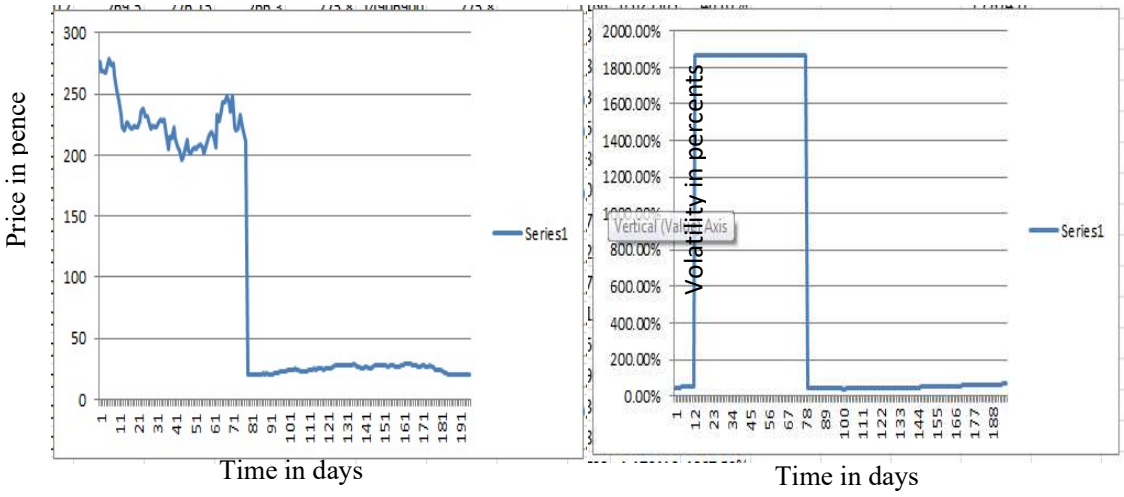
APPENDIX C: FIGURE 12 HSBA CHART AND VOLATILITY 66 DAYS

Share chart 66 days (three months) Volatility 66 days (three months)



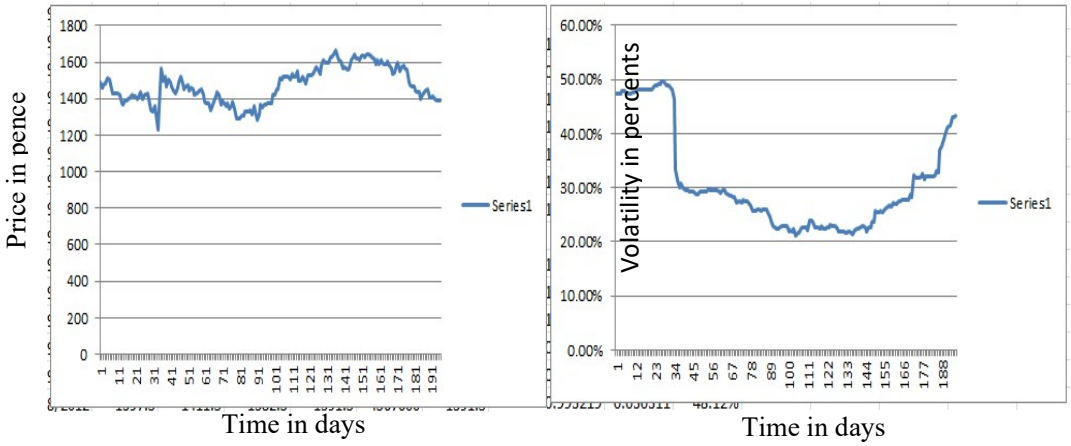
APPENDIX C: FIGURE 13 LLOYDS CHART AND VOLATILITY 66 DAYS

Share chart 66 days (three months) Volatility 66 days (three months)



