Building a software application for the predictive modelling of wind data

National University of Ireland, Cork

Department of Applied Mathematics

Thesis submitted for the degree of Master of Science (Mathematical Modelling and Scientific Computing)

Supervisor: Dr Tom Carroll

Department Head: Prof. Sebastian Wieczorek

Submission Date: September 2015

Joseph Collins

Table of Contents

I, Joseph Collins, certify that this thesis is my own work and I have not obtained a degree in this university or elsewhere on the basis of the work submitted in this thesis.

Joseph Collins

Acknowledgements

A sincere and heartfelt thank you to Tom Carroll not only for the time taken to review draft versions of the thesis and to suggest areas of improvement in the software application but also for the support and encouragement when this mature student decided to return to college.

A huge thank you also to Kieran Mulchrone, coordinator of the M.Sc. programme. His lecturing and general enthusiasm throughout the course has been an inspiration.

Finally, to my three rascals Saoirse (8), Katelyn (5) and Anna Mae (2). You'll start seeing more of your daddy at weekends now.

Abstract

The focus of this thesis will be on a particular facet of the predictive modelling arena, Time Series Models. A motivation for the use of such predictive models will be given (i.e. the Electricity market in Ireland); the main objective of the thesis, however, is to build a robust software application that can be used to assist with the exploratory and time series analytics for differing datasets arising in this context.

Chapter 1 provides an introduction to the original problem setting whilst Chapter 2 gives a brief overview of the software; the remaining chapters are structured so that firstly the user will have an understanding of the statistical methods implemented in that software segment/component and secondly they will have a user manual which illustrates the correct use/operation of the software should the need arise. A link to the software application is provided in Chapter 7.

1. Introduction

The Single Electricity Market, *SEM*, was established on the island of Ireland on the 1st of November 2007. Prior to this, electricity markets in Northern Ireland and the Republic of Ireland were distinct with the latter following a self-dispatch bi-lateral regime. A self-dispatch bi-lateral market is one in which market participants are responsible for sourcing the energy that they will supply to end users; they can do this by either dispatching their own generation or by contracting for electricity bi-laterally.

The SEM is a mandatory gross pool. This means that all energy which is produced by generators is sold into the pool, all energy supplied to end users is taken from the pool. Figure 1-1 illustrates the market structure.

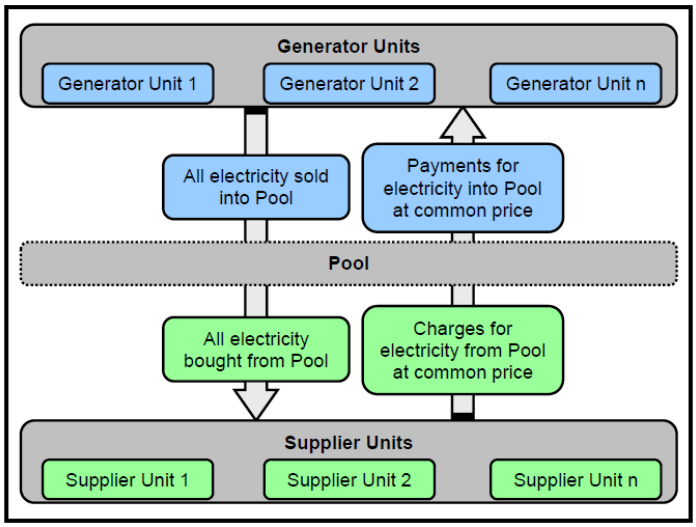

Figure 1-1: Structure of the Single Electricity Market, source [1]

.

The price that generators receive for supplying energy to the pool (and the price that suppliers pay for taking energy out of the pool), units of which are in €/MWh, is called the System Marginal Price, *SMP*. There is a unique SMP for each 30 minute interval (i.e. Trading Period) during the day. These prices are determined by running the Market Scheduling and Pricing, MSP, software. The Single Electricity Market Operator, *SEMO,* is responsible for running the MSP software and overseeing the financial settlement of the electricity market.

The question arises as to how the MSP software determines the half hourly electricity price. A simplification of the underlying algorithm is that it endeavours to schedule generation to meet demand at the lowest possible cost over a single Trading day (a Trading day runs from 6am to 6am). Generators submit bids (i.e. indicate an ability to produce energy at a particular price) to SEMO based on their short run marginal costs. A set of rules called the Bidding Code of

Practice provides guidance as to what generators can/can't do when submitting bids. The algorithm then essentially stacks the generators in order of competitiveness, cheapest bids taking precedence, and schedules the generators until system demand is met. The system price is then set equal to the price of the marginal generating unit (i.e. the most expensive scheduled generator)[1]. Figure 1-2 illustrates the overall approach.

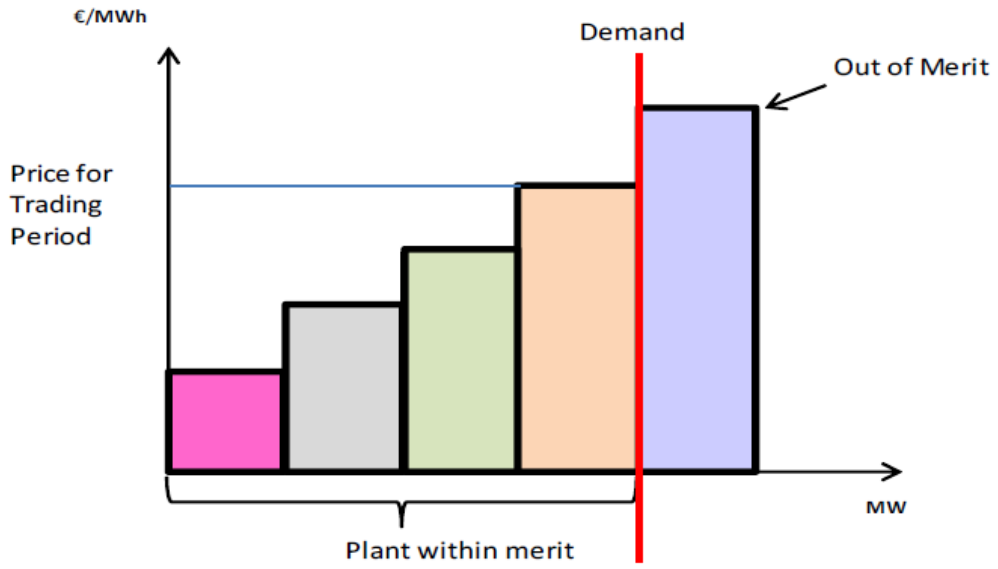

Figure 1-2: illustration of the concept of marginal pricing with the x-axis representing the amount of energy that generators can produce and the y-axis representing the price at which they can produce this energy, source [2].

1.1. Renewable Energy in the SEM

Under the EU directive on the Promotion of the Use of Renewable Energy, Ireland has committed to ensuring that 16% of the total energy consumed in heating, electricity and transport comes from renewable energy sources. To achieve this the Irish Government has set a target that 40% of electricity consumption is to come from renewable sources by 2020.

In recent years the Government has implemented renewable generation support schemes such as the Renewable Energy Feed in Tariff, REFIT, to encourage the build out of renewable energy generators to help ensure the aforementioned targets are met. As a result installed wind capacity in Ireland went from a level of approximately 500MW in 2007 to over 3000MW in 2015. By 2020 it is projected that installed Wind Capacity will reach over 4,600MW on the island of Ireland.

[1] SMP = Shadow Price + Uplift, the Shadow Price is determined on a marginal pricing approach albeit there are notable exceptions and a full reading of the Trading and Settlement Code needs to be undertaken to appreciate the full workings of the pricing software.

Looking forward one can reasonably expect an even greater focus on the predictability of renewable sources of energy in the coming years given the increasing penetration of wind generators in the electricity market. That is, given that the System Operator, *SO*, is responsible for ensuring a stable and secure electricity system, how confident can they be of forecast wind generation levels in the next hour, the next ten hours etc.?

SEM participants, while somewhat shielded from the impacts of increasing renewables on the system under the current pool arrangements will become increasingly concerned with these issues from 2017 onwards with the roll out of the new electricity market, the Integrated Single Electricity Market (I-SEM). The new market model is a result of European directives aimed at ensuring the effective flow and transmission of energy across borders within the European Union. In the I-SEM, generators will assume responsibility for balancing their own portfolios (i.e. self-dispatch). Hence, market participants will increasingly pay attention to the accuracy of renewable energy production forecasts, expected impact of different wind levels on electricity prices etc.

Figure 1-3 illustrates some of the potential concerns. The plot shows the shadow price (a component of the SMP) on two different days in 2014, a plot of system wind levels for both of these days is also given.

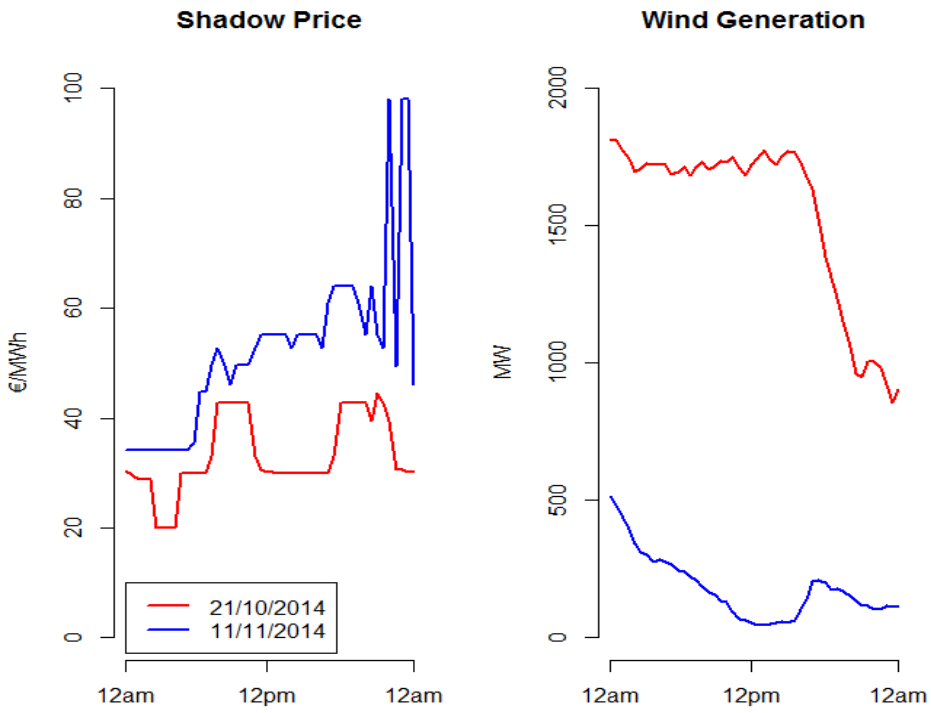

Figure 1-3: plot of shadow prices and wind energy generation levels for two days in 2014.

The SO is required, where possible, to let as much renewable energy generation onto the system as is available (this is called Priority Dispatch). Hence on days with high wind generation levels, the requirement for thermal generation to meet demand is reduced. This will result, on average, with more efficient (cheaper) thermal generators setting the marginal price thereby pushing overall electricity prices down. While Figure 1-3 provides no conclusive evidence that increasing wind levels result in lower prices (other factors such as system demand should also be taken into consideration), patterns similar to those shown in Figure 1-3 have increasingly been observed in recent years. Hence the growing focus on wind energy levels and the accuracy of wind energy forecasts in electricity markets.

1.2. Aims and Objectives

The aims of this thesis are twofold, namely

- A cursory review of time series models used in wind energy forecasting.
- To develop a software application to implement the most salient time series models.

The data utilised throughout the thesis consists of half hourly total wind energy levels (in MW) in the SEM market schedule for years 2010, 2011, 2012, 2013 and 2014 (the latter consisting of observations from 1st January 2014 to 15th November 2014).

While there are a number of open source packages available which provide time series model functionality (R being the most obvious example), all the software is written in C# and is implemented on a standalone basis with no reliance on any external software packages/libraries. To guard against any errors related to algorithm implementation etc., where possible all model/parameter estimates produced via the software application are validated against R to ensure consistency.

1.3. Thesis Outline

The structure of Chapters 3, 4, 5 and 6 will be as follows

- High level overview of the main concepts/approaches encountered in the literature.
- Detailed discussion of the algorithm implementation.
- Illustration of the correct use of the software application using one of the underlying datasets.

Chapter 3 will focus exclusively on examining and transforming the data (the first step in applying time series models). Chapter 4 will examine *Autoregressive Integrated Moving Average*, ARIMA, models. Chapter 5 will introduce *Autoregressive* and *Generalized Autoregressive Conditional*

Heteroscedasticity (ARCH and GARCH) models. In Chapter 6 the transformations and fitted models derived in Chapters 3, 4 and 5 will be used in the production of wind energy forecasts.

2. Software Overview

The software application, *WindApp2015*, is built in the .NET framework and is a Multiple Document Interface (MDI) application. On opening the application, the user is greeted with a blank window similar to that shown in Figure 2-1.

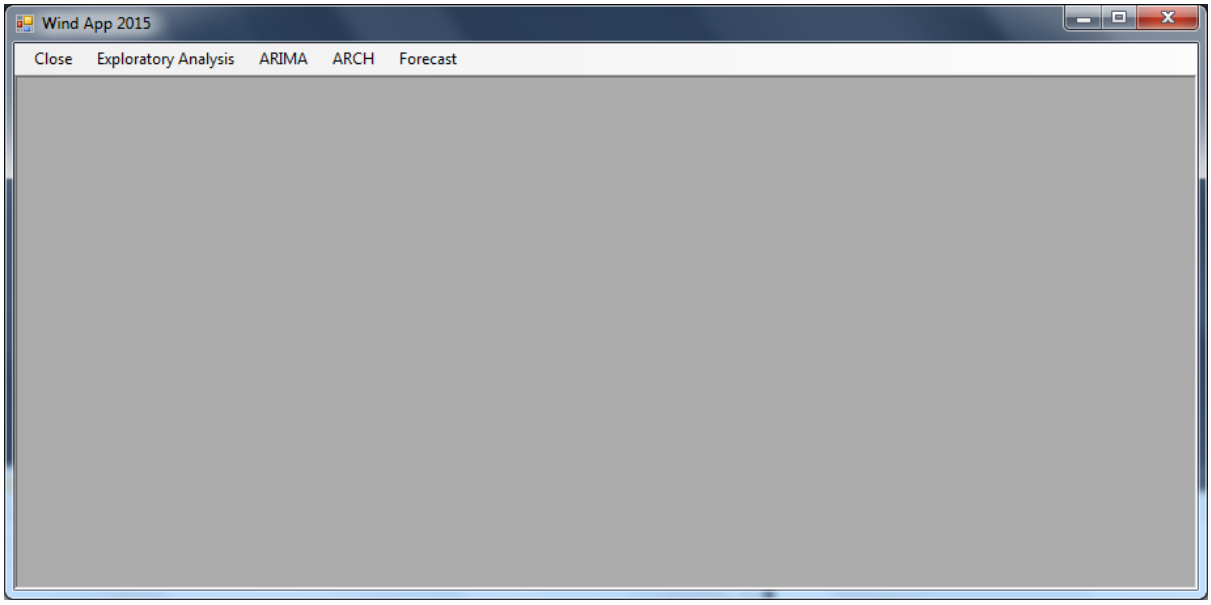

Figure 2-1: opening window of the WindApp2015 application.

The user will have the following options

- *Close* – the *WindApp2015* application will close.
- *Exploratory Analysis* – a new window allowing the user to perform exploratory data analytics, and data transformations, will appear.
- *ARIMA* – a new window which will assist with ARIMA model diagnostic checking and model fitting will open.
- *ARCH* – a new window which will assist in the fitting of ARCH and GARCH type models will appear.
- *Forecast* – a new window which will assist with the production of forecasts will appear.

