Instituto Superior de Ciências do Trabalho e da Empresa

COMPARATIVE STUDY OF ARTIFICIAL NEURAL NETWORK AND BOX-JENKINS ARIMA FOR STOCK PRICE INDEXES

Ângela Marisa Roldão Cancela

Dissertation submitted in fulfilment of the requirements for the degree of Master in Data Analysis Prospecting

Advisor:
Prof. Dr. Diana Mendes, assistant professor, ISCTE Business School, Department of Quantitative Methods

September, 2008

Abstract:

The accuracy in forecasting financial time series, such as stock price indexes, has focused a great deal of attention nowadays. Conventionally, the Box-Jenkins autoregressive integrated moving average (ARIMA) models have been one of the most widely used linear models in time series forecasting. Recent research suggests that artificial neural networks (ANN) can be a promising alternative to the traditional ARIMA structure in forecasting.

This thesis aims to study the efficiency of ARIMA and ANN models for forecasting the value of four Stock Price Indexes, of four different countries (Germany, Italy, Greece and Portugal), during 2006 – 2007, using the data from preceding 15 years.

In order to reach the goal of this study, it is used the *Eviews* software that allows to find an appropriate ARIMA specification, offered also a powerful evaluation, testing and forecasting tools. In order to predict the time series is used the *Matlab* software, which provides a package that allows generating a suitable ANN model.

It is found that ANN provides forecasted results closest to the actual ones when used the logarithmic transformation. The first difference transformation is required in ARIMA but no one founding model is satisfactory. When this transformation is also used with ANN, the forecasted results are less satisfactory.

In fact, it wasn't possible to compare the efficiency of ARIMA and ANN models for forecasting the time series, due to the founding ARIMA models were not satisfactory. A possible solution would be to reduced the input period of 15 years.

Keywords: ARIMA models; Artificial Neural Networks; Backpropagation Algorithm, Stock price index forecasting;

Classifications from JEL Classification System: Time-Series Models (JEL:C22); Forecasting and Simulation (JEL:E17).

Resumo:

Actualmente a precisão na previsão de séries financeiras, tais como Índices Accionistas, têm captado uma enorme atenção. Tradicionalmente, o modelo Box-Jenkins Autorregressivos Integrados de Médias Móveis (ARIMA) é um dos modelos lineares mais utilizados na previsão de séries temporais. Pesquisas recentes têm demonstrado que as Redes Neuronais Artificiais (RNA) podem constituir uma potencial alternativa à tradicional estrutura ARIMA, na previsão.

Esta tese tem por objectivo o estudo da eficiência dos ARIMA e dos modelos de RNA na previsão de quarto índices accionistas de quatro diferentes países (Alemanha, Itália, Grécia e Portugal), desde 2006 a 2007, considerando os 15 anos antecedentes.

De modo a atingir este objectivo, foram utilizados dois *softwares*. Para determinar uma especificação apropriada para os modelos ARIMA foi utilizado o *software Eviews* que dispõe, também, de ferramentas poderosas para avaliar e testar os modelos, possibilitando ainda a previsão através dos mesmos. De forma a encontrar modelos RNA apropriados, para prever as séries em estudo, foi utilizado o *software Matlab.*

As RNA forneceram uma boa precisão na previsão das quatro séries logaritmizadas. Uma vez que os modelos ARIMA requerem estacionaridade das séries, foram utilizadas as séries das primeiras diferenças, no entanto não foi encontrado nenhum modelo que pudesse fornecer uma previsão aceitável. Considerando as séries temporais diferenciadas nas RNA, os resultados da previsão foram menos satisfatórios.

De facto, não foi possível comparar a eficiência dos modelos na previsão dos índices, uma vez que os modelos ARIMA encontrados não foram satisfatórios. Uma hipótese, na tentativa de encontrar modelos satisfatórios seria reduzir o intervalo de 15 anos de *input*.

Palavras-chave: Modelos ARIMA; Redes Neuronais Artificiais; Algoritmo Backpropagation, Previsão de Índices Accionistas;

Classificações do JEL Classification System: Time-Series Models (JEL:C22); Forecasting and Simulation (JEL:E17).

Sumário

A previsão é uma ferramenta preciosa na definição de estratégia nos mercados financeiros e, por consequência, na previsão de índices accionistas. Uma boa previsão permite a tomada de decisões mais ponderadas e justificadas, possibilitando que o investidor individual possa actuar sobre este tipo de mercados. Para o investidor colectivo a utilização desta ferramenta é ainda mais notável, uma vez que, em muitos casos, para além da gestão do próprio capital, permite também garantir as responsabilidades assumidas perante o cliente, como acontece no caso das seguradoras e bancos.

No entanto, dada a complexidade de alguns processos que intervêm nas estratégias definidas, tal como a inflação, é difícil fazer previsão em tempo real. Por esta razão, são, muitas vezes, utilizados modelos lineares.

O modelo Box-Jenkins Autorregressivos Integrados de Médias Móveis (ARIMA) é um dos modelos lineares mais utilizados na previsão de séries temporais, existindo vários artigos sobre a previsão através desses mesmos modelos. Contudo, pesquisas recentes, têm mostrado uma boa performance na previsão através das Redes Neuronais Artificiais (RNA).

Com o intuito de comparar a performance na previsão de índices accionistas, através destes dois modelos, foram utilizados quatro índices accionistas, de quatro diferentes países – Portugal, Grécia, Alemanha e Itália, integrando estes dois últimos o G8. Para tal, considerou-se um histórico de 15 anos a fim de prever o valor dos índices entre 2006 e 2007.

Para determinar uma especificação apropriada para os modelos ARIMA foi utilizado o *software Eviews* que dispõe, também, de ferramentas poderosas para avaliar e testar os modelos, possibilitando ainda a previsão através dos mesmos. De forma a encontrar modelos RNA apropriados, para prever as séries em estudo, foi utilizado o *software Matlab.*

Os resultados obtidos através dos dois modelos foram muito similares, quando aplicados aos diferentes índices. Verificou-se uma boa performance na previsão dos quatro índices entre

2006 e 2007, utilizando os modelos RNA. Todavia, não foi possível comparar os modelos, uma vez que os modelos ARIMA encontrados, não foram satisfatórios.

A estacionaridade das séries é requerida pelos ARIMA, no entanto, no período considerado como *input*, nenhuma das séries era estacionária, pelo que se utilizaram as séries das primeiras diferenças. Apesar da diferenciação, não foi possível obter resultados satisfatórios com os modelos ARIMA encontrados. Resultados similares a estes foram obtidos nas RNA, quando consideradas como *input* cada uma das quatro séries diferenciadas.

É possível que a existência de perda de informação valiosa, no momento da diferenciação das séries em estudo, torne os modelos ARIMA pouco satisfatórios, uma vez que, quando se consideram as séries originais se obtêm resultados satisfatórios nas RNA. Uma hipótese, na tentativa de encontrar modelos satisfatórios seria reduzir o intervalo de 15 anos de *input*.

List of figures:

List of tables:

Contents:

INTRODUCTION

Forecasting future events based on knowledge given in current data has fascinated people throughout history and, consequently, several techniques have been developed to face this problem of predicting the future behaviour of a particular series of events.

Forecasting is a key activity for strategy makers. Given the complexity of the fundamental processes strategy objectives, such as inflation, stock market returns or stock indices, and the difficulty of forecasting in real-time, choice is often taken to simple models. The most well known and widely used methods are based on linear probabilistic models using the autoregressive (AR), the moving average (MA), the autoregressive/moving average (ARMA) and the integrated ARMA models for nonstationary time series.

These models are usually estimated either by least squares or maximum likelihood method. They are quite popular since the procedure involved are easy to understand, and the results are also easy to interpret. However, ordinary least squares methodology is employed to obtain parameters in a simple linear model only. This technique becomes irrelevant when dealing with a complex or a nonlinear model. Besides, strict assumptions concerning the specification of the model and the estimation of parameters have to be satisfied to this method to produce meaningful results.

Recent studies showed that a considerable number of economic relationships are either nonlinear in the parameters or even chaotic. Of course, linear regression analysis cannot be applied with success in these cases. Nonlinear least squares methods seem to be the most used in order to obtain the parameters in nonlinear models. However, one cannot derive a standard formula for parameters in these models. The computational procedures are fairly complex even though it is based on the same principles as in the ordinary least squares.

In last few years some other techniques, such as genetic algorithms and neural networks, have been introduced. They have the ability to approximate nonlinear functions, and parameters are updated adaptively.

The interest in Artificial Neural Networks has increased tremendously lately. Artificial Neural Networks models are applied in many different scientific fields because of their storage and learning capabilities. They provide new ways to solve difficult problems and they are extremely successful in real life situations. There are five main areas in which Artificial Neural Networks models are used: control system, facial and handwriting recognition, medical diagnosis, classification and forecasting.

Trading in stock market indices has gained an extraordinary popularity in major financial markets around the world. The increasing diversity of financial index related instruments, along with the economic growth enjoyed in last few years, has expanded the dimension of global investment opportunity either for individual investors or institutional investors. These index vehicles provide effective means for investors to hedge against potential market risks. They, also, create new profit making opportunities for market speculators and arbitrageurs. Thus, being able to accurately forecast a stock market index has profound implications and significance to researches and practitioners. Therefore we can imagine forecasting as an enormous relevance when we are talking about predict the direction/sign of stock index movement.

In recent years, there has been a growing number of studies looking at the direction or trend of movements of various kinds of financial instruments. Such as Maberly (1986) that explores the relationship between the direction of intraday price change; Wu and Zhang (1997) investigate the predictability of the direction of change in future spot exchange rate; 0'Conner, *et al.* (1997) conduct a laboratory-based experiment and conclude that individuals show different tendencies and behaviours for upward and downward series. This further demonstrates the usefulness of forecasting the direction of change in price level, that is, the importance of being able to classify the future return as gain or a loss.

There are also some studies providing a comparative evaluation of different classification techniques regarding their ability to predict the sign of the index return Such as Leung M., *et* *al.* (2000) whom evaluate the efficacy of several multivariate classification techniques relative to a group of level estimations approaches. They also conduct time series comparisons between the two types of models on the basis of forecast performance and investment return.

The aim of this study is to evaluate the performance of a classical (ARIMA) and an artificial intelligence (ANN) forecasting techniques for four financial time series of four different countries. The studied stock price indexes, belongs to four different countries, Germany and Italy, which are part of the G8 group and Portugal and Greece. Theoretically, both techniques are very similar in that they attempt to discover the appropriate internal representation of the time series data.

In order to reach the goal of this study two softwares will be used. The *Eviews* software will able to find an appropriated ARIMA model in order to predict each one of the four time series, while *Matlab* software will allow to generate the ANN model to each one of the same time series.

This thesis is divided into three Chapters. An overview of the essential features and some technical characteristics of the proposed methodologies are firstly approach on Chapter 1, mentioning its applicability to similar data. An extensive description of how to find an adequate ARIMA model is also done in this chapter. Similarly is discussed basic theory underlying the construction of Neural Networks architecture and shown how to approximate the system with backpropagation technique.

Chapter 2 is divided in three parts. On the first one is done a brief description of the data as well as show some descriptive statistics. The second and the third parts are divided into four sections, doing the selection of an ARIMA and an ANN model, respectively, for each stock price index.

In Chapter 3 are discussed the results obtained in Chapter 2, as well as, it is conclude about this thesis with a perspective on what has been achieved and identifies some prospective topics for further research.

Chapter 1

1 – LITERATURE REVIEW

1.1 – ARIMA MODELS

Early efforts to study time series, particularly in the 19th century, were generally characterized by the idea of a deterministic world. Yule (1927) launched the notion of stochasticity in time series by assuming that every time series can be regarded as the realization of a stochastic process. Based on this simple idea, a number of time series methods have been developed since then.

Researchers such as Slutsky (1937), Walker (1931), Yaglom (1955), and Yule (1927) first formulated the concept of autoregressive (AR) and moving average (MA) models. A considerable body of literature has appeared in the area of time series, dealing with parameter estimation, identification, model checking and forecasting.

Box and Jenkins had integrated the existing knowledge, in 1970, with the book “Time Series Analysis: Forecasting and Control” which has had an enormous impact on the theory and practice of modern time series analysis and forecasting. They also developed a coherent, versatile three-stage iterative cycle for time series identification, estimation and validation, known as the Box–Jenkins approach.

With the introduction of specialized software, it was observed an explosion in the use of autoregressive integrated moving average (ARIMA) models and their extensions in many areas of science. Several papers were written about forecasting discrete time series processes through univariate ARIMA models, transfer function (dynamic regression) models, and multivariate (vector) ARIMA. Often these studies were of empirical nature, using one or more benchmark methods/models as a comparison. They have also been the key to new developments, see for instance Gooijer (2006).

1.1.1 – ARIMA models – Univariate

Although Box-Jenkins methodology shown that various models could, between them, mimic the behaviour of diverse types of series, there was no algorithm to specify a model uniquely. Therefore many techniques and methods have been suggested in order to increase mathematical rigour to the search process of an ARMA model, including Akaike's information criterion (AIC), Akaike's final prediction error (FPE), and the Bayes information criterion (BIC). Often these criteria come down to minimizing (in-sample) one-step-ahead forecast errors, with a penalty term for overfitting.

There are some methods for estimating the parameters of an ARMA model such as the least squares estimator. Although these methods are equivalent asymptotically, in the sense that estimates tend to the same normal distribution, there are large differences in finite sample properties, which may influence forecasts. In order to minimize the effect of parameter estimation errors on the probability limits of the forecasts it is recommended the use of full maximum likelihood.

More recently, Kim (2003) considered parameter estimation and forecasting of AR models in small samples. He found that (bootstrap) bias-corrected parameter estimators produce more accurate forecasts than the least squares estimator. Landsman and Damodaran (1989) presented evidence that the James-Stein ARIMA parameter estimator improves forecast accuracy relative to other methods, under Mean Square Error (MSE) loss criterion.

If a time series is known to follow a univariate ARIMA model, forecasts using disaggregated observations are, in terms of MSE, at least as good as forecasts using aggregated observations. However, in practical applications, there are other factors to be considered, such as missing values in disaggregated series or outliers when the ARIMA model parameters are estimated. When the model is stationary, Hotta and Cardoso Neto (1993) showed that the loss of efficiency using aggregated data is not large, even if the model is not known. Thus, prediction could be done either by disaggregated or aggregated models.

Usual univariate ARIMA modelling has been shown to produce one-step-ahead forecasts as accurate as those produced by competent modellers (Hill and Fildes, 1984; Libert, 1984; Poulos, *et al.*, 1987).

Rather than adopting a single AR model for all forecast horizons, Kang (2003) empirically investigated the case of using a multi-step-ahead forecasting AR model selected separately for each horizon. The forecasting performance of the multi-step-ahead procedure appears to depend on, among other things, optimal order selection criteria, forecast periods, forecast horizons, and the time series to be forecast.

1.1.2 – ARIMA models – Multivariate

A multivariate generalization of the univariate ARIMA model is the vector ARIMA (VARIMA) model. The population characteristics of VARMA processes appear to have been first derived by Quenouille (1957). VARIMA models can accommodate assumptions on exogeneity and on contemporaneous relationships, they offered new challenges to forecasters and policymakers.
Vector autoregressions (VARs) constitute a special case of the more general class of VARMA models. In essence, a VAR model is a fairly unrestricted (flexible) approximation to the reduced form of a wide variety of dynamic econometric models. VAR models can be specified in a number of ways. In general, VAR models tend to suffer from "overfitting" with too many free insignificant parameters. As a result, these models can provide poor out of sample forecasts, even though within-sample fitting is good; see, e.g., Liu, *et al.* (1994) and Simkins (1995). Instead of restricting some of the parameters in the usual way, Litterman

(1986) and others imposed a prior distribution on the parameters, expressing the belief that many economic variables behave like a random walk.
The Engle and Granger (1987) concept of cointegration has raised various interesting questions regarding the forecasting ability of error correction models (ECMs) over unrestricted VARs.

1.2 – NEURAL NETWORKS MODELS

The exceptional characteristic of the human brain it is the ability to learn from the past, which is facilitated by the complex system of sending and receiving electrical impulses among neurons. These facts has fascinated numerous researchers and led to the creation of a cognitive science, also known as artificial intelligence. The construction of a network, known as an artificial neural network (ANN) simulates brain characteristics. Neural derives from neuron, and artificial from the fact that it is not biological. Unlike the brain, the ANN performs discrete operations, which are made possible with electronic computers' ability to swiftly perform complex operations.

In 1943, McCulloch and Pitts developed the first computing machines that intend to simulate the structure of the biological nervous system and could perform logic functions through learning. The output resulting from this network was a combination of logic functions, which were used to transmit information from one neuron to another. This eventually led to the development of the binary probit model. According to this model, the neural unit could either swhich on or off depending on whether the function was activated or not. A threshold determines the activation of the system: if the input is greater than the threshold, then the neuron is activated and equal to 1, otherwise it is 0.

Donald Hebb (1949) introduced the first learning law. This learning law, also known as Hebb learning rule, is based on simultaneous combinations of neurons capable of strengthening the connection between them.

The work of McCulloch and Pitts influenced another researchers such as, Rosenblatt (1962), that developed advanced models that had ability to learn, of which the best known is the "perceptron". It is a single feedforward network. The output obtained from this single layer is the weighted sum of different inputs.

A significant advance in Neural Networks came in 1967 with the introduction of the smooth sigmoid function as activation function by Cowan. This function has the ability to approximate any nonlinear function. Instead of switching from off to on like the perceptron, the activation function assists the output function to turn on gradually as it is activated. In 1974, Werbos published the "backpropagation" learning method. The backpropagation technique enables the determination of parameter values for which the error is minimized. Prior to the introduction of back-propagation method, it was difficult to determine multiple parameter values.

Since the 1940s, Artificial Neural Network (ANN) have been used in various applications in engineering. As artificial intelligence have improved, they also began to be used in the solution of medical, military and astronomical problems. Recently, ANN have been regularly applied to the research area of finance, such as stock market prediction, and bankruptcy perdition of economic agents. But the largest application of neural network in economics and finance is found in the area of forecasting time series, due to their ability to classify a set of observations into different categories, and their forecasting aptitude.

1.3 – PROPERTIES OF FINANCIAL TIME SERIES

The predictability of most common financial time series such as stock prices is a controversial issue and has been questioned in scope of the efficient market hypothesis (EMH). The EMH states that the current market price reflects the assimilation of all the information available. This means that given the information, no prediction of future changes in the price can be made. As new information enters the system the unbalanced state is immediately discovered and quickly eliminated by a correct change in market price.

In recent years, the EMH became a controversial issue due to many reasons. On one side, it was shown in some studies that excess profits can be achieved using only past price data, on the other side it is very difficult to test the strong form due to lack of data.

Another reason that helps the predictability being a complex task are some statistical properties of the time series that are different at different points in time. For example the volatility (standard deviation) is different during different periods. There are periods, where the index values fluctuates greatly on a single day, and more calm periods. And, of course, financial time series are influenced by business cycle.

We can't forgotten that financial time series are usually very noisy, i.e., there is a large amount of random (unpredictable) day-to-day variations. The events, such as interest rate changes, announcements of macroeconomic news as well as political events are random and unpredictable and contribute to the noise in the time series. The noisy nature of financial time series makes it difficult to distinguish a good prediction algorithm from a bad one.

However, forecasting is a key activity to define strategies in various fields, nowadays is crucial when trading in stock market indices. It has broadened the dimension of global investment opportunity to both individual and institutional investors, as provide an effective means for investors to hedge against potential market risks and create new profit making opportunities for market speculators and arbitrageurs.

In many forecasting situation, more than one component is needed to capture the dynamics in a series to be forecast. Usually, a time series could be defined as:

$$y_t = T_t + S_t + C_t + \varepsilon_t \qquad (1.3.1)$$

where T is the *trend* component, S is the *seasonal* component, C is the *cyclical* component and ε is the *residue* component, which is white noise. Typically, it is assumed that each component is uncorrelated with all other components at all leads and lags.

1.4 – MODELS USED

Traditionally, the autoregressive integrated moving average (ARIMA) model has been one of the most widely used models in time series prediction. Recent research activities in forecasting with ANN suggest that ANN can be a promising alternative to the traditional ARIMA structure. These linear models and ANN are often compared with mixed conclusions in terms of the superiority in forecasting performance.

The ARIMA time series models allow Y_t, random variable, to be explained by past, or lagged values of Y itself and stochastic error terms.

An ANN consists of several layers of nodes: an *input* layer, an *output* and zero or more *hidden* layers. The input layer consists of one node for each independent variable, while the output layer consists of one or more nodes that correspond to the final decision. The hidden layers lie in between, and each consists of several nodes that receive inputs from nodes in layer below them and feed their outputs to nodes in layers above.

There are several papers dealing with the comparison of different forecasting methods. In what follows, it is briefly review some of them.

In 2000, Leung *et al.* evaluated the efficacy of several multivariate classification techniques relative to a group of level estimation approaches, conducted time series comparisons between the two types of models and the basis of forecast performance and investment return. The classification models used, by them, to predict direction/sign of stock index movement based on probability, include linear discriminant analysis, probit, logit and probabilistic neural network. Their empirical experimentation suggests that probabilistic neural network was the best performer among the forecasting models evaluated in their study.

