UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

JOÃO HENRIQUE FERREIRA FLORES

ARMA-CIGMN - An Incremental Gaussian Mixture Network for time series analysis and forecasting

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Prof. Dr. Paulo Martins Engel
Advisor

Porto Alegre, 6, March 2015

CIP – CATALOGING-IN-PUBLICATION

Flores, João Henrique Ferreira

ARMA-CIGMN - An Incremental Gaussian Mixture Network for time series analysis and forecasting / João Henrique Ferreira Flores. – Porto Alegre: PPGC da UFRGS, 2015.

90 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2015. Advisor: Paulo Martins Engel.

1. Neural networks. 2. Time series. 3. Forecasting. 4. ARIMA. 5. IGMN. 6. ARMA-CIGMN. I. Engel, Paulo Martins. II. Título.

*"Everything must be made as simple as possible.
But not simpler."*
— EINSTEIN, ALBERT

ACKNOWLEDGEMENTS

First, I like to thanks my advisor, Dr. Paulo Martins Engel. Mainly because of his particular views in the work and his major contributions. My graduate formation as an statistician may have helped in the formal presentations, but the language use and the implications of that in the field of computer science was all his. And most of the questions and inspirations that made this work come together. And, for all that, I'm thankful. Also, the Instituto de Informática, UFRGS and the Instituto de Matemática, with regards for the Departamento de Estatística, for the opportunity they gave me and the believe they had on me.

My parents and relatives, specially, Ms. Dulcy Ferreira Flores and Mr. João Alcides Flores, my parents. They helped me finding time. Some say that time is priceless, and so are their contribution. I'm really thankful for that.

My colleagues and friends. All of them helped me in some or other way, but some of them made more important contributions and must be cited here. Edigleison, Renato and Rafael Pinto for their views and contributions over the algorithm and the model evaluation. The discussions we had made this work more solid, and I have to thank them. Rafael Santos, also an statistician, for helping me in the statistical department. He also have major contributions to this work. And all my other colleagues and friends that are not cited here, but also had helped.

Finally, my family. My wife, Patrice Basso and my daughter Helena. I got married and had a daughter during this work and that alone may explain why these two women are so important in my life now. While I'm thankful for all cited above, and this, the reason I'm able to made this work in the first place was her and now, it is for my daughter too. For them, I dedicate this work.

ABSTRACT

This work presents a new model of neural network for time series analysis and forecasting: the ARMA-CIGMN (Autoregressive Moving Average Classical Incremental Gaussian Mixture Network) model and its analysis. This model is based on modifications made to a reformulated IGMN, the Classical IGMN (CIGMN). The CIGMN is similar to the original IGMN, but based on a classical statistical approach. The modifications to the IGMN algorithm were made to better fit it to time series. The proposed ARMA-CIGMN model demonstrates good forecasts and the modeling procedure can also be aided by known statistical tools as the autocorrelation (acf) and partial autocorrelation functions (pacf), already used in classical statistical time series modeling and also with the original IGMN algorithm models. The ARMA-CIGMN model was evaluated using known series and simulated data. The models used for comparisons were the classical statistical ARIMA model and its variants, the original IGMN and two modifications over the original IGMN: (i) a modification similar to a classical ARMA (Autoregressive Moving Average) model and (ii) a similar NOE (Nonlinear Output Error) model. It is also presented a reformulated IGMN version with a classical statistical approach, which is needed for the ARMA-CIGMN model.

Keywords: Neural networks, time series, forecasting, ARIMA, IGMN, ARMA-CIGMN.

ARMA-CIGMN - Uma Rede Incremental de Mistura Gaussiana para análise e previsão de séries temporais

RESUMO

Este trabalho apresenta um novo modelo de redes neurais para análise e previsão de séries temporais: o modelo ARMA-CIGMN (do inglês, *Autoregressive Moving Average Classical Incremental Gaussian Mixture Network*) além dos resultados obtidos pelo mesmo. Este modelo se baseia em modificações realizadas em uma versão reformulada da IGMN. A IGMN Clássica, CIGMN, é similar à versão original da IGMN, porém baseada em uma abordagem estatística clássica, a qual também é apresentada neste trabalho. As modificações do algoritmo da IGMN foram feitas para melhor adpatação a séries temporais. O modelo ARMA-CIGMN demonstra boa capacidade preditiva e a modelagem ainda pode ser auxiliada por conhecidas ferramentas estatísticas como a função de autorrelação (acf, do original em inglês *autocorrelation function*) e a de autocorrelação parcial (pacf, do original em inglês *partial autocorrelation function*), já utilizadas em modelagem de séries temporais e nos modelos da IGMN original. As comparações foram feitas utilizando-se séries conhecidas e dados simulados. Foram selecionados para comparação os modelos estatísticos clássicos ARIMA (do inglês, *Autoregressive Integrated Moving Average*), a IGMN original e duas modificações feitas ainda na IGMN original:(i) um modelo similar ao modelo ARMA (do inglês, *Autoregressive Moving Average*) clássico e (ii) um modelo similar ao modelo NOE (do inglês, *Nonlinear Output Error*). Também é apresentada um versão reformulada da IGMN, usando a abordagem clássica da estatística, necessária para o desenvolvimento do modelo ARMA-CIGMN.

Palavras-chave: redes neurais, séries temporais, previsão, ARIMA, IGMN, ARMA-CIGMN.

LIST OF ABBREVIATIONS AND ACRONYMS

ACF	Autocorrelation Function
ADANN	Automatic Design of Artificial Neural Networks
ANN	Artificial Neural Networks
ARCH	Autoregressive Conditional Heteroskedasticity
ARIMA	Autoregressive Integrated Moving Average
ARMA-CIGMN	Autoregressive Moving Average Classical Incremental Gaussian Mixture Network
GARCH	Generalized Autoregressive Conditional Heteroskedasticity
IGMN	Incremental Gaussian Mixture Network
MA	Moving Average
MAE	Mean Absolute Error
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NARX	Nonlinear Autoregressive with Exogenous Variables
NMSET	Normalized Mean Squared Error with the Trivial Solution
NOE	Nonlinear Output Error
OLS	Ordinary Least Squares
PACF	Partial Autocorrelation Function
CIGMN	Classical Incremental Gaussian Mixture Network
PSO	Particle Swarm Optimization
RBF	Radial Base Function
RMSE	Root Mean Squared Error
SARIMA	Seasonal Autoregressive Integrated Moving Average
TDNN	Time Delay Neural Network

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

1.1 Initial comments

The process of forecasting is considered a fundamental tool for planning in any given area. However, the act of forecasting can be done in many different ways. Guessing tomorrow temperature, the possible traffic or even the time of arrival are common problems based on forecasts. In a more scientific approach, to make a forecast, we need data and a model (or models). Nowadays, data is found more easily, and more vastly. These data, when sequentially observing some results over a period of time are called time series. To analyze and to forecast based on these data, these time series, there is a large number of models (MORETTIN; TOLOI, 2006; HAMILTON, 1994). All models have some theoretical assumptions for working correctly. The problem, in most cases, is not to obtain the data, or to compute the parameters of a model, but which model to use. Particularly, in univariate time series.