The following sections present a high level overview of the main classes used, how each of the aforementioned windows (MDI child forms) are produced etc. More detailed explanations including examples of how to use the software are given in Chapters 3, 4, 5 and 6.

2.1. Statistics Class

The *Statistics* class is responsible for providing the vast majority of quantitative analytics in the software application. The *Statistics* class implements methods for calculating summary statistics such as the minimum, maximum, mean, standard deviation, skewness, kurtosis etc. of a dataset.

It has the ability to help in the production of histograms (automatically identifying the bins and their location), it can assist with producing random samples from Normal and Weibull distributions and it can also assist with finding the best fit Weibull parameters to a particular data set.

Time series methods implemented in the *Statistics* class include the autocorrelation and partial autocorrelation functions, the ability to difference a time series (i.e. to remove any trends in the underlying data, see [3]), algorithms for fitting ARIMA models and for estimating the associated model residuals.

Other methods of note in the *Statistics* class include a suite of time series models, namely ARCH and GARCH, in which the conditional variance (i.e. conditional on the information up to a particular point in time) is allowed to vary.

When using a *Statistics* object it will frequently be the case that the underlying data is saved in an attribute of the Statistics class titled *D* (shorthand for Data). It is simply a list of type *double* and is defined as *List<double> D*. Other properties of the Statistics class which will be used for storing data include

- *TransformedData* – list of type double, saves the transformed data (see Chapter 3).
- *StandardisedTransformedData* – list of type double, saves the transformed and standardised data (see Chapter 3).
- *DifferencedData* – list of type double, saves the differenced data (see Chapter 4).

2.2. Other Classes

Some of the other classes which comprise the *WindApp2015* application (these classes will primarily be used by methods of the *Statistics* class) include

- *GElim* – class which performs Gaussian elimination i.e. solving systems of simultaneous equations.
- *DR* – short hand for differentiation and root finding class, the class implements Newton Raphson and other algorithms.
- *system_solver* – class which can be used to get the inverse of 2 dimensional arrays, it will be used by a number of methods in the *DR* class.
- *Matrix* – class which performs matrix multiplication, addition, subtraction etc. It will be used by a number of methods in the *DR* class.
- *Vector* – class which performs vector multiplication, addition, subtraction etc. It will also be used by a number of methods in the *DR* class.

- *FunctionMatrix* – this is a class, or construct, which will be used when examining multidimensional systems. It can be used to create matrices of functions (of one or more variables). It is used in the *Statistics* class to fit Autoregressive models (see the *ARML* method), it is also used in a number of methods of the *DR* class.
- *Likelihood* – class that is built specifically to assist with implementing Autoregressive maximum likelihood methods in the *Statistics* class.
- *Model* – after fitting one of the ARIMA/ARCH/GARCH models via the *Statistics* class, the model output (parameter estimates, residuals, autocorrelation residuals etc.) will be saved in an instance of the *Model* class.

2.3. Linking to External Data

The *WindApp2015* application assumes that the dataset of interest is saved in a Microsoft Excel format. If this is not the case (i.e. the dataset is saved in a Microsoft SQL Server database for example) then some modifications to the methods outlined in subsequent paragraphs will be required. It is also assumed that the Excel data is saved in a time ordered sequence i.e. cell A1 is the first observation, cell B1 is the second observation etc.

Each of the MDI child forms referenced in Section 2.4 will have a *Load Data* button which automatically appears once the form opens, see Figure 2-2. Once the user clicks the button they will be able to locate the Excel file where the dataset is saved and on clicking *Open* the *ListBox* which was previously blank, as seen in Figure 2-2, will then be populated with a list outlining the name of the Excel workbook and the worksheets within that workbook, see Figure 2-3. If the user wants to perform an analysis on the data associated with any particular workbook/worksheet, they just need to click on the relevant line in the *ListBox* and the dataset will be loaded into the *WindApp2015* application.

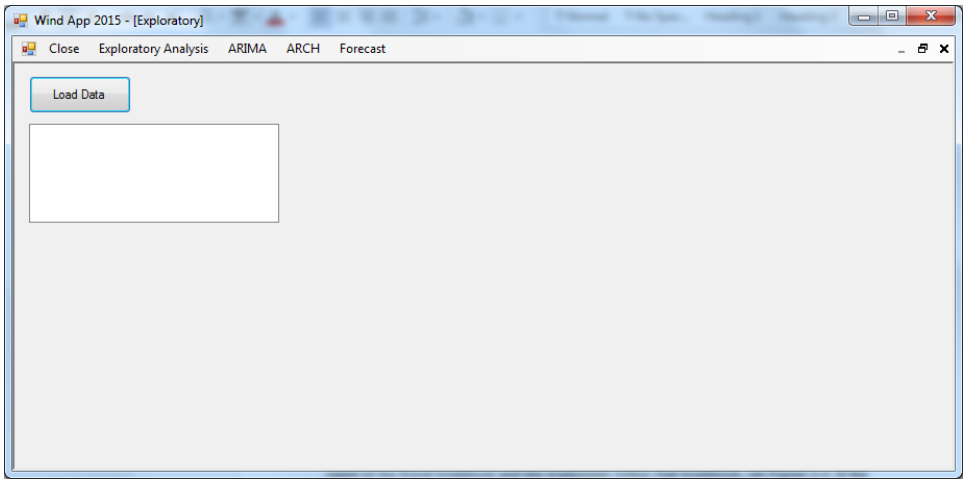

Figure 2-2: initial window that appears for each of the MDI child forms.

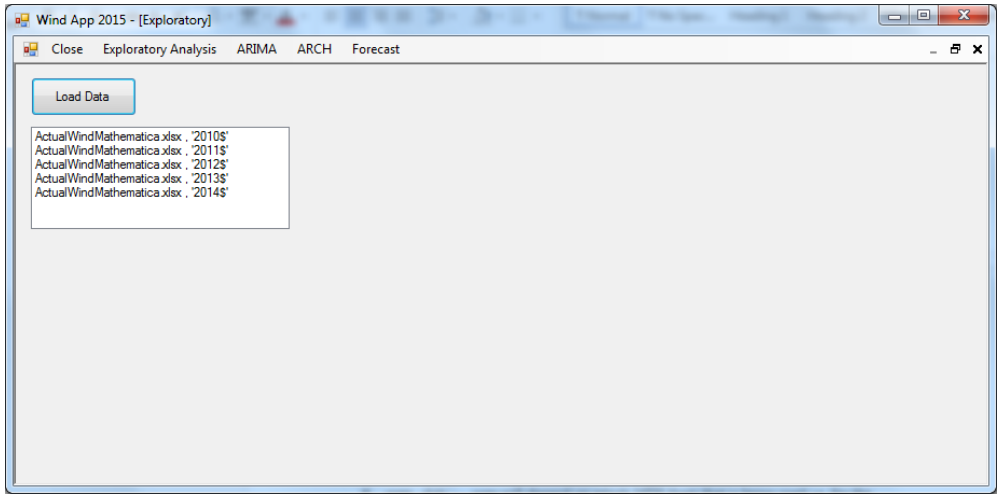

Figure 2-3: ListBox as it appears once an external dataset has been loaded to the WindApp2015 application.

Within each MDI child form (see Section 2.4), the methods which enable the aforementioned functionality include

- *name_click( ) – name* will depend on which MDI form that is being used i.e. for the *Exploratory Analysis* window the method is called *explrbt_click*( ). The method will utilise the .NET *OpenFileDialog*( ) command to locate the Excel file and to make a note of values such as the location of the file, the filename etc. Next a *populatelistbox( )* method will be called, using information from the previous step the method will ensure that the *ListBox* is populated with the workbook name and all the worksheets that comprise the particular workbook.
- *name_SelectedValueChanged ( )* – any time the user clicks on a particular line in the *ListBox* this method is called and it loads the data of interest from the relevant workbook and worksheet into the *WindApp2015* application.

The *name_click* and *name_SelectedValueChanged* functions utilise *OledbConnection*, *OleDbDataAdapater*, *DataSet* and *DataTable* objects.

2.4. MDI Child Forms

The MDI child forms which are created via the MDI parent form (*mainForm)* are as follows

- *explrForm* – used to generate the *Exploratory Analysis* window.
- *ARIMA* – used to generate the *ARIMA* window.
- *ARCH* – used to generate the *ARCH* window.
- *Forecast* – used to generate the *Forecast* window.

Additional details on the operation of each of these forms/windows will be provided in subsequent chapters.

3. Transforming the Data

Chapter 3 will primarily be concerned with exploratory analysis of the underlying data sets. The software package will assist the user in

- Presenting time series and histogram plots of the underlying data.
- Identifying whether or not the data follows a Weibull type distribution, if it does what are the best fit model parameters?
- Choosing a method to transform the data so that it better approximates a Gaussian distribution (for which a suite of time series models have been developed).
- Identifying any other adjustments that need to be made to the data i.e. do we need to account for seasonality etc.?

Section 3.1 will give a brief outline of the theoretical underpinnings outlined in Brown, Katz and Murphy [4]. Section 3.2 will provide an in-depth review of how the best fit parameters for the Weibull distribution are obtained. Section 3.3 will briefly discuss how to generate samples from Normal and Weibull distributions. Section 3.4 will provide a detailed explanation of how to use the software by examining a particular dataset.

3.1. Weibull Distribution and Wind Data

The probability density function of a Weibull random variable is given by

$$f(x;\lambda,\kappa) = \begin{cases} \frac{\kappa}{\lambda}\left(\frac{x}{\lambda}\right)^{\kappa-1} e^{-\left(\frac{x}{\lambda}\right)^{\kappa}}, x \geq 0 \\ 0, x < 0 \end{cases}. \quad 3.1$$

Both κ and λ are greater than 0; κ is called the shape parameter and λ is called the scale parameter. An attractive feature of a Weibull distribution is that, depending on the choice of parameters κ and λ, the distribution can assume a wide variety of forms (i.e. shapes). According to Dubey [5], when the shape parameter κ is close to 3.6, the Weibull distribution is similar in form to a Gaussian distribution. Brown, Katz and Murphy [4] use this as one possible approach to transforming the underlying wind speed data so that it approximates a Gaussian distribution. That is, if a random variable follows a Weibull distribution with parameters κ and λ, then that random variable raised to the power of m is still a Weibull random variable with shape parameter given by $\frac{\kappa}{m}$ and scale parameter given by λ^m. Hence, in order to transform the underlying data so that it approximates a Gaussian distribution choose a value of m which satisfies

$$m = \frac{\kappa}{3.6}. \quad 3.2$$

Thus, for wind datasets, assuming the underlying data follows a Weibull type distribution, the exercise becomes one of determining, via maximum likelihood estimation, the shape parameter κ (see Section 3.2 for full details). Once this is available, then the data can be transformed according to Equation 3.2 and the resulting data should approximate a Gaussian distribution.

Brown, Katz and Murphy also describe an alternative procedure for choosing the transformation. The method follows that suggested by Hinkley [6] and is iterative in nature. A summary of the procedure is as follows:

1. Given a positive real number m, for each data point, d_i, in the time series calculate d_i^m.
2. For the transformed time series, calculate the mean, median and standard deviation (sd) of the time series. Using these values then calculate $\frac{mean-median}{sd}$.
3. Repeat steps 1 and 2 for different values of m.
4. The transformation is then the value of m for which the value of $\frac{mean-median}{sd}$ is closest to zero.

Once the data has been transformed, then further standardisation may be required [7]. For example, for wind datasets the hypothesis may be that wind levels are higher in Winter months than Summer months or that overnight wind levels are typically lower than those observed during the day. To account for this the underlying data should standardised. That is given the transformed time series observations, $\widetilde{d_i}$, split the data into groups according to the hypothesis; for each group calculate the mean and standard deviation, then for each transformed observation in a particular group subtract the mean value of that group and then divide by the standard deviation of that group i.e.

$$\frac{\widetilde{d_i} - mean\left(group\ \widetilde{d_i}\right)}{standard\ deviation\left(group\ \widetilde{d_i}\right)}. \qquad 3.3$$

At this stage, the transformed and standardised dataset should approximate a Normal distribution. The question arises as to how the user can check for normality. A number of different statistical tests exist, but two quick visual checks include a histogram and a quantile-quantile plot (Q-Qplot) of the transformed and standardised data. More details on the latter (i.e. the Q-Qplot) are given in Section 3.2.

3.2. Weibull Parameter Estimation

Given a random sample $x_1, x_2, \ldots, x_n$ from a Weibull distribution with shape parameter κ and scale parameter λ, maximum likelihood estimation is used to determine the *best fit* parameters to

the observed data. Full details can be found in [8] and [9], but in summary the likelihood of the entire sample, $L(\kappa, \lambda)$, is given by

$$L(\kappa, \lambda) = \prod_{i=1}^{n} f(x_i; \lambda, \kappa) = \prod_{i=1}^{n} \frac{\kappa}{\lambda}\left(\frac{x_i}{\lambda}\right)^{\kappa-1} e^{-\left(\frac{x_i}{\lambda}\right)^{\kappa}}. \tag{3.5}$$

To determine the maximum likelihood estimate of the parameter λ (denote the maximum likelihood estimate as $\hat{\lambda}$), first take the natural logarithm of Equation 3.5, then differentiating with respect to λ and equating to zero yields

$$\begin{aligned} \frac{\partial L(\kappa, \lambda)}{\partial \lambda} &= \frac{\kappa n}{\lambda^{\kappa+1}}\left(\frac{1}{n}\sum_{i=1}^{n} x_i^{\kappa} - \lambda^{\kappa}\right) = 0 \\ \Rightarrow \hat{\lambda} &= \left(\frac{1}{n}\sum_{i=1}^{n} x_i^{\kappa}\right)^{\frac{1}{\kappa}}. \end{aligned} \tag{3.6}$$

Next, taking the logarithm of Equation 3.5, taking the derivative with respect to κ and replacing λ with its maximum likelihood estimate, $\hat{\lambda}$, yields

$$\frac{\partial L(\kappa, \lambda)}{\partial \kappa} = n\left(\frac{\sum_{i=1}^{n} x_i^{\kappa} \ln x_i}{\sum_{i=1}^{n} x_i^{\kappa}} - \frac{1}{n}\sum_{i=1}^{n} \ln x_i - \frac{1}{\kappa}\right) = 0. \tag{3.7}$$

In the software application *WindApp2015* the *weibull_parameter_estimate()* method of the *Statistics* class implements a Newton Raphson root finding approach to determine the value of κ which satisfies Equation 3.7. That is, given a time series $x_1, x_2, \ldots, x_n$, letting

$$\begin{aligned} f(\kappa) &= \frac{\sum_{i=1}^{n} x_i^{\kappa} \ln x_i}{\sum_{i=1}^{n} x_i^{\kappa}} - \frac{1}{n}\sum_{i=1}^{n} \ln x_i - \frac{1}{\kappa} \\ f'(\kappa) &= \frac{1}{\kappa^2} - \frac{\left(\sum_{i=1}^{n} x_i^{\kappa} \ln x_i\right)^2}{\left(\sum_{i=1}^{n} x_i^{\kappa}\right)^2} + \frac{\sum_{i=1}^{n} x_i^{\kappa} (\ln x_i)^2}{\sum_{i=1}^{n} x_i^{\kappa}}. \end{aligned} \tag{3.8}$$

Then given an estimate, κ_i, for κ, an update, κ_{i+1}, is given by

$$\kappa_{i+1} = \kappa_i - \frac{f(\kappa_i)}{f'(\kappa_i)}. \tag{3.9}$$

After iterating on Equation 3.9 (until the difference between successive estimates lies below a specified threshold), then the maximum likelihood estimate of λ can be found by simply replacing κ with $\hat{\kappa}$ in Equation 3.6. The initial guess, κ_0, at which the iteration outlined in Equation 3.9 commences is given by [10]

$$\left(\frac{standard\ deviation\ of\ \ x_1, x_2, \ldots, x_n}{mean\ of\ \ x_1, x_2, \ldots, x_n}\right)^{-1.086}. \tag{3.10}$$

3.3. Random Samples and Q-Qplots

It will be seen in Section 3.4 that the software application generates random samples from a Normal distribution and from a Weibull distribution. How is this achieved given that there are no built-in Normal and Weibull random number generators in the .NET platform? To generate random numbers from a Normal distribution the Box-Muller transform [11] is implemented in the *normal_sample()* method of the *Statistics* class. In this method, two Uniform random variables (uniform on the interval [0, 1]), denoted by U_1 and U_2 are generated using the *Random* class (part of the .NET framework). Next variables X_1 and X_2 are calculated as follows

$$\begin{aligned} X_1 &= (-2\ln U_1)^{0.5}\ \mathrm{Cos}(2\pi U_2) \\ X_2 &= (-2\ln U_1)^{0.5}\ \mathrm{Sin}(2\pi U_2). \end{aligned} \tag{3.11}$$

X_1 and X_2 will be a pair of random numbers from the same standard Normal distribution; hence only one of these values is required to produce a random Gaussian sample. If we want to sample from a Normal distribution with mean μ and standard deviation σ, simply apply the following transformation

$$\mu + \sigma X_2. \tag{3.12}$$

In a similar manner, given a random sample from a Uniform [0, 1] distribution, denoted by U_1, a random sample from a Weibull distribution with shape parameter κ and scale parameter λ can be derived as follows [12]

$$\lambda(-\ln(U_1))^{\frac{1}{\kappa}}. \tag{3.13}$$

Equation 3.13 is implemented in the *weibull_sample()* method of the *Statistics* class.