Ho *et al.* (2002), drew a comparative study between ANN and Box-Jenkins ARIMA modelling in time series prediction. The neural network architectures evaluated by them, in this study, were the multiplayer feedforward network and the recurrent neural network

(RNN). They concluded that both ARIMA and RNN models outperform the feedforward model; in terms of lower predictive errors and higher percentage of correct reversal detection. They also investigated the effect of varying the damped feedback weights in recurrent net, and they found that RNN at the optimal weighting factors gave more satisfactory performances than ARIMA model.

Phua, *et al.* (2003) applied generically evolved neural networks models to predict the Straits Times Index of the Stock Exchange of Singapore. Their studies show that satisfactory results can be achieved when applying these techniques.

Kamruzzaman and Sarker (2003) investigated three ANN based forecasting models to predict six foreign currencies against Australian dollar using historical data and moving average technical indicators, and a comparison was made with traditional ARIMA model. All the ANN based models outperformed ARIMA model measured on five performance metrics used by them. Their results demonstrate that ANN based model can forecast the exchange rates closely.

Following the papers cited above and taking in consideration the growing interest in forecasting, in this study it is expected to evaluate the performance of an ARIMA and an ANN model when applied for predicting four time series representing stock price indexes.

1.4.1 – Classical methods for time series forecasting

On a classical time series forecasting it is assumed that the future value is a linear combination of historical data. There are several time series forecasting models, however the most highly popularized is Box-Jenkins ARIMA model which was successfully applied in financial time series forecasting, and also as a promising tool for modelling the empirical dependencies between successive time between failures.

Before broaching the structure of the ARIMA model it is done a brief overview of basic concepts of linear time series analysis such as stationarity, seasonality, unit-root nonstationarity and a short reference to the most classical common types of time series forecasting process.
A random or stochastic process is a collection of random variables ordered in time. If a random variable*, Y,* it is continuous will be denote as *Y(t),* and if it is discrete will be denoted by Y_t. A type of stochastic process that received a great deal of attention and scrutiny by time series analysts is the stationary stochastic process.

Stationarity. Broadly speaking, a time series is said to be stationary if the mean, the variance and the autocovariance (at various lags) remain the same no matter at what point we measure them, that is, they are constant over the time.

A time series Y_t is said to be strictly stationary if the joint distribution of ($Y_{t_1}, Y_{t_2}, \ldots, Y_{t_k}$) and of ($Y_{t_1-t}, Y_{t_2-t}, \ldots, Y_{t_k-t}$) are identical for all *t*, where $k \in /N$, $t_i \in /N$ and $i = 1, \cdots, k,$ i.e., strict stationary requires that the joint distribution of ($Y_{t_1}, Y_{t_2}, \ldots, Y_{t_k}$) is invariant under time shift.

This is a very strong condition that is hard to verify empirically. A weaker version of stationarity is often assumed. A time series Y_t is weakly stationary if both the mean of Y_t and the covariance between Y_t and Y_{t-s} are time-invariant, $s \in /N$. Thus Y_t is weakly stationary if

- $E(Y_t) = \mu$, which is a constant and
- $Cov(y_t, y_{t-s}) = \gamma_s$, is called lag-s autocovariance of Y_t.

Stationarity is very important in time series because if a time series is nonstationary (time varying mean or a time-varying variance or both) it's behaviour only can be study for the period under consideration. Each set of time series data will therefore be a particular episode. Thus, for the purpose of forecasting nonstationary time series may be of little practical value.

Autocorrelation function (ACF). The correlation coefficient between two random variables *X* and *Y*, measures the strength of linear dependence between *X* and *Y*, and take values between *-1* and *1*. It is defined as

$$\rho_{X,Y} = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}} = \frac{E[(X-\overline{X})(Y-\overline{Y})]}{\sqrt{E(X-\overline{X})^2 E(Y-\overline{Y})^2}},$$

where $\overline{X}$ is the mean of *X* and $\overline{Y}$ the mean of *Y*. If $\rho_{X,Y} = 0$, then *X* and *Y* are independent random variables.

Consider a weakly stationary time series Y_t. When the linear dependence between Y_t and it's past values Y_{t-i} is of interest, the concept of correlation is generalized to autocorrelation. The correlation coefficient between Y_t and Y_{t-s} is called the lag-*s* autocorrelation of Y_t and is commonly denoted by ρ_s, which under the weak stationarity assumption is a function defined by

$$\rho_s = \frac{\sum_{t=s+1}^{T}(Y_t - \overline{Y})(Y_{t-s} - \overline{Y})}{\sum_{t=1}^{T}(Y_t - \overline{Y})^2}.$$

As a matter of fact, a linear time series model can be characterized by its Autocorrelation function (ACF), and linear time series modelling makes use of the sample ACF to capture the linear dynamic of the data.

If ρ_1 is nonzero, it means that the series is first order serially correlated. If ρ_s dies off more or less geometrically with increasing lag s, it is a sign that the series obeys a low-order autoregressive process. If ρ_s drops to zero after a small number of lags, it is a sign that the series obeys a low-order moving-average process.

White noise. A time series Y_t is called white noise if Y_t is a sequence of independently and identically distributed (*iid*) random variables with finite mean and variance. In particular, if Y_t is normally distributed with mean zero and variance σ^2 and no serial correlation, then it is said to be Gaussian white noise. For a white noise series, all the ACFs are zero. In practice, if all sample ACFs are close to zero, then the series is a white noise series.

Random walk process. A random walk process is generally an approach used in the equity market to describe, for example, the behaviour of stock prices or exchange rates. This process continually drifts from any expected value in a specific period of time. In this approach it is not considered any constant value or constant variance over time.

Usually in the literature we can distinguish between two types of random walk process: random walk without a drift (1.4.1) (i.e., no constant or intercept term) and random walk with a drift (1.4.2) (i.e., a constant term is present):

$$y_t = y_{t-1} + \varepsilon_t \tag{1.4.1}$$

$$y_t = \alpha + y_{t-1} + \varepsilon_t \tag{1.4.2}$$

where y_0 is a real number denoting the starting value of the process, and ε_t is a white noise series and α is known as drift parameter.

If a *trend* in a time series process is completely predictable and not variable, it is said a deterministic trend, whereas if it is not predictable, it is said a stochastic trend.

Unit Root Process. Writing a random walk process as

$$y_t = \rho\, y_{t-1} + \varepsilon_t\ ,\ -1 \le \rho \le 1 \tag{1.4.3}$$

then if ρ=1, we have a random walk process without a drift. Moreover, if ρ is in fact *1*, we face what is known as the unit root problem, that is, a situation of nonstationarity; thus in this case the variance of Y_t is nonstationary. The name "unit root" is due the fact that $\rho=1$.

However, if $|\rho|<1$, then it can be shown that the time series Y_t is stationary. In practice, it is important to find out if a time series process has a unit root or not.

In the following definitions, the constants will be defined by α_i with $i \in /N$ or by β_i with $i \in /N$.

Autoregressive process. Consider the following equation

$$Y_t = \alpha_0 + \alpha_1 Y_{t-1} + \varepsilon_t \tag{1.4.4}$$

where ε_t is white noise, i.e., a random number from a distribution with following properties:

- $E(\varepsilon_t) = 0$
- $Var(\varepsilon_t) = \sigma^2$ and
- $Cov(\varepsilon_{t-s}, \varepsilon_t) = 0\,, \quad if\ s \neq 0$

Cov means Covariance and is a measure of association between two variables, that is,

$$Cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y}) \tag{1.4.5}$$

Consequently, white noise is a sequence of randomly distributed real numbers with zero mean and no association between numbers drawn at different points of time.

As the current value, Y_t, is expressed in terms of the past value, Y_{t-1}, the equation (1.4.4) is called AR(1) model or first order autoregressive model, where it is assumed that Y_{t-1} and ε_t are independent.

The extension of the equation presented before to two consecutive past values would lead to an AR(2) model, that is

$$Y_t = \alpha_0 + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \varepsilon_t = \alpha_0 + \sum_{i=1}^{2}\alpha_i Y_{t-i} + \varepsilon_t \tag{1.4.6}$$

More generally, we say that a variable Y_t is autoregressive of order *p*, AR(*p),* with $p \in /N$, if it is a function of its *p* past values and can be expressed as

$$Y_t = \alpha_0 + \sum_{i=1}^{p}\alpha_p Y_{t-p} + \varepsilon_t \tag{1.4.7}$$

Moving Average process. When the current value, Y_t, of a time series is expressed in terms of current and previous values of the stochastic error term ε_t, we are talking about the first order moving average, MA(1), that is

$$Y_t = \beta_0 \varepsilon_t + \beta_1 \varepsilon_{t-1} \tag{1.4.8}$$

where ε_t, is a purely random term with mean zero and variance σ^2.

When (1.4.8) contains the *q* most recent values of the stochastic error term, we have a MA(*q*) model, which is given by the following equation:

$$Y_t = \beta_0 \varepsilon_t + \beta_1 \varepsilon_{t-1} + \cdots + \beta_q \varepsilon_{t-q} = \sum_{i=0}^{q} \beta_i \varepsilon_{t-i} \tag{1.4.9}$$

Autoregressive Moving Average process. An Autoregressive Moving Average process, ARMA model, is a combination between the autoregressive model and a moving average model. In general, an ARMA (*p,q*) is a combination of an AR(*p*), equation (1.4.7), and MA(*q*), equation (1.4.9), and can be written as

$$Y_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i Y_{t-i} + \sum_{i=1}^{q} \beta_i \varepsilon_{t-i} \tag{1.4.10}$$

In practice, one applies the ARMA process not to the time series, but to transformed time series. It is often the case, that the time series of differences is stationary in spite of the nonstationarity of the underlying process. Stationary time series can be well estimated by the ARMA model. That leads to the definition of the ARIMA model.

Autoregressive Integrated Moving Average process. Popularly known as the Box-Jenkins methodology, but technically known as the ARIMA methodology, these methods emphasis is not on constructing single or simultaneous equation models, but on analysing the probabilistic, or stochastic, properties of economic time series on their own under the philosophy "let the data speak for themselves". Unlike the regression models, in which Y_t is explained by *n* regressor X_1, X_2, …, X_n, the ARIMA time series models allow Y_t to be explained by past, or lagged values of Y itself and stochastic error terms. For these reason, these models are sometimes called theoretic models because they are not derived from any economic theory – and economic theories are often basis of simultaneous equation models.

The previous mentioned models are based on assumptions that the time series involved are (weakly) stationary. But it is known that many economic time series are nonstationary, that is, they are *integrated*. If a time series is integrated of order *1*, i.e., *I(1),* it's first differences are *I(0),* that is, stationary. Similarly, if a time series is *I(2),* it's second difference is *I(0).* In general, if a time series is *I(d),* after differencing it *d* times we obtain an *I(0)* series.

Therefore if we needed to difference a time series *d* times in order to make it stationary and then apply the ARMA*(p,q)* process to it, we say that the original time series is an ARIMA*(p,d,q)* process, that is, it is an autoregressive integrated moving average time series. The first task in estimating ARIMA model is to specify *p* (the number of autoregressive terms), *d* (the number of first differences or other transformation) and *q* (the number of moving-average terms). Once these parameters have been chosen, the job of specifying the ARIMA equation is complete, and the computer estimation package will then calculate estimates of the appropriate coefficients. Because the error term in the moving average process is, of course, not observable, a non linear estimation technique must be used instead of OLS.

For the forecasting purposes the usage of ARIMA model can be summarized by the flowchart shown in the following figure.

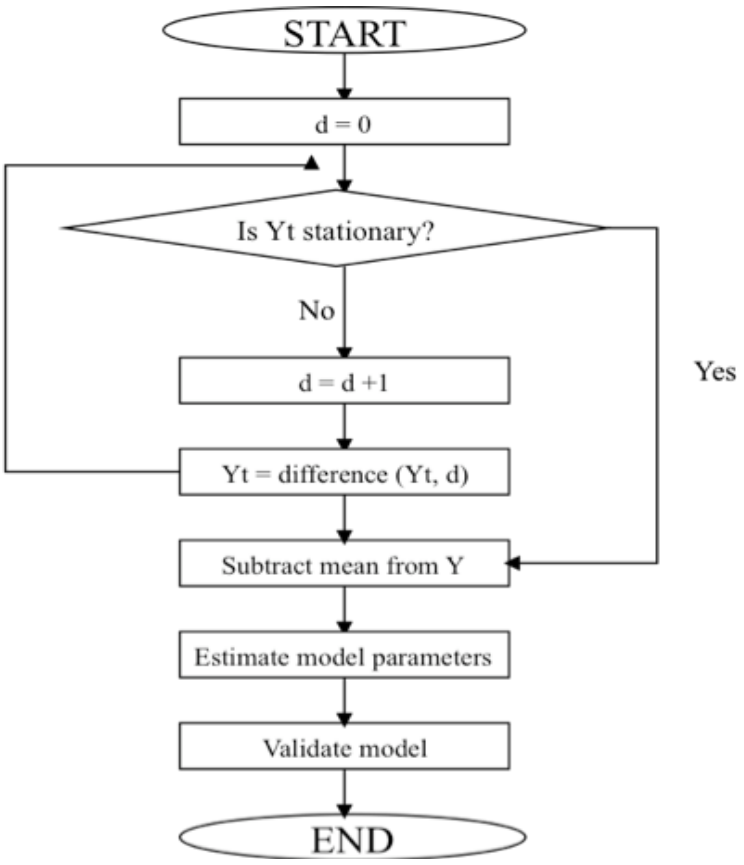


Figure 1.4.1: Summarized proceeding to modulate an ARIMA

In the first step, the differencing operator is applied to the time series until it becomes stationary. The determination whether Y_t is stationary or not can be performed by visualizing the correlogram of Y_t. It's rare to find in economic examples a situation calling for *d > 1*. If the ACF of the dependent variable approaches zero as the number of lags increases ("the correlogram converges to zero"), then the series is stationary; a nonstationary series will show little tendency for the ACFs to decrease in size as the number of lags increase. After an examination of the plot and the ACF makes the researcher feel comfortable that enough transformations have been applied to make the resulting series stationary. The next step is to choose integer values for *p* and *q*. After the *d* value is determined, any non-zero mean is removed from the time series (the mean is calculated and subtracted from each element of the time series).

Next, the parameters *p* and *q* are estimated. The number of autoregressive terms *(p)* and moving average terms *(q)* to be introduced are typically determined at the same time. These are chosen by finding the lowest *p* and *q* for each residuals of the estimated equation devoid to the autoregressive and moving-average components. This is done by:

- First point: choosing an *initial (p,q)* set (making them as small as is reasonable),
- Second point: estimating equations (1.4.7) and (1.4.9) for that *(p,q)* and then chosen above, and
- Third point: testing the residuals of the estimated equation to see if they are free of autocorrelations. If they aren't, then either *p* or *q* is increased by one, and the process is begun again.

To complete first point, choosing an initial *(p,q)* set, an ARIMA(0,*d*,0) is run and the residuals from this estimate are analysed with two statistical measures: the ACF, and a new measure, the partial ACF (PACF), which is similar to ACF and we expect that will hold the effects of the other lagged residuals constant. That is, the partial ACF, for the k^{th} lag is the correlation coefficient between e_t and e_{t-k}, holding constant all other residuals. Almost every different ARIMA model has a unique ACF/partial ACF combination; the theoretical ACF and partial ACF patterns are different for different ARIMA(*p,d,q*) specifications. In particular, the last lag before the PACF moves toward zero with an exponential decay is typically a good value for *p,* and the last lag before the ACF moves toward zero with an exponential decay is typically a good value for *q*.
Once the (*p,d,q*) parameters have been chosen, the equation is estimated by the computer's ARIMA equation package. Since the moving average process error term cannot actually be observed, an ARIMA estimation with $q > 0$ requires the use of a nonlinear estimation procedure rather than OLS, if $q=0$, OLS can be used.

After estimating an initial *(p,d,q)* combination, calculate and inspect the ACF's and PACF's of the residuals. If these ACF's and PACF's are all significantly different from zero (use the t-test if in doubt) then the equation can be considered as in the final specification (free from autoregressive and moving average components). If one or more of the PACF's or ACF's are significantly different from zero, then increase *p* (if a PACF is significant) or *q* (if ACF is significant) by one and reestimate the model. This process will be performed as long as the residuals have no autoregressive or moving-average components.

The last step, validation, is used for forecasting proposes. As a final check, some researchers compare the variance of ARIMA*(p,d,q)* with those of ARIMA*(p+1,d,q)* and ARIMA*(p,d,q+1).* If ARIMA*(p,d,q)* has the lowest variance, then it should be considered the final specification. Also useful is a Q-statistic, a measure of whether the first *k* ACF's are (jointly) significantly different from zero. If the Q-statistic, which is usually printed out by ARIMA computer packages, is less critical chi-square value, then the model can be considered free from autoregressive and moving average components.

1.4.2 – Artificial Neural Networks for time series forecasting

Neural network methods are coming from the brain science of cognitive theory and neurophysiology. Thus Artificial Neural networks (ANN) are characterized by pattern of connections among the various network layers, the numbers of neurons in each layer, the learning algorithm and the neurons activation functions. A neural network is a set of connected input units (also termed neurons, nodes or variables) X_0, X_1, X_2, …, X_n and one or more output variables such as Y_0, Y_1, Y_2, …, Y_n where each connection has an associated parameter indicating the strength of this connection, the so called weight. During the learning phase, the network learns by adjusting the weights so as to be able to correctly predict or classify the output target of a given set of inputs samples.
Usually in ANN modelling one of the inputs, $X_i, i \in \{1,\ldots,n\}$, known as bias, have to be included and assigned with value 1.

There exits a huge variety of different ANN types, but the most commonly used are the multi-layer feedforward and the recurrent networks.

1.4.2.1 – Multi-layer Feedforward Neural Networks

In the multi-layer feedforward ANN the information is transmitted from the input layer to the output layer. It does not allow any internal feedback of information. As in the human brain, the signals flow in one direction, from the input to the output layer. Feedforward networks are guaranteed to reach stability (Kermanshahi, *et.al.,* 2002).

For each training sample, the input variables are fed simultaneously into a layer of units making up the input layer. The weighted outputs of these units are, in turn, fed simultaneously to a second layer of units known as a hidden layer. The hidden layer's weighted outputs can be input to another hidden layer, and so on. The weight outputs of the last hidden layer are inputs to units making up the output layer, which issues the network's prediction for a given set of samples.

A Neural Network system with 2 hidden units is shown in following figure.

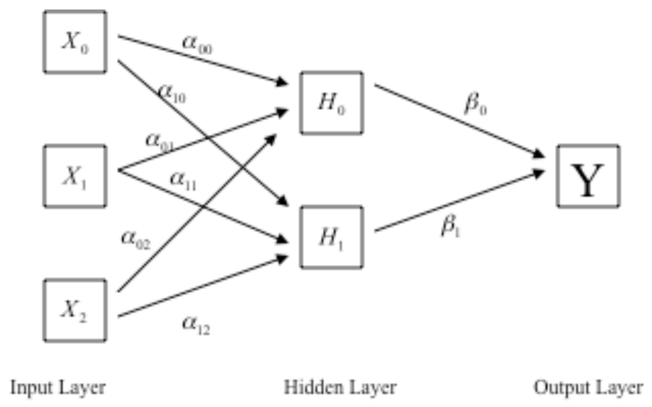


Figure 1.4.2: Feedforward Neural Networks (single hidden layer network)

Input-hidden units relationship can be expressed as:

$$H_j = \sum_{i=0}^{n} \alpha_{ij} X_i \tag{1.4.11}$$

where H_j are hidden variables, X_i are input variables and the α_{ij}, are networks weights that regulate flow from input to hidden variables.

Similarly, output variables are weighted sum of hidden units:

$$Y_j = \sum_{j=0}^{m} \beta_j H_j \qquad (1.4.12)$$

where H_j are hidden variables, Y_j are output variables and β_j, are connection strength between hidden and output variables.
Substituting (1.4.11) in (1.4.12), we have

$$Y_j = \sum_{j=0}^{m} \sum_{i=0}^{n} \beta_j \left(X_i \alpha_{ji} \right) \ . \qquad (1.4.13)$$

which gives the complete relationship between input, hidden and output variables.

Not all relationships in economics and finance are direct. The hidden layers capture all non-direct relationships between input and output variables. All intermediate variables are represented in ANN as unknown units. Due the lack of information about the variables represented in the hidden layers, ANN is considering by most researchers as a “Black Box”.

The number of hidden layers and units in each hidden layer are related with the ability of network to approximate more complex functions, however networks with complex structures don’t perform necessarily better. These networks seem more sensitive to noise, which in turn obstructs the learning process. Although more hidden units in the system normally result in better forecast, too many hidden units may lead to over-fitting on sample data. The decision concerning the number of hidden layers is still based on trial and error.

In what follows in this thesis, the parameters and weights are estimated during the training process using the Backpropagation algorithm, one of the most powerful algorithm related to ANN.