With the advance in data storage, the easiness to obtain time series data, and the software tools available, the problem is to choose the "best" model. For example, if a series is considered nonlinear because of its variance (an heteroskedastic series), models of the ARCH/GARCH (autoregressive conditional heteroskedasticity and general autoregressive conditional heteroskedasticy model, respectively) family are recommended (ENDERS, 1995; HAMILTON, 1994), but it is not, by any case, the only solution. And, even if the ARCH/GARCH family is chosen, there are more than one hundred variations of the ARCH/GARCH models (MORETTIN; TOLOI, 2006; HAMILTON, 1994; ENDERS, 1995). Other statistical models have similar problems.

There are, basically, four different types of time series: (i) linear, (ii) nonlinear in respect to its mean, (iii) nonlinear in respect to its variance and (iv) nonlinear in respect to its both mean and variance (MORETTIN; TOLOI, 2006). There are different models to each of these characteristics, not to mention the classical models, the nonparametric models and the Bayesian approach. There are, also, different approaches on known models like in (HAMZAÇEBI; AKAY; KUTAY, 2009; DELLANA; WEST, 2009; COSTA; BRAGA; MENEZES, 2007; WANG; LU, 2006; CRONE; HIBON; NIKOLOPOULOS, 2011; DONATE; SANCHEZ; MIGUEL, 2012), just to name a few.

One way to solve the problem of model selection is to use a more general model. A model that, if not optimal, is robust, or suboptimal, in some sense. Models are considered robust if one or more of the assumptions are not met, but the parameters can still be well adjusted. We can mention the Artificial Neural Network (ANN) models (similar to an statistical nonparametric model), the seasonal autoregressive integrated moving average model (SARIMA, classical statistical approach), the Bayesian approach of some classical models among a few others. Of all these approaches, one of the most usual method is the nonparametric approach, which includes the ANN models.

The nonparametric approach has less assumptions than the others and even these assumptions are considered rather comprehensive, mostly sample based instead of a population based, like in the classical approach or somehow subjective, like in the Bayesian approach. Because of this, the nonparametric models can be adapted to nearly every situation. One issue related to nonparametric models, however, is that this adaptation does not follow a set of rules, a process to help the user. Also, in many cases, it falls as subjective as in the Bayesian approach or as a trial-and-error models (ZHANG, 2001; ZHANG; BERARDI, 2001; ZHANG, 2003; QI; ZHANG, 2001; ZHANG; KLINE, 2007; YAN, 2012). Most papers presenting ANN models adjusted to time series do not explain the used method, but only the final model. And even this final model is not a final model at all, but one among others that has, in average or not, better results. There is another problem with most models in the nonparametric approach: the randomness of the model parameters (HAYKIN, 2001). In the Multilayer Perceptron (MLP) for example, based on the used training method, for each time that a model is generated, it is a different model. Not all nonparametric methods are heavily dependent on the initial parameters, but some of them are. Also, nonparametric models are more likely to present overfitting.

The model overfitting has been a problem almost exclusively of the nonparametric approach. Basically, an ANN model, more generally, can be adjusted to any given function (HAYKIN, 2001). The problem is to forecast using this model. For example, a Multilayer Perceptron (MLP) model can be easily fitted to any time series model so well, that the error between the model generated data and the real data is negligible. However, if we want to forecast, based on this model, the results are, most likely, awfully poor. The problem of overfitting is more practical than theoretical. We have a perfect model fitted to the training data and poorly forecast data. This problem is usually avoided by some programming tools, as the early stop function available in some softwares, as seen in Figure 1.1.

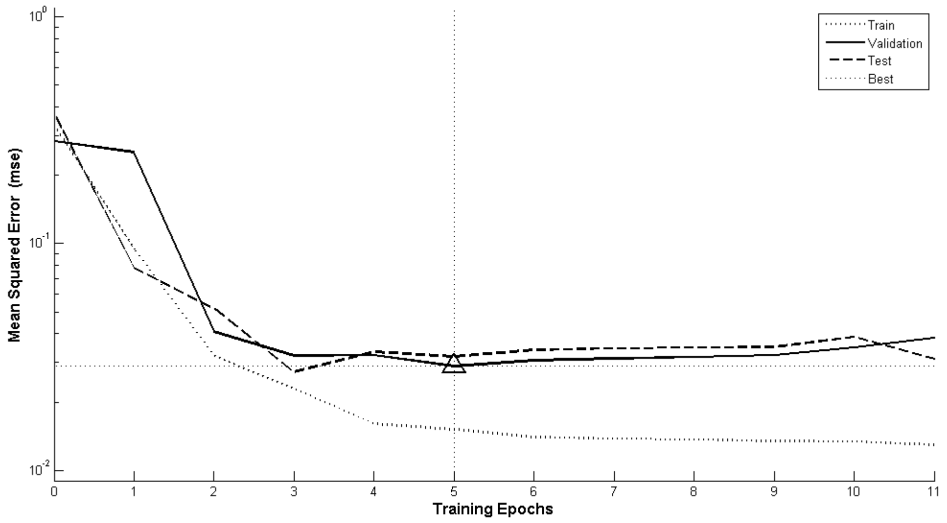


Figure 1.1: A graphical example of the early stopping procedure on MATLAB Neural Networks Toolbox.

Observing Figure 1.1, it is possible to see that the training error keeps getting smaller after each epoch and that the validation error get smaller but after epoch 5 begins to grow. So, for this example, after epoch 5, the model is possibly overfitted. This problem can also be avoided by some heuristics and the separation of data in groups for training, adjusting and verification of the model. The randomness of the models and the overfitting are problems that the classical models do not present (MORETTIN; TOLOI, 2006).

The classical models, mainly the SARIMA models have their own set of assumptions and are considered robust enough in respect to some of these assumptions (MORETTIN; TOLOI, 2006; ENDERS, 1995). The classical models, commonly, do not present the same randomness that the nonparametric models and are less prone to overfitting. For example, the SARIMA models have an algorithm, a series of analysis to help the final user to obtain a parsimonious model. This algorithm is based on Box and Jenkins works (BOX; JENKINS, 1976; BOX; JENKINS; REINSEL, 1994), this is why these models are also known as Box-Jenkins models (ENDERS, 1995). It is based on graphical analysis and statistical tests to help the user. However, these graphical tools and the statistical tests not only help the user, they also prevent the model overfitting (HAMILTON, 1994; ENDERS, 1995; MORETTIN; TOLOI, 2006).