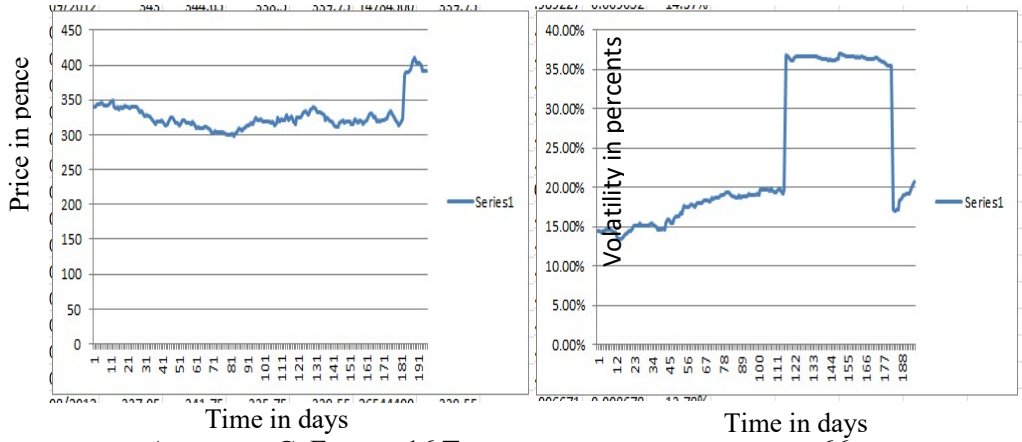
APPENDIX C: FIGURE 14 ROYAL BANK OF SCOTLAND CHART AND VOLATILITY 66 DAYS

Share chart 66 days (three months) Volatility 66 days (three months)



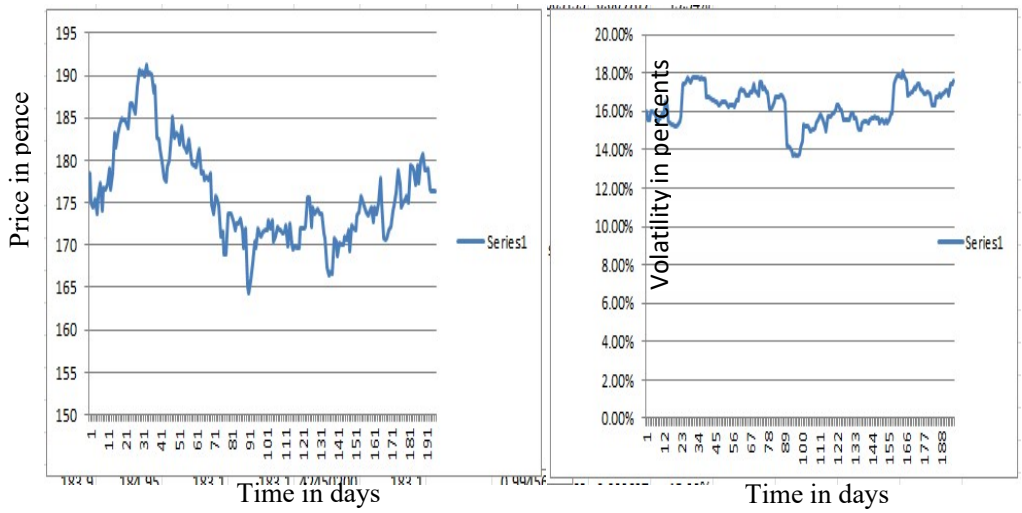
APPENDIX C: FIGURE 15 STANDARD CHARTERED CHART AND VOLATILITY 66 DAYS

Share chart 66 days (three months) Volatility 66 days (three months)



APPENDIX C: FIGURE 16 TESCO CHART AND VOLATILITY 66 DAYS

Share chart 66 days (three months) Volatility 66 days (three months)



APPENDIX C: FIGURE 17 VODAFONE CHART AND VOLATILITY 66 DAYS

Appendix D: Descriptive Statistics

APPENDIX D: TABLE 1 DESCRIPTIVE STATISTICS 13/01/2013-31/01/2013

13/01/2014/01/13 to 31/01/2014	
Close price	
Mean	281.1266667
Standard Error	2.234581385
Median	278.9
Mode	#N/A
Standard Deviation	8.654496492
Sample Variance	74.90030952
Kurtosis	-1.23404476
Skewness	0.419105607
Range	27.15
Minimum	269.35
Maximum	296.5
Sum	4216.9
Count	15
Largest	296.5
Smallest	269.35
Confidence Level/Range	0.176526718
Confidence Level(95.0%)	4.792700394