Finally, in Section 3.4 (and elsewhere in the thesis) Q-Qplots will be encountered. Q-Q stands for quantile-quantile plot i.e. a form of probability plot. To generate a Q-Qplot for two datasets/distributions simply graph (x, y) pairs where the x-coordinate represents a particular quantile for the first distribution and the y-coordinate represents the corresponding quantile for the second distribution. If the distributions/datasets are the same, then the graph of (x, y) pairs should lie on a straight line with slope 1. If there is some linear relationship between the two datasets/distributions, again they should lie close to a straight line, but the slope will depend on the linear relationship. Q-Qplots will be used as a diagnostic tool for checking normality assumptions.

3.4. Software: Exploratory Analysis example

As mentioned in Section 1.2, the data used for illustrative purposes in this thesis are half hourly Wind Energy levels in the SEM market schedule for 2010, 2011, 2012, 2013 and 2014. On opening the *WindApp2015* application, clicking the *Load Data* button and selecting one of the lines in the *ListBox* the user is automatically presented with a number of plots and summary statistics as seen in Figure 3-1.

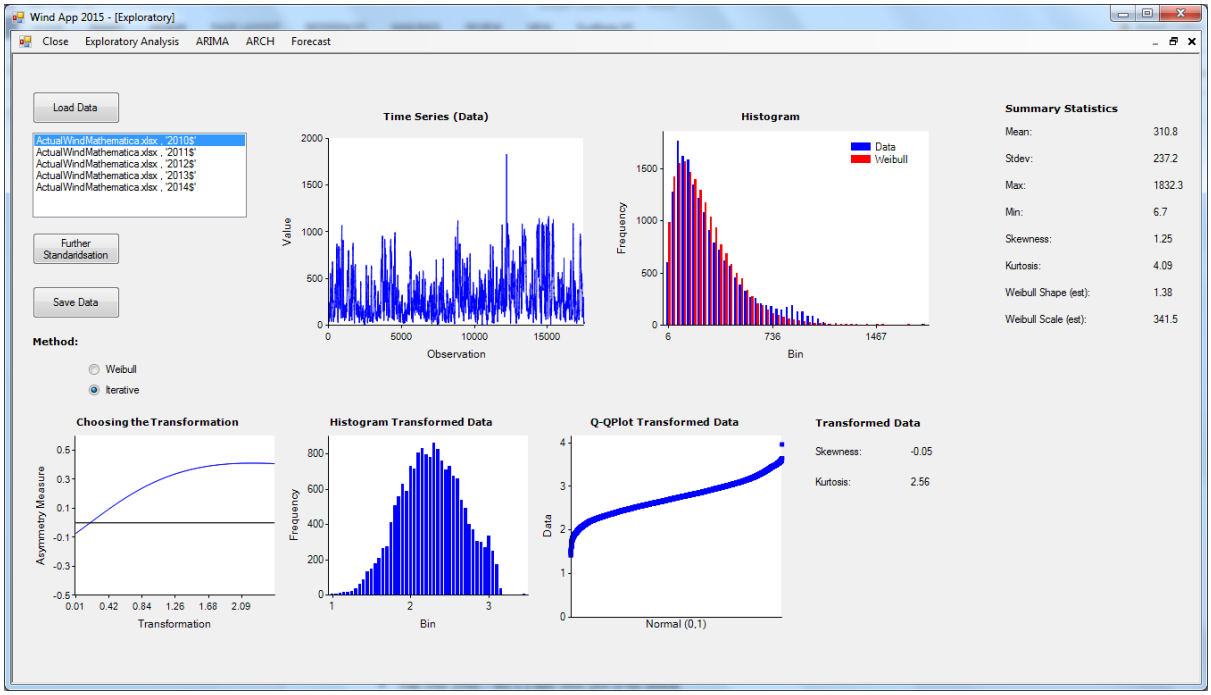

Figure 3-1.

Working from left to right, looking at the first row and then the second row, the items and information shown are as follows

- *Time Series (Data)* – this is a time series plot of the dataset.
- *Histogram* – the blue bins represent the dataset. If the dataset is assumed to follow a Weibull distribution and the best fit parameters $\hat{\kappa}$ and $\hat{\lambda}$ are estimated, then the red bins represent the histogram associated with random draws from a Weibull distribution with shape and scale parameters $\hat{\kappa}$ and $\hat{\lambda}$.
- *Summary Statistics* – summary statistics for the dataset including minimum value, maximum value, standard deviation, mean, skewness, kurtosis, Weibull best fit parameters $\hat{\kappa}$ and $\hat{\lambda}$.
- *Choosing the Transformation* – as outlined in Section 3.1 the dataset can be transformed using either an iterative approach or using a Weibull related transformation. If the *Iterative* radio button is clicked, then the dataset will be transformed using the iterative approach

and the graph titled *Choosing the Transformation* plots the asymmetry measure (see Step 4 page 14) for each different value of the power transform, m. If the Weibull radio button is clicked, the *Choosing the Transformation* plot will not appear and the data will be transformed according to the Weibull approach.

- *Histogram Transformed Data* – shows a histogram of the transformed dataset.
- *Q-Qplot Transformed Data* – a Q-Qplot of the transformed data whereby the quantiles of the transformed data are plotted against the corresponding quantiles from a Normal distribution with mean 0 and variance 1.
- *Transformed Data* – presents the skewness and kurtosis for the transformed dataset.

If the user wants to investigate the need for standardising the data then clicking on the *Further Standardisation* button will produce the window shown in Figure 3-2

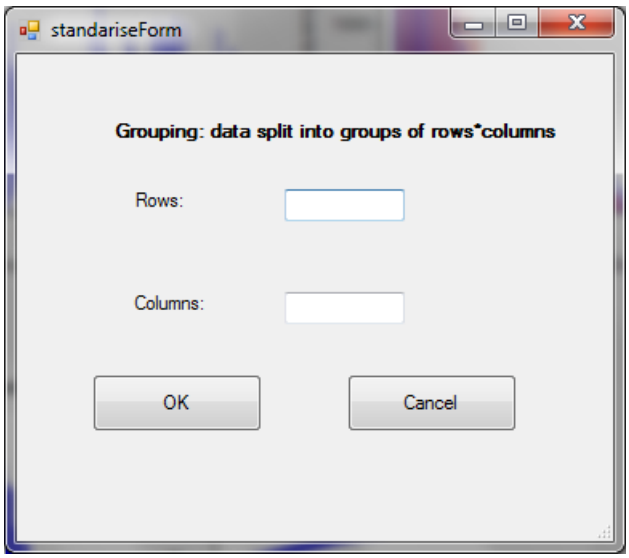

Figure 3-2.

To illustrate how this window is used, imagine that the data in the underlying worksheet is a 365 by 48 matrix of wind observations. The (i, j)th entry in this matrix corresponds to the observation on day i and trading period j.

If the hypothesis is that wind values are generally lower at night than during the day, one way of examining this would be to enter 365 in the *Rows* textbox and 1 in the *Columns* textbox. After clicking the *OK* button the transformed data will be split into sub-blocks of dimension 365 by 1 (i.e. it effectively splits the transformed data in columns), the mean and standard deviation of each group is then calculated. The mean of each group is then plotted in the *Transformed Data: Mean of groups* graph as seen in Figure 3-3. Using the transformed data, standardisation then takes place using the aforementioned criterion (i.e. for each observation subtract the mean of the

group and then divide by the standard deviation of that group). The skewness and kurtosis of the transformed and standardised data are then shown in the *Standardised Data* section.

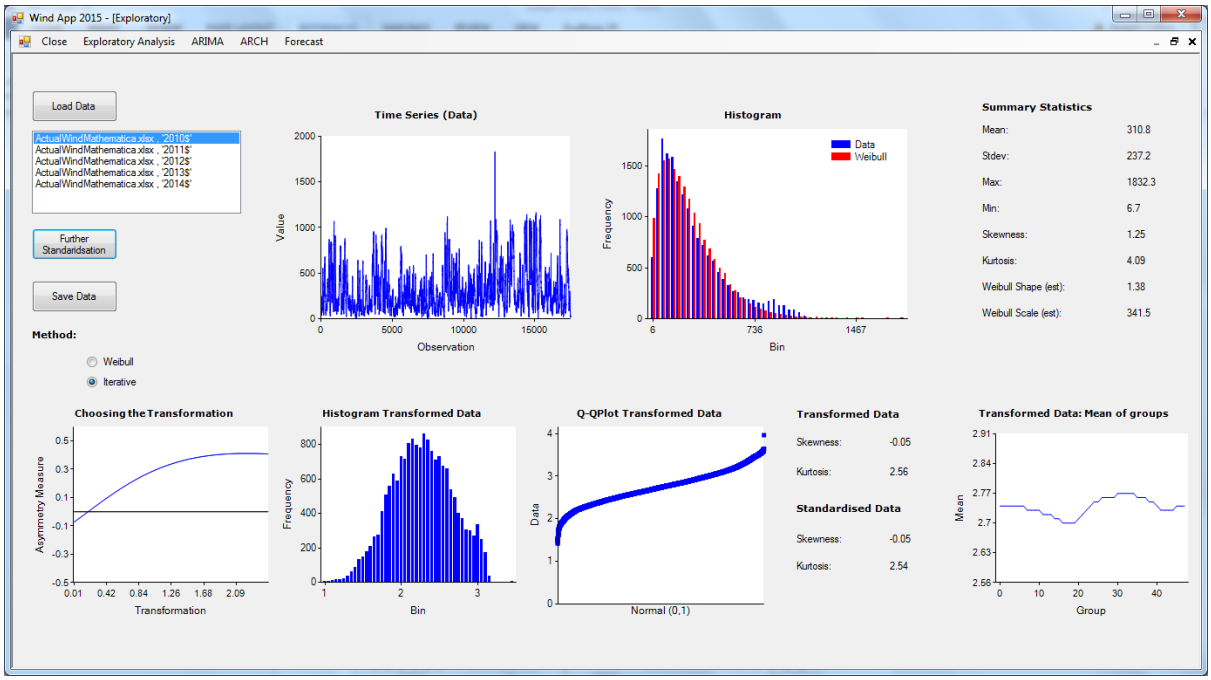

Figure 3-3: standardising the data after it has already been transformed.

At this stage all the exploratory data analysis has been performed. If the user wants to save the transformed and standardised data to the relevant Workbook they can click on the *Save Data* button. A copy of the aforementioned datasets (i.e. original observations, transformed data, transformed and standardised data) and the transformation method and transformation parameters will also be saved to separate CSV files. Some of the information saved in the CSV files (i.e. the transformation method and the transformation parameters) will be of use in subsequent software components.

The methods of the Statistics class used to implement the features outlined in the previous paragraphs include

- *avg( ), mean( ), median( ), stdev( ), skew( ), kurt( ), maximum( ), minimum( ).*
- *histogramBins( ), histogramValues( )* – used in the construction of histograms.
- *iterative_measurement( ), closestzero( ), gen_list( )* – used in the iterative transformation approach, see Section 3.1.
- *weibull_parameter_estimate( ), weibull_initial_guess( )* – see Section 3.2.
- *weibull_sample( ), normal_sample( )* – see Section 3.3.
- *standardise( )* – method used to help with the standardisation of the transformed data, see Figure 3-3.

The dataset underlying Figure 3-1 and Figure 3-3 is half hourly wind levels from 2010. Examining both of these figures, are there any noticeable features/characteristics?

From the *Time Series (Data)* plot it is hard to notice any discernible pattern apart from the wide range in values. Looking at the *Histogram* graph it appears that the histogram of observed values is a reasonable fit to a Weibull distribution. Looking at the *Summary Statistics* values, it can be seen that the best fit Weibull parameters $\hat{\kappa}$ and $\hat{\lambda}$ have values of 1.38 and 341.5 respectively. Using the iterative transformation approach, it can be seen that raising the data to the power of 0.18 has the lowest asymmetry measure (i.e. *Choosing the Transformation* plot). Looking at the *Histogram Transformed Data*, it can be seen that the transformed data is a much better approximation to a Normal distribution but the right tails appear to cut off suddenly. The tails of the Q-Qplot (*Q-QPlot Transformed Data*) deviate from a straight line raising further questions about normality of the transformed data. Finally, the plot *Transformed Data: Mean of Groups* seems to indicate that there is a difference between overnight and daytime wind levels; care needs to be taken when interpreting this graph (i.e. while visually it may appear that there is a distinct pattern, to confirm that this is the case tests for statistical significance should be undertaken).

4. Autoregressive Integrated Moving Average Models

Chapter 4 focuses on applying a class of models, namely *time series* models, to a dataset. The software will enable the user to

- Use diagnostics to help identify an appropriate time series model to apply to the dataset.
- Determine whether the data needs to be *differenced*.
- Automatically fit a number of different time series models to the dataset and highlight which model provides the best fit to the data.
- Allow the user to fit a specific time series model of their choosing.
- Present model parameter estimates and a residuals analysis for any of the fitted models the user is interested in.

Section 4.1 will provide an introduction to time series models, Section 4.2 will outline some of the different approaches utilised in estimating/fitting the time series model parameters. The chapter will conclude with an illustration of how to use the software in conjunction with a brief summary of the main classes and methods used in the analysis.