The SARIMA models are based on only 6 different parameters for modeling, and these parameters can be computed via two graphical tools. These graphical tools are the autocorrelation function (acf) and the partial autocorrelation function (pacf) that help the user to set the parameters and to verify the model. Other classical models share these same properties to some extent. The other desirable characteristic of the classical models, also based on their assumptions, is that they follow a probabilistic approach. Based on this, confidence intervals can be computed, among other statistical tests.

The problem is that, even for the robust models, some assumptions can be considered too restrictive in some cases. The SARIMA models assume that the time series is linear and stationary. A series can be nonlinear in three different ways, as shown before. If a series is nonlinear in respect to its mean, this can be solved by the integration (or seasonal integration) parameter, but only in some cases. And, because of this, there are hundreds of different classical models, nearly one for each specific situation.

Summing up, on the one hand we have nonparametric models with lesser assumptions, but too subjective to model. On the other hand, we have the classical models, with more assumptions, less subjective, but with so many different models to choose. A good solution would be a mixture of the nonparametric and classical models characteristics. The Incremental Gaussian Mixture Network (IGMN) is supposed to be this model. In different works (HEINEN, 2011; PINTO, 2011; FLORES; PINTO; ENGEL, 2012), the IGMN has been used to model time series. Initially, it was used with the same approach as some of the ANN models, mainly the MLP, the Radial Base Function (RBF) and the Time-Delay Neural Network (TDNN) showing good results (HEINEN, 2011). Later, with some variations (PINTO, 2011), and finally with an approach similar to the classical models (FLORES; PINTO; ENGEL, 2012). The IGMN has shown that it can model different time series, linear and nonlinear, and, to some extent, can mimic the process used to model, based on the acf and pacf, as the classical models. In addition the IGMN does not suffer from the randomness of most ANN models and, because of its probability definitions, also shares most of the properties found in classical models, as confidence intervals among others. However, even the IGMN fails to model the moving average (MA) component of a time series and also, the IGMN models are still adjusted based on the same principles as others ANN models, mainly by trial and error and with some subjective knowledge about the data.

1.2 Main objectives and contributions

The IGMN has already been presented in other works, mainly in HEINEN (2011). However, the process of generating a model has not yet been well presented for the IGMN. An important issue is that IGMN has not been developed exclusively for use with time series, even that its capabilities for this task have been demonstrated on (PINTO, 2011). Another problem presented in (FLORES; PINTO; ENGEL, 2012) is about the difficulty that the IGMN has to model a moving average (MA) process. Based on these facts and some preliminary results, the main goal of this work is to present a new model, a modification on the IGMN algorithm, that we call the ARMA-CIGMN, which stands for Autoregressive Moving Average Classical Incremental Gaussian Mixture Network. The ARMA-CIGMN takes into account the MA component of a time series using a classical statistical approach of the IGMN algorithm. We present and test this model on some known series and simulated data against classical models and other versions of the IGMN.

A secondary objective is to improve the online incremental one-shot learning using the proposed ARMA-CIGMN model and the autocorrelation function (acf) and partial autocorrelation function (pacf). These tools and methods are similar to what is used in SARIMA modeling, i.e., to help the user to achieve a parsimonious model. This goal can be seen on some results found in (FLORES; PINTO; ENGEL, 2012) using the original IGMN models and, in this work, the same process was applied to the proposed ARMA-CIGMN model.

Another contribution of his work is a theoretical analysis of the IGMN algorithm showing that it follows a Bayesian approach. However, in a Bayesian approach, the $\boldsymbol{\theta}$ parameters of the MA components can not be estimated online, depending on sampling techniques, like the Gibbs sampling (CONGDON, 2003). On the other hand, the estimate of the $\boldsymbol{\theta}$ parameters used in this work and demonstrated in (ENDERS, 1995; HAMILTON, 1994) is based on an online procedure and it is only possible under the classical statistical approach. Thereafter, an important contribution of this work is the reformulation of the IGMN algorithm using a classical approach, resulting in the Classical IGMN, or simply CIGMN.

It is also a goal of this work to present the IGMN, and CIGMN, as an all around model. It is important to note that it is not an objective of this work to present IGMN as a model with a minimal prediction error or to compare with other models using some prediction, or forecasting, error comparison metric. These comparisons are made to show that the IGMN model is good as any specific model, but without the problem of choosing one specific model for each specific task, mainly in the univariate time series approach. Some results presented in this work reinforce this goal.

1.3 Outline

This text is organized as follows. Chapter 2 presents the classical approach of time series, as a background to future references and to establish the algorithm that later will be adapted to the IGMN on time series modeling. This chapter also presents the most common classical model, the SARIMA model class, the regression models on time series and a background on Artificial Neural Networks (ANN) models, in general.

Chapter 3 shows the IGMN, the classical statistical approach for the IGMN and the proposed ARMA-CIGMN model. It presents its formulation, the parameters used to help the computational work and some important analysis. In this chapter it is also shown that the IGMN can be considered as a combination of multiple linear regression models, which allows IGMN to be considered a nonlinear model, and the earlier modifications on the original IGMN.

Chapter 4 presents the experiments and results. These experiments, and the results that follow, show some important characteristics of the different models applied.

Finally, Chapter 5 concludes this work summarizing the results and contributions and presenting future works, observations and recommendations.

2 TIME SERIES MODELING

2.1 Initial Comments

The most common definition of a time series is, basically, a series of data observed and registered over a definite period of time. It is understandable that these data points are not independent to each other. This probable correlation among data adds some dependence, which is used to develop the different models (MORETTIN; TOLOI, 2006). The most common statistical tools to analyze data make the assumption of independent data and so these correlations affect these methods. Thereafter, there is a need to elaborate models that can deal with the assumption of correlated data. These correlations can be due to a random factor, for example, a Markov Chain similar effect (HAMILTON, 1994; ENDERS, 1995). For example, in the classic Air Passengers data (BOX; JENKINS, 1976), the passengers over months can be dependent of past months passengers or because of one Markov Chain path. Suppose that the number of passengers on June be dependent on January's passengers, because these passengers told their families about the convenience of a flight and these families took vacations, raised money and trade the bus to an airplane. But this can also be because of some Markov Chain that presents the probability of people who flew in January to fly again in June. The problem with the Markov Chain is the growth tendency of a time series. This problem and its consequences will be addressed in the coming section. Nevertheless, both these definitions, the temporal correlation and the random process, add up to the classical statistical models.

It is important to note that, some assumptions are not exclusive to the classical statistical models, or even to other models. One of the assumptions that persists in almost every time series models is the notion of a data generating process, most similar to a differential equation with stochastic components (ENDERS, 1995). The other assumptions are more a delimitation of this work: the discrete time representation and the univariate data. To demonstrate it, let us assume that y is a quantitative variable, discrete or continuous, where $y_k, y_{k+h}, y_{k+2h}, \ldots, y_t$ is a sample of a time series. The time correlations are used so that a value of, let us say, y_{k+7h} can be explained only by the past values, i.e., $y_k, y_{k+h}, \ldots, y_{k+6h}$ or some combination of these values. The h intervals, as the constant k are discrete. This is a sample of a time series of size T, that represents the population. The models presented here are all based on these assumptions: an unknown data generating process, but that can be estimated, the discrete time representation and the univariate component.