APPENDIX D: TABLE 2 BARC.L DESCRIPTIVE STATISTICS 23/12/2013 - 10/01/2014

23/12/2013 to10/01/ 2014	
Close price	
Mean	273.2967
Standard Error	1.832644
Median	271.95
Mode	265.45
Standard Deviation	7.097798
Sample Variance	50.37874
Kurtosis	-1.17451
Skewness	0.431526
Range	20.05
Minimum	264.35
Maximum	284.4
Sum	4099.45
Count	15
Largest	284.4
Smallest	264.35
Confidence Level/Range	0.196041
Confidence Level(95.0%)	3.93063

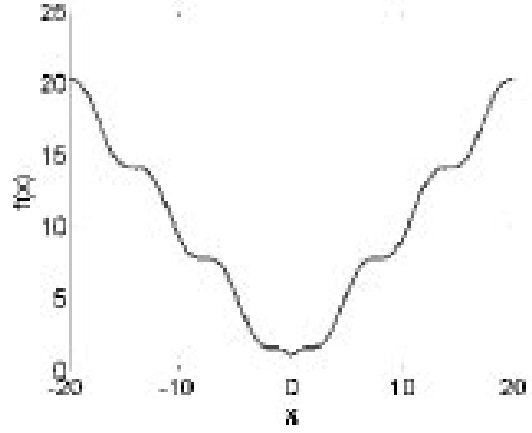
Appendix E: Test Functions

F1

$$|x| + \cos(x)$$

minimum: $f(0) = 1$

for $-\infty \le x \le \infty$

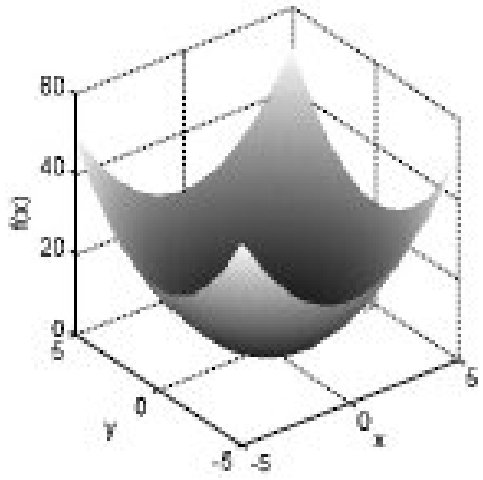


F3

$$\sum_{n=1}^{N} x_n^2$$

minimum: $f(0,0) = 1$

for $-\infty \le x \le \infty$

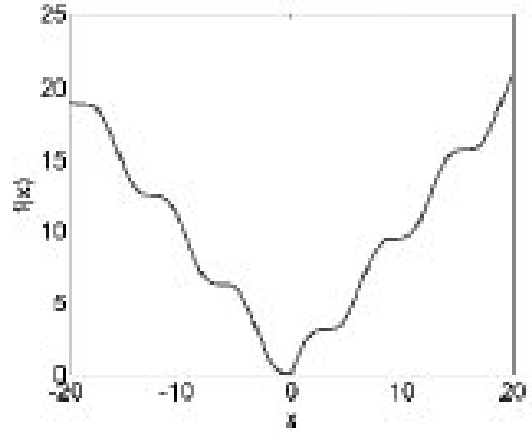


F2

$$|x| + \sin(x)$$

minimum: $f(0) = 0$

for $-\infty \le x \le \infty$

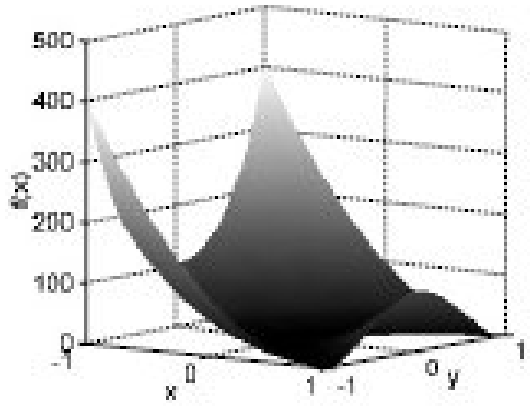


F4

$$\sum_{n=1}^{N-1} \left\{ 100\left[x_{n+1} - x_n^2\right]^2 + \left[1 - x_n\right]^2 \right\}$$