4.1. Time Series Models Overview

Given a sequence of random variables

$$R_1, R_2, R_3, \dots, R_n, \dots \tag{4.1}$$

an *autoregressive, AR,* process of order p is one given by

$$R_t = \phi_1 R_{t-1} + \phi_2 R_{t-2} + \cdots \phi_p R_{t-p} + \varepsilon_t. \tag{4.2}$$

In Equation 4.2, ε_t represents an independent Normal random variable with mean 0 and a variance of σ^2, the ϕ_i are fixed parameters. In a similar manner a *moving average, MA*, process of order q is specified by

$$R_t = \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots \theta_q \varepsilon_{t-q} + \varepsilon_t \tag{4.3}$$

whereas an *autoregressive moving average, ARMA,* process of order (p, q) is simply a combination of Equations 4.2 and 4.3. That is

$$R_t = \phi_1 R_{t-1} + \phi_2 R_{t-2} + \cdots \phi_p R_{t-p} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots \theta_q \varepsilon_{t-q} + \varepsilon_t. \tag{4.4}$$

Such models are typically used to model observations that are equally spaced in time. The random variables $R_1, R_2, R_3, \dots, R_n$ are said to be *weakly stationary* if the expected value[2] and

[2] It is assumed that the mean is 0. If the mean is nonzero, then a constant is added to the right hand side of Equations 4.2, 4.3 and 4.4.

variance of each random variable is the same and if the autocovariance between any of the two variables only depends on the lag, or gap, between them. The assumption that the underlying time series is weakly stationary is critical when utilising time series models.

The *autocorrelation coefficient* (acf) and *partial autocorrelation coefficient* (pacf) can be used as diagnostic tools for determining which particular ARMA model is likely to provide the best fit to the dataset. The autocorrelation coefficient at lag k for a series of observations $r_1, r_2, \dots, r_n$, denoted by ρ_k is given by

$$\rho_k = \frac{\sum_{i=1}^{n-k}(r_i - \overline{r})(r_{i+k} - \overline{r})}{\sum_{i=1}^{n}(r_i - \overline{r})^2}. \tag{4.5}$$

In contrast, the partial autocorrelation gives the partial correlation of a time series with its own lagged values controlling for the values of the time series at all shorter lags. More formally, the partial autocorrelation coefficient at lag k for a dataset, ψ_k, can be calculated from the Durbin relations [7] as follows

$$\psi_k = \frac{\rho_k - \sum_{j=1}^{k-1}\psi_{k-1,j}\rho_{k-j}}{1 - \sum_{j=1}^{k-1}\psi_{k-1,j}\rho_{k-j}}, k > 1$$
$$\psi_{k,j} = \psi_{k-1,j} - \psi_k\psi_{k-1,k-j}. \tag{4.6}$$

In order to identify the particular ARMA model that is likely to provide the best fit to the dataset, the procedure is to examine the plot of the acf and pacf for different lags k. If the observations follow a *white noise* process (i.e. sequence of independent and identically distributed, *i.i.d*, random variables with mean 0 and variance σ^2), then one can typically expect 5% or less of the autocorrelations or partial autocorrelations to be significant[3]. If, however, the observations are from an AR type process, then the acf will decay whereas the pacf will cut-off after lag p (p being the order of the AR process). In a similar manner, for MA processes the acf will cut-off after lag q (q being the order of the MA process) while the pacf will decay. For ARMA processes the acf and pacf gradually decrease.

The acf for a sample can also assist in identifying whether or not the data requires *differencing* in order to achieve stationarity. Given a sequence of observations $r_1, r_2, \dots, r_n$ the first order differenced time series is given by

$$r_2 - r_1, r_3 - r_2, \dots, r_n - r_{n-1}. \tag{4.7}$$

The second order differenced Time Series is obtained in a similar manner i.e.

[3] The 5% significance level can be calculated by comparing the acf or pacf to $\frac{1.96}{\sqrt{n}}$ where n is the number of observations in the sample

$$(r_3 - r_2) - (r_2 - r_1), \dots, (r_n - r_{n-1}) - (r_{n-1} - r_{n-2}). \quad 4.8$$

For a non-stationary time series[4], for example one in which there is an underlying trend in the data, the acf will decrease slowly to 0 [13]. If on taking a first order difference it is observed that the acf quickly decreases to 0 it can be assumed that the transformed (differenced) data is stationary. In this instance, given a first order difference was used, the value of *i,* the integrated component of an *ARIMA(p, i, q)* model, is 1.

In the *Statistics* class, the following methods help to implement some of the aforementioned calculations/techniques

- *sampleautocorrelation(), samplepartialautocorrelation(), sampleautocovariance()*[5] – methods to calculate the sample acf, pacf and sample autocovariance.
- *difference()* – method which can be used to difference a time series. The method will use an attribute of the *Statistics* class, *Diff*, to specify the order of differencing.

4.2. Parameter Estimation and Residuals Analysis

Given a sequence of observations $r_1, r_2, \dots, r_n$ and an ARIMA model as outlined in Section 4.1, how are the model parameter estimates derived? An in-depth review of the differing approaches can be found in [14], [15], [16] and [17]; Sections 4.2.1 to 4.2.5 provide a brief synopsis of some of the methods.

4.2.1. Yule Walker

Given an AR model outlined by Equation 4.2, the autocovariance is given by

$$\gamma(\tau) = \begin{cases} \phi_1\gamma(\tau-1) + \phi_2\gamma(\tau-2) + \cdots + \phi_p\gamma(\tau-p), & \tau = 1,2,\dots,p \\ \phi_1\gamma(1) + \phi_2\gamma(2) + \cdots + \phi_p\gamma(p) + \sigma^2, & \tau = 0 \end{cases}. \quad 4.9$$

The system of equations defined by 4.9 is referred to as the *Yule Walker equations*. On replacing the autocovariance function with its sample equivalent, solving the system of simultaneous equations will yield the parameter estimates $(\hat{\phi}_1, \hat{\phi}_2, \dots, \hat{\phi}_p, \hat{\sigma}^2)$.

In the *WindApp2015* application, the following methods in the statistics class help to implement the Yule Walker approach

- *yulewalker*() – solves the system of equations specified by 4.9 with the sample autocovariance replaced by the sample autocorrelation. The method uses the *GElim* class to solve the simultaneous equations.

[4] As outlined in [13], a number of statistical significance tests for stationarity exist. These tests are not implemented in the *WindApp2015* application; however see functions such as *adf.test*, *kpss.test* and *ndiffs* in R.

[5] The sample autocovariance is calculated by $\sum_{i=1}^{n-k}(r_i - \overline{r})(r_{i+k} - \overline{r})$.

- *yulewalkerresiduals()* – given a set of observations and estimates $(\hat{\phi}_1, \hat{\phi}_2, \dots, \hat{\phi}_p)$ for an AR(p) model, this method will calculate the associated residuals.
- *yulewalkervariance()* – similarly, given a set of observations and parameter estimates $(\hat{\phi}_1, \hat{\phi}_2, \dots, \hat{\phi}_p)$ this method will calculate $\hat{\sigma}^2$.

4.2.2. Innovations Algorithm

The *Innovations Algorithm* is a recursive procedure for estimating the parameters of an MA(q) process. The algorithm (detailed in [15]) uses the following recursion relations

$$\hat{\theta}_{m,m-k} = \hat{v}_k^{-1}\left[\hat{\gamma}(m-k) - \sum_{j=0}^{k-1} \hat{\theta}_{m,m-j}\hat{\theta}_{k,k-j}\hat{v}_j\right], k = 0, \dots, m-1$$
$$\hat{v}_m = \hat{\gamma}(0) - \sum_{j=0}^{m-1} \hat{\theta}_{m,m-j}\hat{\theta}_{m,m-j}\hat{v}_j. \quad 4.10$$

Here $\hat{\gamma}(i)$ represents the sample autocovariance at lag i. The algorithm is as follows

1. Using $\hat{v}_0 = \hat{\gamma}(0)$ iterate on Equation 4.10 for $m = 1,2, \dots n$ where n is some integer (see Step 2).
2. The MA(q) model parameters are then estimated by the values $\hat{\theta}_{m,j}$ (with j ≤ q) where m is sufficiently large such that the values of $\hat{\theta}_{m,j}$ have stabilised.

The following methods of the *Statistics* class help to implement the Innovations Algorithm in the *WindApp2015* application

- *innovations()* – the method is supplied with a dataset, the order q of the MA model and the number of iterations of the algorithm (i.e. the n in Step 1 above). On reaching a specified stability criterion, the values of $\hat{\theta}_{m,j}$ and $\hat{v}_m$ are returned. If the stability criterion is not met, then default values of -1000 are returned.
- *innovationsresiduals()* – given a set of observations and estimates $(\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q)$ for an MA(q) model, this method will calculate the associated residuals.

4.2.3. Hannan Rissanen Procedure

The Hannan Rissanen procedure is used to fit ARMA(p, q) models to a dataset. The steps in the procedure are as follows (again full details can be found in [15])

1. An AR(k) model, with k being sufficiently large is fitted to the dataset i.e.

$$r_t = \chi_1 r_{t-1} + \chi_2 r_{t-2} + \cdots \chi_k r_{t-k} + \varepsilon_t. \quad 4.11$$

2. Using the observations $r_1, r_2, \dots, r_n$ and parameter estimates $(\hat{\chi}_1, \hat{\chi}_2, \dots, \hat{\chi}_k)$ calculate the residuals i.e.

$$\hat{\varepsilon_t} = r_t - \hat{\chi}_1 r_{t-1} - \hat{\chi}_2 r_{t-2} - \cdots - \hat{\chi}_k r_{t-k}. \quad 4.12$$

3. Using the residuals obtained from step 2, given the ARMA(p, q) model specified by

$$r_t = \phi_1 r_{t-1} + \phi_2 r_{t-2} + \cdots \phi_p r_{t-p} + \theta_1 \hat{\varepsilon}_{t-1} + \theta_2 \hat{\varepsilon}_{t-2} + \cdots + \theta_q \hat{\varepsilon}_{t-q} + \varepsilon_t \quad 4.13$$

use least squares regression[6] to determine estimates $(\hat{\phi}_1, \hat{\phi}_2, \dots, \hat{\phi}_p, \hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q)$.

4. $\hat{\sigma}^2$ is estimated by dividing the residual sum of squares by the number of terms in the sum.

In the *WindApp2015* software application, the following methods of the *Statistics* class implement the above functionality

- *HannanRissanen()* – implements the Hannan Rissanen procedure, least squares is performed via the *leastsquares()* method.
- *leastsquares()* – uses classes of type *Matrix* and *Vector* to implement least squares regression[7].
- *ARMAresiduals()* – given a set of observations and estimates $(\hat{\phi}_1, \hat{\phi}_2, \dots, \hat{\phi}_p, \hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q)$ for an ARMA(p, q) model this method will calculate the associated residuals. This function will also be used by methods in Section 4.2.5.
- *HannanRissanenVariance()* – determines the value $\hat{\sigma}^2$ as outlined in step 4 above.

4.2.4. Maximum Likelihood for AR(p) processes

When using maximum likelihood to determine parameter estimates for AR models either a conditional or exact likelihood approach can be used (a similar choice exists for MA and ARMA models). In the *Wind2015App* a conditional likelihood approach is followed. That is, conditional on observations $r_1, r_2, \dots, r_p$ the likelihood for the sample is given by

$$L(\theta) = p_\theta(r_{p+1}, \dots, r_n | r_1, \dots, r_p) = \prod_{t=p+1}^{n} p_\theta(r_t | r_{t-p}, \dots, r_{t-1}). \quad 4.14$$

On taking the log of the likelihood (under the assumption that the R_t conditional on the previous t-1 observations is normally distributed)

[6] Values $(\phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q)$ which minimise $\sum_{t=Max(p,q)+1}^{n} (r_t - \phi_1 r_{t-1} \cdots - \phi_p r_{t-p} - \theta_1 \hat{\varepsilon}_{t-1} \cdots - \theta_q \hat{\varepsilon}_{t-q})^2$.

[7] Given a linear regression model described by $y = X\beta + \varepsilon$, the estimate of $\hat{\beta}$ is $(X'X)^{-1}X'y$.

$$logL(\theta) = -\frac{n-p}{2}log(2\pi) - (n-p)log\sigma - \sum_{t=p+1}^{n}\left[\frac{(r_t - \phi_1 r_{t-1} - \cdots \phi_p r_{t-p})^2}{2\sigma^2}\right]. \quad 4.15$$

In order to determine the maximum likelihood estimates[8] of the parameters$(\phi_1, \ldots, \phi_p)$, on taking partial derivatives with respect to $\phi_1, \ldots, \phi_p$ and equating to zero a system of p equations of the following form result

$$\sum_{t=p+1}^{n}(r_t - \phi_1 r_{t-1} - \cdots \phi_p r_{t-p})r_{t-i} = 0. \quad 4.16$$

The i[th] equation results from taking the partial derivative with respect to ϕ_i. Solving the above system using a multi-dimensional Newton Raphson approach will yield the maximum likelihood estimates $(\hat{\phi}_1, \hat{\phi}_2, \ldots, \hat{\phi}_p)$.

A similar approach can be used to determine the maximum likelihood estimate of σ^2. That is, on differentiating with respect to σ^2 and letting the resulting equation equal 0 it can be seen that

$$\hat{\sigma}^2 = \frac{1}{n-p}\sum_{t=p+1}^{n}(r_t - \phi_1 r_{t-1} \cdots - \phi_p r_{t-p})^2. \quad 4.17$$

In the *Statistics* class of the *WindApp2015* application the following methods help to implement the aforementioned techniques:

- *ARML()* – given a set of observations, an initial guess for $(\hat{\phi}_1, \hat{\phi}_2, \ldots, \hat{\phi}_p)$ is obtained using the *yulewalker()* method. Next, instances of the *DR* and *FunctionMatrix* classes are used to find an estimate of $(\hat{\phi}_1, \hat{\phi}_2, \ldots, \hat{\phi}_p)$ which satisfies Equation 4.16 to within a specified tolerance.
- *ARMLresiduals()* – given a set of observations and estimates $(\hat{\phi}_1, \hat{\phi}_2, \ldots, \hat{\phi}_p)$ for an AR(p) model, this method will calculate the associated residuals.
- *ARMLVariance()* – given the residuals from an AR(p) model, the function will return an estimate of $\hat{\sigma}^2$ based on Equation 4.17.

4.2.5. Maximum Likelihood for MA and ARMA processes

Similar to the AR(p) maximum likelihood approach, when determining the maximum likelihood parameter estimates for MA and ARMA models a conditional likelihood approach is utilised in the *WindApp2015* application.

[8] Values $(\hat{\phi}_1, \hat{\phi}_2, \ldots, \hat{\phi}_p)$ which maximise the value of Equation 4.15.

Assuming a sequence of observations $r_1, r_2, \dots, r_n$ follow an MA(q) process as outlined in Equation 4.3 with $\varepsilon_1 = \varepsilon_2 = \cdots = \varepsilon_q = 0$, then the log likelihood is given by

$$logL(\theta) = -\frac{n}{2}log(2\pi) - \frac{n}{2}log\sigma^2 - \sum_{t=q+1}^{n} \frac{\varepsilon_t^2}{2\sigma^2}. \quad 4.18$$

The ε_t's can be found by iterating on Equation 4.3. It can be seen however that on substituting these values of ε_t into Equation 4.18 the resulting non-linear equation becomes complicated in nature and as a result it is difficult to find the maximum likelihood parameter estimates $\left(\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q, \hat{\sigma}^2\right)$ by taking partial derivatives[9], setting the resulting equations to zero and finding the root of the system of equations.