The models presented here also share other characteristics. The time series analysis models are commonly used to: (i) forecasts, (ii) component identification and (iii) parameters analysis. However, it is usual to apply a model only for forecasting. The models to be presented in this work are focused on their forecast capabilities and the process of estimation of their parameters. Forecasting is the most common use of time series models (MORETTIN; TOLOI, 2006; SHUMWAY; STOFFER, 2000) and is often applied, to model selection (FLORES, 2006; MORETTIN; TOLOI, 2006; INOUE; KILIAN, 2006; QI; ZHANG, 2001; GRANGER; KING; WHITE, 1995; SIN; WHITE, 1996). A model can be very well adjusted to a data series but with poor forecasts. It is a common rule to divide the data series in two sequential groups: (i) modeling and (ii) forecasting. There is no global optimal rule to divide the data series. This separation depends on the chosen model and even on the model parameters and restrictions. But it is usual that the modeling group must be bigger than the forecasting group. The modeling group can be defined as all the data series except the last k values, where k is the size of the forecasting group. For example, a series of size 100, could have the modeling group defined as the first 90 values and the last 10 as the forecasting group. This evaluation of the forecasts is necessary to prevent the problem of overfitting (FLORES, 2006; SHUMWAY; STOFFER, 2000). A model can be considered overfitted when the prediction errors are much smaller than the forecasting errors. This could indicate that the model is well fitted to the modeling group, but has difficulties when forecasting (MORETTIN; TOLOI, 2006; ENDERS, 1995).

In the sequence, different approaches to model a time series will be presented. In this work, the time series models presented are: (i) the SARIMA classical statistical model family and (ii) the ANN models. The classical statistical models are presented because of its formulation and the ANN models as a background for the later presentation of the IGMN, CIGMN and the ARMA-CIGMN model.

2.2 Classical Statistical Models

The classical statistical models make assumptions of the classical statistics or the parametric statistics. Classical because of the statistical inference foundation, to assume that a parameter of a population is constant and unknown and that the sample contains all the information needed to estimate this parameter (ROHATGI, 2003; MORETTIN; TOLOI, 2006). The models presented here make the assumption of a stochastic component, a random error.

In these models, the assumption is that the stochastic component behaves according to a Gaussian Probabilistic Distribution (MORETTIN; TOLOI, 2006; ENDERS, 1995). In specific cases, the Student's t Distribution is used. Using the description made earlier, T being the time series population, an stochastic process is a family $X = x(t), t \in T$, such that, for each $t \in T$, $x(t)$ is a random variable, also denoted by x_t. So, t is a sample of T. For every t, x_t is a random variable defined over Ω, the probability space. This definition allows that, for any t, x_t follows a different probabilistic function, discrete or continuous. However, in this work, we consider x_t equally distributed for all t, unless otherwise noted.

There is not just one classical statistical model, but many models. In this work, only the most usual models will be presented. One of the critics made about the classical models approach is the large number of models, some of them for very specific situations.

2.2.1 The Box-Jenkins models (SARIMA)

Probably, the most common classical model is based on the Box-Jenkins algorithm (BOX; JENKINS, 1976; BOX; JENKINS; REINSEL, 1994). The Box-Jenkins approach for estimating a time series model is in the form of:

$$x_t = \mu + \phi_1 x_{t-1} + \ldots + \phi_p x_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q} \tag{2.1}$$

The model represented by (2.1) is called an autoregressive moving average model or simply an ARMA(p,q) model, where p and q represent the autoregressive and moving average orders, respectively. The parameter vector $\boldsymbol{\phi}$ represents the autoregressive components (AR(p)), the parameter vector $\boldsymbol{\theta}$ represents the moving average components (MA(q)) and μ is the intercept or mean. It is considered that the noise ϵ_t is the stochastic component of the model. The $\epsilon_{t-1}, \epsilon_{t-1}, \ldots, \epsilon_{t-q}$ are past values, obtained by (2.2).

$$\epsilon_{t-q} = x_{t-q} - \hat{x}_{t-q} \tag{2.2}$$

where x_{t-q} is the real observed value and $\hat{x}_{t-q}$ is the value estimated by the model in time $t - q$.

It is important to notice that ϵ_t represents, on this approach, a white noise process. The white noise process can be defined by (2.3), (2.4) and (2.5).

$$E[\epsilon_t] = E[\epsilon_{t-1}] = \ldots = 0 \tag{2.3}$$

$$Var(\epsilon_t) = Var[\epsilon_{t-1}] = \ldots = \sigma^2 \tag{2.4}$$

$$Cov(\epsilon_t, \epsilon_{t-s}) = Cov(\epsilon_{t-j}, \epsilon_{t-j-s}) = \cdots = 0, \forall\, j \text{ and } s \tag{2.5}$$

An ARMA(p,q) model assumes that the time series is stationary. For this, let us assume an ARMA(2,1) model, according to (2.6).

$$x_t = \phi_0 + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \epsilon_t + \theta_1 \epsilon_{t-1} \tag{2.6}$$

Since the magnitude of the intercept μ does not affect the stationary conditions, consider $\mu = 0$. To find a particular solution, using the method of undetermined coefficients, the solution is (ENDERS, 1995):

$$x_t = \sum_{i=0}^{\infty} \alpha_i \epsilon_{t-i} \tag{2.7}$$

For (2.7) to be a solution of (2.6), the α_i must satisfy:

$$\begin{aligned} &\alpha_0 \epsilon_t + \alpha_1 \epsilon_{t-1} + \alpha_2 \epsilon_{t-2} + \ldots = \phi_1(\alpha_0 \epsilon_{t-1} + \alpha_1 \epsilon_{t-2} + \alpha_2 \epsilon_{t-3} + \ldots) + \\ &\phi_2(\alpha_0 \epsilon_{t-2} + \alpha_1 \epsilon_{t-3} + \alpha_2 \epsilon_{t-4} + \ldots) + \epsilon_t + \theta_1 \epsilon_{t-1} \end{aligned} \tag{2.8}$$

To match the coefficients on the terms containing $\epsilon_t, \epsilon_{t-1}, \epsilon_{t-2}, \ldots$, it is necessary to set:

$$\begin{aligned} &1.\ \alpha_0 = 1 \\ &2.\ \alpha_1 = \phi_1 \alpha_0 + \theta_1 \quad \Rightarrow \alpha_1 = \phi_1 + \theta_1 \\ &3.\ \alpha_i = \phi_1 \alpha_{i-1} + \phi_2 \alpha_{i-2} \quad \text{for all } i \geq 2 \end{aligned} \tag{2.9}$$