minimum: $f(1,1) = 0$

for $-\infty \le x_n \le \infty$



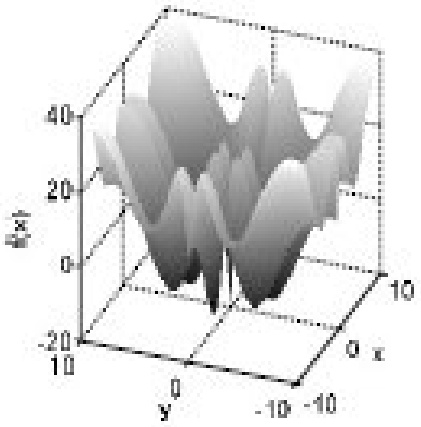
APPENDIX E: FIGURE 1 TEST FUNCTION 1-4

F5

$$\sum_{n=1}^{N}|x_n|-10\cos\left(\sqrt{|10x_n|}\right)$$

minimum: $f(x)=1$ at $x=0$

for $-\infty \le x_n \le \infty$

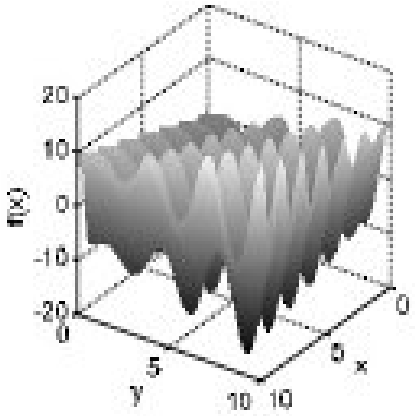


F7

$$x\sin(4x)+1.1y\sin(2y)$$

minimum: $f(0.9039,0.8668)=-18.5547$

for $0 \le x, y \le 10$

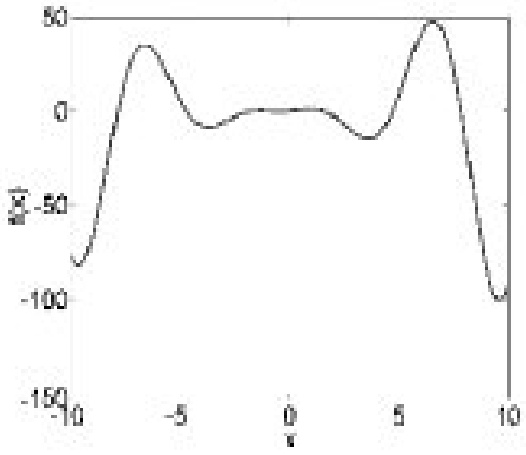


F6

$$(x^2+x)\cos(x)$$

minimum: $f(9.6204)=-100.22$

for $-10 \le x \le 10$

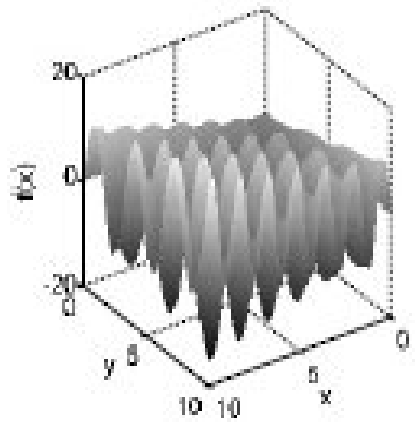


F8

$$y\sin(4x)+1.1x\sin(2y)$$

minimum: $f(0.9039,0.8668)=-18.5547$

for $0 \le x, y \le 10$



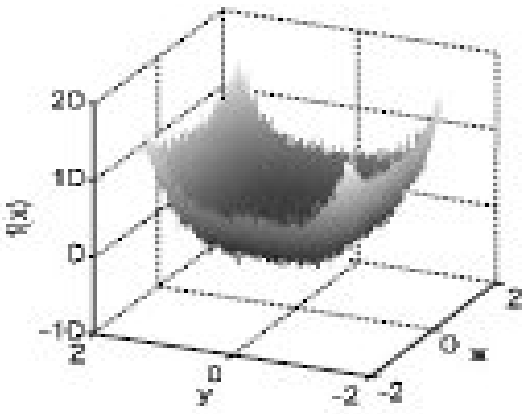
APPENDIX E: FIGURE 2 TEST FUNCTIONS 5-8