In the *WindApp2015* application a simpler, iterative, grid search approach is implemented. The procedure is as follows

1. Given a set of observations, produce an initial guess for the parameters $\left(\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q, \hat{\sigma}^2\right)$ by running the *innovations()* method.
2. Using the estimate of $\left(\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q, \hat{\sigma}^2\right)$ determine neighbouring grid points. For example, if we are in 2 dimensions (i.e. interested in finding best fit parameters $\hat{\theta}_1$ and $\hat{\sigma}^2$) then the grid points of interest include $\left(\hat{\theta}_1 - \delta, \hat{\sigma}^2 - \delta\right), \left(\hat{\theta}_1 - \delta, \hat{\sigma}^2\right), \left(\hat{\theta}_1 - \delta, \hat{\sigma}^2 + \delta\right), \left(\hat{\theta}_1, \hat{\sigma}^2 - \delta\right), \left(\hat{\theta}_1, \hat{\sigma}^2\right), \left(\hat{\theta}_1, \hat{\sigma}^2 + \delta\right), \left(\hat{\theta}_1 + \delta, \hat{\sigma}^2 - \delta\right), \left(\hat{\theta}_1 + \delta, \hat{\sigma}^2\right), \left(\hat{\theta}_1 + \delta, \hat{\sigma}^2 + \delta\right)$, where δ is the step size.
3. For each grid point in step 2, derive the related $\hat{\varepsilon}_t$'s by iterating on Equation 4.3. Using Equation 4.18 calculate the log-likelihood for each grid point.
4. The updated guess for $\left(\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q, \hat{\sigma}^2\right)$ is the grid point which gives the maximum log likelihood in step 3.
5. Repeat steps 2,3 and 4 until either
    a. The difference in the maximum likelihoods between successive iterations is less than a specified tolerance.
    b. The number of iterations exceeds a particular threshold e.g. 500.

If the algorithm does not converge (i.e. the maximum likelihood between successive iterations does not reduce below a specified threshold within a specified number of iterations), then a default value of -1000 is returned for each parameter.

[9] Using Equation 4.18 take partial derivatives with respect to $\theta_1, \dots, \theta_q, \sigma^2$.

A similar process is followed when determining the maximum likelihood estimates for an ARMA model. The only difference is that Equation 4.18 becomes

$$logL(\theta) = -\frac{n-p}{2}log(2\pi) - \frac{n-p}{2}log\sigma^2 - \sum_{t=q+1}^{n}\frac{\varepsilon_t^2}{2\sigma^2} \quad 4.19$$

and the $\hat{\varepsilon}_t$'s are generated by iterating on Equation 4.4.

Methods of note in the *statistics* class include

- *grid()* – this is a recursive function which is used to help generate the grid points referred to in step 2.
- *MAMLalternative(), ARMAMLalternative()*[10] – implements the algorithm outlined in steps 1-5.
- *MAMLresiduals(), ARMAMLresiduals()* – given a set of observations and parameter estimates from the relevant MA or ARMA model, these methods will calculate the associated residuals.

4.2.6. Residuals Analysis

After choosing a particular ARIMA model to fit to a dataset, estimating the model parameters and calculating the residuals, it is critical that a residuals analysis is performed. Visual checks include simply plotting the acf and pacf of the residuals for different lags k; examining a histogram of the residuals and a Q-Qplot of the residuals. If the fitted model is appropriate, one can typically expect to see no significant observations in the plot of the acf and pacf for the residuals; in a similar manner the histogram and Q-Qplot of the residuals should exhibit normal behaviour.

If the above is not the case, then the validity/appropriateness of fitting a particular ARIMA model to the dataset is called into question.

While examining the residuals a useful additional check is to examine the autocorrelation of the squared residuals for different lags. If there are significant autocorrelations in the squared residuals, then it is usually indicative that a *Generalised Autoregressive Conditional Heteroscedasticity* (GARCH) framework is appropriate. Additional details can be found in Section 5.

[10] Methods *MAML()* and *ARMAML()* also exist; the difference between *MAML()* and *MAMLalternative()* for example is that the former produces estimates $(\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q)$ whereas the latter produces estimates $(\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q, \hat{\sigma}^2)$.

4.3. Software: ARIMA Example

On opening the *WindApp2015* application the user clicks on the *ARIMA* tab and a new blank window appears. As usual, the user clicks on the *Load Data* button and selects the relevant Workbook and Excel spreadsheet. In this example we will utilise the transformed 2010 wind level data derived in Section 3.4. On selecting the worksheet of interest in the *ListBox* the user is presented with

- *Sample ACF and PACF* – a plot of the acf and pacf of the chosen dataset for lags 1 to 100. The plot also includes two horizontal lines at heights $-\frac{1.96}{\sqrt{n}}$ and $+\frac{1.96}{\sqrt{n}}$ which correspond to 5% significance levels (n is the number of points in the dataset).
- *Differencing* – as outlined in Section 4.1, if there is an underlying trend inherent in the data then it may be possible to remove the trend by differencing. If the acf decays slowly (and it has a number of significant observations), then it is usually indicative that the data needs to be differenced. The *Differencing* button will enable the user to difference the data.
- *Run ARIMA models* – on clicking this button various AR, MA and ARMA models will automatically be fitted to the dataset.

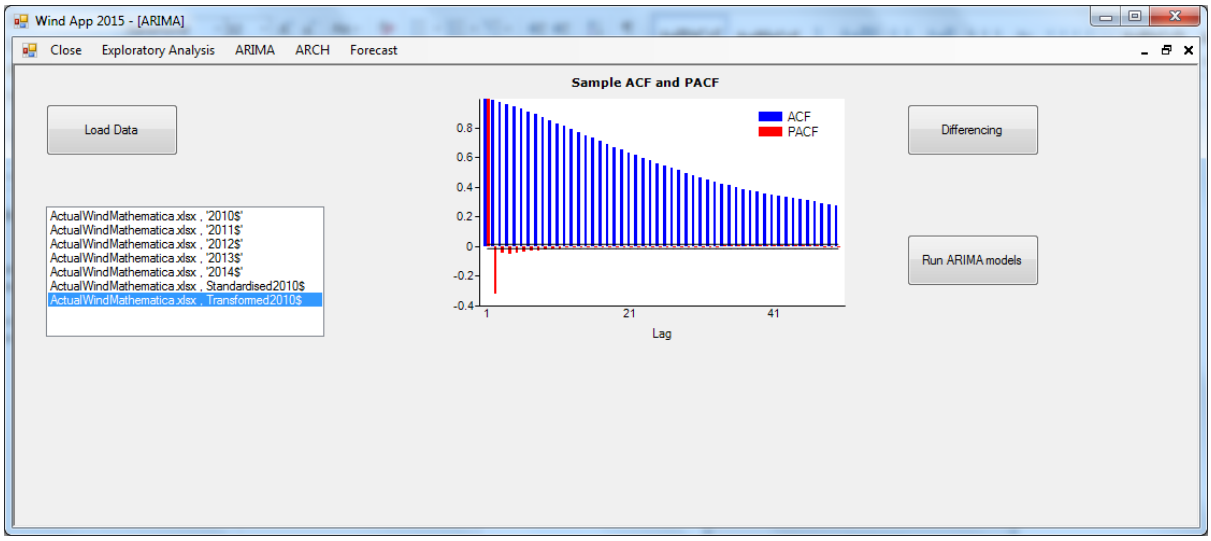

Figure 4-1.

Looking at Figure 4-1 it can be seen that the acf decays slowly and that there are also a large number of significant observations[11]. Hence, to difference the data, on clicking the *Differencing* button the user is presented with the following form

[11] If there is no requirement for differencing (i.e. the plot of the ACF indicates stationarity), then on clicking the *Run ARIMA models* button the user is presented with the option of mean adjusting the time series. If the user clicks *yes* then each data point, d_i, becomes $d_i - \bar{d}$ where $\bar{d}$ is the mean of the observations. This helps ensure an intercept of 0 in Equations 4.2, 4.3 and 4.4.

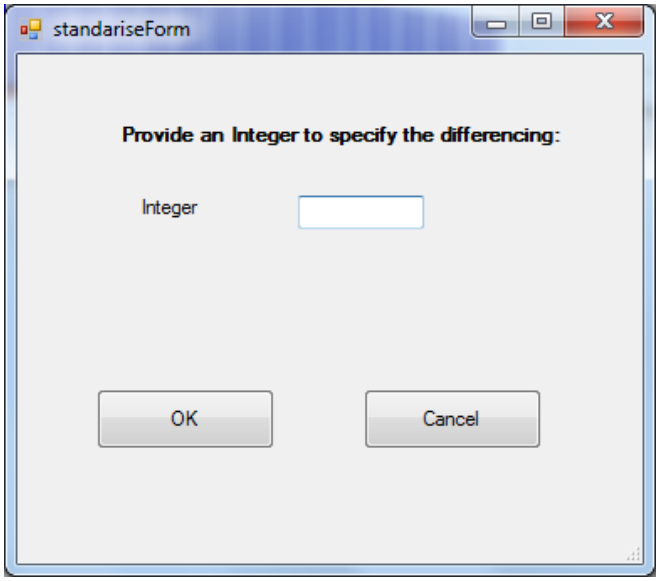

Figure 4-2.

Here the user enters an integer to specify the order of differencing with which they wish to apply to the data[12]. For most datasets a difference of order 1 or 2 will suffice, in this example the user enters 1 for example and on clicking the *OK* button they are presented with the following

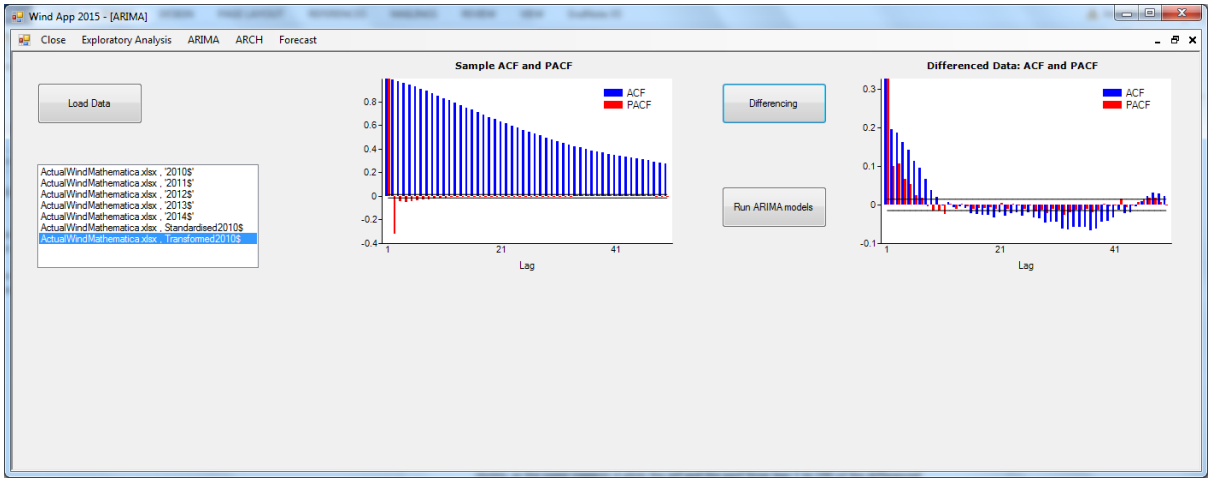

Figure 4-3.

In Figure 4-3 it can be seen that an additional plot titled *Differenced Data: ACF and PACF* is shown. As the name suggests, it plots the acf and the pacf from lags 1 to 100 of the differenced data. The gradual decline of the acf and pacf is indicative of an ARMA type process, it is worth noting however that significant autocorrelations at lags greater than 20 is indicative of a potential need for further differencing.

On clicking the *Run ARIMA models* button the user is presented with (see Figure 4-4)

[12] Using the *forecast* package in R the *ndiffs* command can be used to determine how many differences are required to make the time series stationary. See http://www.inside-r.org/packages/cran/forecast/docs/ndiffs

- *DataGridView* – the datagridview presents the various AR, MA and ARMA models that have been fitted to the dataset. The datagridview also displays the sum of squared residuals, the *Akaike Information Criterion* (AIC)[13] and the $\hat{\sigma}^2$ estimate for each model. The model with the lowest sum of squared residuals is automatically highlighted in blue. To view the model parameter estimates and residual diagnostics associated with one of the other fitted models the user simply double clicks on the model of interest in the datagridview.
- *Residuals ACF* – a plot of the acf of the residuals for the model of interest.
- *Histogram Residuals* – a histogram of the residuals for the model of interest.
- *Model Parameter Estimates* – a table showing the model parameter estimates.
- *Residual Statistics* – summary statistics of the model residuals.
- *Q-QPlot Model Residuals* – a Q-Qplot of the residuals for the model of interest.
- *Squared Residuals ACF* – a plot of the acf for the squared residuals, such plots can be used to detect the presence of GARCH type disturbances in the underlying data.
- *Fit Specific Model* – this button allows the user to fit an ARMA(p, q) model of their choosing to the data. The model parameters are estimated via maximum likelihood; if the parameter estimation process converges the model will be added to the datagridview and the parameter estimates and residual diagnostics will be displayed.
- *Save ARIMA models* – on clicking this button a summary of the various models will be saved to a csv file in a similar location to that of the original Excel spreadsheet. Details recorded include the transformation type and values if applicable (i.e. Weibull/Iterative, shape, scale), model fitting procedure (i.e. Yule Walker etc.), number of AR parameters, number of MA parameters, number of differences, AIC, sum of squared residuals, $\hat{\sigma}^2$ and model parameter estimates.

[13] In *WindApp2015* AIC is the value given by $n\ ln(\hat{\sigma}^2) + 2T$ where n is the number of observations and T is the number of parameters from the ARMA(p, q) model that need to be estimated. This definition is taken from Equation 6 in [7]

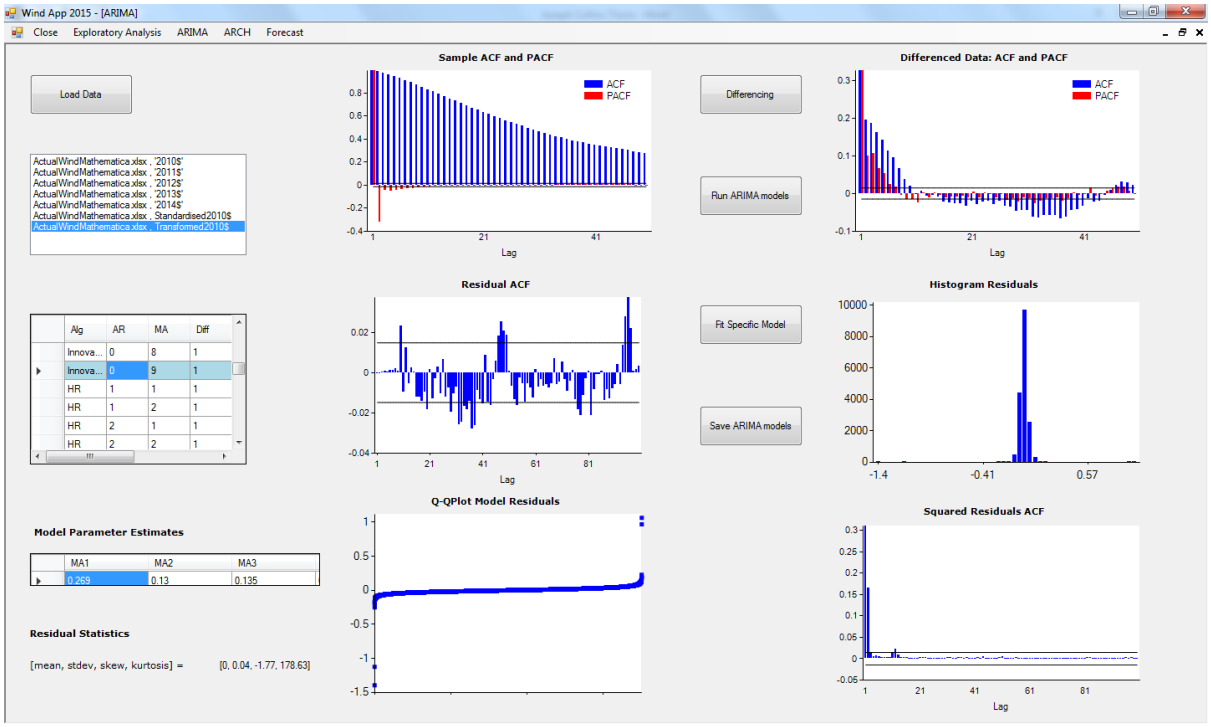

Figure 4-4.