If the characteristic roots of (2.6) are within the unit circle, the α_i must constitute a convergent sequence (ENDERS, 1995). To verify that the x_t generated by (2.7) is stationary, take the expectation to form $E[x_t] = E[x_{t-i}] = 0$, for all t and i. So, the mean is finite and independent of time. The ϵ_t was defined as a white noise process, so the variance of x_t is constant and also independent of time, so follows:

$$Var(x_t) = E[(\alpha_0\epsilon_t + \alpha_1\epsilon_{t-1} + \alpha_2\epsilon_{t-2} + \ldots)^2] = \sigma^2 \sum_{i=0}^{\infty} \alpha_i^2 \tag{2.10}$$

So, $Var(x_t) = Var(x_{t-s})$, for all t and s. Finally, the covariance between x_t and x_{t-s} is:

$$Cov(x_t, x_{t-s}) = \sigma^2(\alpha_s + \alpha_{s+1}\alpha_1 + \alpha_{s+2}\alpha_2 + \ldots)^2 \tag{2.11}$$

Then, the $Cov(x_t, x_{t-s})$ is also constant and independent of t, for all t and s. If the characteristic roots of (2.1) do not lie within the unit circle, the sequence of α_i will not converge and so, the x_t sequence (ENDERS, 1995; HAMILTON, 1994).

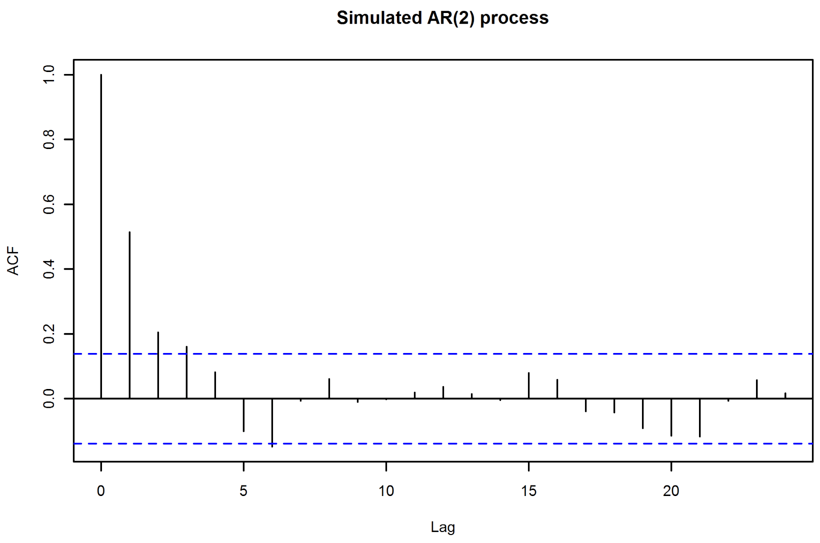
Based of this preliminary analysis, two graphical and statistical tools are used for modeling classical models (ENDERS, 1995; MORETTIN; TOLOI, 2006; HAMILTON, 1994), (i) the autocorrelation function (acf) and (ii) the partial autocorrelation function (pacf). These functions show the intensity of the temporal autocorrelation, each on its own manner. A model is considered adjusted when both, acf and pacf, show no significant autocorrelation, i.e., when ϵ_t behaves like a white noise process using a user defined lag L.

The acf is obtained from the linear correlation of each x_t value of the series to the others in different lags, as x_{t-1}, x_{t-2} and so on, until the lag L defined. The pacf, computes a similar function, but removing the interference of other values. After the modelling, these graphs are computed using the residuals, the past prediction errors ϵ_i, $i = t$,$t-1$,. . .,$t-L$. Unless noted otherwise, all acf and pacf graphs presented in this work are made using the residuals. In the acf, for example, the correlation between x_t and x_{t-2} suffers the interference of x_{t-1}. The pacf removes that interference. Each of these functions not only has its own interpretation, but also a combined interpretation. For example, observe Figure (2.1).

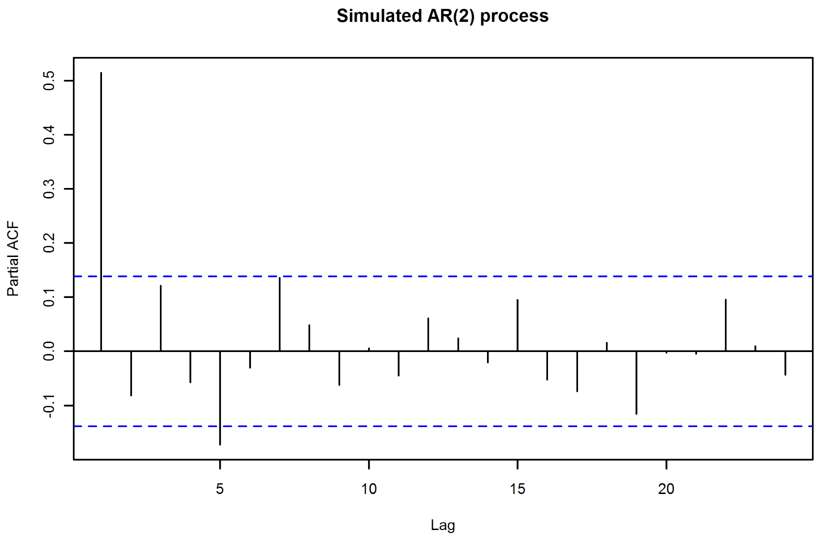
In the graphs of Figure (2.1) one can note two different correlations beyond the statistical significance limits, the dashed lines. By definition, the acf graph shows every lag correlation, even on lag zero, which is not taken into account. In Figure (2.1) it is also possible to observe significant correlations in the pacf graph, even with a second order autoregressive model. This is due to common sample variations and the possible interaction of the autoregressive (AR) term in the moving average (MA) term. Not only, but also because of this, it is recommended to use both functions.

By definition, autoregressive components can be observed when the pacf is statistically different from zero at the lags $m = 1, 2, \ldots, p$ and zero thereafter. In this cases an AR(p) is used, according to (2.12), where ϕ are the autoregressive parameters, x_t is the observation on time t and ϵ_t is the white noise on time t.

$$x_t = \phi_1 x_{(t-1)} + \phi_2 x_{(t-2)} + \ldots + \phi_p x_{(t-p)} + \epsilon_t \tag{2.12}$$



(a)



(b)

Figure 2.1: Example of a acf(a) and pacf(b) of a second order autoregressive(AR) model - AR(2)

The seasonal autoregressive component has a statistically equal zero pacf, except in the lags $m = s, 2s, \ldots, Ps$, in which case an AR$_s$(P) model is obtained, according to (2.13), where s represents that seasonal effect.

$$x_t = \phi_s x_{(t-s)} + \phi_{2s} x_{(t-2s)} + \ldots + \phi_{Ps} x_{(t-Ps)} + \epsilon_t \tag{2.13}$$

The autoregressive behavior refers to the correlation that exists between the current and the past values. However, some precautions are necessary. The limits of acceptable correlation (represented by the dashed line in Figure (2.1)) are constructed assuming the data, or residuals, follow a Gaussian probability distribution. If this assumption can not be verified, the amplitude of the limits becomes different and the limits must be calculated on a nonparametric approach. In our experiments, the applied series are also previously used in other works using the classical statistical approach, so it is not necessary to take these verifications into account.