F9

$$\left[\sum_{n=1}^{N} n x_n^4\right] + N_n(0,1)$$

minimum: varies

for $-\infty \le x \le \infty$

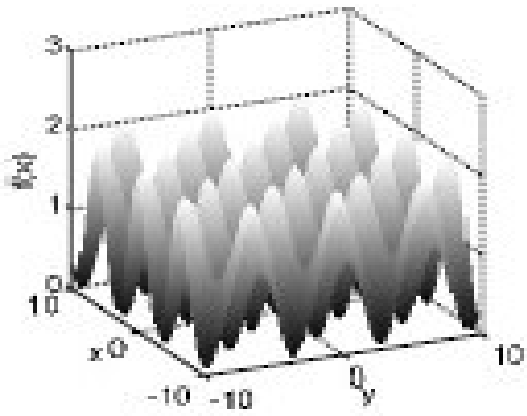


F11

$$1 + \sum_{n=1}^{N} \frac{x_n^2}{4000} - \prod_{n=1}^{N} \cos(x_n)$$

minimum: $f(0,0) = 0$

for $-\infty \le x_n \le \infty$

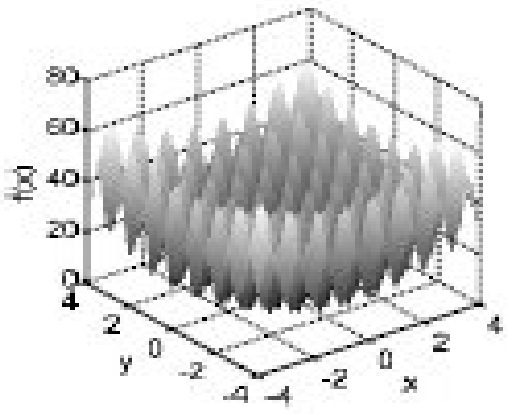


F10

$$10N + \sum_{n=1}^{N} [x_n^2 - 10\cos(2\pi x_n)]$$

minimum: $f(0,0) = 0$

for $-\infty \le x_n \le \infty$

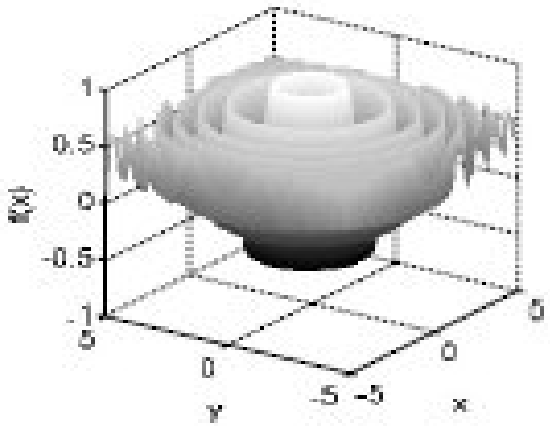


F12

$$0.5 + \frac{\sin^2\sqrt{x^2+y^2} - 0.5}{1 + 0.1(x^2+y^2)}$$

minimum: $f(1.897, 1.006) = -0.5231$

for $-\infty \le x, y \le \infty$



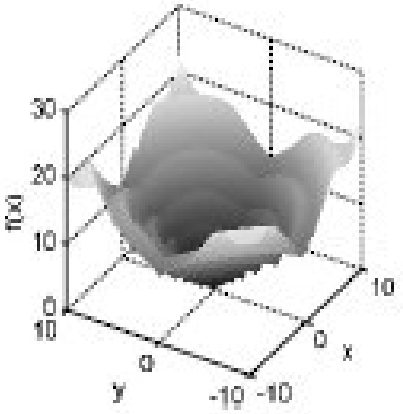
APPENDIX E: FIGURE 3 TEST FUNCTIONS 9-12