Examining Figure 4-4, it can be seen that for the transformed 2010 wind data, on taking a difference of order 1, an MA(9) model produces the lowest sum of squared residuals (i.e. this is the model highlighted in blue in the datagridview)[14]. While the histogram and Q-Qplot of the residuals from the MA(9) model seem to approximate a Normal distribution, the size and number of outliers is a cause for concern. Similarly, the acf of the model residuals do not appear to follow that of a white noise process (i.e. some of the autocorrelations appear to be either consistently positive or negative).

Taking a cursory look at the fitted models in the datagridview some notable features include

- More parsimonious models such as ARMA(p, q) with $1 \leq p, q \leq 3$ have a sum of squared residuals close to that of the aforementioned MA(9) model.
- ARMA(p, q) models, with both p and q > 0, that have been fitted via Hannan Rissanen can produce model parameter estimates that are significantly different to those produced via the maximum likelihood approach.
- Not all models which are fit via maximum likelihood will converge within the specified number of iterations. For example the ARMA(1, 3) model is not presented in the datagridview as it did not converge within 500 iterations.

[14] Note that the AR(9) model which has been fit via maximum likelihood produces a sum of squared residuals which is almost in line with that given by the MA(9) model.

- For both MA and AR models which have been fitted via either the innovations or Yule Walker method, it can be seen that the parameter estimates are close to, or coincide with, those produced via maximum likelihood approaches. This is as expected as it is these estimates (i.e. either the innovations or Yule Walker estimates) that are used as the initial guess when determining the maximum likelihood estimates.

For completeness, Figure 4-5 shows the transformed 2010 wind data when a difference of order 2 is applied. For the differenced dataset it can be seen that an ARMA(3, 3) model produces the lowest sum of squared residuals.

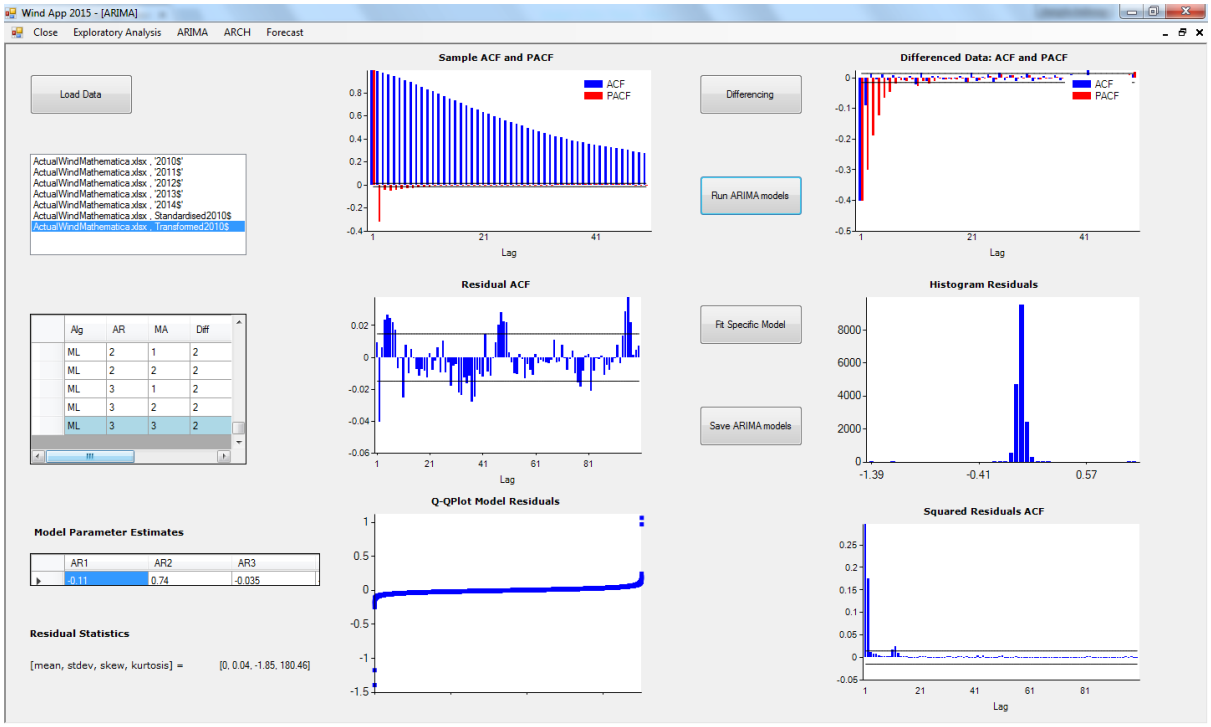

Figure 4-5.

5. GARCH

Chapter 5 provides a cursory introduction to *Autoregressive Conditional Heteroscedasticity* (ARCH) and *Generalised Autoregressive Conditional Heteroscedasticity* (GARCH) models. The attractive feature of this suite of models is that they can capture particular features of a time series/dataset which traditional ARIMA models cannot. One of the primary areas for the application of such models has been in financial markets: when examining asset returns it is often the case that periods of large changes in the underlying asset price tend to be followed by other large changes and, similarly, small changes in the price of an asset tended to be followed by other small changes. Engle [18] and subsequently Bollerslev [19] developed a suite of time series models in which the conditional variance was allowed to vary over time; this class of models incorporates volatility clustering which as discussed is frequently observed in financial markets.

Section 5.1 will provide a brief description of the ARCH and GARCH framework, Section 5.2 will outline the process by which the model parameters can be estimated and Section 5.3 will provide an illustration of how to use the *WindApp2015* software to calibrate such models.

5.1. ARCH and GARCH Overview

Assume that R_t follows an ARMA(p, q) process as outlined in Equation 5.1 with the ε_t's as usual having mean 0 and variance σ^2. Thus,

$$R_t = \phi_1 R_{t-1} + \cdots + \phi_p R_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t. \tag{5.1}$$

The ARCH(q) model is given by

$$\begin{aligned} \varepsilon_t | F_{t-1} &= v_t \sqrt{h_t} \\ h_t &= \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \alpha_2 \varepsilon_{t-2}^2 + \cdots + \alpha_q \varepsilon_{t-q}^2. \end{aligned} \tag{5.2}$$

In Equation 5.2, $\varepsilon_t | F_{t-1}$, means ε_t conditional on the information up to and including time t-1. The variable v_t, in 5.2, is normally distributed with mean of 0 and unit variance. Another way of specifying the model is as follows

$$\begin{aligned} R_t | F_{t-1} &\sim N(X_t \beta, h_t) \\ h_t &= \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \alpha_2 \varepsilon_{t-2}^2 + \cdots + \alpha_q \varepsilon_{t-q}^2 \\ \varepsilon_t &= R_t - X_t \beta. \end{aligned} \tag{5.3}$$

$X_t \beta$ in Equation 5.3 could be an ARIMA model similar to Equation 5.1, it could be a constant or it could be a combination of lagged endogenous or exogenous variables. For the purposes of the *WindApp2015* software application we will assume that $X_t \beta$ follows an ARIMA model.

Note that in Equation 5.3, $\alpha_0 > 0$ and $\alpha_i \geq 0$ for i=1, 2,…, q. As outlined in [16], an additional requirement (for stationarity) is that

$$\alpha_1 + \alpha_2 + \cdots + \alpha_q < 1. \tag{5.4}$$

In a similar manner the GARCH(p, q) model is defined by

$$\begin{aligned} R_t|F_{t-1} &\sim N(X_t\beta, h_t) \\ h_t &= \alpha_0 + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 + \sum_{i=1}^{p} \delta_i h_{t-i} \\ \varepsilon_t &= R_t - X_t\beta. \end{aligned} \tag{5.5}$$

From [16] it can be seen that $\alpha_0 > 0$, $\alpha_i \geq 0$ for i=1, 2,…, q and $\delta_i \geq 0$ for i=1, 2,…, p. An additional requirement, again similar to that specified in Equation 5.4, is that

$$(\alpha_1 + \delta_1) + (\alpha_2 + \delta_2) + \cdots + (\alpha_p + \delta_p) < 1. \tag{5.6}$$

Section 21.2 of [16] also provides an explanation as to why the acf (for different lags k) of the squared residuals of an ARIMA model are examined in order to check for the presence of GARCH type disturbances.

Various extensions of ARCH and GARCH exist (i.e. IGARCH, EGARCH, ARCH-M etc.). However, for the purposes of the *WindApp2015* application we will restrict our interest to ARCH and GARCH type processes.

5.2. Parameter Estimation

Akin to Sections 4.2.4 and 4.2.5, in order to determine the best fit ARCH and GARCH parameters specified in Equations 5.3 and 5.5, the *WindApp2015* application uses conditional maximum likelihood estimation.

Conditioning on the first q observations, the log-likelihood for the ARCH(q) model is given by

$$logL(\theta) = -\frac{n}{2}log(2\pi) - \frac{1}{2}\sum_{t=q+1}^{n} log(h_t) - \frac{1}{2}\sum_{t=q+1}^{n} \frac{(r_t - X_t\beta)^2}{h_t}. \tag{5.7}$$

In order to determine the values $(\hat{\alpha}_0, \hat{\alpha}_1, \dots, \hat{\alpha}_q)$ which maximise the log-likelihood given by Equation 5.7 we implement an iterative grid-search approach similar to that used to determine maximum likelihood parameter estimates for MA and ARMA models. The procedure is as follows

1. Given a set of observations and ARMA(p, q) maximum likelihood parameter estimates $(\hat{\phi}_1, \hat{\phi}_2, \dots, \hat{\phi}_p, \hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q)$, calculate the associated residuals, $\hat{\varepsilon}_t$.

2. Produce an initial guess for the parameters $(\hat{\alpha}_0, \hat{\alpha}_1, \dots, \hat{\alpha}_q)$ by setting $\hat{\alpha}_0$ equal to 0.5 and letting

$$\alpha_i = \frac{1 - 0.05}{q}, i = 1,2, \dots q. \quad 5.8$$

3. Using the estimates $(\hat{\alpha}_0, \hat{\alpha}_1, \dots, \hat{\alpha}_q)$ determine neighbouring grid points. For each grid point calculate the related h_t's as follows

$$h_t = \hat{\alpha}_0 + \hat{\alpha}_1 \hat{\varepsilon}_{t-1}^2 + \cdots + \hat{\alpha}_q \hat{\varepsilon}_{t-q}^2. \quad 5.9$$

Determine the log-likelihood for each grid-point using Equation 5.7. The updated guess for $(\hat{\alpha}_0, \hat{\alpha}_1, \dots, \hat{\alpha}_q)$ is the grid point which gives the maximum log-likelihood.

4. Repeat Step 3 until either

    a. The difference in the maximum likelihoods between successive iterations is less than a specified tolerance.
    b. The number of iterations exceeds a particular threshold e.g. 500.

If the algorithm does not converge (i.e. the maximum likelihood between successive iterations does not fall below a specified threshold within a specified number of iterations), then a default value of -1000 is returned for each parameter.

The process for determining the maximum likelihood estimates $(\hat{\alpha}_0, \hat{\alpha}_1, \dots, \hat{\alpha}_q, \hat{\delta}_1, \dots, \hat{\delta}_p)$ of a GARCH(p, q) model is similar, with the exception that Equation 5.9 is replaced by

$$h_t = \hat{\alpha}_0 + \hat{\alpha}_1 \hat{\varepsilon}_{t-1}^2 + \cdots + \hat{\alpha}_q \hat{\varepsilon}_{t-q}^2 + \hat{\delta}_1 h_{t-1} + \cdots + \hat{\delta}_p h_{t-p} \quad 5.10$$

Methods of the *Statistics* class which help implement the above functionality include

- *grid()* – as outlined in Section 4.2.5, *grid()* is a recursive function which helps generate the set of grid points referred to in Step 3.
- *archML(), garchML()* – implements the algorithm outlined in Steps 1-4, these methods will use the *archht()* and *garchht()* functions respectively.
- *archht(), garchht()* – these methods calculate the h_t's using either Equation 5.9 or Equation 5.10 as appropriate.

5.3. Software: ARCH and GARCH Example

On opening the *WindApp2015* application the user clicks on the *ARCH* tab and a new blank window appears, as before the user clicks on the *Load Data* button and selects the relevant Workbook and Excel spreadsheet. If the user selects a worksheet for which no ARIMA models

have previously been calibrated and saved then a warning message appears and no information is displayed. In this example we will continue to work with the transformed 2010 wind level data seen in Sections 3.4 and 4.3. On selecting the relevant worksheet from the *ListBox* the user is presented with (see Figure 5-1)

- *Fitted ARIMA Models* – this datagridview presents a summary of the ARIMA models that were calibrated in Section 4.3. If the user wants to choose a particular ARIMA model to represent the $X_t\beta$ in either Equation 5.3 or 5.5 they simply highlight the row of interest in the datagridview.
- *Fit ARCH and GARCH* – on clicking this button various ARCH and GARCH models will be applied to the dataset. As highlighted in the previous bullet point, the choice of $X_t\beta$ is given by the row highlighted in the *Fitted ARIMA models* datagridview.

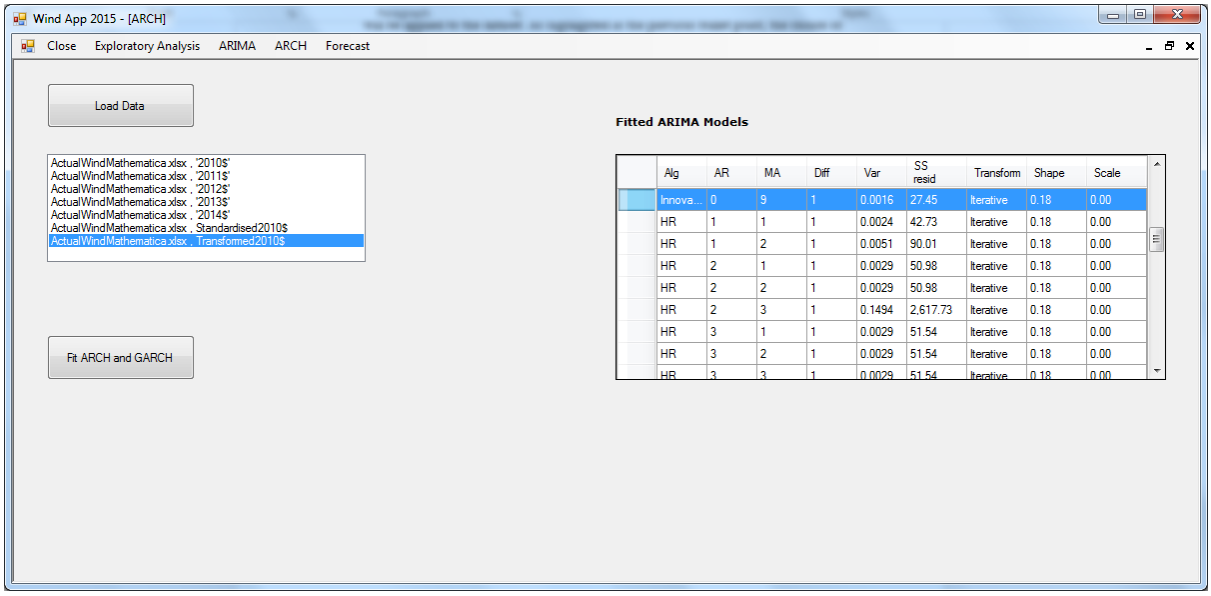

Figure 5-1.