The other model component to be detected is the moving average (MA) component. This component can be identified using, mainly, the acf. When the acf is statistically different from zero on lags $m = 1, 2, \ldots, q$ and equal zero thereafter, the model is a MA(q), according (2.14), where θ are the moving average parameters and ϵ_t the residuals on time t.

$$x_t = \theta_1 \epsilon_{(t-1)} + \theta_2 \epsilon_{(t-2)} + \ldots + \theta_q \epsilon_{(t-q)} + \epsilon_t \tag{2.14}$$

It is important to note, however, that the ϵ_t is a white noise, but the $\epsilon_{(t-1)}, \epsilon_{(t-2)}, \ldots, \epsilon_{(t-q)}$ are not, as they are already observed in previous periods, as seen on (2.2).

The seasonal moving average component has an acf statistically equal to zero, except on lags $m = s, 2s, \ldots, Qs$, in which case a MA$_s$(Q) model is suggested, according to Equation (2.15).

$$x_t = \theta_s \epsilon_{(t-s)} + \theta_{2s} \epsilon_{(t-2s)} + \ldots + \theta_{Qs} \epsilon_{(t-Qs)} + \epsilon_t \tag{2.15}$$

The Seasonal Autoregressive Integrated Moving Average(SARIMA) model combines the effects of an AR(p), an AR$_s$(P), a MA(q), a MA$_s$(Q) component and a differentiation (or integrated) component. This model is presented as a SARIMA(p,i,q)$\times$(P,I,Q)$_s$, where the i component stands for integrated component, I for the seasonal integrated component and s the seasonal time period, as seen before. A SARMA model, according to (2.16), is similar to a SARIMA model, but without the integrated component. The integrated, or seasonal integrated, components are used in classical statistical approach when the time series is not stationary, namely a time series with a non constant mean, as seen on (2.16).

$$\begin{aligned} x_t = {} & \phi_1 x_{(t-1)} + \ldots + \phi_p x_{(t-p)} + \phi_s x_{(t-s)} + \ldots + \phi_{Ps} x_{(t-Ps)} + \\ & \theta_1 \epsilon_{(t-1)} + \ldots + \theta_q \epsilon_{(t-q)} + \theta_s \epsilon_{(t-s)} + \ldots + \theta_{Qs} \epsilon_{(t-Qs)} + \epsilon_t \end{aligned} \tag{2.16}$$

However, in practice, the mean $\bar{x}$, variance s^2 and the correlation r_s are obtained from a sample and the real, theoretical values are unknown. For this, let us assume, as before, a sample of size t, where:

$$\bar{x} = \frac{1}{t}\sum_{i=1}^{t} x_i \tag{2.17}$$

$$s^2 = \frac{1}{t-1}\sum_{i=1}^{t} (x_t - \bar{x})^2 \tag{2.18}$$

$$r_s = \frac{\sum_{i=s+1}^{t} (x_t - \bar{x})(x_{t-s} - \bar{x})}{\sum_{i=1}^{t} (x_t - \bar{x})^2} \text{, for each value of } s = 1, 2, 3, \ldots \tag{2.19}$$

With these definitions, and based on the model assumptions, we have two similar statistical tests to verify if the computed autocorrelations and partial autocorrelations are statistically significant or a result of a random effect, using a definite lag-window, s. The first test is based on the Box-Pierce Q-statistic (BOX; JENKINS, 1976).

$$Q = t\sum_{i=1}^{s} r_i^2, \text{ where } Q \sim \chi_s^2 \tag{2.20}$$

where χ_s^2 represents a Chi-square distribution with s degrees of freedom.

However, the calculation of the Box-Pierce Q-statistic in (2.20), works poorly, even in moderately large samples (ENDERS, 1995). Ljung and Box (MORETTIN; TOLOI, 2006) use another form to calculate the Q-statistic, that has reportedly, superior performance on small samples, as shown in (2.21).

$$Q = t(t+2)\sum_{i=1}^{s} \frac{r_i^2}{t-i}, \text{ where } Q \sim \chi_{(s-p-q)}^2 \tag{2.21}$$

The calculation of the Q-statistic presented in (2.21) also follows a Chi-square distribution (χ^2), but with different degrees of freedom. In (2.21) the degrees of freedom are based not only on the lag-window s, but on the parameters of an ARMA model, p and q.

Paremeter estimation

Using the acf and pacf to identify the expected model, the next step is to estimate the model parameters. This can be done with different methods, but only the maximum likelihood will be presented. The method will be presented for an ARMA(p,q) model. Let us assume a vector $\mathbf{\Theta} \equiv (\boldsymbol{\phi}, \boldsymbol{\theta}, \sigma^2)$, $\boldsymbol{\phi} = (c, \phi_1, \phi_2, \ldots, \phi_p)$ where c is any finite constant, and $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_q)$. Suppose we have observed a sample of size $t(x_1, x_2, \ldots, x_t)$. The approach will be to calculate the probability density

$$f_{x_t, x_{t-1}, \ldots, x_1}(x_t, x_{t-1}, \ldots, x_1; \mathbf{\Theta}) = (2\pi)^{-\frac{t}{2}}\sigma^{-t} exp\left\{-\frac{\sum_{i=1}^{t} x_i^2}{2\sigma^2}\right\} \tag{2.22}$$

which might loosely be viewed as the probability of observing this particular sample. The maximum likelihood estimate (MLE) for $\mathbf{\Theta}$ is the value for which this sample is most likely to have been observed; that is, the value of $\mathbf{\Theta}$ that maximizes (2.22). Finding the MLE involves two different steps: (i) calculate the likelihood function (2.22) and (ii) compute the value (or values) of $\mathbf{\Theta}$ that maximizes this function.