1.4.2.2 – Transfer Function in ANN

The use of transfer function has been indispensable in ANN. Let's assume in the equation (1.4.13) a function *f*, the transfer function, defined by

$$Y_j = f\left(\sum_{j=0}^{m}\sum_{i=0}^{n}\beta_j\left(X_i\alpha_{ji}\right)\right). \qquad (1.4.14)$$

The transfer function is responsible for transferring the weighted sum of inputs to some value that is given to the next neuron. There are several types of transfer functions, among which

Binary transfer function, $f(z) = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}, \ z \in \Re$ and

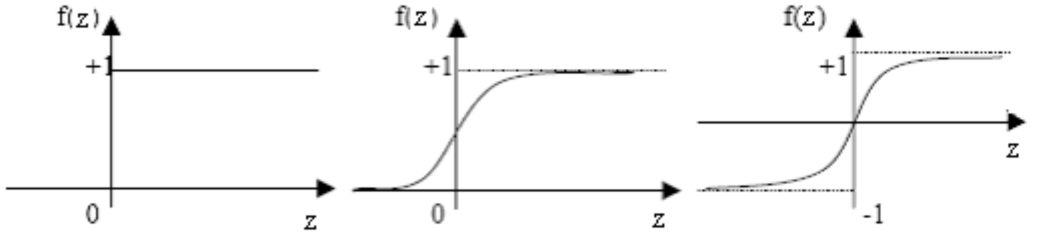
Sigmoid transfer function, $f(z) = \dfrac{1}{1+e^{-z}}$, $z \in \Re$.

Note that the return value of this functions lies in the interval [0,1], and in consequence these functions cannot be used in ANN to approximate functions, which can also take negative values (e.g. returns time series).

As shown below, in Figure 1.4.3, the binary transfer function and the sigmoid transfer function, are bound between 0 and 1. The difference lies on the fact that the logistic function turns on gradually when activated. When the logistic function approaches 0, the function is almost insensitive to impulse received from input layer, meaning output is inactivated.

The tangent hyperbolic transfer function, $f(z) = \dfrac{\sinh(z)}{\cosh(z)} = \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$, $z \in \Re$,

has the advantage of having the output interval, [-1,1], thus, it can be used in ANN that need to approximate functions that can take on negative values (e.g. stock index differences).



Binary transfer function **Sigmoid transfer function** **Hyperbolic transfer function**

Figure 1.4.3: Transfer functions

A wrong functional form of the model under investigation leads to incorrect estimation of parameters. ANN model builders are less concerned about the shape of the function, as specified by the magnitude of the functional parameters.

The Neural Networks model parameters are obtained by using the learning algorithms. The success or failure of these learning algorithms depends on whether or not the transfer function is differentiable. The most common nonlinear functions used are the sigmoid function and the tangent hyperbolic function. When the network contains hidden layers the sigmoid function is preferred to binary one, since the latter renders a model difficult to train. In this instance, the error obtained during the training process (which serves in the estimation of parameters) is constant, hence the gradient does not exist or is zero, making the learning process impossible when used some learning algorithms, such as backpropagation. With the sigmoid function it is possible to tell whether the change in weights is good or bad, because a small change in the weights will generate some change in output. With the step unit function, a small change in weights will usually generate no change in output.

If the relationship between input units and hidden units is nonlinear, a typical ANN specification consists of approximating the relation by the logistic function. We have then

$$H_j = \frac{1}{1 + e^{-\sum_{i=0}^{n} \alpha_{ij} x_i}}. \qquad (1.4.15)$$

Placing (1.4.15) in (1.4.12), we have

$$Y_j = \sum_{j=0}^{m} \frac{\beta_j}{1+e^{-\sum_{i=0}^{n}\alpha_{ij}x_i}} = f\left(\sum_{j=0}^{m}\beta_j \sum_{i=0}^{n} H\left(X_i \alpha_{ij}\right)\right) \tag{1.4.16}$$

where *f(x)* is an identity function, *m* is the number of hidden units *n* is the number of input units, α_{ij} are the connection strengths between the input layer and hidden layer, β_j represent network weights that connect the hidden layer to the output layer and *H(X)* is a logistic function connecting the output layer to the input layer.

1.4.2.3 – Learning Rule

The power of Neural Network models depends to a large extent on the way their layer connection weights are adjusted over time. Usually the weights of the ANN must be adjusted using some learning algorithm in order for the ANN to be able to approximate the target function with a sufficient precision.

A learning rule is defined as a method that modifies the weights and bias of a network. This method is also known as a training algorithm. The reasoning behind training is that the weights are updated in a way that will facilitate the learning of patterns inherent to the data. Data are divided into two sets, a training set and a test set. The training set serves to estimate weights in the model. Hence, the learning process is a crucial phase in ANN modelling. Usually the test set, is consisting of 10% to 30% of the total data set. The algorithm used to estimate the values of parameters, depends on the type of ANN under investigation.

1.4.2.4 – Backpropagation learning algorithm

Backpropagation is by far the most popular neural network algorithm that has been used to perform training on multi-layer feedforward neural networks. It is a method for transmission responsibility for mismatches to each of the processing elements in the network by propagating the gradient of the activation function back through the network to each hidden layer down to the first hidden layer. The weights $(\omega_1,\omega_2,\ldots,\omega_n)$ are then modified so as to minimize the mean squared error between the network prediction and the actual target.

In other words, the network processes an output, which depends on the networks weights, (that initially are assigned random values – usually between *-1* and *1*), input units, hidden units and the transfer function. The difference between the process and actual output (target), known as network error is propagating backward into the network. This error is used to update weights. This process is repeated until the total network error is minimized.

The error function can be defined as

$$E(\vec{\omega}) = \frac{1}{2}\sum_{t=1}^{N}(T_t - Y_t)^2 \tag{1.4.17}$$

where T_t and Y_t are the targeted output value and the process output in the t^{th} iteration, respectively, i.e., $E(\vec{\omega})$ is the sum of prediction errors for all training examples.

Prediction errors of individual training examples are in turn equal to the sum of the differences between output values produced by the ANN and the desired (correct) values. $\vec{\omega}$ is the vector containing the weights of the ANN. The goal of a learning algorithm is to minimize $E(\vec{\omega})$ for a particular set of training examples. There are several ways to achieve this, one common way is using the gradient descendent method.

1.4.2.4.1 – Stochastic gradient descent backpropagation learning algorithm

The gradient descendent method, can be described by 4 steps:

- First Step: choose some random initial values for the model parameters.
- Second Step: calculate the gradient of the error function $\nabla E(\vec{\omega})$:

$$\nabla E(\omega) = \left(\frac{\partial E(\omega)}{\partial \omega_1}, \frac{\partial E(\omega)}{\partial \omega_2}, \ldots, \frac{\partial E(\omega)}{\partial \omega_n} \right).$$

- Third Step: change the model parameters so that we move a short distance in the direction of the greatest rate of decrease of the error, i.e., in the direction of -$\nabla E(\vec{\omega})$
- Fourth Step: Repeat second and third steps until $\nabla E(\vec{\omega})$ gets close to zero. If the gradient of E is negative, we must increase ω to move ”forward” towards the minimum. If E is positive, we must move ”backwards” to the minimum.

First, a neural network is created and initialized (weights are set to small random numbers). Then, until the termination condition (e.g. the mean squared error of the ANN is less than a certain error threshold) is met, all training examples are ”taught” by the ANN. Inputs of each training example are fed to the ANN, and processed from the input layer, over the hidden layer(s) to the output layer. In this way, vector Y_t of output values produced by the ANN is obtained (third step). In the next step, the weights of the ANN must be adjusted. Basically, this happens when the weight update value, $\Delta\omega$, must ”move” the weight in the direction of steepest descent of the error function *E* with respect to the weight, or the partial derivative of *E* with respect to the weight:

$$\Delta\omega = -\eta \frac{\partial E}{\partial \omega_{ji}} \qquad (1.4.18)$$

where η is the learning rate that determines the size of the step that it is used for “moving” towards the minimum of *E*. If it is too small, then the convergence to optimal point may be small, leading to a slow convergence of ANN. Conversely, if it is too large, the algorithm may not converge at all. Usually $\eta \in \Re, \quad 0 \le \eta \le 1$.

Note that ω_{ji} can influence the ANN only through net, i.e., the weighted sum of inputs for unit j, therefore, we can use the chain rule to write

$$\frac{\partial E}{\partial \omega_{ji}} = \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial \omega_{ji}} \quad . \qquad (1.4.19)$$

Notes that

$$\frac{\partial net_j}{\partial \omega_{ji}} = (net_j)d\omega_{ji}$$

$$(net_j)d\omega_{ji} = \left(\sum_z \omega_{jz}x_{jz}\right)d\omega_{ji} = \left(\omega_{j0}x_{j0} + \cdots + \omega_{jz}x_{jz}\right)d\omega_{ji}$$
$$= \left(0x_{j0} + \cdots + 1x_{ji} + \cdots + 0x_{jz}\right)$$
$$= x_{ji} \quad .$$

So the equation (1.4.19) it is reduced to

$$\frac{\partial E}{\partial \omega_{ji}} = \frac{\partial E}{\partial net_j} x_{ji} \ . \tag{1.4.20}$$

The weights of hidden nodes are also updated. All the expressions are the same for both output and hidden nodes, with the exception of the derivation of $\delta_j = \frac{\partial E}{\partial net_j}$.

In the case of output nodes, net_j, they can influenced ANN only through Y_j. Similarly it was done above, with ω_{ji}, we can chain rule to write:

$$\frac{\partial E}{\partial net_j} = \frac{\partial E}{\partial Y_j}\frac{\partial Y_j}{\partial net_j} \quad . \tag{1.4.21}$$

Note that

$$\frac{\partial E}{\partial Y_j} = \left(\frac{1}{2}\sum_z (T_j - Y_j)^2\right)dY_j = \frac{1}{2}\left((T_0 - Y_0)^2 + \cdots + (T_z - Y_z)^2\right)dY_j$$
$$= \frac{1}{2}\left(0 + \cdots + (-2T_j + 2Y_j) + \cdots + 0\right)$$
$$= -(T_j - Y_j)$$

$$\frac{\partial Y_j}{\partial net_j} = \frac{\partial transfer(net_j)}{\partial net_j} = transfer'\,(net_j) \ .$$

So the equation (1.4.21) it is reduced to

$$\frac{\partial E}{\partial net_j} = -(T_j - Y_j).transfer'(net_j) \quad . \tag{1.4.22}$$

Considering the previous equations, the weight update $\Delta\omega$ for output nodes can be expressed as:

$$\begin{aligned}\Delta\omega = -\eta\frac{\partial E}{\partial \omega_{ji}} = -\eta\frac{\partial E}{\partial net_j}\frac{\partial net_j}{\partial \omega_{ji}} &= \\ = -\eta\frac{\partial E}{\partial Y_j}\frac{\partial Y_j}{\partial net_j}x_{ji} &= \\ = -\eta\big(-(T_j - Y_j)\big)\big(transfer'(net_j)\big)x_{ji} &= \\ = \eta\big(T_j - Y_j\big)\big(transfer'(net_j)\big)x_{ji} & \quad .\end{aligned}$$

For hidden nodes, this derivation must take into account the indirect ways in which ω_{ji} can influence the network outputs, consequently *E*.

There are many improvements of this algorithm such as momentum term, weight decay etc.. However, multilayer feedback in combination with stochastic gradient descent learning algorithm is the most popular ANN technique used in practice. Another important feature of this learning algorithm is that it assumes a quadratic error function, hence it assumes there is only one minimum. In practice, the error function can have - apart from the global minimum - multiple local minima. There is a danger for the algorithm to land in one of the local minima and thus not be able to reduce the error to highest extent possible by reaching a global minimum.

Chapter 2

2 – INPUT DATA, MODELS SELECTION AND RESULTS

2.1 – DATA AND SOME DESCRIPTIVE STATISTICS

The data, used in this study are extracted from Thomson Financial Datastream and correspond to stock price indexes for four countries namely: Germany (BD), Italy (IT), Greece (GR) and Portugal (PT). The observations are daily time series and are ranged from February 1st 1991 to February 1st, 2006, which yields a total of n = 4175 observations for each series.

In the following table it is possible to see some descriptive statistics of the four time series, including the Jarque-Bera Test of Normality. We do reject the null hypothesis that the residuals are normally distributed in all time series.

	BD	GR	IT	PT
Mean	4 792 663	1 066 769	1 695 499	1 341 943
Median	4 483 100	9 210 900	1 608 080	1 289 900
Maximum	1 009 670	3 360 220	3 502 260	2 755 400
Minimum	2 314 700	1 787 500	6 237 800	5 883 000
Std. Dev.	1 888 623	7 059 177	7 323 893	5573703
Skewness	0.645120	0.972055	0.453212	0.379909
Kurtosis	2 473 827	3 267 931	1 981 211	1 952 944
Jarque-Bera	3 377 535	6 699 748	3 234 814	2 911 453
Probability	0.000000	0.000000	0.000000	0.000000
Sum	2000937	4453759	7078708	560261.1
Sum Sq. Dev.	1.49E+08	2.08E+09	2.24E+09	12 967 018
Observations	4175	4175	4175	4175

Figure 2.1: Descriptive statistics

As these time series represent stock price indexes, the original data were transformed using the natural logarithm, also in order to reduce the effect of outliers. One of the main advantage of such transformation is the stationary property; which permit to not lose the important information from trading, and it is also transforming the data to stabilize the variance.

Thus, the daily time series were coded as:
LOGBD= natural logarithm of the Stock price index of Germany: TOTMKBD(PI),
LOGITL= natural logarithm of the Stock price index of Italy: TOTMKIT(PI),
LOGGR= natural logarithm of the Stock price index of Greece: TOTMKGR(PI) and
LOGPT= natural logarithm of the Stock price index of Portugal: TOTMKPT(PI).

There are several testes for stationary, but the most encountered in the literature are those related with graphical analyses, correlograms and the unit root test

Figure 2.2, depict the nonstationary shape of each one of the four time series, it is going to be analyse. These series varies randomly over time and there is no global trend or seasonality observed. All of them have a quite similar behaviour during all observed period, with a steepest decrease of Italy in the 90th justifying the socio-political conditions and with a consistent increasing for Greece after 90th which can be justified by the collapse of communism in the Balkanians. A decreasing for all of them after 2000 is justifying by the terrorist attacks on September 2001, as well as ,the subsequent terrorist attacks.

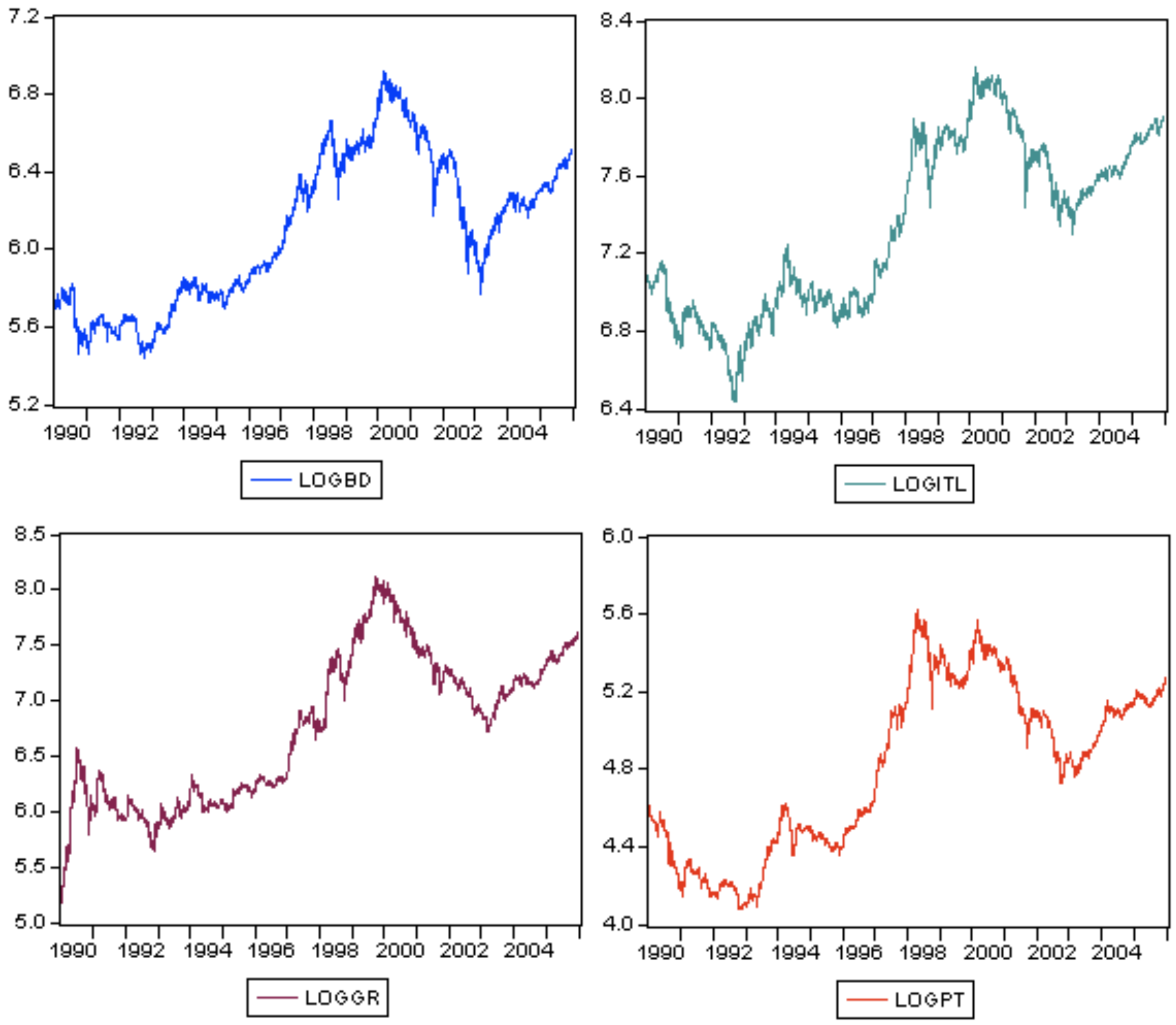
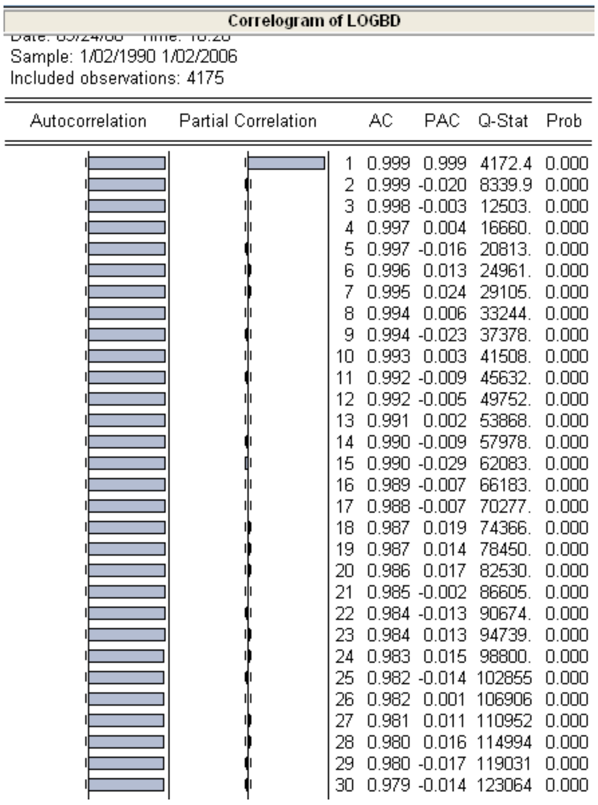


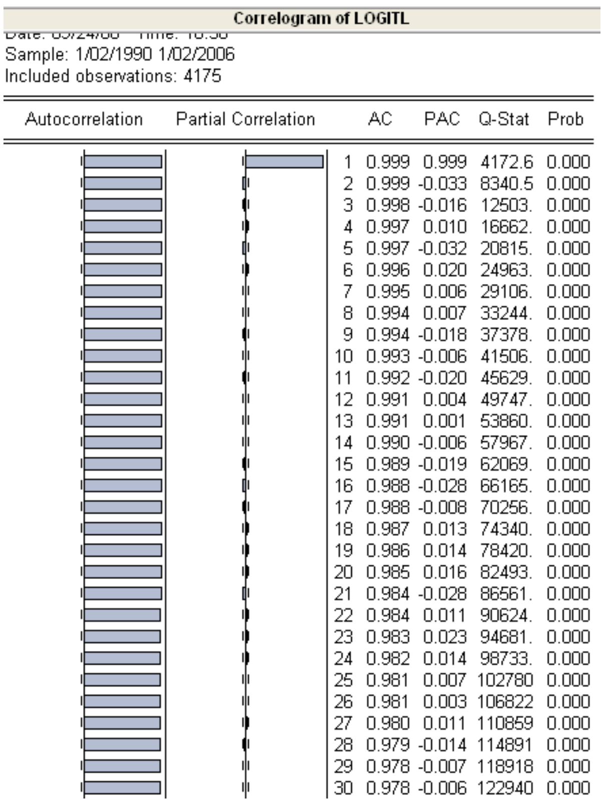
Figure 2.2: Graphical representation of natural logarithm of the four Stock price index.

When regarding a correlogram we can find if a particular time series is stationary. For this purpose it is consider the correlogram of natural logarithm series (figure 2.3 and 2.4) and, the correlogram of first difference of natural logarithm series on figures 2.6 and 2.7, where it is shown the correlogram up to 30 lags for each time series.