On clicking the *Fit ARCH and GARCH* button we are presented with Figure 5-2. It is similar to Figure 5-1 with the addition of a *Fitted ARCH/GARCH Models* datagridview. This datagridview shows the number of GARCH p and q parameters in each model in addition to the AIC for that model. Here the AIC value is given by[15]

$$AIC = 2k - 2ln(L). \tag{5.11}$$

where k is the number of free parameters and L is the likelihood.

[15] Equation 16 in [20]. This contrasts to the AIC calculation utilised in Section 4.3 and described in Footnote 13.

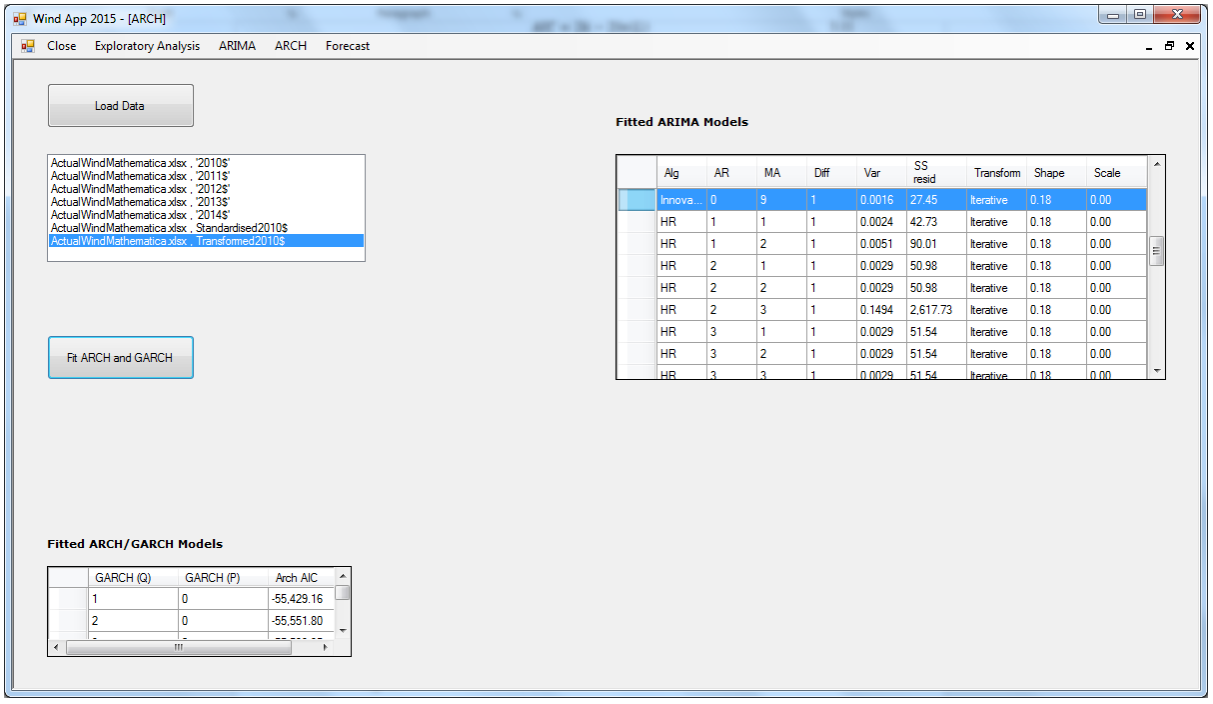

Figure 5-2.

On clicking one of the models in the *Fitted ARCH/GARCH Models* datagridview we are presented with (see Figure 5-3)

- *ARIMA Parameters* – this datagridview presents the ARIMA parameter estimates $(\hat{\phi}_1, \hat{\phi}_2, \dots, \hat{\phi}_p, \hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q)$ used in $X_t\beta$.
- *ARCH/GARCH Parameters* – this datagridview presents the ARCH/GARCH parameter estimates $(\hat{\alpha}_0, \hat{\alpha}_1, \dots, \hat{\alpha}_q, \hat{\delta}_1, \dots, \hat{\delta}_p)$.
- *ACF of v(t) from ARCH/GARCH model*– using the model parameter estimates $(\hat{\phi}_1, \hat{\phi}_2, \dots, \hat{\phi}_p, \hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q)$ and $(\hat{\alpha}_0, \hat{\alpha}_1, \dots, \hat{\alpha}_q, \hat{\delta}_1, \dots, \hat{\delta}_p)$ the $\hat{\varepsilon}_t$'s and h_t's for the dataset are calculated, the v_t values are derived via Equation 5.2 and they are expected to be i.i.d with mean of 0 and unit variance. The plot gives the acf of the v_t's for different lags k.
- *Save ARCH/GARCH models* – on clicking this button a summary of the various models will be saved to a csv file in a similar location to that of the original Excel spreadsheet.

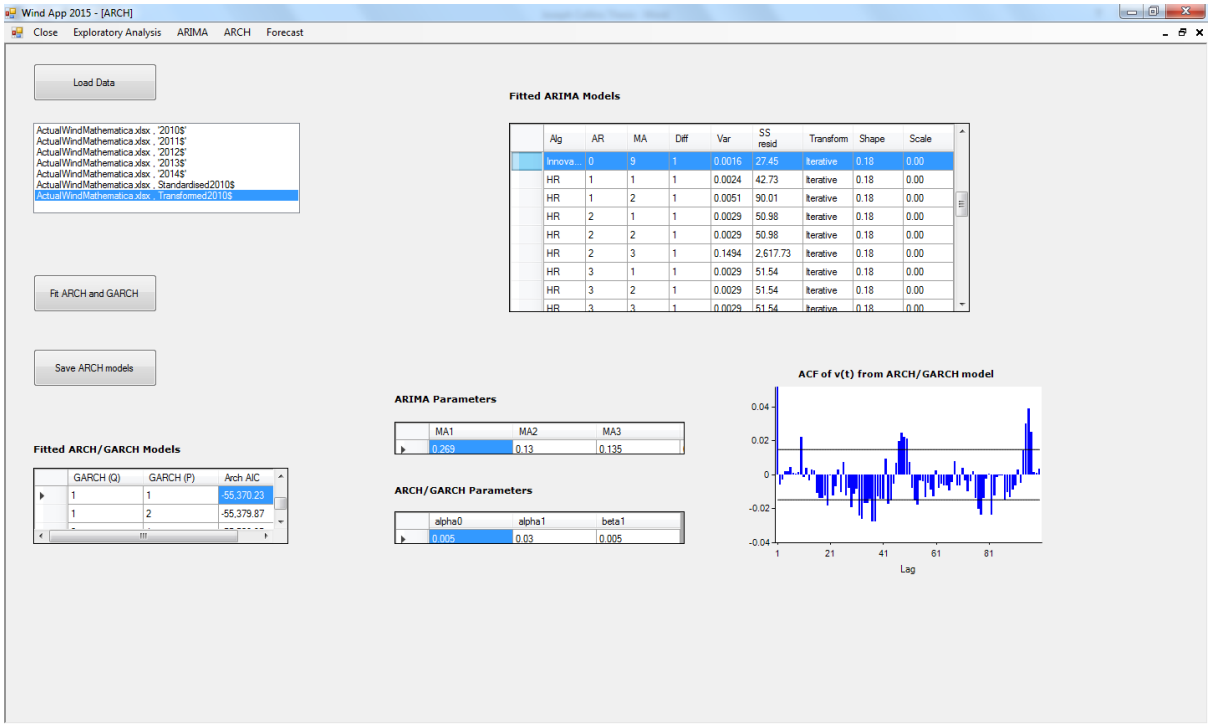

Figure 5-3.

Looking at the acf plot in Figure 5-3 it can be seen that there is still a significant observation at lag 1 (albeit lag's 1 and 2 in Figure 4-4 were significant). Also of note is the number of significant observations from lag 30 onwards.

Similar to commentary in Section 4.2.6, the acf plot raises questions as to the appropriateness of a GARCH(1, 1) model being fitted to the transformed and differenced 2010 wind data.

6. Forecasting

Given the data transformations and model parameter estimates from Sections 3, 4 and 5 in this chapter a brief outline of how these calibrated models can be utilised to make predictive forecasts is provided. Section 6.1 describes the theoretical approach to producing predictive forecasts for ARMA/ARCH/GARCH models while Section 6.2 provides an example of how to use the relevant *WindApp2015* software component.

6.1. Prediction

Given a sequence of observations $r_1, r_2, \dots, r_t$, an estimate for the future value of the random variable R_{t+n}, where n is the number of steps in the future, is required. This estimate will be denoted by $\hat{r}_{t+n|t}$. It is known that the *optimal predictor* n steps ahead, denoted by $\tilde{r}_{t+n|t}$, is given by the conditional expectation of R_{t+n} given all the current information. That is

$$\tilde{r}_{t+n|t} = E(R_{t+n}|F_t). \tag{6.1}$$

From [21] it can be seen that the optimal predictor n steps ahead, $\tilde{r}_{t+n|t}$, has an attractive property in that it minimises the value of

$$MSE(\hat{r}_{t+n|t}) = E\left[\left(R_{t+n} - \hat{r}_{t+n|t}\right)^2\right]. \tag{6.2}$$

Given an ARMA(p, q) model of the form

$$R_{t+n} = \phi_1 R_{t+n-1} + \cdots \phi_p R_{t+n-p} + \theta_1 \varepsilon_{t+n-1} + \cdots \theta_q \varepsilon_{t+n-q} + \varepsilon_{t+n.} \tag{6.3}$$

taking the expectation conditional on information up to and including time t yields

$$\tilde{r}_{t+n|t} = \phi_1 \tilde{r}_{t+n-1|t} + \cdots \phi_p \tilde{r}_{t+n-p|t} + \theta_1 \tilde{\varepsilon}_{t+n-1|t} + \cdots \theta_q \tilde{\varepsilon}_{t+n-q|t} + \tilde{\varepsilon}_{t+n|t}, l = 1, \dots \tag{6.4}$$

It is also the case that

$$\begin{aligned} \tilde{r}_{t+n|t} &= r_{t+n} \; for \; n \leq 0 \\ \tilde{\varepsilon}_{t+n|t} &= \begin{cases} 0 \; for \; n > 0 \\ \varepsilon_{t+n} \; for \; n \leq 0 \end{cases}. \end{aligned} \tag{6.5}$$

By iterating on Equations 6.4 and 6.5 predictive forecasts for ARMA(p, q) processes can be made. For an AR(1) model for example, it can be seen that

$$\tilde{r}_{t+n|t} = \phi_1^n r_t. \tag{6.6}$$

In a similar manner, for an MA(1) process, using equations 6.4 and 6.5 yields[16]

$$\tilde{r}_{t+1|t} = \theta_1 \varepsilon_t. \tag{6.7}$$

[16] A characteristic of MA(q) processes is that non trivial (i.e. non zero) forecasts can only be made q steps into the future.

In the *Statistics* class of the *WindApp2015* software the *predictnstepsahead()* method produces estimates $\tilde{r}_{t+n|t}$ using Equations 6.4 and 6.5.

The *WindApp2015* application also provides the user with the ability to simulate a particular ARMA/ARCH/GARCH process. The methods of the *Statistics* class which help to implement this functionality include

- *ARMAsimulatenstepsahead()* – given a set of observations and ARMA (p, q) parameter estimates $\left(\hat{\phi}_1, \hat{\phi}_2, \dots, \hat{\phi}_p, \hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q, \hat{\sigma}^2\right)$, the method will produce a simulated set of time series observations from a specified point in time.
- *ARCHsimulatenstepsahead()* – given a set of observations, ARMA (p, q) parameter estimates $\left(\hat{\phi}_1, \hat{\phi}_2, \dots, \hat{\phi}_p, \hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q, \hat{\sigma}^2\right)$ and ARCH(q) parameter estimates $\left(\hat{\alpha}_0, \hat{\alpha}_1, \dots, \hat{\alpha}_q\right)$, the method will produce a simulated set of time series observations from a specified point in time.
- *GARCHsimulatenstepsahead()* – given a set of observations and ARMA (p, q) parameter estimates, in addition to GARCH(q, p) parameter estimates $\left(\hat{\alpha}_0, \hat{\alpha}_1, \dots, \hat{\alpha}_q, \hat{\delta}_1, \dots, \hat{\delta}_p\right)$, the function will simulate the time series model from a user defined point in time.

For ARMA(p, q) simulations, the *ARMAsimulatenstepsahead()* function simply takes a random draw from a Normal distribution with mean 0 and variance $\hat{\sigma}^2$. Given a set of observations $r_1, r_2, \dots, r_t$, the first simulated observation, denoted by $\dot{r}_{t+1}$, can be obtained using Equation 6.3 with the value of ε_{t+1} set equal to the aforementioned random draw. Estimates $\dot{r}_{t+2}, \dot{r}_{t+3}, \dots, \dot{r}_{t+k}$, are obtained in a similar manner.

For ARCH(q) and GARCH(p, q) simulations, again a random draw from a Normal distribution is required, however this time the distribution has a mean of 0 and a variance of 1. Given the set of observations $r_1, r_2, \dots, r_t$, taking Equation 5.2 with the value of v_{t+1} set equal to the random draw from the Normal distribution, a value of ε_{t+1} can then be calculated. The first simulated observation, $\dot{r}_{t+1}$, is then determined using Equation 6.3. If additional simulated time series observations are required (i.e. $\dot{r}_{t+2}, \dot{r}_{t+3}, \dots, \dot{r}_{t+k}$) the process is simply repeated.

It was seen in Section 4.1 that, even after transforming the data, some time series datasets may require differencing in order to achieve stationarity. If such differencing occurs, then the related time series model forecasts/simulations will be for the differenced data and not for the transformed dataset. However, it is a relatively straightforward process to convert a forecast/simulation for the differenced data into a forecast/simulation for the transformed

dataset. In the *Statistics* class, the *predictnstepsaheadDD(), ARMAsimulatenstepsaheadDD(), ARCHsimulatenstepsaheadDD()* and *GARCHsimulatenstepsaheadDD()* methods work with the differenced data to produce forecasts and simulations for the transformed dataset.

Finally, if the data has been transformed as outlined in Section 3.1 (and possibly mean adjusted as described in Footnote 11) then the *unwind()*[17] function of the *Statistics* class is used to convert to forecast/simulation into the units of the original dataset. Note that if additional standardisations have been applied to the original dataset as discussed in Section 3.1 (for example hypothesising that there is a difference between overnight and daytime wind levels), the *unwind()* function does not currently have the capability to adjust for this.

6.2. Software: Forecast Example

On clicking the *Forecast* tab a new window appears displaying a *Load Data* button and a blank *ListBox*. The user clicks on the *Load Data* button, locates the relevant workbook and then clicks *OK*. The user then selects the worksheet of interest from the *ListBox* and a datagridview titled *Fitted ARIMA/ARCH/GARCH Models* will appear, similar to shown in Figure 6-1. Note that if no ARIMA/ARCH/GARCH models have been calibrated and saved (i.e. Sections 4.3 and 5.3) then a warning message will appear and no information will be displayed.