The ARMA(p,q) model takes the form of

$$\begin{aligned} x_t = c + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} \\ + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} \ldots + \theta_q \epsilon_{t-q} + \epsilon_t \end{aligned} \tag{2.23}$$

Replacing x_t from Equation (2.23) into Equation (2.22), then we have the MLE, as shown in Equation (2.24):

$$\begin{aligned} L(\mathbf{\Theta}|\mathbf{x}_0, \mathbf{e}_0) = (2\pi)^{-\frac{t}{2}} \sigma^{-t} exp \left\{ -\frac{1}{2\sigma^2} \left(\sum_{i=1}^{t} \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots \right.\right. \\ \left.\left. + \phi_p x_{t-p} + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \ldots + \theta_q e_{t-q} \right)^2 \right\} \end{aligned} \tag{2.24}$$

Which is the likelihood function for an ARMA(p,q) model conditioned on both x's and ϵ's. Taking initial values for $\mathbf{x}_0 \equiv (x_0, x_{-1}, x_{-2}, \ldots, x_{-p+1})$ and $\boldsymbol{\epsilon}_0 \equiv (\epsilon_0, \epsilon_{-1}, \epsilon_{-2}, \ldots, \epsilon_{-q+1})$ as given, the sequence $\{\epsilon_1, \epsilon_2, \ldots, \epsilon_t\}$ can be calculated from $\{x_1, x_2, \ldots, x_t\}$ by iterating on:

$$\begin{aligned} \epsilon_t = x_t - c - \phi_1 x_{t-1} - \phi_2 x_{t-2} - \ldots - \phi_p x_{t-p} \\ - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} \ldots - \theta_q \epsilon_{t-q}, \end{aligned} \tag{2.25}$$

The conditional log likelihood is then:

$$\begin{aligned} l = log(L(\mathbf{\Theta})) = log f_{x_t, x_{t-1}, \ldots, x_1 | \mathbf{x}_0, \boldsymbol{\epsilon}_0}(x_t, x_{t-1}, \ldots, x_1 | \mathbf{x}_0, \boldsymbol{\epsilon}_0; \mathbf{\Theta}) = \\ -\frac{t}{2} log(2\pi) - \frac{t}{2} log(\sigma^2) - \sum_{i=1}^{t} \frac{\epsilon_t^2}{2\sigma^2} \end{aligned} \tag{2.26}$$

One common approach is to set initial x's and ϵ's equal to their expected values. That is, set $x_s = c/(1 - \phi_1 - \phi_2 - \ldots - \phi_p)$ for $s = 0, -1, -2, \ldots, -p+1$ and set $\epsilon_s = 0$ for $s = 0, -1, -2, \ldots, -q+1$), and then proceed with the iteration in (2.26) for all sample. Alternatively, (BOX; JENKINS, 1976) recommended setting ϵ's to zero but y's equal to their actual values. Thus, iteration using (2.25) is started at time step $t = p + 1$ with $x_1, x_2, \ldots, x_p$ set to the observed values and:

$$\epsilon_p = \epsilon_{p-1} = \epsilon_{p-2} = \ldots = \epsilon_{p-q+1} = 0 \tag{2.27}$$

Then the conditional likelihood calculated as in (2.28).

$$\begin{aligned} l = log(L(\Theta)) = \\ log f(x_t, x_{t-1}, \ldots, x_{p+1} | x_p, x_{p-1}, \ldots, x_1, \epsilon_p = 0, \ldots, \epsilon_{p-q+1} = 0) = \\ -\frac{t-p}{2} log(2\pi) - \frac{t-p}{2} log(\sigma^2) - \sum_{i=p+1}^{t} \frac{\epsilon_t^2}{2\sigma^2} \end{aligned} \tag{2.28}$$

As in the case of the moving average models, these approximations should be used only if all values of z satisfy (2.29).

$$1 + \theta_1 z + \theta_2 z^2 + \ldots + \theta_q z^q = 0 \tag{2.29}$$

The same process can be applied, with few adaptations, for the other models. Using the Kalman filter (HAMILTON, 1994) is the simplest approach to compute the exact likelihood function. Numerical optimization can also be used. Methods as grid search, steepest ascent, Newton-Raphson, Davidon-Fletcher-Powel (HAMILTON, 1994) among others can be used.

2.2.2 Linear regression models on time series

A simple linear regression model can be described as in Equation (2.30).

$$y_i = \beta_0 + x_i \beta_1 + \epsilon_i \tag{2.30}$$

where y_i represents a dependent variable (or vector), x_i an independent variable (or observable), β_0 is an estimate parameter of the model constant, β_1 is an estimate parameter or the linear coefficient, ϵ_i is an independent identically distributed (i.i.d.) probabilistic error, commonly a Gaussian distributed error and i is a sample index, where $i = 1, 2, 3, \ldots, n$.

The main objective of the regression model is to express the dependence among y_i results and x_i observations, or simply, the conditional expectancy of Y on X, i.e., $E[Y|X]$. The model presented on Equation (2.30) is the theoretical, or population based, model. With the β_i estimates, denoted as $\hat{\beta}_i$, the model becomes as defined on Equation (2.31).

$$y_i = \hat{\beta}_0 + x_i \hat{\beta}_1 + e_i \tag{2.31}$$

where $\hat{\beta}$ is the β estimate and e_i the residual $e_i = y_i - \hat{y}_i$.

The $\hat{\beta}_i$ of these models are obtained via a well known method, the ordinary least squares (OLS). The OLS estimate of β_i is the value of β_i that minimizes the residuals sum of squares (RSS), as presented on Equation (2.32).

$$RSS = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - x_i \hat{\beta}_1)^2 \tag{2.32}$$

The substitution made on Equation (2.32), is based on the prediction equation, as in Equation (2.33).

$$\hat{y}_i = \hat{\beta}_0 + x_i \hat{\beta}_1 \tag{2.33}$$

where $\hat{y}_i$ is the prediction of y_i based on the observable x_i, a constant $\hat{\beta}_0$ and the linear coefficient $\hat{\beta}_1$.

In general, a multivariate model, with m dependent variables y, k independent variables x and a sample size of n observations can be represented as in Equation (2.34).

$$\begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,m} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n,1} & y_{n,2} & \cdots & y_{n,m} \end{bmatrix} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,k} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & x_{n,2} & \cdots & x_{n,k} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{k-1} \end{bmatrix} \tag{2.34}$$

Then, the OLS estimate for the $(k-1)$ β parameters, is shown on Equation (2.35).

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{2.35}$$

The OLS is a best linear unbiased estimator, or simply BLUE (HAMILTON, 1994; NETER et al., 1996; GUJARATI, 2006), based on some assumptions: (i) errors have a zero expectation, (ii) errors are uncorrelated and (iii) errors are homoscedastic, as shown in Equations (2.36) through (2.38).

$$E[\epsilon_i] = 0; \tag{2.36}$$
$$Var(\epsilon_i) = \sigma^2, \forall i, \sigma^2 < \infty; \tag{2.37}$$
$$Cov(\epsilon_i, \epsilon_j) = 0, \forall i \neq j \tag{2.38}$$

As expected, it is possible to make statistical tests to verify these assumptions over the sample residuals e_i. It is important to notice that no assumptions are made on the probabilistic distribution for the errors ϵ_i. These assumptions are needed for, basically, hypothesis tests and confidence intervals for the parameters.