Correlogram of LOGBD

Sample: 1/02/1990 1/02/2006
Included observations: 4175

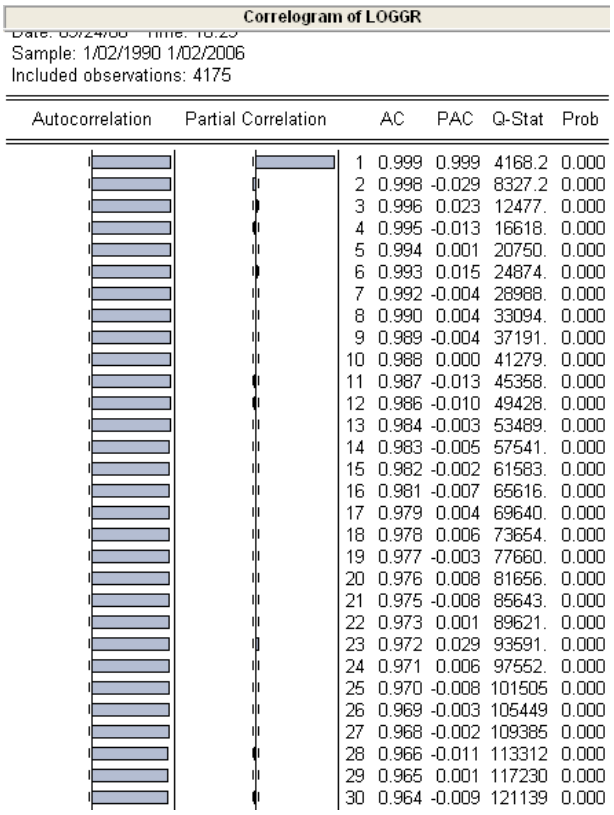
Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	0.999	0.999	4172.4	0.000
		2	0.999	-0.020	8339.9	0.000
		3	0.998	-0.003	12503.	0.000
		4	0.997	0.004	16660.	0.000
		5	0.997	-0.016	20813.	0.000
		6	0.996	0.013	24961.	0.000
		7	0.995	0.024	29105.	0.000
		8	0.994	0.006	33244.	0.000
		9	0.994	-0.023	37378.	0.000
		10	0.993	0.003	41508.	0.000
		11	0.992	-0.009	45632.	0.000
		12	0.992	-0.005	49752.	0.000
		13	0.991	0.002	53868.	0.000
		14	0.990	-0.009	57978.	0.000
		15	0.990	-0.029	62083.	0.000
		16	0.989	-0.007	66183.	0.000
		17	0.988	-0.007	70277.	0.000
		18	0.987	0.019	74366.	0.000
		19	0.987	0.014	78450.	0.000
		20	0.986	0.017	82530.	0.000
		21	0.985	-0.002	86605.	0.000
		22	0.984	-0.013	90674.	0.000
		23	0.984	0.013	94739.	0.000
		24	0.983	0.015	98800.	0.000
		25	0.982	-0.014	102855	0.000
		26	0.982	0.001	106906	0.000
		27	0.981	0.011	110952	0.000
		28	0.980	0.016	114994	0.000
		29	0.980	-0.017	119031	0.000
		30	0.979	-0.014	123064	0.000


Correlogram of LOGITL

Sample: 1/02/1990 1/02/2006
Included observations: 4175

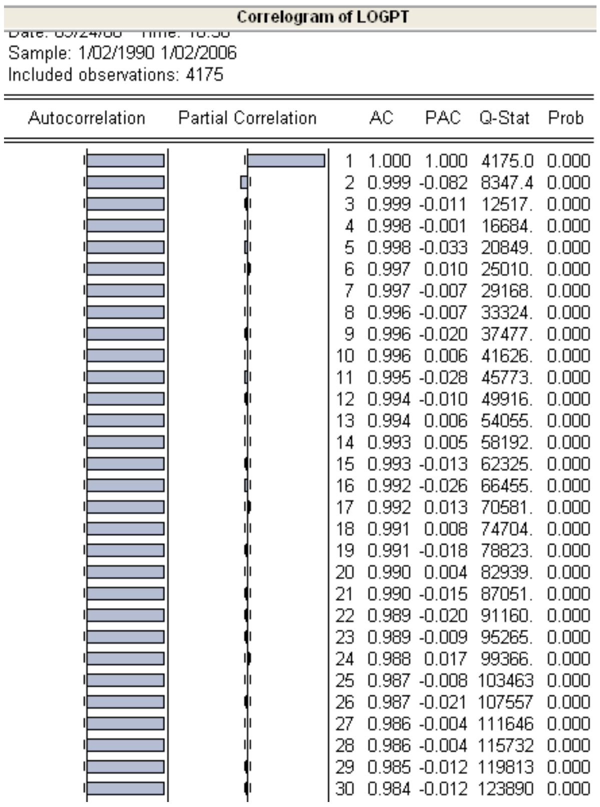
Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	0.999	0.999	4172.6	0.000
		2	0.999	-0.033	8340.5	0.000
		3	0.998	-0.016	12503.	0.000
		4	0.997	0.010	16662.	0.000
		5	0.997	-0.032	20815.	0.000
		6	0.996	0.020	24963.	0.000
		7	0.995	0.006	29106.	0.000
		8	0.994	0.007	33244.	0.000
		9	0.994	-0.018	37378.	0.000
		10	0.993	-0.006	41506.	0.000
		11	0.992	-0.020	45629.	0.000
		12	0.991	0.004	49747.	0.000
		13	0.991	0.001	53860.	0.000
		14	0.990	-0.006	57967.	0.000
		15	0.989	-0.019	62069.	0.000
		16	0.988	-0.028	66165.	0.000
		17	0.988	-0.008	70256.	0.000
		18	0.987	0.013	74340.	0.000
		19	0.986	0.014	78420.	0.000
		20	0.985	0.016	82493.	0.000
		21	0.984	-0.028	86561.	0.000
		22	0.984	0.011	90624.	0.000
		23	0.983	0.023	94681.	0.000
		24	0.982	0.014	98733.	0.000
		25	0.981	0.007	102780	0.000
		26	0.981	0.003	106822	0.000
		27	0.980	0.011	110859	0.000
		28	0.979	-0.014	114891	0.000
		29	0.978	-0.007	118918	0.000
		30	0.978	-0.006	122940	0.000

Figure 2.3: The correlogram of LOGBD and the correlogram of LOGGR


Correlogram of LOGGR

Sample: 1/02/1990 1/02/2006
Included observations: 4175

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	0.999	0.999	4168.2	0.000
		2	0.998	-0.029	8327.2	0.000
		3	0.996	0.023	12477.	0.000
		4	0.995	-0.013	16618.	0.000
		5	0.994	0.001	20750.	0.000
		6	0.993	0.015	24874.	0.000
		7	0.992	-0.004	28988.	0.000
		8	0.990	0.004	33094.	0.000
		9	0.989	-0.004	37191.	0.000
		10	0.988	0.000	41279.	0.000
		11	0.987	-0.013	45358.	0.000
		12	0.986	-0.010	49428.	0.000
		13	0.984	-0.003	53489.	0.000
		14	0.983	-0.005	57541.	0.000
		15	0.982	-0.002	61583.	0.000
		16	0.981	-0.007	65616.	0.000
		17	0.979	0.004	69640.	0.000
		18	0.978	0.006	73654.	0.000
		19	0.977	-0.003	77660.	0.000
		20	0.976	0.008	81656.	0.000
		21	0.975	-0.008	85643.	0.000
		22	0.973	0.001	89621.	0.000
		23	0.972	0.029	93591.	0.000
		24	0.971	0.006	97552.	0.000
		25	0.970	-0.008	101505	0.000
		26	0.969	-0.003	105449	0.000
		27	0.968	-0.002	109385	0.000
		28	0.966	-0.011	113312	0.000
		29	0.965	0.001	117230	0.000
		30	0.964	-0.009	121139	0.000


Correlogram of LOGPT

Sample: 1/02/1990 1/02/2006
Included observations: 4175

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	1.000	1.000	4175.0	0.000
		2	0.999	-0.082	8347.4	0.000
		3	0.999	-0.011	12517.	0.000
		4	0.998	-0.001	16684.	0.000
		5	0.998	-0.033	20849.	0.000
		6	0.997	0.010	25010.	0.000
		7	0.997	-0.007	29168.	0.000
		8	0.996	-0.007	33324.	0.000
		9	0.996	-0.020	37477.	0.000
		10	0.996	0.006	41626.	0.000
		11	0.995	-0.028	45773.	0.000
		12	0.994	-0.010	49916.	0.000
		13	0.994	0.006	54055.	0.000
		14	0.993	0.005	58192.	0.000
		15	0.993	-0.013	62325.	0.000
		16	0.992	-0.026	66455.	0.000
		17	0.992	0.013	70581.	0.000
		18	0.991	0.008	74704.	0.000
		19	0.991	-0.018	78823.	0.000
		20	0.990	0.004	82939.	0.000
		21	0.990	-0.015	87051.	0.000
		22	0.989	-0.020	91160.	0.000
		23	0.989	-0.009	95265.	0.000
		24	0.988	0.017	99366.	0.000
		25	0.987	-0.008	103463	0.000
		26	0.987	-0.021	107557	0.000
		27	0.986	-0.004	111646	0.000
		28	0.986	-0.004	115732	0.000
		29	0.985	-0.012	119813	0.000
		30	0.984	-0.012	123890	0.000

Figure 2.4: The correlogram of LOGITL and the correlogram of LOGPT

On Eviews software the Q stats reported in correlogram represents the Ljung-Box Q statistics, Q_{LB}, and their *p*-values are also reported there. Under the null hypotheses of no residual autocorrelation, Q_{LB} is approximately distributed as chi-square distribution with *m* degrees of freedom, χ^2_m. In practise, the selection of *m* is done to balance competing criteria. On one hand, *m* should not be too small because, after all, we are trying to do a joint test on a large part of autocorrelation function, and on other hand, as *m* grows relative to *T*, the quality of the distributional approximations we have invoked deteriorates. In practise, focusing on *m* in the neighbourhood of $\sqrt{T}$ is often reasonable. (Diebold, F., 1997).

$$Q_{LB} = T(T+2)\sum_{j=1}^{s}\frac{\rho_j^2}{T-j}, \tag{2.1.1}$$

where $Q \sim \chi^2(m)$.

Looking at the column labelled AC (which is the sample ACF) and the first diagram, labelled Autocorrelation, where the solid vertical line represents the zero axis., the most salient feature is that the autocorrelation is very high, even up to a lag of 30 quarters. Even when we choose a 200 laggs length (one-quarter of the length of the time series would be 1044, but the maximum allowed in the Eviews software is 200) the coefficient is bigger than 0,84. That is, it falls out of the confidence intervals, which is given by $[-2/\sqrt{T}, 2/\sqrt{T}]$, where *T* is the number of observations. Thus our confidence interval is $[-2/\sqrt{T}, 2/\sqrt{T}]$ = $[-2/\sqrt{4175}, 2/\sqrt{4175}] = [-0.031, 0.031]$.

Other important fact that stands out from the correlogram is, the sample partial autocorrelation function (the column labelled PAC) which, after the first lag drops dramatically, and also the resulting correlogram (second diagram, labelled Partial Autocorrelation). The autocorrelation starts at a very high value and declines very slowly as the lag lengthens, which means that we are deal with a typical correlogram of a nonstationary time series.

Once done the graphical observation and the correlogram evaluation, we can say that are integrated of order 1. But we also can test the stationarity through the *unit root test,* which has recently become popular in econometrics, being quite efficient and easy to be interpreted.

There are several tests that allowed to aboard the unit root issue. One of the most well known is the *Augmented Dickey-Fuller* (ADF) test. That is based on the following equation:

$$\Delta y_t = \beta_1 + \beta_2 t + (\rho - 1) y_{t-1} + \sum_{i=1}^{m} \alpha_i \Delta y_{t-i} + \varepsilon_t , \qquad (2.1.2)$$

where ε_t is a pure white noise error term and where $\Delta y_{t-1} = y_{t-1} - y_{t-2}$, $\Delta y_{t-2} = y_{t-2} - y_{t-3}$, etc.. The number of lagged difference terms to include is often determined empirically, where the idea is to include enough terms so that (2.1.2) is serially uncorrelated. Usually it is used the *Akaike Information Criterion,* AIC, or the *Schwarz or Bayesian Information Criterion,* SBC, in order to have so many lags as need for $\{\varepsilon_t\} \sim \mathrm{WN}(0,\sigma^2)$.

The ADF test follows the same asymptotic distribution as Dickey-Fuller (DF) statistic, (the τ (tau) statistic) and the null hypothesis is $\rho = 1$, that is, if there is a unit root – then the time series is nonstationary. The alternative hypothesis is that ρ is less than 1, that is, the time series is stationary. The case of ρ being bigger than 1 is out of possible, because in that case the original time series will be explosive.

In table 2.1 it is possible see the results of ADF test and the values of AIC and SBC for the three different forms (model with a drift, model with drift and with trend, and a model without both.

	Augmented Dickey-Fuller							
	Test for unit root in Level				Test for unit root in 1st difference			
Null Hypothesis:	**LOGBD has a unit root**				**D(LOGBD) has not a unit root**			
Include in test equation:	ADF	P-value*	AIC	SBC	ADF	P-value*	AIC	SBC
-> Drift	-0.8910	-2.8620	-6.1497	-6.1452	-6.1028	-2.8620	**-6.15001**	**-6.1470**
-> Drift and Trend	-1.4900	-3.4109	-6.1496	-6.1435	-6.1021	-3.4109	-6.14953	-6.1450
-> None	0.9583	-1.9409	-6.1500	-6.1469	-6.1019	-1.9409	-6.15024	-6.1487
Null Hypothesis:	**LOGITL has a unit root**				**D(LOGITL) has not a unit root**			
Include in test equation:	ADF	P-value*	AIC	SBC	ADF	P-value*	AIC	SBC
-> Drift	-0.6539	-2.8620	-5.9242	-5.9196	-6.0721	-2.8620	**-5.92455**	**-5.9215**
-> Drift and Trend	-1.9143	-3.4109	-5.9245	-5.9185	-6.0719	-3.4109	-5.92415	-5.9196
-> None	0.9300	-1.9409	-5.9245	-5.9215	-6.0713	-1.9409	-5.92480	-5.9233
Null Hypothesis:	**LOGGR has a unit root**				**D(LOGGR) has not a unit root**			
Include in test equation:	ADF	P-value*	AIC	SBC	ADF	P-value*	AIC	SBC
-> Drift	-1.8918	-2.8620	-5.3622	-5.3577	-5.5492	-2.8620	**-5.36187**	-5.3588
-> Drift and Trend	-1.9288	-3.4109	-5.3620	-5.3559	-5.5499	-3.4109	-5.36159	-5.3570
-> None	1.6951	-1.9409	-5.3617	-5.3587	-5.5443	-1.9409	-5.36149	**-5.3600**
Null Hypothesis:	**LOGPT has a unit root**				**D(LOGPT) has a not unit root**			
Include in test equation:	ADF	P-value*	AIC	SBC	ADF	P-value*	AIC	SBC
-> Drift	-0.4385	-2.8620	-6.6896	-6.6850	-5.6123	-2.8620	-6.6900	-6.6870
-> Drift and Trend	-1.4680	-3.4109	-6.6897	-6.6836	-5.6123	-3.4109	-6.6896	-6.6851
-> None	1.0066	-1.9409	-6.6900	-6.6870	-5.6112	-1.9409	**-6.6902**	**-6.6887**

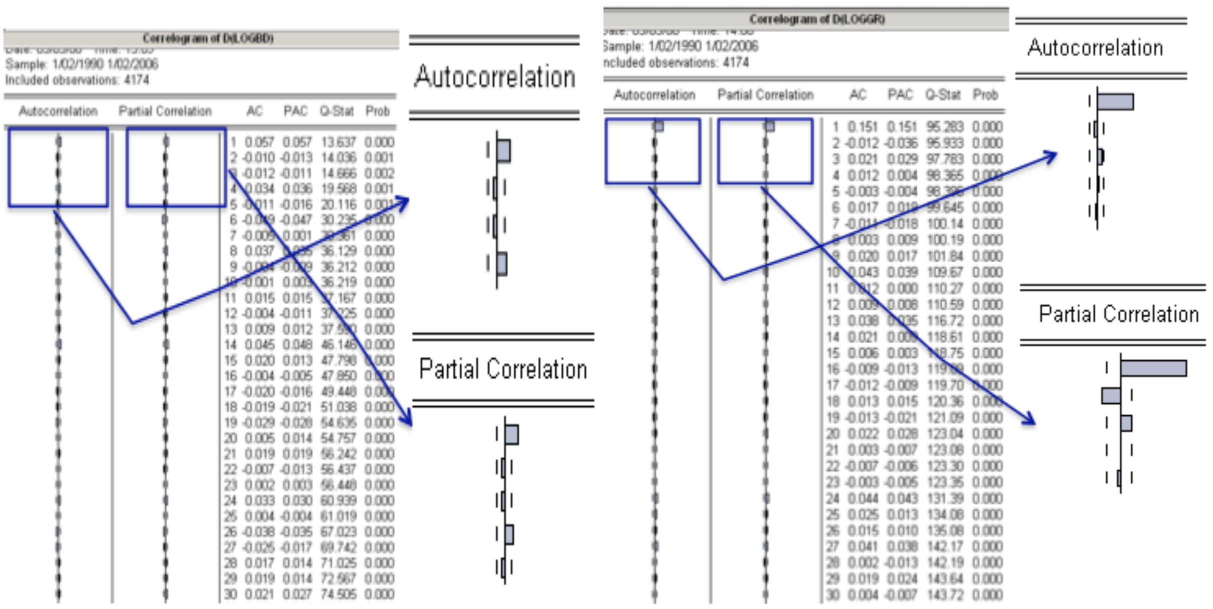
*critical values at 5% level

Table 2.1: The resume of *Eviews* results for the Unit Root Test

Once again, in all time series we don't reject, in levels, the null hypothesis, that is, there is a unit root – the time series are nonstationary, although we reject this hypothesis in first difference model. Similar results were obtain using Phillips-Perron test.

We can also see this from the estimated ACF and PACF correlograms given in the Figures 2.5 and 2.6, which illustrated the correlograms of the first differences series. Now we have a much different pattern for ACF and PACF. Despite, some of the ACF's and of the PACF's laggs seem statistically different from zero the majority are not statistically different from zero.

It is denote DLOGBD for the first differences of LOGBD, the DLOGILT to denote the first differences of LOGITL, similarly DLOGGR for the first differences of LOGGR and finally DLOGPT to denote the first differences of LOGPT.