F13

$$(x^2+y^2)^{0.25}\sin\left\{30\left[(x+0.5)^2+y^2\right]^{0.1}\right\}+|x|+|y|$$

minimum: $f(0,0)=0$

for $-\infty \le x, y \le \infty$

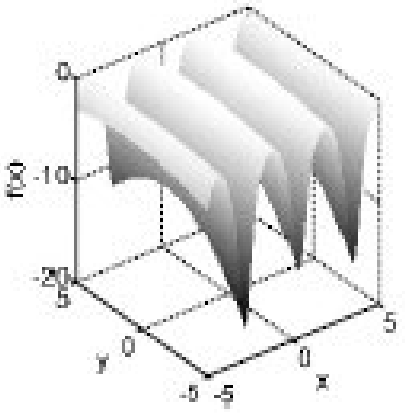


F15

$$-e^{-0.2\sqrt{x^2+y^2}}-e^{0.5(\cos 2x+\cos 2y)}$$

minimum: $f(-2.7730,-5)=-16.947$

for $-5 \le x, y \le 5$

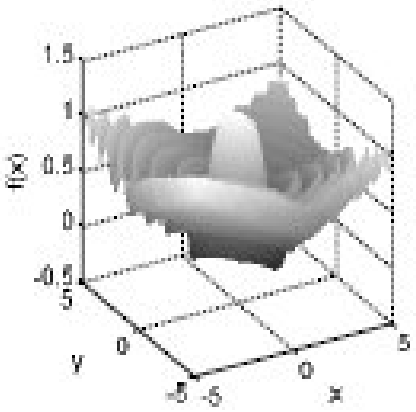


F14

$$J_0(x^2+y^2)+0.1|1-x|+0.1|1-y|$$

minimum: $f(1,1.6606)=-0.3356$

for $-\infty \le x, y \le \infty$

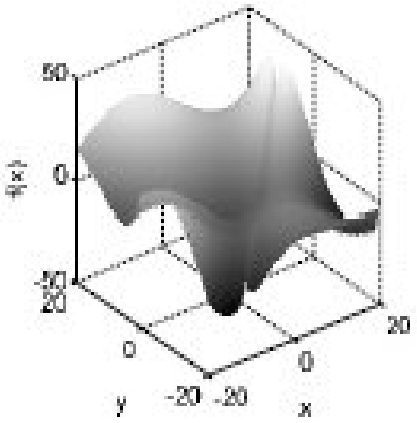


F16

$$-x\sin\left(\sqrt{|x-(y+9)|}\right)-(y+9)\sin\left(\sqrt{|y+0.5x+9|}\right)$$

minimum: $f(-14.58,-20)=-23.806$

for $-20 \le x, y \le 20$



APPENDIX E: FIGURE 4 TEST FUNCTIONS 13-16

List of Symbols

μ: mean or drift

m: annualised μ

σ: standard deviation or volatility

σ^2: variance

t: current time

t_i: sampling time *i*

T: specified period of time

τ: length of time interval in years

N: number of observations

P: significance score value

S: price of asset refers to financial share price

S_i: share price at the end of *i* th interval ($i = 0,1,2..,n$)

R^2: coefficient of determination

Δ: small change in *x* for any variable *x*

α: significance level

β: measure of risk

u: control signal

v: velocity of change

w_{ij}: neural network input weights

W_i: independent and identically distributed random variables

List of Abbreviations

ANN – Artificial Neural Network

BPNN – Back Propagation Neural Network

DFT – Digital Fourier Transform

FTSE – Financial Times Stock Exchange

FFT – Fast Fourier Transform

MLP – Multi Layer Perceptron

NN – Neural Network

PSO – Particle Swarm Optimization

IDFT – Inverse Digital Fourier Transform

PID – Proportional Integral Derivative controller

WDT – Wavelet Digital Transform

BARC.L – Barclays PLC stock exchange symbol

HSBC.L – HSBC stock exchange symbol

LLOY.L – Lloyds stock exchange symbol

MKS.L – Marks and Spencer stock exchange symbol

RBS.L – Royal Bank of Scotland symbol