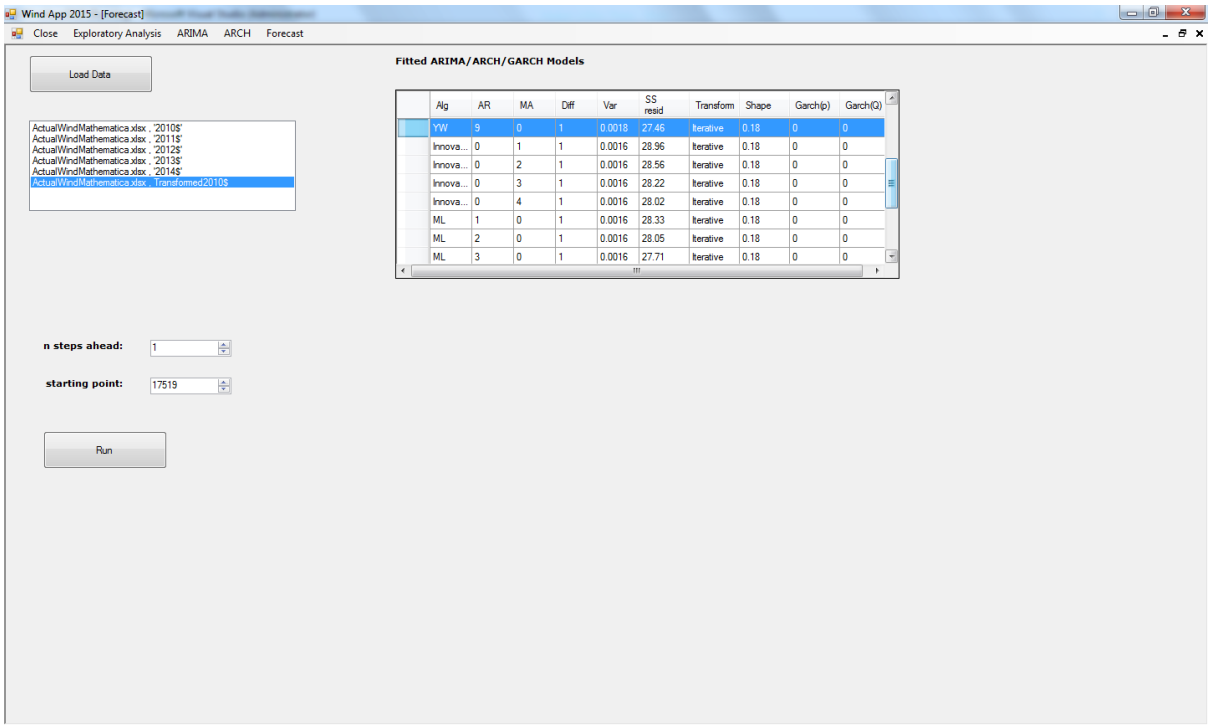

Figure 6-1

[17] The *unwind()* function automatically caps any forecast/simulation values at 3500; the reasoning being that at the current time the total wind energy in the SEM schedule (in MW) would not exceed this threshold.

The datagridview will display the high level details of the various ARIMA/ARCH/GARCH models that have previously been fit to the dataset. The *n steps ahead* and *starting point* numeric buttons allow the user to specify (a) how many periods ahead they would like to forecast/simulate (b) from what particular point in time they would like the forecast/simulation to begin.

For example, on choosing the transformed 2010 wind data and setting the value of *n steps ahead* and *starting point* numeric buttons to 20 and 17344 respectively, clicking the *Run* button produces the graphs shown in Figure 6-2.

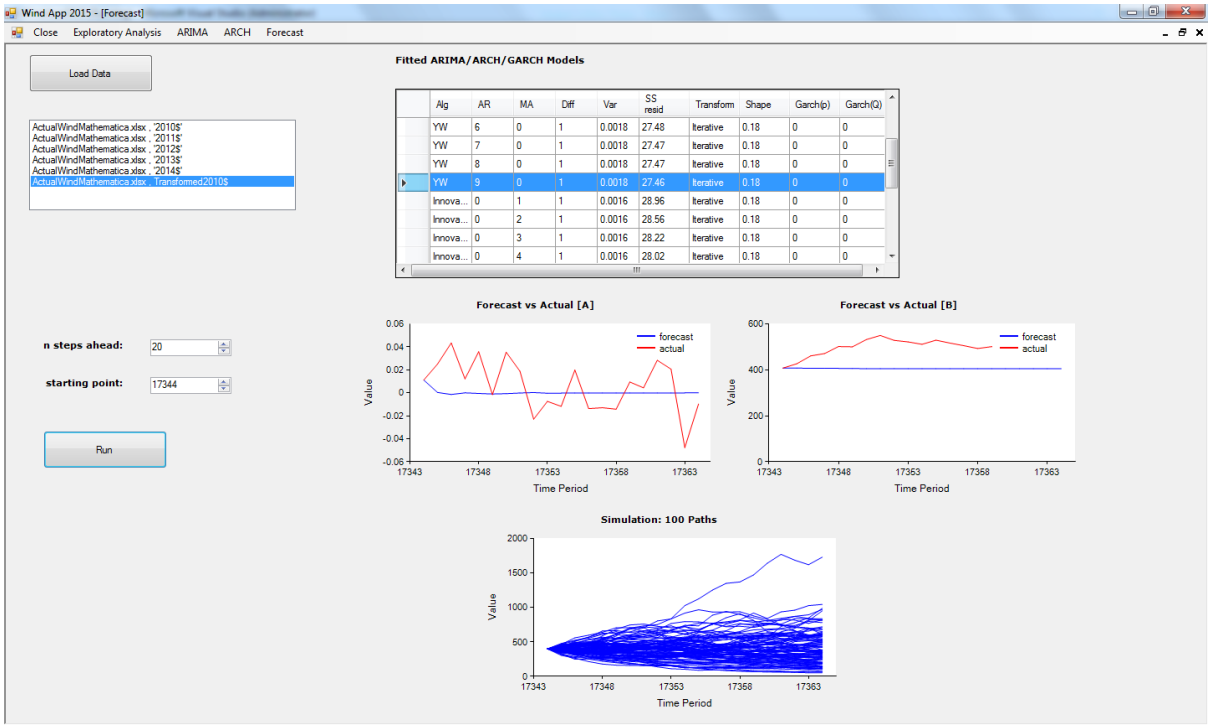

Figure 6-2.

The graphs show the forecasts/simulations that derive from the model selected by the user from the *Fitted ARMA/ARCH/GARCH Models* datagridview. The model relating to the graphs shown in Figure 6-2 for example is an AR(9) model, fitted via Yule-Walker, in which the data was transformed via an iterative process and subsequently had a difference of order 1 applied. The plots are as follows

- *Forecast vs Actual (A)* – the graph displays a plot of the forecast versus actual values starting from a time point defined by the user for a specified number of steps into the future. The units will be those against which the ARIMA model was calibrated. For example, if the dataset was transformed and differenced, the forecast and actual values will be in equivalent units to that of the transformed and differenced data.

- *Forecast vs Actual (B)* – similar to the *Forecast vs Actual (A)* graph except the forecast and actual values will be in units equivalent to that of the original dataset. If the original dataset was not transformed, or subsequently was not differenced, then this graph will not be displayed.
- *Simulation: 100 Paths* – as the name suggests, starting from a point in time specified by the user, the graphs shows 100 simulated paths for the ARIMA/ARCH/GARCH model chosen by the user. The units will be those of the original dataset.

Figure 6-3 presents an example of an MA(9) process, fitted via maximum likelihood, with the conditional variance defined by a GARCH(1, 1) model, in which the data was transformed via an iterative process and subsequently had a difference of order 1 applied. Forecasts and simulations commence at time point 15,244 for 20 steps into the future.

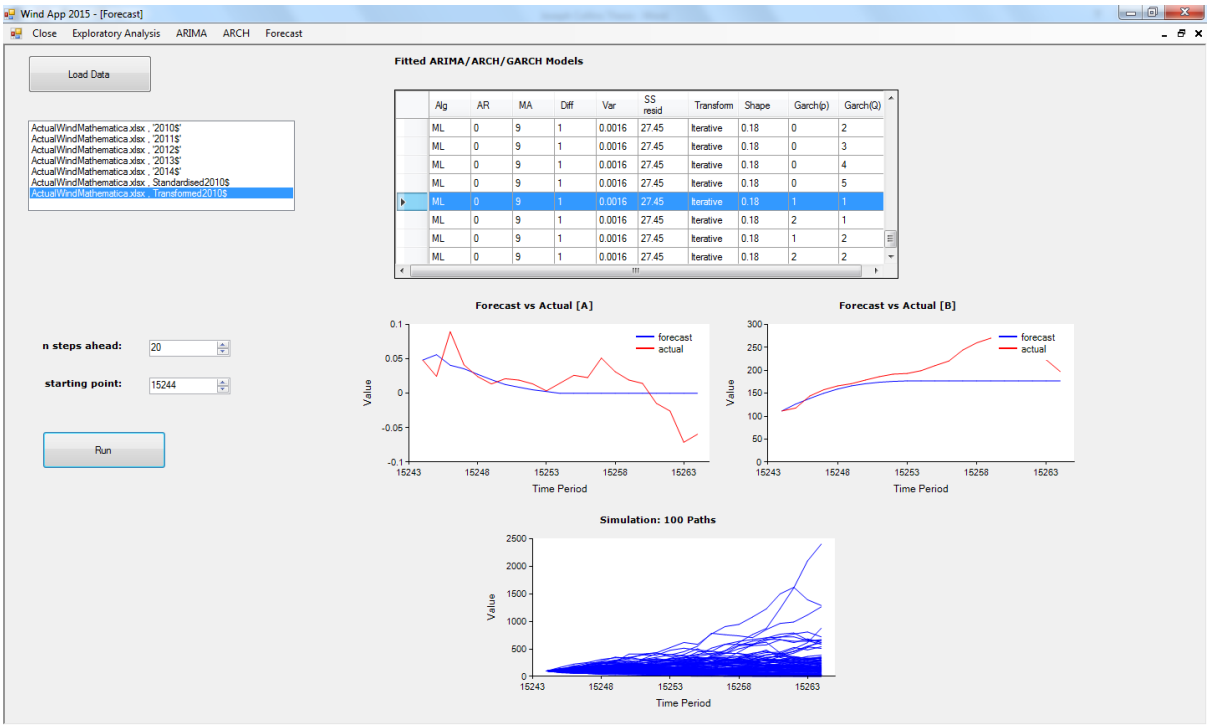

Figure 6-3.

7. Conclusion

This dissertation can be viewed as a *user manual* with illustrations as to the correct use of the *WindApp2015* software. This *user manual* also provides an overview of the mathematical and statistical theory underlying the various software components.

The primary objective of this thesis was to build a software application which could be used to apply time series modelling techniques to different datasets. The *WindApp2015* application, developed in C#, automatically calibrates various ARIMA, ARCH and GARCH models to the dataset provided by the user. These calibrated models can in turn be used to provide predictive forecasts and simulations.

At each stage in the development of the *WindApp2015* application the C# implementation of the various statistical algorithms were cross checked against R for consistency purposes. With the exception of the Hannan Rissanen algorithm, the *WindApp2015* parameter estimates were in-line, or closely approximated, those given by R. For the Hannan Rissanen algorithm, occasionally the *WindApp2015* parameter estimates would coincide with R, but frequently they would not. This author is unsure as to whether this is an issue with the choice of the order k (see Step 1 in Section 4.2.3) inherent in the related code segments or whether there is some more fundamental underlying code issue. A source of comfort for this author is that the alternative approach to estimating model parameters for ARMA(p, q) models with both p and q > 0 (i.e. maximum likelihood) produced estimates which were consistent with R.

The dataset utilised throughout this *user manual* is the total wind energy levels (in MW) in the SEM market schedule for the years 2010, 2011, 2012, 2013 and 2014. It was seen that in order to apply ARMA models to such data the first step was the transformation (and or standardisation) of the data[18]. Subsequently it was seen that the transformed data is likely to require differencing prior to calibrating the various ARMA models. The residuals from the calibrated models exhibit some characteristics that question the validity of the underlying model assumptions. Time series models in which the conditional variance was allowed to vary with time were presented in Chapter 5 whilst Chapter 6 showed how these time series models could be used to make forecasts of future wind levels in the SEM market schedule.

On a standalone basis the *Statistics* class of the *WindApp2015* software application contains C# implementations of various statistical/mathematical algorithms which may be of interest to

[18] Alternative datasets may/may not, require transformation and/or standardisation.

users. Undoubtedly, improvements to the WindApp2015 application could be made[19] but an attractive feature of the software is that it combines various diagnostic and model fitting/forecasting routines in one simple, user friendly package.

Finally, a copy of the *WindApp2015* application will shortly be made available on *GitHub*. For a link to the relevant *GitHub* repository see the *WindApp2015* project section on the author's *LinkedIn* page (https://ie.linkedin.com/pub/joseph-collins/b6/252/abb). Alternatively, a copy of the application and the underlying source files can be provided on request by emailing joseph_anthony_collins@yahoo.co.uk.

[19] For example, when given a dataset, only calibrate the models to the most recent observations. Alternatively, split the dataset into *training*, *cross validation* and *test* sets before fitting the various models.

8. Bibliography

[1] Trading and Settlement Code - Helicopter Guide, Version 2.0, 7th October 2013.

[2] https://www.cer.ie/docs/000262/cer11075.pdf

[3] https://www.otexts.org/fpp/8/1

[4] Barbara G. Brown, Richard W. Katz and Allan H. Murphy, "Time Series Models to Simulate and Forecast Wind Speed and Wind Power", 1984, *Journal of Climate and Applied Meteorology*, Vol 23 Issue 8.

[5] Dubey, S.D., 1967b, "Normal and Weibull distributions", *Naval Research Logistics Quarterly*, 14, 69-79.

[6] Hinkley, D., 1977, "On quick choice of power transformation", *Applied Statisti*cs, 26, 67-69.

[7] J.L. Torres, A. Garcia, M. De Blas, A. De Francisco, 2005, "Forecast of hourly average wind speed with ARMA models in Navarre (Spain)", *Solar Energy*, Volume 79, Issue 1, 65-77.

[8] Mohammad A. Al-Fawzan, 2000, "Methods for Estimating the Parameters of the Weibull Distribution", May 2000. (http://interstat.statjournals.net/YEAR/2000/articles/0010001.pdf)

[9] http://www.weibull.com/hotwire/issue148/hottopics148.htm

[10] C.G. Justus, W.R. Hargraves, Amir Mikhail and Denis Graber, 1978, "Methods for Estimating Wind Speed Frequency Distributions", *Journal of Applied Meteorology*, Volume 17, Issue 13, 350-353.

[11]G.E.P. Box and M. E. Muller, 1958, "A note on the generation of random deviates", *Annals of Mathematical Statistics*, Volume 29, Number 2, 610-611.

[12] https://www.taygeta.com/random/weibull.html

[13] https://www.otexts.org/fpp/8/1

[14] George E.P.Box, Gwilym M.Jenkins, Gregory C.Reinsel. "*Time Series Analysis: Forecasting and Control*, 4th Edition, Wiley.

[15] Peter J. Brockwell, Richard A. Davis, 1991, "*Time Series: Theory and Methods, Second Edition"*. Springer Series in Statistics.

[16] James D.Hamilton, 1994, "*Time Series Analysis*", Prince University Press.

[17] http://media.wolfram.com/documents/TimeSeriesDocumentation.pdf

[18] R.Engle, 1982, "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of UK Inflation", *Econometrica*, 50, 987-1008.

[19] T.Bollerslev, 1986, "Generalized Autoregressive Conditional Heteroscedasticity", *Journal of Econometrics*, 31, 307-27.

[20] Heping Liu, Ergin Erdem, Jing Shi, 2011, "Comprehensive evaluation of ARMA-GARCH(-M) approaches for modelling the mean and volatility of wind speed", *Applied Energy*, 88, 724-732.

[21] J.Collins, May 2004, "Time Series Models and the ISEQ Index: ARCH in Context", *A thesis submitted to the National University of Ireland for the degree of M.Sc.*