Correlogram of D(LOGBD)

Sample: 1/02/1990 1/02/2006
Included observations: 4174

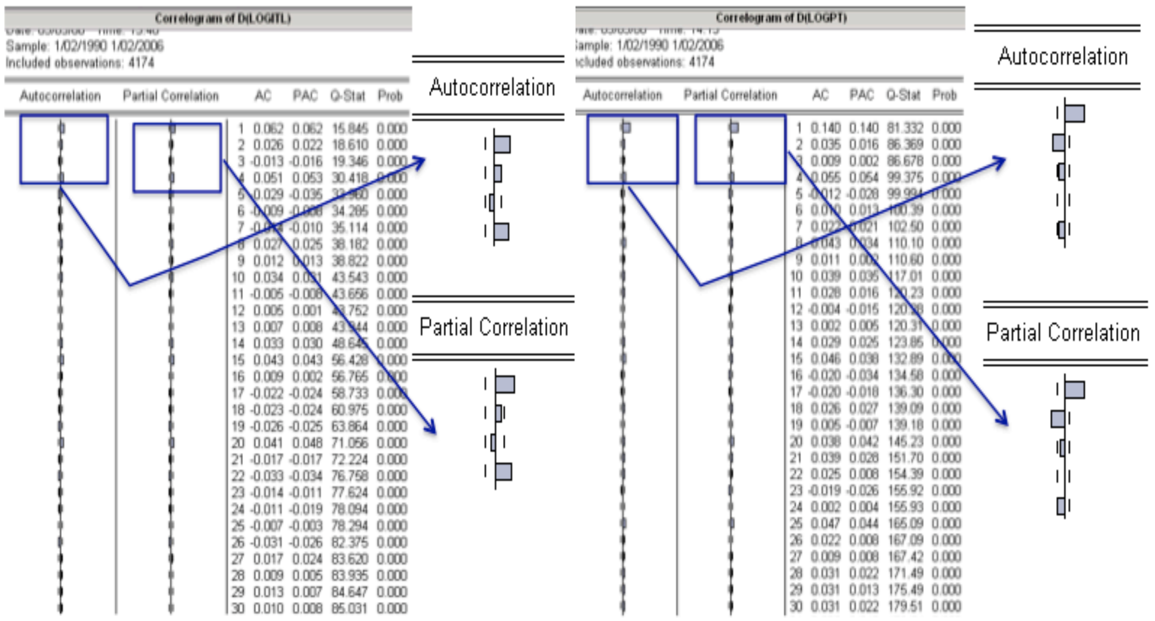
	AC	PAC	Q-Stat	Prob
1	0.057	0.057	13.637	0.000
2	-0.010	-0.013	14.036	0.001
3	-0.012	-0.011	14.666	0.002
4	0.034	0.036	19.568	0.001
5	-0.011	-0.016	20.116	0.001
6	-0.049	-0.047	30.235	0.000
7	-0.005	0.001	30.361	0.000
8	0.037	0.035	36.129	0.000
9	-0.004	-0.009	36.212	0.000
10	-0.001	0.003	36.219	0.000
11	0.015	0.015	37.167	0.000
12	-0.004	-0.011	37.225	0.000
13	0.009	0.012	37.590	0.000
14	0.045	0.048	46.146	0.000
15	0.020	0.013	47.798	0.000
16	-0.004	-0.005	47.850	0.000
17	-0.020	-0.016	49.448	0.000
18	-0.019	-0.021	51.038	0.000
19	-0.029	-0.028	54.635	0.000
20	0.005	0.014	54.757	0.000
21	0.019	0.019	56.242	0.000
22	-0.007	-0.013	56.437	0.000
23	0.002	0.003	56.448	0.000
24	0.033	0.030	60.939	0.000
25	0.004	-0.004	61.019	0.000
26	-0.038	-0.035	67.023	0.000
27	-0.025	-0.017	69.742	0.000
28	0.017	0.014	71.025	0.000
29	0.019	0.014	72.567	0.000
30	0.021	0.027	74.505	0.000

Correlogram of D(LOGGR)

Sample: 1/02/1990 1/02/2006
Included observations: 4174

	AC	PAC	Q-Stat	Prob
1	0.151	0.151	95.283	0.000
2	-0.012	-0.036	95.933	0.000
3	0.021	0.029	97.783	0.000
4	0.012	0.004	98.365	0.000
5	-0.003	-0.004	98.396	0.000
6	0.017	0.019	99.645	0.000
7	-0.011	-0.018	100.14	0.000
8	0.003	0.009	100.19	0.000
9	0.020	0.017	101.84	0.000
10	0.043	0.039	109.67	0.000
11	0.012	0.000	110.27	0.000
12	0.009	0.008	110.59	0.000
13	0.038	0.035	116.72	0.000
14	0.021	0.008	118.61	0.000
15	0.006	0.003	118.75	0.000
16	-0.009	-0.013	119.09	0.000
17	-0.012	-0.009	119.70	0.000
18	0.013	0.015	120.36	0.000
19	-0.013	-0.021	121.09	0.000
20	0.022	0.028	123.04	0.000
21	0.003	-0.007	123.08	0.000
22	-0.007	-0.006	123.30	0.000
23	-0.003	-0.005	123.35	0.000
24	0.044	0.043	131.39	0.000
25	0.025	0.013	134.08	0.000
26	0.015	0.010	135.08	0.000
27	0.041	0.038	142.17	0.000
28	0.002	-0.013	142.19	0.000
29	0.019	0.024	143.64	0.000
30	0.004	-0.007	143.72	0.000

Figure 2.5: The correlograms of DLOGBD and of DLOGITL



Correlogram of D(LOGITL)

Sample: 1/02/1990 1/02/2006
Included observations: 4174

	AC	PAC	Q-Stat	Prob
1	0.062	0.062	15.845	0.000
2	0.026	0.022	18.610	0.000
3	-0.013	-0.016	19.346	0.000
4	0.051	0.053	30.418	0.000
5	-0.029	-0.035	33.960	0.000
6	-0.009	-0.008	34.285	0.000
7	-0.014	-0.010	35.114	0.000
8	0.027	0.025	38.182	0.000
9	0.012	0.013	38.822	0.000
10	0.034	0.031	43.543	0.000
11	-0.005	-0.008	43.656	0.000
12	0.005	0.001	43.752	0.000
13	0.007	0.008	43.944	0.000
14	0.033	0.030	48.645	0.000
15	0.043	0.043	56.428	0.000
16	0.009	0.002	56.765	0.000
17	-0.022	-0.024	58.733	0.000
18	-0.023	-0.024	60.975	0.000
19	-0.026	-0.025	63.864	0.000
20	0.041	0.048	71.056	0.000
21	-0.017	-0.017	72.224	0.000
22	-0.033	-0.034	76.758	0.000
23	-0.014	-0.011	77.624	0.000
24	-0.011	-0.019	78.094	0.000
25	-0.007	-0.003	78.294	0.000
26	-0.031	-0.026	82.375	0.000
27	0.017	0.024	83.620	0.000
28	0.009	0.005	83.935	0.000
29	0.013	0.007	84.647	0.000
30	0.010	0.008	85.031	0.000

Correlogram of D(LOGPT)

Sample: 1/02/1990 1/02/2006
Included observations: 4174

	AC	PAC	Q-Stat	Prob
1	0.140	0.140	81.332	0.000
2	0.035	0.016	86.369	0.000
3	0.009	0.002	86.678	0.000
4	0.055	0.054	99.375	0.000
5	-0.012	-0.028	99.994	0.000
6	0.010	0.013	100.39	0.000
7	0.022	0.021	102.50	0.000
8	0.043	0.034	110.10	0.000
9	0.011	0.002	110.60	0.000
10	0.039	0.035	117.01	0.000
11	0.028	0.016	120.23	0.000
12	-0.004	-0.015	120.28	0.000
13	0.002	0.005	120.31	0.000
14	0.029	0.025	123.85	0.000
15	0.046	0.038	132.89	0.000
16	-0.020	-0.034	134.58	0.000
17	-0.020	-0.018	136.30	0.000
18	0.026	0.027	139.09	0.000
19	0.005	-0.007	139.18	0.000
20	0.038	0.042	145.23	0.000
21	0.039	0.028	151.70	0.000
22	0.025	0.008	154.39	0.000
23	-0.019	-0.026	155.92	0.000
24	0.002	0.004	155.93	0.000
25	0.047	0.044	165.09	0.000
26	0.022	0.008	167.09	0.000
27	0.009	0.008	167.42	0.000
28	0.031	0.022	171.49	0.000
29	0.031	0.013	175.49	0.000
30	0.031	0.022	179.51	0.000

Figure 2.6: The correlograms of DLOGGR and of DLOGPT.

In this way, after differencing the LOGBD, LOGGR, LOGITL and LOGPT time series, we obtain stationary time series.

2.2 – THE ARIMA(*p,d,q*) MODEL SELECTION

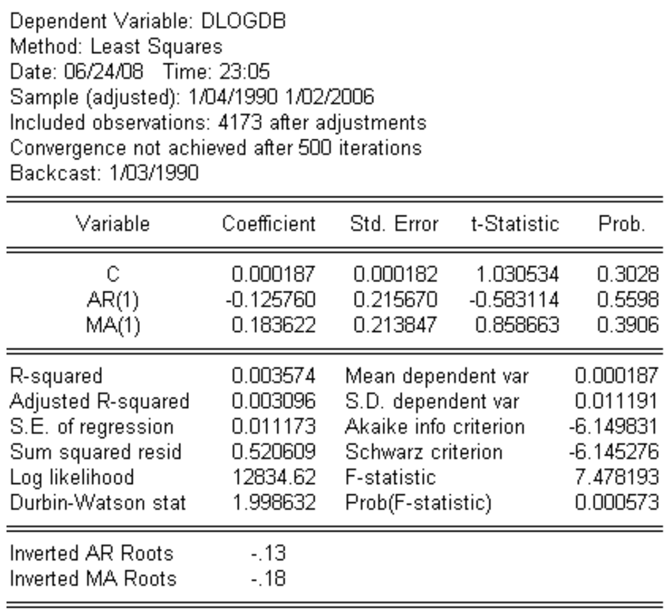
If one extends the model by allowing the AR polynomial to have 1 as a characteristic root, then the model becomes the ARIMA model. As it was seen, the four time series are nonstationary and thid fact implies to use the first-difference form data. Now, the time series are integrated of order *1*, i.e., *I(1),* which implies that we will have *d=1* in ARIMA*(p,d,q)* model for all time series, even if we consider a model with drift and/or with trend or without one or both.

The next step is to decide how many autoregressive *(p)* and moving average *(q)* parameters are necessary to give up an effective but still parsimonious[1] model of the process. The correlograms given in last two figures enable us to find these parameters. The decision is not straightforward and in less typical cases requires not only experience but also a good deal of experimentation with alternative models.

2.2.1 –ARIMA(*p,1,q*) model selection for the Stock price index of Germany

On DLOGBD correlogram, present in Figure 2.5, it is possible to note significant correlation and PAC at lag 1. In *Eviews,* it was run the equation specified as DLOGBD C AR(1) MA(1) (which estimates the model with a nonlinear algorithm, and helps controlling for nonlinearities if they exist in the LOGBD)

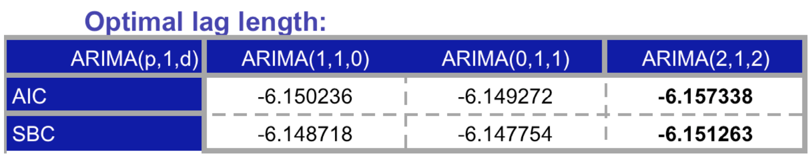
[1] By a parsimonious model we mean that it has the fewest parameters and greatest number of degrees of freedom among all models that fit the data. In practice, the numbers of the *p* or *q* parameters are very rarely greater than 2.



```
Dependent Variable: DLOGDB
Method: Least Squares
Date: 06/24/08   Time: 23:05
Sample (adjusted): 1/04/1990 1/02/2006
Included observations: 4173 after adjustments
Convergence not achieved after 500 iterations
Backcast: 1/03/1990
==========================================================================
        Variable        Coefficient    Std. Error    t-Statistic    Prob.
==========================================================================
           C             0.000187      0.000182      1.030534     0.3028
         AR(1)          -0.125760      0.215670     -0.583114     0.5598
         MA(1)           0.183622      0.213847      0.858663     0.3906
==========================================================================
R-squared                0.003574    Mean dependent var        0.000187
Adjusted R-squared       0.003096    S.D. dependent var        0.011191
S.E. of regression       0.011173    Akaike info criterion    -6.149831
Sum squared resid        0.520609    Schwarz criterion        -6.145276
Log likelihood           12834.62    F-statistic               7.478193
Durbin-Watson stat       1.998632    Prob(F-statistic)         0.000573
==========================================================================
Inverted AR Roots          -.13
Inverted MA Roots          -.18
==========================================================================
```

Figure 2.7: ARIMA(1,1,1) estimation output

Despite AR and MA roots are inside the unit circle, so the stability condition is satisfied, the coefficients are all insignificants. Then it was estimated the ARIMA(1,1,0) and ARIMA(0,1,1), in both cases only intercept was insignificant so it was removed from the models.

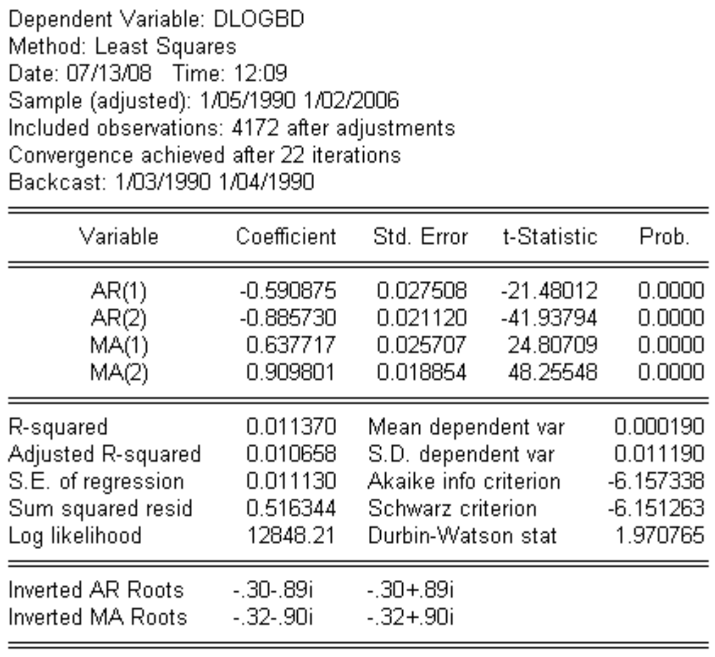


Optimal lag length:

ARIMA(p,1,d)	ARIMA(1,1,0)	ARIMA(0,1,1)	ARIMA(2,1,2)
AIC	-6.150236	-6.149272	**-6.157338**
SBC	-6.148718	-6.147754	**-6.151263**

Table 2.2: The *Eviews* values for the Information criteria for ARIMA(*p,1,q*) for DLOGBD

Information criteria favor an ARIMA(2,1,2) specification although the marginal change is very small. These results are not showed for ARIMA(2,1,0) neither for ARIMA(2,1,1), neither for ARIMA(0,1,2) or ARIMA(1,1,2) because their estimation reveals insignificant coefficients.



```
Dependent Variable: DLOGBD
Method: Least Squares
Date: 07/13/08   Time: 12:09
Sample (adjusted): 1/05/1990 1/02/2006
Included observations: 4172 after adjustments
Convergence achieved after 22 iterations
Backcast: 1/03/1990 1/04/1990

      Variable        Coefficient    Std. Error    t-Statistic     Prob.

        AR(1)          -0.590875      0.027508     -21.48012      0.0000
        AR(2)          -0.885730      0.021120     -41.93794      0.0000
        MA(1)           0.637717      0.025707      24.80709      0.0000
        MA(2)           0.909801      0.018854      48.25548      0.0000

R-squared               0.011370    Mean dependent var      0.000190
Adjusted R-squared      0.010658    S.D. dependent var      0.011190
S.E. of regression      0.011130    Akaike info criterion  -6.157338
Sum squared resid       0.516344    Schwarz criterion      -6.151263
Log likelihood          12848.21    Durbin-Watson stat      1.970765

Inverted AR Roots      -.30-.89i     -.30+.89i
Inverted MA Roots      -.32-.90i     -.32+.90i
```

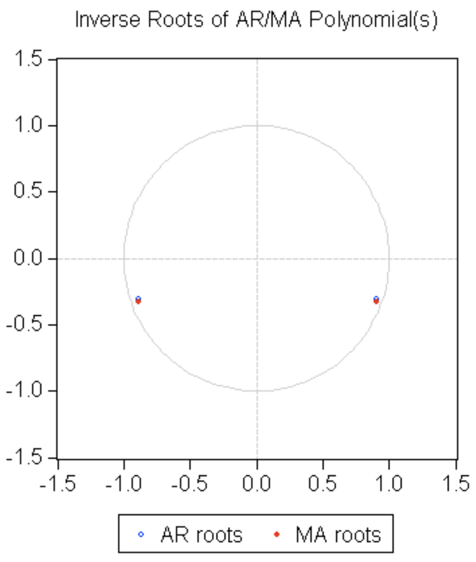


Figure 2.8: ARIMA(2,1,2) estimation outputs for DLOGBD

Although the small R-squared, here, all coefficients are significant and the stability is confirmed (AR and MA roots are inside the unit circle).

Let Y_t denote the DLOGBD, then our tentatively identified ARIMA model is

$$\hat{Y}_t = -0{,}5909Y_{t-1} - 0{,}8857Y_{t-2} + 0{,}6377\varepsilon_{t-1} + 0{,}9098\varepsilon_{t-2} \qquad (2.2.1)$$

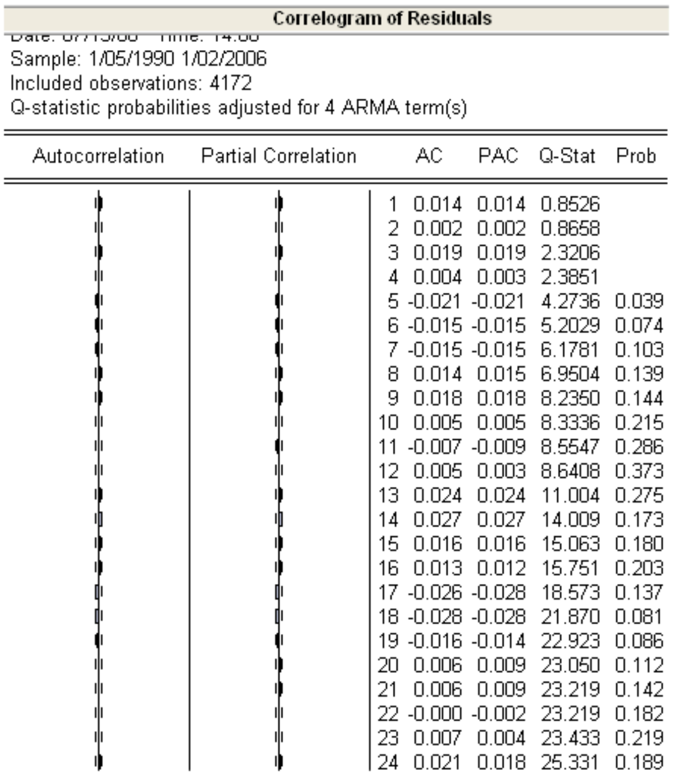
$$\text{se} = (0{,}0275) \quad (0{,}0211) \quad (0{,}0257) \quad (0{,}0189)$$

$$\text{t} = (-21{,}48) \quad (-41{,}938) \quad (21{,}807) \quad (48{,}255)$$

$$R^2 = 1{,}1\%.$$

Diagnostic testing for residual autocorrelation

An requirement for an ARIMA time series model is that the estimated residual time series is approximately white noise.

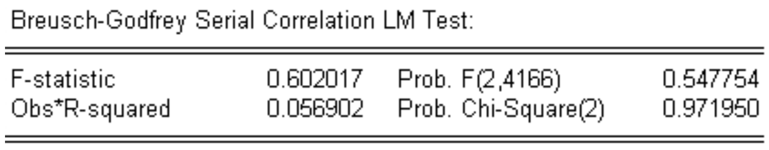


```
Correlogram of Residuals
Date: [illegible]   Time: [illegible]
Sample: 1/05/1990 1/02/2006
Included observations: 4172
Q-statistic probabilities adjusted for 4 ARMA term(s)

 Autocorrelation   Partial Correlation      AC     PAC    Q-Stat   Prob

                                        1  0.014  0.014  0.8526
                                        2  0.002  0.002  0.8658
                                        3  0.019  0.019  2.3206
                                        4  0.004  0.003  2.3851
                                        5 -0.021 -0.021  4.2736  0.039
                                        6 -0.015 -0.015  5.2029  0.074
                                        7 -0.015 -0.015  6.1781  0.103
                                        8  0.014  0.015  6.9504  0.139
                                        9  0.018  0.018  8.2350  0.144
                                       10  0.005  0.005  8.3336  0.215
                                       11 -0.007 -0.009  8.5547  0.286
                                       12  0.005  0.003  8.6408  0.373
                                       13  0.024  0.024  11.004  0.275
                                       14  0.027  0.027  14.009  0.173
                                       15  0.016  0.016  15.063  0.180
                                       16  0.013  0.012  15.751  0.203
                                       17 -0.026 -0.028  18.573  0.137
                                       18 -0.028 -0.028  21.870  0.081
                                       19 -0.016 -0.014  22.923  0.086
                                       20  0.006  0.009  23.050  0.112
                                       21  0.006  0.009  23.219  0.142
                                       22 -0.000 -0.002  23.219  0.182
                                       23  0.007  0.004  23.433  0.219
                                       24  0.021  0.018  25.331  0.189
```

Figure 2.9: Correlogram of residuals of ARIMA(2,1,2) model with DLOGBD

Residual autocorrelations are pretty small, the correlograms of both autocorrelation and partial autocorrelation give the impression that the residuals estimated from ARIMA(2,1,2) are nearly a white noise.

The Ljung-Box statistics and their *p*-values also look good. Thus, the null hypothesis that there is no residual autocorrelation is not rejected. The same result is obtain when applied the Breusch-Godfrey (LM) test, we do reject the null hypothesis (there is no serial correlation in error terms):



```
Breusch-Godfrey Serial Correlation LM Test:

F-statistic          0.602017    Prob. F(2,4166)          0.547754
Obs*R-squared        0.056902    Prob. Chi-Square(2)      0.971950
```

Figure 2.10: The results of Breusch-Godfrey (LM) test

Diagnostic testing for normality of residuals

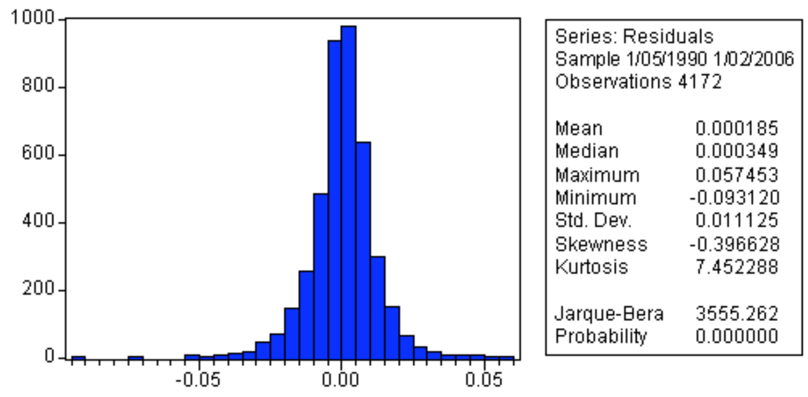
To facilitate the interpretation, for example for the parameter values, the estimated residuals should be approximately normal.

Although the previous tests shown that, the residuals are nearly white noise, we can also check the Jarque-Bera test, which is a goodness-of-fit measure of departure from normality, based on the sample kurtosis (*K*) and skewness (*S*). The test statistic Jarque-Bera (JB) is defined as

$$JB = \frac{n}{6}\left(S^2 + \frac{(K-3)^2}{4}\right), \tag{2.2.2}$$

where *n* is the number of observations.

The statistic JB has an asymptotic, χ_2^2, chi-square distribution with two degrees of freedom, and can be used to test the null hypothesis, that is, the data are from a normal distribution. The null hypothesis is a joint hypothesis of the skewness being zero and the excess kurtosis being 0, since samples from a normal distribution have an expected skewness of 0 and an expected excess kurtosis of 0 (which is the same as a kurtosis of 3).



```
Series: Residuals
Sample 1/05/1990 1/02/2006
Observations 4172

Mean          0.000185
Median        0.000349
Maximum       0.057453
Minimum      -0.093120
Std. Dev.     0.011125
Skewness     -0.396628
Kurtosis      7.452288

Jarque-Bera   3555.262
Probability   0.000000
```

Figure 2.11: Histogram and some statistics of residuals of ARIMA(2,1,2) model with DLOGDB

The null hypothesis of Jarque-Bera test is rejected. The rejection of the normality null hypothesis may indicate that there are some outlying observations or that the error process is not homoskedastic.

After several tests, there was no ARIMA(p,1,q) model better than the previous one, despite the rejection of normality.

2.2.2 –ARIMA(*p*,1,*q*) model selection for the Stock price index of Greece

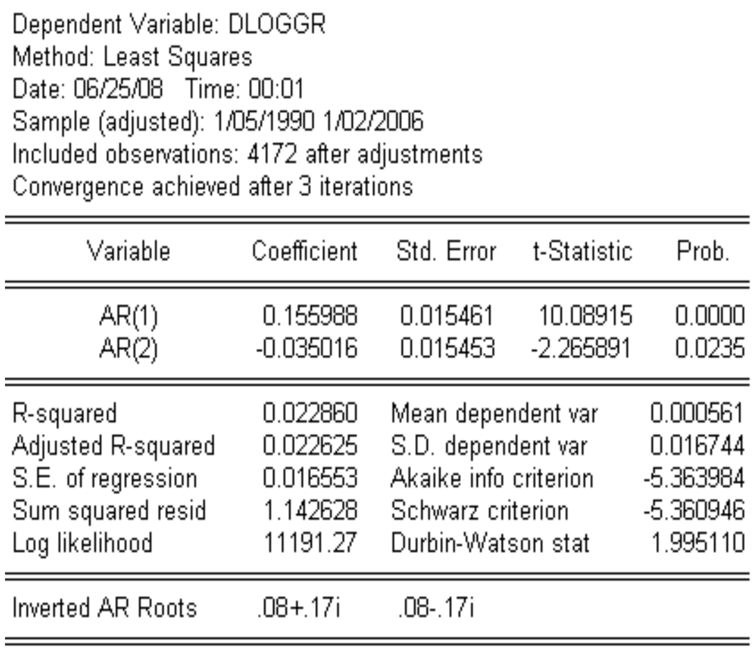
Several equations were run in Eviews in order to find which parameters *p* and *q* performed better the ARIMA(*p*,1,*q*) model for the DLOGGR time series.

Optimal lag length:

ARIMA(p,1,d)	ARIMA(1,1,0)	ARIMA(2,1,0)	ARIMA(1,1,2)	ARIMA(0,1,1)
AIC	-5.361491	-5.363984	**-5.366595**	-5.362092
SBC	-5.359973	-5.360946	**-5.362040**	-5.360574

Table 2.3: The *Eviews* values for the Information criteria for ARIMA(*p,1,q*) for DLOGGR

Although the ARIMA(1,1,2) looks good it is no better than ARIMA(2,1,0) because as it has an inverse autoregressive root of 0,99 and an inverse moving-average root of 0,99, so they are likely to be statistically indistinguishable from those of our earlier-estimate ARIMA(0,1,1), and this has worst values in AIC and SBC.



```
Dependent Variable: DLOGGR
Method: Least Squares
Date: 06/25/08   Time: 00:01
Sample (adjusted): 1/05/1990 1/02/2006
Included observations: 4172 after adjustments
Convergence achieved after 3 iterations

        Variable        Coefficient   Std. Error   t-Statistic    Prob.

          AR(1)          0.155988     0.015461     10.08915      0.0000
          AR(2)         -0.035016     0.015453     -2.265891     0.0235

R-squared                0.022860   Mean dependent var      0.000561
Adjusted R-squared       0.022625   S.D. dependent var      0.016744
S.E. of regression       0.016553   Akaike info criterion  -5.363984
Sum squared resid        1.142628   Schwarz criterion      -5.360946
Log likelihood           11191.27   Durbin-Watson stat      1.995110

Inverted AR Roots       .08+.17i     .08-.17i
```

Figure 2.12: estimation output of ARIMA(2,1,0) with DLOGGR

Let Y_t denote the DLOGGR, then our tentatively identified ARIMA model is

$$\hat{Y}_t = 0{,}156Y_{t-1} - 0{,}035Y_{t-2} \qquad (2.2.3)$$

se = (0,01546) (0,01545)

t = (10,0892) (-2,2659)

R^2 =2,3%.

Very similar results were obtained on the diagnostic tests for residual time series, as it will be discussed in a subsequent chapter.

2.2.3 –ARIMA(*p*,1,*q*) model selection for the Stock price index of Italy

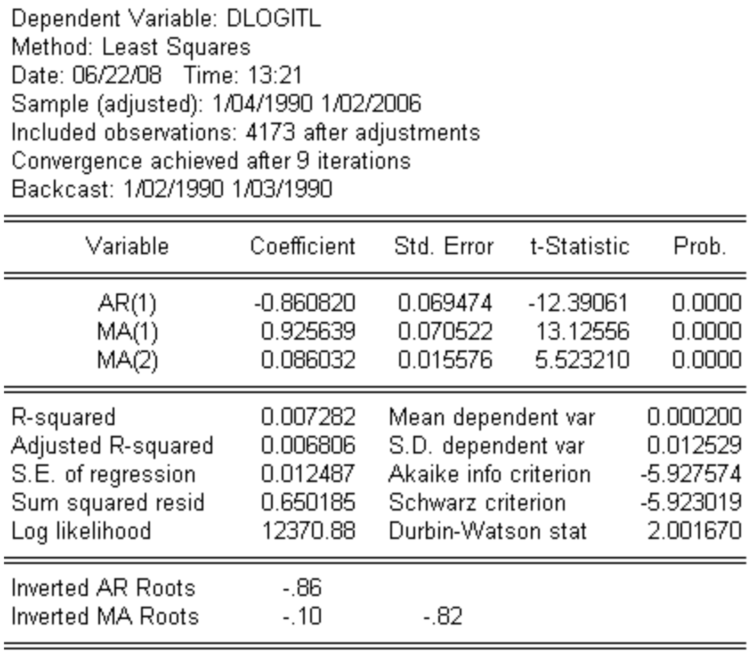
As we can see in the following table, the information criteria favor the ARIMA(1,1,0) and the ARIMA(1,1,2) to model the DLOGITL. After observing some points such as the correlogram of residuals, the results of Breusch-Godfrey (LM) and the Ajusted R-squared we decided for the ARIMA(1,1,2) model.

Optimal lag length:

ARIMA(p,1,d)	ARIMA(2,1,1)	ARIMA(1,1,0)	ARIMA(1,1,2)	ARIMA(0,1,1)
AIC	-5.927273	-5.924799	**-5.927574**	-5.924781
SBC	-5.922717	**-5.923281**	-5.923019	-5.923263

Table 2.4: The *Eviews* values for the Information criteria for ARIMA(*p,1,q*) for DLOGITL

For DLOGITL the fitted model was an ARIMA(1,1,2).



```
Dependent Variable: DLOGITL
Method: Least Squares
Date: 06/22/08   Time: 13:21
Sample (adjusted): 1/04/1990 1/02/2006
Included observations: 4173 after adjustments
Convergence achieved after 9 iterations
Backcast: 1/02/1990 1/03/1990
==========================================================
        Variable      Coefficient   Std. Error  t-Statistic   Prob.
==========================================================
          AR(1)       -0.860820     0.069474   -12.39061    0.0000
          MA(1)        0.925639     0.070522    13.12556    0.0000
          MA(2)        0.086032     0.015576    5.523210    0.0000
==========================================================
R-squared              0.007282   Mean dependent var     0.000200
Adjusted R-squared     0.006806   S.D. dependent var     0.012529
S.E. of regression     0.012487   Akaike info criterion -5.927574
Sum squared resid      0.650185   Schwarz criterion     -5.923019
Log likelihood         12370.88   Durbin-Watson stat     2.001670
==========================================================
Inverted AR Roots         -.86
Inverted MA Roots         -.10          -.82
==========================================================
```

Figure 2.13: ARIMA(1,1,2) estimation output with DLOGITL

Let Y_t denote the DLOGITL, then our tentatively identified ARIMA model is

$$\hat{Y}_t = -0{,}8608Y_{t-1} + 0{,}086\varepsilon_{t-1} + 0{,}9256\varepsilon_{t-2} \qquad (2.2.4)$$

se = (0,0695)   (0,0705)   (0,0016)

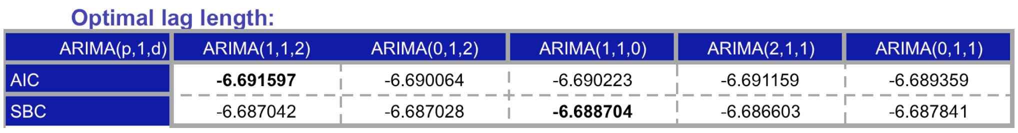
t = (-12,3906)   (13,1256)   (5,5232)

R^2 =0,7%.

Once again, the results obtained on the diagnostic tests for residual time series were, very similar to the previews.

2.2.4 – ARIMA(*p*,1,*q*) model selection for the Stock price index of Portugal

In the case of Portugal Stock price index it appears to be appropriated the use of an ARIMA(1,1,2) and an ARIMA(1,1,0) model as the best in terms of the information criteria.

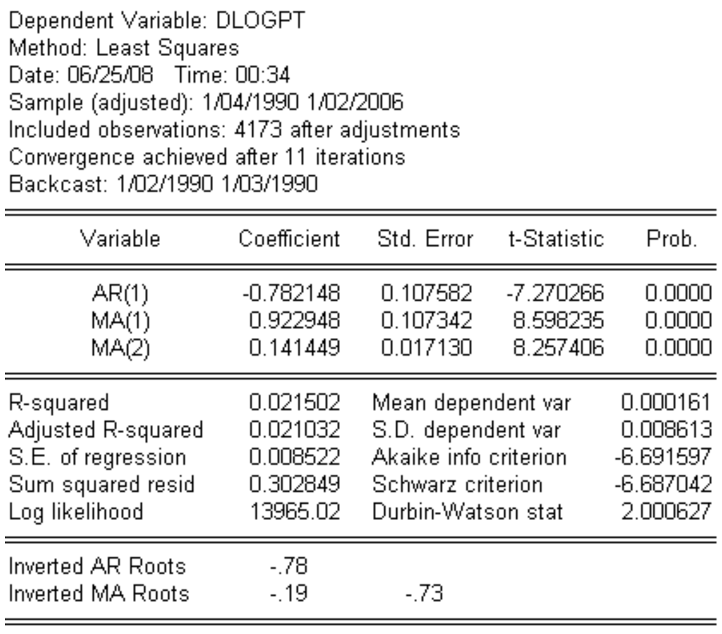


Optimal lag length:

ARIMA(p,1,d)	ARIMA(1,1,2)	ARIMA(0,1,2)	ARIMA(1,1,0)	ARIMA(2,1,1)	ARIMA(0,1,1)
AIC	**-6.691597**	-6.690064	-6.690223	-6.691159	-6.689359
SBC	-6.687042	-6.687028	**-6.688704**	-6.686603	-6.687841

Table 2.5: The *Eviews* values for the Information criteria for ARIMA(*p,1,q*) for DLOGPT

After observing some points such as the stability condition, that is satisfied in both of the referred models, we decided for the ARIMA(1,1,2) model which presents a bigger Adjusted R-squared.



```
Dependent Variable: DLOGPT
Method: Least Squares
Date: 06/25/08   Time: 00:34
Sample (adjusted): 1/04/1990 1/02/2006
Included observations: 4173 after adjustments
Convergence achieved after 11 iterations
Backcast: 1/02/1990 1/03/1990
=========================================================================
        Variable       Coefficient    Std. Error    t-Statistic    Prob.
=========================================================================
          AR(1)         -0.782148      0.107582     -7.270266     0.0000
          MA(1)          0.922948      0.107342      8.598235     0.0000
          MA(2)          0.141449      0.017130      8.257406     0.0000
=========================================================================
R-squared                0.021502    Mean dependent var       0.000161
Adjusted R-squared       0.021032    S.D. dependent var       0.008613
S.E. of regression       0.008522    Akaike info criterion   -6.691597
Sum squared resid        0.302849    Schwarz criterion       -6.687042
Log likelihood           13965.02    Durbin-Watson stat       2.000627
=========================================================================
Inverted AR Roots          -.78
Inverted MA Roots          -.19          -.73
=========================================================================
```

Figure 2.14: ARIMA(1,1,2) estimation output with DLOGPT

Let Y_t denote the DLOGPT, then our tentatively identified ARIMA model is

$$\hat{Y}_t = -0{,}7821 Y_{t-1} + 0{,}9229\varepsilon_{t-1} + 0{,}1414\varepsilon_{t-2} \qquad (2.2.5)$$

se = (0,1076)   (0,1073)   (0,0171)

t = (-7,2701)   (8,5982)   (8,2574)

R^2 =2,2%.

Diagnostic tests for residual time series reveal very similar results to the three previews models.

2.3 – THE ANN MODEL SELECTION

After using *Eviews* in Box-Jenkins ARIMA model, we will use *Matlab* in ANN modelling, through it's M-files, which are commands of *Matlab* that are stored as ordinary text files.

An M-file can be either a function with input and output variables or a list of commands. That was what S. Mohammadi and H. Abbasi-Nejad, both from faculty of Economics of University of Tehran, drew in ANNEARLY M-file.

This M-file forecasts time series with minimum Root Mean Squares Error (RMSE) used as stopping criteria in the network. The RMSE is a kind of generalized standard deviation. It pops up whenever you look for differences between target and the output. It is determined by the root square of Mean Square Error (MSE). Smaller values of RMSE indicate higher accuracy in forecasting.

A measure of dispersion between the target and the output is the MSE:

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(T_t - Y_t)^2, \tag{2.2.6}$$

where T_t is the actual or targeted output value of the t^{th} iteration and, Y_t is the computed output also of the t^{th} iteration.

Through *newff*, MATLAB command's, it is created a feedforward backpropagation network. This command is defined as in (2.2.7):

$$newff = (Y, [S_1\, S_2 \ldots S_n], \{TF_1\, TF_2 \ldots TF_n\}, BTF, BLF, PF) \tag{2.2.7}$$

where

- Y is a time series that has a vertical vector form.
- S_i is the size of the i^{th} layer, for n layers.
- TF_i is the transfer function of the i^{th} layer, for n layers. It is used the 'tansig' , transfer function by default.
- BTF is the backpropagation network training function (default = 'trainlm').
- BLF is the backpropagation weight/bias learning function (default = 'learngdm').
- PF is the performance function (grafic), by default is used 'mse' and returns an n layer feed-forward backpropagation network.

In ANNEARLY M-file the authors initialized the net, as a function dependent of the number of lags (lag) and hidden layers (lay), with the subsequent command. The user, such as, the learning rate parameter, defines these two arguments.

$$newff = (PR(1:lag,:),[lay \ \ 1],\{'\tan sig','purelin'\},'trainlm') \quad (2.2.8)$$

where

- $PR(1:lag,:)$ define the limits of entire $pattern$,
- $[lay \ \ 1]$ define the number of neurons in each layer,
- $\{'\tan sig','purelin'\}$ are the transfer functions and
- $'trainlm'$ is the training function

The remaining training parameters were defined as follows:

- Ratio to increase learning rate =1,05
- Ratio to decrease learning rate =0,7
- Maximum number of epochs to train=1000
- Epochs between displays = 100
- Performance goal = $1e-5$

By the last, the user also defines the percentage of observations for training set and for be forecasted and finally the learning rate.

Here, in ANN, as we have to define a percentage of observations for training, we decided to have the same number of observations, 4175, as we have in ARIMA for training, so we had to increased the series in 464 observations. Thus, our input series have now 4639 observations, 90% for training and 10% for comparison in forecasting.

The learning rate is supposed constant throughout training. The performance of the algorithm is very sensitive to the proper setting of the learning rate. If the learning rate is set too high, the algorithm can oscillate and become unstable. If the learning rate is too small, the algorithm takes too long to converge.

The selection of hidden layers is more like an art rather than mathematical method. When the number of hidden layers units is small, the correlation of the output and input cannot be studied well and the errors increased. Moreover, when the number of hidden layers units is more than adequate, even an irrelevant noise is studied to the correlation of both, and the error

grows accordingly. There are some methods for obtain the number of hidden layer units, however, there is no general solution for this problem (Kermanshahi, *et al.,* 2002). Therefore, we decided to start with one hidden layer units and go to fifteen layers units.
Several models were run for the four series, considering different number of hidden layers, and different numbers of lags, through *annearly:*

annearly(y, maxlag, nhiden, trset, HPF,lr)

where

- *y* is our time series
- *maxlag* is the maximum number of lag that should be entered in model.
- *nhiden*, is the number of hidden layer units.
- *Trset* match to our 90% of observations for training
- *HPF* match to our 10% of observations that should be forecasted and
- *lr* is the learning rate.

2.3.1 – Feedforward backpropagation network for the stock price index of Germany

Beginning with one unit/neuron in the hidden layer, the ANNEARLY M-file was run considering one and then two lags, with different learning rates.

Considering the subsequent results, where were run forty-two times the ANNEARLY M-file, thus it were produced the same forty-two networks. The lowest MinRMSE in each run is showed in the following table, under the number of units/neurons in the hidden layer. In blue, we have the lowest MinRMSE of all forty-two runs.

Maximum number of Lags	1			2		
Learning Rate	0.01	0.05	0.1	0.01	0.05	0.1
minRMSE whith 1 unit in hidden layer	0.0037581419	0.0037581419	0.0037581419	0.0037581419	0.0037581419	0.0037581419
minRMSE whith 2 unit in hidden layer	0.0037279355	0.0037581419	0.0037504469	0.0037495728	0.0037406300	0.0037581419
minRMSE whith 3 unit in hidden layer	0.0037279936	0.0037222259	0.0037333224	0.0037308917	0.0037297729	0.0037380953
minRMSE whith 4 unit in hidden layer	0.0037311613	0.0037326867	0.0037285520	0.0037321193	0.0037268508	0.0037407152
minRMSE whith 5 unit in hidden layer	0.0037279509	0.0037279509	0.0037332598	0.0037355458	0.0037332931	0.0037264517
minRMSE whith 10 unit in hidden layer	0.0037150110	0.0037313202	0.0037351494	0.0037331096	0.0037230894	0.0037055845
minRMSE whith 15 unit in hidden layer	0.0037031753	0.0037292407	0.0037285428	0.0037201358	0.0037136935	0.0037289880

Table 2.6: Forecasting results of ANN model for LOGBD time series

As we can see in the previous table, the Minimum RMSE (minRMSE) doesn't change a lot when considered different lags and different learning rates. Although the lowest MinRMSE is obtained when consider the model with fifteen units in the hidden layer, one lag and the learning rate equal to 0.01.

Through these results we have the best model to forecast the LOGBD time series when we have the subsequent parameters:

annearly(LOGBD, 1, 15, 90, 10, 0.01)

It is find a very good performance of the ANN to predict the LOGBD series, between February 1st, 2006 and October 12th, 2007, 464 by using those parameters.

The Figure 2.15, show us that there is a minimum error between the original series (*yy*) and the forecasted one (*yhatopt*). Thus the residuals are very small, as we can see in the following figure, the majority are near to zero, falling into the interval $]-0.05, 0.05[$.

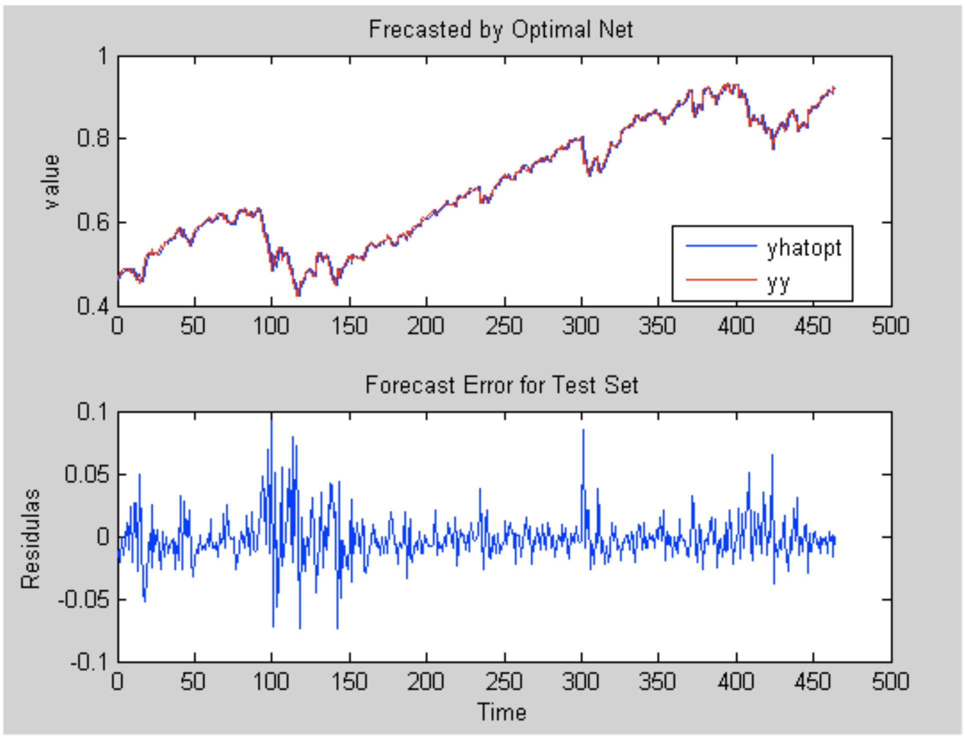


Figure 2.15: Forecasting outputs of LOGBD trough ANN – in *Matlab*

Although ANN does not oblige to have stationary time series, as in ARIMA, it was decided to run the ANNEARLY M-file considering $Y = DLOGBD$, that is, the same input time series as it was used in ARIMA.

Maximum number of Lags	1			2		
Learning Rate	0.01	0.05	0.1	0.01	0.05	0.1
minRMSE whith 1 unit in hidden layer	0.0370743256	0.0371934286	0.0369961692	0.0370175126	0.0371206381	0.0370109358
minRMSE whith 2 unit in hidden layer	0.0368174532	0.0369904157	0.0370082018	0.0370216480	0.0367962597	0.0370169906
minRMSE whith 3 unit in hidden layer	0.0370192490	0.0370219586	0.0370216443	0.0369073243	0.0369889428	0.0369023903
minRMSE whith 4 unit in hidden layer	0.0369318260	0.0369164060	0.0368519412	0.0367469585	0.0369396931	0.0367627548
minRMSE whith 5 unit in hidden layer	0.0369173301	0.0368843710	0.0369885765	0.0368559776	0.0369335455	0.0368575431
minRMSE whith 10 unit in hidden layer	0.0367879004	0.0368973957	0.0369015683	0.0366706737	0.0367912453	0.0368915034
minRMSE whith 15 unit in hidden layer	0.0366421473	0.0368176280	0.0367013897	0.0368061950	0.0368394277	0.0367878635

Table 2.7: Forecasting results of ANN model for DLOGBD time series

As we can see, in every networks the value of minRMSE is higher when considering the stationary time series, what is obvious because when we differencing a time series we lose some of the information wider laying in the original data.

Thus the accuracy obtained between the original series, $Y = DLOGBD$, (*yy*) and the forecasted one (*yhatopt*) is much lower than it was when considering $Y = LOGBD$, as we can confirm regarding the Figure 2.16. Consequently the residuals are falling into a bigger interval, with some escape tendency.

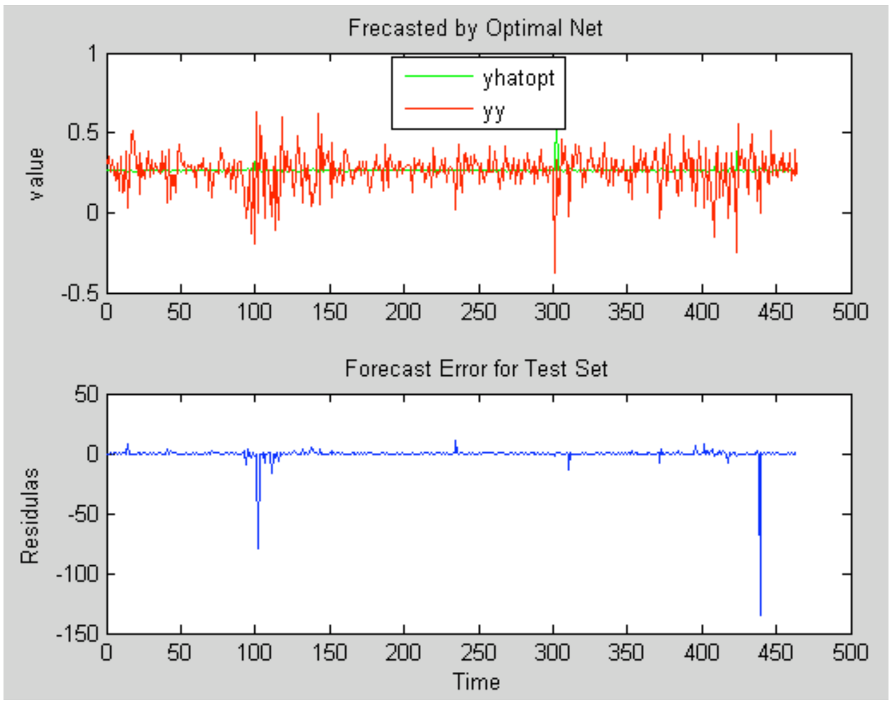
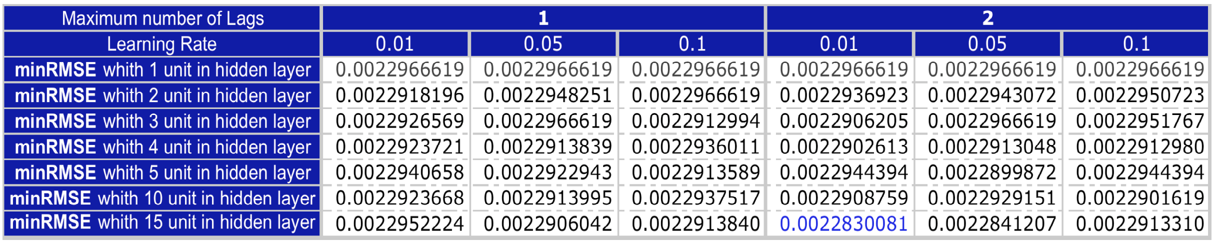


Figure 2.16: Forecasting outputs of DLOGBD trough ANN – in *Matlab*

2.3.2 – Feedforward backpropagation network for the Stock price index of Greece

For the case of Greece, the ANNEARLY M-file was run considering the same parameters as we used before, thus it were produced the same forty-two networks as for $Y = LOGBD$:



Maximum number of Lags	1			2		
Learning Rate	0.01	0.05	0.1	0.01	0.05	0.1
minRMSE whith 1 unit in hidden layer	0.0022966619	0.0022966619	0.0022966619	0.0022966619	0.0022966619	0.0022966619
minRMSE whith 2 unit in hidden layer	0.0022918196	0.0022948251	0.0022966619	0.0022936923	0.0022943072	0.0022950723
minRMSE whith 3 unit in hidden layer	0.0022926569	0.0022966619	0.0022912994	0.0022906205	0.0022966619	0.0022951767
minRMSE whith 4 unit in hidden layer	0.0022923721	0.0022913839	0.0022936011	0.0022902613	0.0022913048	0.0022912980
minRMSE whith 5 unit in hidden layer	0.0022940658	0.0022922943	0.0022913589	0.0022944394	0.0022899872	0.0022944394
minRMSE whith 10 unit in hidden layer	0.0022923668	0.0022913995	0.0022937517	0.0022908759	0.0022929151	0.0022901619
minRMSE whith 15 unit in hidden layer	0.0022952224	0.0022906042	0.0022913840	0.0022830081	0.0022841207	0.0022913310

Table 2.8: Forecasting results of ANN model for LOGGR time series

The minRMSE, illustrated in Table 2.8, was obtained when considering the network with a learning rate of 0.01, fifteen units in the hidden layer and two lags. Once more the minRMSE doesn't change a lot when considered different lags and different learning rates.

Regarding Figure 2.17 it is possible to notice that the residuals are too small, also quite small, converging to zero. In this way, there was obtained a good accuracy between the original series, $Y = LOGGR$, (*yy*) and the forecasted one (*yhatopt*).

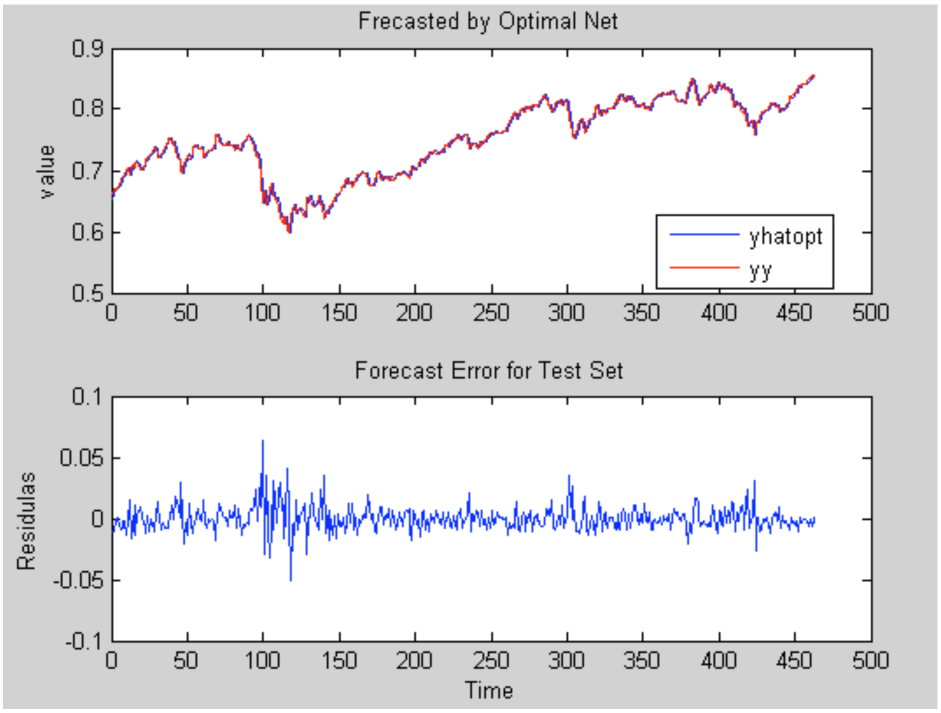


Figure 2.17: Forecasting outputs of LOGGR trough ANN – in *Matlab*

In order to have the same input time series as we had in ARIMA, the same forty-two networks were generated with the differencing time series, $Y = DLOGGR$, with the following results:

Maximum number of Lags	1			2		
Learning Rate	0.01	0.05	0.1	0.01	0.05	0.1
minRMSE whith 1 unit in hidden layer	0.0268651995	0.0269436546	0.0268769543	0.0268217099	0.0268515568	0.0268588373
minRMSE whith 2 unit in hidden layer	0.0268212650	0.0268103396	0.0268577151	0.0268566565	0.0268082251	0.0268623630
minRMSE whith 3 unit in hidden layer	0.0267903092	0.0268320810	0.0268561431	0.0268623979	0.0268520331	0.0268193285
minRMSE whith 4 unit in hidden layer	0.0268316981	0.0268316981	0.0268592776	0.0268225354	0.0268195664	0.0268477812
minRMSE whith 5 unit in hidden layer	0.0268324912	0.0268473766	0.0268398434	0.0268193324	0.0268585580	0.0267718126
minRMSE whith 10 unit in hidden layer	0.0267528630	0.0267359970	0.0266451021	0.0267320789	0.0267524435	0.0266886953
minRMSE whith 15 unit in hidden layer	0.0267703382	0.0266818265	0.0266844722	0.0267046163	0.0267164327	0.0267037432

Table 2.9: Forecasting results of ANN model for DLOGGR time series

The value of minRMSE is always bigger when considering $Y = DLOGGR$ instead of $Y = LOGGR$ in the network with the same parameters. Which means that there is a lower accuracy in forecasting, as it is possible to confirm in the following figure.

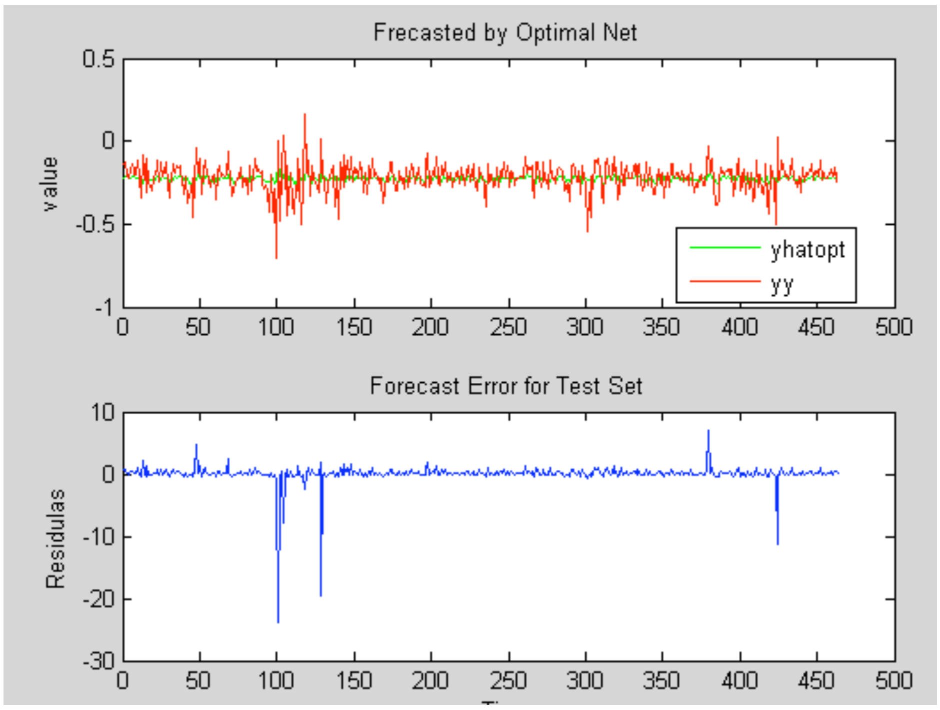


Figure 2.18: Forecasting outputs of DLOGGR trough ANN – in *Matlab*

2.3.3 – Feedforward backpropagation network for the Stock price index of Italy

Once again, using the same parameters, the forty-two networks were now generate with $y = LOGITL$.

Maximum number of Lags	1			2		
Learning Rate	0.01	0.05	0.1	0.01	0.05	0.1
minRMSE whith 1 unit in hidden layer	0.0030374649	0.0030374649	0.0030374649	0.0030374649	0.0030374649	0.0030374649
minRMSE whith 2 unit in hidden layer	0.0030348884	0.0030335610	0.0030355917	0.0030374649	0.0030334513	0.0030322501
minRMSE whith 3 unit in hidden layer	0.0030314956	0.0030196713	0.0030242598	0.0030276753	0.0030267950	0.0030248832
minRMSE whith 4 unit in hidden layer	0.0030262959	0.0030317621	0.0030329971	0.0030241501	0.0030331922	0.0030331922
minRMSE whith 5 unit in hidden layer	0.0030272960	0.0030313846	0.0030287432	0.0030204282	0.0030305123	0.0030149285
minRMSE whith 10 unit in hidden layer	0.0030278397	0.0030277002	0.0030269911	0.0030274892	0.0030237725	0.0030177334
minRMSE whith 15 unit in hidden layer	0.0030275814	0.0030282962	0.0030286239	0.0030299350	0.0030193967	0.0030214669

Table 2.10: Forecasting results of ANN model for LOGITL time series

Regarding these results we find the best model to forecast the LOGITL time series when we consider five units in the hidden layer, two lags and the learning rate equal to 0.1:

annearly(LOGITL, 2, 5, 90, 10, 0.1)

Once again the residuals obtained are very close to zero. In this way, there was obtained a good forecasting accuracy:

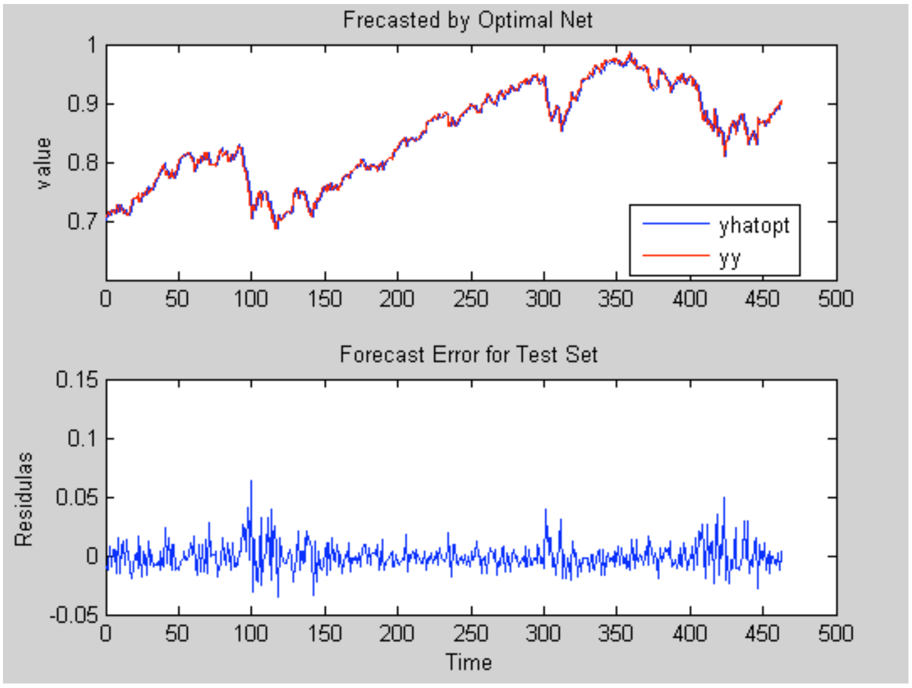


Figure 2.19: Forecasting outputs of LOGITL trough ANN – in *Matlab*

Different results were obtained when considering, in the same networks, the differencing time series of LOGITL, $Y = DLOGITL$. This means higher values for minRMSE, consequently worst evaluation on the forecasting results accuracy.

Maximum number of Lags	1			2		
Learning Rate	0.01	0.05	0.1	0.01	0.05	0.1
minRMSE whith 1 unit in hidden layer	0.0357775360	0.0353564913	0.0357490392	0.0353602808	0.0356719202	0.0357385855
minRMSE whith 2 unit in hidden layer	0.0354148590	0.0354148590	0.0353556742	0.0353622173	0.0353783944	0.0355085865
minRMSE whith 3 unit in hidden layer	0.0356777133	0.0355063888	0.0354804866	0.0354605317	0.0353938299	0.0354851275
minRMSE whith 4 unit in hidden layer	0.0354166262	0.0355092420	0.0354964646	0.0354964646	0.0353558337	0.0355182464
minRMSE whith 5 unit in hidden layer	0.0355153579	0.0353597239	0.0354048547	0.0354247332	0.0354042234	0.0355094184
minRMSE whith 10 unit in hidden layer	0.0354981898	0.0353449708	0.0354949252	0.0354947024	0.0353629143	0.0353293328
minRMSE whith 15 unit in hidden layer	0.0353951597	0.0354042845	0.0354362303	0.0353760425	0.0354273835	0.0352166198

Table 2.11: Forecasting results of ANN model for DLOGITL time series

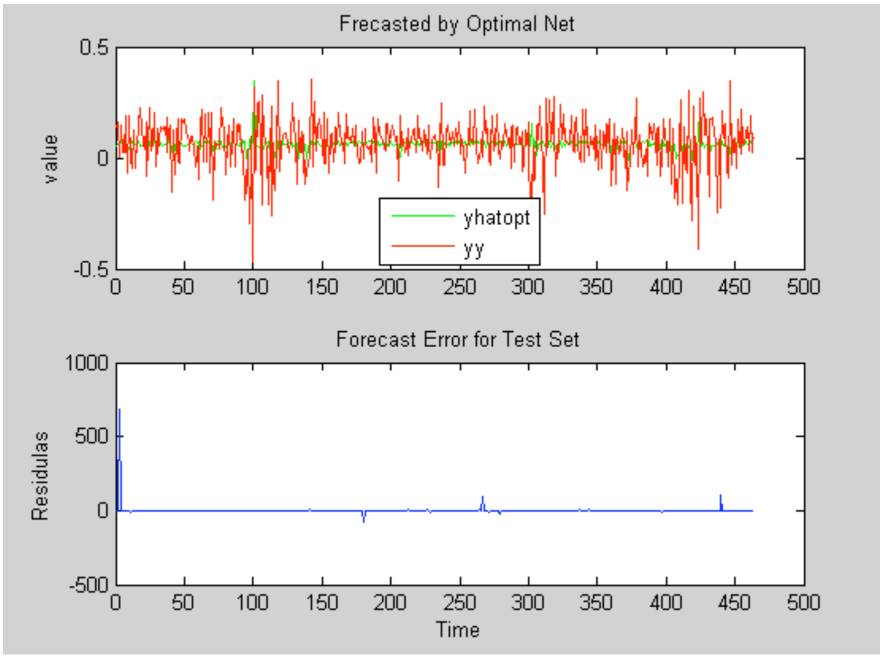


Figure 2.20: Forecasting outputs of DLOGITL trough ANN – in *Matlab*

2.3.4 – Feedforward backpropagation network for the Stock price index of Portugal

In order to obtain the best network to forecast LOGPT, we compared the minRMSE achieved in each of the forty-two networks:

Maximum number of Lags	1			2		
Learning Rate	0.01	0.05	0.1	0.01	0.05	0.1
minRMSE whith 1 unit in hidden layer	0.0027289261	0.0027289261	0.0027289276	0.0027114706	0.0027114706	0.0027114706
minRMSE whith 2 unit in hidden layer	0.0027289261	0.0027289261	0.0027289261	0.0026928459	0.0027096774	0.0027004639
minRMSE whith 3 unit in hidden layer	0.0026841758	0.0027289261	0.0027094148	0.0027060842	0.0027114699	0.0026732878
minRMSE whith 4 unit in hidden layer	0.0027289261	0.0026784847	0.0027104822	0.0027045744	0.0026647336	0.0027041924
minRMSE whith 5 unit in hidden layer	0.0027289261	0.0027289261	0.0027170144	0.0027027949	0.0027288678	0.0026637323
minRMSE whith 10 unit in hidden layer	0.0027170145	0.0027019084	0.0026597739	0.0027114706	0.0026734408	0.0026629642
minRMSE whith 15 unit in hidden layer	0.0026329884	0.0027289261	0.0027289261	0.0026887206	0.0027114706	0.0027114706

Table 2.12: Forecasting results of ANN model for LOGPT time series

It was select the network with fifteen units in the hidden layer, one lag and a learning rate equal to 0.01, as the best network to forecast the LOGPT time series:

annearly(LOGPT, 1, 15, 90, 10, 0.1)

Figure 2.12, shows the forecasting results using the network with lower value of minRMSE, and the behavior of the residuals.

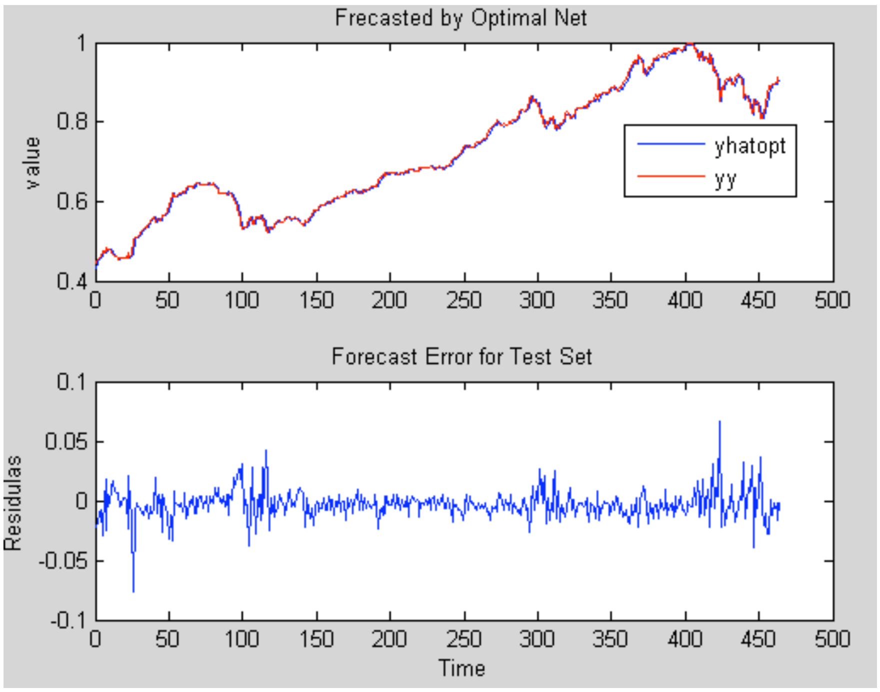


Figure 2.21: Forecasting outputs of LOGPT trough ANN – in *Matlab*

Once again, it is obtained a good performance on the forecasting results using a feedforward backpropagation network for the $Y = LOGPT$ time series.

In order to have the same input time series as we had in ARIMA, the same forty-two feedforward backpropagation networks were generated with the differencing time series, $Y = DLOGPT$, with the following results:

Maximum number of Lags	1			2		
Learning Rate	0.01	0.05	0.1	0.01	0.05	0.1
minRMSE whith 1 unit in hidden layer	0.0305449206	0.0305448635	0.0305289964	0.0304998271	0.0304608693	0.0305025050
minRMSE whith 2 unit in hidden layer	0.0304446827	0.0304292602	0.0304309646	0.0304472279	0.0302000725	0.0304375047
minRMSE whith 3 unit in hidden layer	0.0304756429	0.0303998379	0.0304872835	0.0302627599	0.0303047586	0.0301384593
minRMSE whith 4 unit in hidden layer	0.0303759405	0.0303357729	0.0303880441	0.0301756701	0.0302314717	0.0302752796
minRMSE whith 5 unit in hidden layer	0.0303413257	0.0302912518	0.0302592876	0.0302847697	0.0302761010	0.0302506541
minRMSE whith 10 unit in hidden layer	0.0302665651	0.0303128599	0.0302809394	0.0301230384	0.0302729317	0.0300454026
minRMSE whith 15 unit in hidden layer	0.0303354261	0.0302845011	0.0302578903	0.0300047016	0.0300134613	0.0300134613

Table 2.13: Forecasting results of ANN model for DLOGPT time series

The minRMSE is always higher considering the differencing time series, $Y = DLOGPT$, when compared to the minRMSE obtained with the $Y = LOGPT$ time series. Therefore the results on the forecasting accuracy are worst, as can be seen in Figure 2.20.

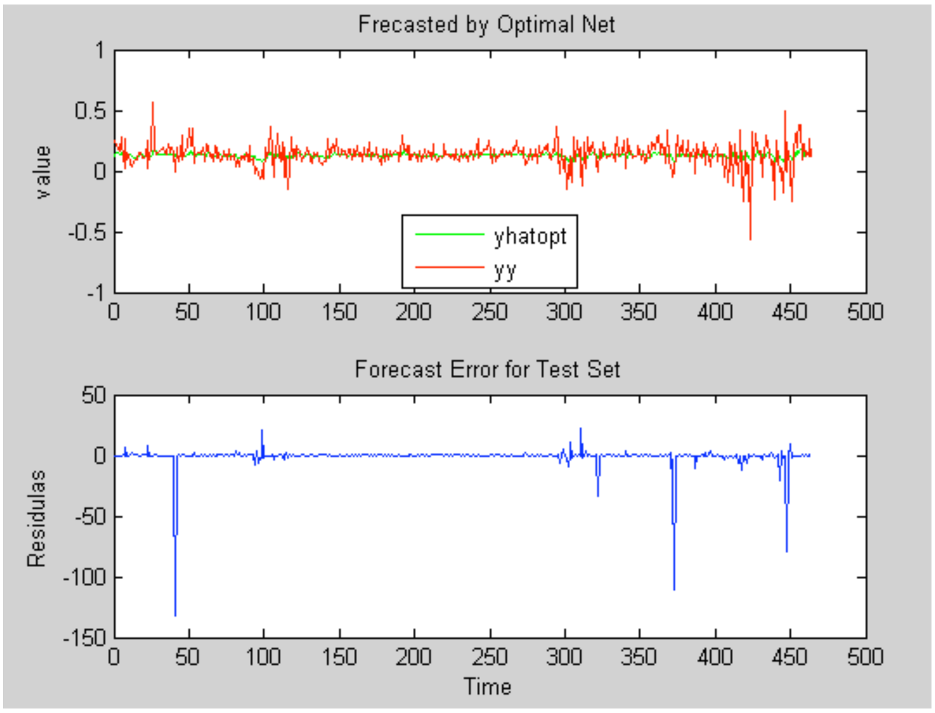


Figure 2.22: Forecasting outputs of DLOGPT trough ANN – in *Matlab*

Chapter 3

3 – DISCUSSIONS AND CONCLUSIONS

In this work it has been proposed to compare the prediction accuracy between the classical ARIMA model and the Artificial Neural Networks.

The predictability of financial time series, such as stock prices indexes is a complex task, due to several reasons.

Considering all mentioned reasons, during this work perhaps the most controversial is the fact that financial time series are usually very noisy, that is, there is a large amount of random (unpredictable) day-to-day variations. The events, such as interest rate changes, announcements of macroeconomic news as well as political events are random and unpredictable and, contribute to the noise in the time series.

3.1 – THE ARIMA RESULTS

To identify the appropriate ARIMA model for a time series, it is necessary identifying the order(s) of differencing needing to stationarize the series. As the four time series were integrated of order *1*, our input time series, were the first differenced series of the four logaritmized stock price indexes.

In order to select an ARIMA for each one of the four differenced logaritmized stock price indexes, some tests were taken in account to testing the signification of coefficients and to test the residuals. Then two evaluation criteria were considering, the Akaike's information criterion (AIC) and the Bayes information criterion (BIC) to evaluated the best ARIMA.

Both R-squared and Ajusted R-squared obtained, in each identified ARIMA model for each one of the four differenced and logaritmized stock price indexes, were too low. Some tests applied to the estimated residuals show that they are not white noise, but close to white noise, the models obtained are not satisfactories, and despite of several tries to respecify the models, the results were more or less the same, so we couldn't use any ARIMA model to forecast with the proposed accuracy.

When a time series is differencing there are a loss of valuable information. A valuable long-term relationship, between the endogenous variable and explanatory variables, is lost. Such long-run relation can only be stated in the level form and not in first-difference form. This increases the hard task of finding a satisfactory ARIMA model, since it seems that a high rate of nonlinearity should be considered in a more realistic model.

Due to the noisy nature of financial time series, nonstationarity and nonlinearity it can be even harder to forecast a stock prices index using linear models, such as ARIMA. ARIMA models are linear models, but in the real-word time series are rarely pure linear combination.

3.2 – THE ANN RESULTS

The ANN model used in this study was a feedforward network with a backpropagation algorithm. That is one of the most commonly used by its valuable properties, which are seen as big advantages.

The backpropagation training is mathematically designed to minimize the RMSE across all training patterns. RMSE had been used as a performance measure. This measure would allow to compare the performance between ARIMA and ANN models, if some ARIMA model were considering satisfactory.

In the case of the neural network, the proceeding for finding the best network design is somewhat arbitrary. There is no known systematic procedure for setting the parameters values used to design the network.

Thus it was experimented to forecast each one of the four logaritmized stock price indexes with forty-two different networks designs, considering different units/neurons in the hidden layer, three different learning rates and considering a maximum of two lags.

As ANN models can learn from experience, generalize and "see through" noise and nonstationarity. They are robust enough to recognize patterns, which have been obscured by noise.

The "first step" was to evaluate the forty-two networks with the four logaritmized stock price indexes. Latter, as in ARIMA the input time series must be stationary, it was decided to evaluate, also, the same forty-two networks with the four differenced logaritmized stock price indexes.

Considering the "first step", we can conclude that the values of the RMSE obtained in each one of the time series was very similar. In the following table, we can see the minimum and the maximum value of the RMSE, considering all forty-two networks, for each one of these time series.

	Considering all forty-two networks	
Predict Time Series	The minimum RMSE	The maximum RMSE
Logaritmized stock price index of Germany, Y=LOGBD	0.0037031753	0.0037581419
Logaritmized stock price index of Grecee, Y=LOGGR	0.0022830081	0.0022966619
Logaritmized stock price index of Italy, Y=LOGITL	0.0030149285	0.0030374649
Logaritmized stock price index of Portugal, Y=LOGPT	0.0026329884	0.0027289276

Table 3.1: The minimum and the maximum value of the RMSE, considering all forty-two networks

Independent of the learning rates, the number of lags considered or, the number of neurons take it in account in the hidden layer, the value of minRMSE does not change more than 3,7%. Despite it is being possible notice a lower value of RMSE when considered 10 or 15 neurons than when considered less neurons in the hidden layer, the difference of values is smaller.

In the previous Chapter, were showed some figures with the forecast accuracy considering the best network design, for each one of the four previous time series. As it was reported, there is a good forecast accuracy in all of them. There is a very satisfactory match between the original series (*yy*) and the forecasted one (*yhatopt*). Therefore, the residuals are very small in all four forecasted time series, the majority of the residuals are close to zero, falling into a very small interval $]-0.05, 0.05[$.

Similar to other forecasting methods, ANN have problems and limitations. The accuracy of the prediction greatly depends on historical data. Thus, as it was already mentioned, when a time series is differencing there are loss of valuable information, then even using ANN models the prediction accuracy must be worst.

This can be notice when were considered as input each one of the four differenced logaritmized stock price indexes. Where the value of the RMSE increase near 900%, when compared to the value obtain in the same networks when the input time series are not differenced time series.

It is possible to observe this increase, when we compare the values of the Table 3.1 with the values of the Table 3.2.

	Considering all forty-two networks	
Predict Time Series	The minimum RMSE	The maximum RMSE
Differenced logaritmized stock price index of Germany, Y=DLOGBD	0.0366421473	0.0371934286
Differenced logaritmized stock price index of Grecee, Y=DLOGGR	0.0266451021	0.0269436546
Differenced logaritmized stock price index of Italy, Y=DLOGITL	0.0352166198	0.0357775360
Differenced logaritmized stock price index of Portugal, Y=DLOGPT	0.0300047016	0.0305449206

Table 3.2: the minimum and the maximum value of the RMSE, considering all forty-two networks

Also here, the value of minRMSE does not change a lot, independent of the learning rates or the number of lags considered, or the number of neurons takes in account in the hidden layer. Therefore, with these input time series, it was not possible to find a good network design in order to have an acceptable prediction.

To have this perception we only need to regard the figure, with the forecast accuracy considering the best network design, for each one of the four previous time series reported in the previous Chapter. There are big differences between the original series (*yy*) and the forecasted one (*yhatopt*). Therefore the residuals are much higher in all four forecasted time series, the interval where the residuals falling into is also large.r

The results for the ANN model used in this study have demonstrated that ANN methods can be used effectively in prediction of a stock price index. They seem to perform very well with financial data and can make great contributions to provide an effective instrument for investors to hedge against potential market risks. Creating new profit making opportunities for market speculators and arbitrageurs. This could also be a powerful instrument to many enterprises such as banks and insurances that deal day by day with stock market indexes.

3.3 – ARIMA *VERSUS* ANN AND FUTURE WORKS

In this study it was expected to have an accurate forecasting for a future period using the both models in consideration. But, due to the previous referred reasons, it was impossible to obtain validate forecast using ARIMA models. In spite of the obtained results this study aware us to other important aspects.

While ANN requires a good amount of observations for training, to guarantee an appropriated learning in order to recognize a wide range of patterns, ARIMA requires stationarity. This could be controversial, financial data are greatly influenced by economical, political, international, and even natural events, thus it is a hard task to find a large period with stationarity in such data. The input data of this study encloses 4175 observations (15 years), and it was prove that the four input time series were not stationaries.

Otherwise the ARIMA's methodology foresees the nonstationarity, being its first step, apply the differencing operator to the time series until it becomes stationary. However, by differencing there is a loss of valuable long-term relationship between the endogenous variable and explanatory variables. And with 4175 observations in the input time series this loss becomes even bigger; as it is possible to verify because, also, with ANN it was not achieved an acceptable prediction considering as input the differenced time series.

An alternative could be to reduce the number of observations in the input time series, in order to find a period with sufficient observations for ANN training, and to reduce those loss of valuable information when applying the differencing operator in ARIMA. This could be taking in account in future works.

Another suggestion to future works, with intent to solve those loss of information, would be find out whether endogenous variable and explanatory variables are cointegrated. Cointegration currently enjoys high acclaim in econometric modeling literature.

In the course of the realization of this thesis, in spite of some founded difficulties, some knowledge was consolidated and other new knowledge was acquired. During the literature review, it was possible to be aware to the sort of models that can be used with this kind of data, as well as, the advantages and the disadvantages of the use of those models. It was notice that, in the recent papers, it is given most importance to nonlinear models and to new models that combine the classical techniques with the modern ones, in the financial field. As the financial field is an important field in actuarial working area, it is significant to be conscious of the development of new techniques and feel, in practise, the difficulties and the advantages between the classical techniques and the recent ones.

References

Azoff, E. M. (1994), *Neural Network Time Series Forecasting of Financial Markets*, West Sussex, John Wiley & Sons.

Brockwell, P. J. and R. A. Davis (1998), Introduction to Time Series and Forecasting, New York, Springer-Verlag.

Cowan, J. (1967), *A Mathematical Theory of Central Nervous Activity*, unpublished Ph.D. dissertation, University of London.

Diebold, F. (1997), *Elements of Forecasting*, South-Westren College Publishing.

Engle, R. F., and C. W. J. Granger (1987), *Co-integration and error correction: Representation, estimation, and testing*. Econometrica, 55, 1057 – 1072.

Franses, P. (1998), *Time series for business and economic forecasting*, Cambridge University Press.

Gately, E. (1996), Neural Networks for Financial Forecasting, New-York, John Wiley & Sons.

Gooijer J. and R. Hyndman (2006), *25 years of time series forecasting*, International Journal of Forecasting 22, 443– 473.

Hebb, D.O. (1949), *The organization of Behavior*, New York: John Wiley & Sons.

Hill, G., and R. Fildes (1984), *The accuracy of extrapolation methods: An automatic Box–Jenkins package SIFT*. Journal of Forecasting 3, 319 – 323.

Ho S., Xie M. and T. Goh (2002), *A comparative study of neural and Box-Jenkins ARIMA modelling in time series prediction*, Computers & Industrial Engineering 42, 371-375

Hotta, L. K., & J. Cardoso Neto (1993), *The effect of aggregation on prediction in ARIMA models*, Journal of Time Series Analysis 14, 261 – 269.

Kamruzzaman, J. and R. Sarker R (2003), *Forecasting of currency exchange rates using ANN: A case study*, Proceedings of 2003 International Conference On Neural Networks & Signal Processing, Proceedings, Vol. 1 And 2: 793-797

Kang, I. B. (2003), *Multi-period forecasting using different models for different horizons: An application to U.S. economic time series data*, International Journal of Forecasting 19, 387-400

Kermanshahi, B. and H. Iwamiya (2002), *Up to year 2020 load forecasting using neural nets*, Electrical Power & Energy Systems 24, 789-797.

Kim, J. H. (2003), *Forecasting autoregressive time series with biascorrected parameter estimators,* International Journal of Forecasting 19, 493 – 502.

Layton, A., L. V. Defris and B. Zehnwirth (1986), *An international comparison of economic leading indicators of telecommunication traffic*, International Journal of Forecasting 2, 413-425

Libert, G. (1984), *The M-competition with a fully automatic Box–Jenkins procedure*, Journal of Forecasting 3, 325 – 328.

Leung M., H. Daouk and A. Chen (2000), *Forecasting stock indices: a comparison of classification and level estimation models*, International Journal of Forecasting 16, 173-190.

Liu, T. R., M. E. Gerlow and S. H. Irwin (1994), *The performance of alternative VAR models in forecasting exchange rates*, International Journal of Forecasting 10, 419 – 433.

Litterman, R. B. (1986), *Forecasting wit h Bayesian vector autoregressions—Five years of experience*, Journal of Business and Economic Statistics 4, 25 – 38.

Maberly, E. (1986), *The informal content of the interday price change with respect to ctock index futures,* Journal of Futures Markets 6, 385-395.

McCulloch, W.S. and W. Pitts (1943), *A Logical Calculus of the Ideas Immanent in Nervous Activity*, Bulletin of Mathematical Biophysics 5, 115-133.

O'Connor, M., W. Remus, and K. Griggs (1997), *Going up-going down: How good are people at forecasting trends and changes in trends?*, Journal of Forecasting 16, 165-176.

Phua P., D. Ming and W. Lin (2001), *Neural Network with generically evolved algorithms for stocks prediction*, Asia-Pacific Journal of Operational Research 18, 103-107

Poulos, L., A. Kvanli, and R. Pavur (1987), *A comparison of the accuracy of the Box–Jenkins method with that of automated forecasting methods*, International Journal of Forecasting 3, 261 – 267.

Quenouille, M. H. (1957, Second Edition), *The analysis of multiple time-series,* London, Griffin.

Rosenblatt, F. (1962), *Principles of Neurodynamics*, New York, Spartan.

Simkins, S. (1995), *Forecasting with vector autoregressive (VAR) models subject to business cycle restrictions*, International Journal of Forecasting 11, 569 – 583.

Slutsky E. (1937), *The summation of random causes as the source of cyclic process*, Econometrica, 5.

Walker G. (1931), *On periodicity in series of related terms*, Proceedings of the Royal Society, A131.

Weigend A. and N. A. Gershenfeld (1994), *Time Series Prediction: Forecasting the Future and Understanding the Past* (Santa Fe)

Werbos, P. (1974), *Beyond Regression: New Tools For Predicting and Analysis in the Behavioral Sciences*, Ph.D. thesis, Cambridge, MA: Harvard University Committee on Applied Mathematics.

Wu, Y. and H. Zhang (1997) *Forward premiums as unbiased predictors of future currency depreciation: A non-parametric analysis*, Journal of international Money and Finance 16, 609-623.

Yaglom, A. (1955), *The correlation theory of processes whose nth difference constitute a stationary process*, American Mathematical Society.

Yule, G. U. (1927), *On the method of investigating periodicities in disturbed series, with special reference to Wolferts sunspot numbers*, Philosophical Transactions of the Royal Society London, Series A, 226, 267– 298.