Using a regression model as defined above for time series, requires just simple manipulations on the formulation, but more complex ones on the assumptions. Let us assume an AR(p) component. We may present the regression model as in Equation (2.39), which is very similar to an AR model.

$$x_t = \beta_0 + \beta_1 x_{t-1} + \beta_2 x_{t-2} + \ldots + \beta_p x_{t-p} + \epsilon_i \tag{2.39}$$

However, by using the OLS to estimate the parameters of the model presented in Equation (2.39), some issues arise. In the OLS estimate presented on Equation (2.35) we need that $(\mathbf{X}^T\mathbf{X})$ exists and be invertible. This can only occur when the values in $\mathbf{X}$ are not linearly dependent, or statistically correlated. In a model constructed using past values, x_{t-1}, x_{t-2}, x_{t-3} and so on it is very likely that some correlation appears. This correlation is worst in small samples. Without this independence, the OLS parameters $\hat{\boldsymbol{\beta}}$ return a biased estimate of $\boldsymbol{\beta}$ for any autoregression, and the follow up statistical tests (t test and F test) can only be justified asymptotically (HAMILTON, 1994; NETER et al., 1996).

The problem with the MA parameters in regression is more complex, as the $\boldsymbol{\theta}$ can not be obtained via an OLS estimate because it depends on another OLS estimate. Let us assume an MA(q) model as in Equation (2.40) [1].

$$x_t = \theta_1 \epsilon_{(t-1)} + \theta_2 \epsilon_{(t-2)} + \ldots + \theta_q \epsilon_{(t-q)} + \epsilon_t \tag{2.40}$$

[1] The similar equation can be seen on Equation (2.14) and it is presented here only for readability

where $\epsilon_{(t-i)} = x_{(t-i)} - \hat{x}_{(t-i)}$.

To estimate the model $\boldsymbol{\theta}$, we need the past ϵ, where the past ϵ can only be obtained with the real observed values x_t and model estimates $\hat{x}_t$ and, finally, the estimated $\hat{x}_t$ can only be obtained with the estimated $\boldsymbol{\theta}$, resulting in a recurrent dependence. Summing up, the regression models can be used on time series, but work better on large samples, using the OLS estimate, and only to AR(p) components, as the MA(q) components suffer from recurrence. In the next chapter, this problem will be further discussed and a solution will be presented.

2.3 Artificial Neural Networks Models

Besides the classical statistical models, another approach to time series analysis are the nonparametric models. While there is a wide range of different nonparametric models, the Artificial Neural Networks (ANN) models are the most common (RIPLEY, 1996). ANNs have been widely used for a diversity of forecasting problems, including linear and nonlinear time series and present excellent results in terms of forecasting for both. The problems with the use of ANNs are: (i) the randomness of the model and (ii) the search for an optimal configuration. Each one of these issues would not be so relevant if the other one does not occur.

For example, in the work of (ZHANG, 2001) the author made many tests using Multilayer Perceptron (MLP) models in different simulated linear time series, using the ARIMA models family, as an MA(1), MA(2) among others, and concludes that the MLP models can work very well, if not better, than classical ARIMA models on linear series. In the same work it is said that for nonlinear time series, the ANNs have already been tested and also presented excellent forecasting capabilities. But all the comparisons were made using different types of forecasting errors and using too many different MLP configurations to find the ones that the author recommends.

The ANNs, in general, do not use any tools to assist the user to find the optimal network configuration. They heavily depend on systematic methods, where the most common is the genetic algorithm. This can be seen in the works of (CRONE; HIBON; NIKOLOPOULOS, 2011), (DONATE; SANCHEZ; MIGUEL, 2012) and (LIN WANG; CHEN, 2015). In (CRONE; HIBON; NIKOLOPOULOS, 2011) the authors report and analyse the results from the NN3 competition, a competition of time series forecasting. Besides being an empirical result, all mentioned models and configurations were obtained using some systematic search method to find the best configurations, including genetic and evolutionary algorithms. In the work of (DONATE; SANCHEZ; MIGUEL, 2012) the authors present a different approach based on Automatic Design of Artificial Neural Networks (ADANN), among other models. The main goal of the work is to develop an accurate automatic method to design ANNs. Finally, in (LIN WANG; CHEN, 2015), an improved adaptive differential evolution (ADE) method is proposed using back-propagation neural network (BPNN) to stablish better connection weights and thresholds. This reinforces the idea that, mainly, the ANN models presented in different works were obtained by using some automatic method.

For other works, as in (ZOUNEMAT-KERMANI; TESHNEHLAB, 2007; CHIU; CHEN, 2009; DELLANA; WEST, 2009), they use a trial-and-error method, which consists in using a large number of different configurations and test all in the series. In (ZOUNEMAT-KERMANI; TESHNEHLAB, 2007), the authors use an adaptive neuro-fuzzy network, with different configurations. The authors tried 20 different configurations, although only 10 are shown in the results. In (CHIU; CHEN, 2009), the authors use a fuzzy-based support vector machine with a high dimensional input layer (a total of 61 inputs) and genetic algorithms. The authors indicate that the genetic algorithms used in the work are to optimize the model parameters, but in fact, as the parameters can be set to zero (0), they are used to configure the input layer. The authors also tested different configurations for the input layer. In (DELLANA; WEST, 2009), the authors also use various different configurations for forecasting and, as in the previous works, choose one that offers the better results for the used series.

As said before, the use of automatic methods or trial-and-error is not an issue by itself. Even the classical ARIMA models are normally presented with different configurations or using some automatic method to find the used parameters. The issue is the lack of tools to help the user to achieve, if not the best configuration, one that suits the problem well enough. The question that arises is: what criterium can be used to define well enough? Almost all automatic methods need to optimize a certain criterium. In the ANN models for time series, this criterium is the forecasting errors. But what forecasting errors? The forecasting errors can be calculated in many different ways and with many different results. The Mean Squared Error (MSE), the Root Mean Squared Error (RMSE) and the Mean Absolute Percentage Error (MAPE) are very popular, but are not the only ones. In the work of (ZHANG, 2001), a different version of MAPE is used, as in the works of (DELLANA; WEST, 2009). In fact, in the work of (QI; ZHANG, 2001), the authors present 17 different error metrics. Based only on this work, the user may have to choose among 17 different metrics along the different automatic methods to find a suitable model. In (VOYANT et al., 2015) the authors propose to use heterogeneous transfer functions on a MLP modelling to improve forecasting. However, even then, the user will be confronted with another issue of the ANNs, the randomness of the model.

The majority of ANNs models suffer from the randomness of that parameters, which is, basically, the fact that the same configuration applied for the same training data achieves different results (HAYKIN, 2001; RIPLEY, 1996). Even with some modifications on the algorithm, as in the works of (COSTA; BRAGA; MENEZES, 2007; ZHANG, 2003; WANG; LU, 2006), the problem persists. Most works show the results using a k-fold cross validation or presenting the best model, based on some error metric, as in (COSTA; BRAGA; MENEZES, 2007; WANG; LU, 2006; DELLANA; WEST, 2009; KHASHEI; BIJARI, 2010, 2014; ZHANG, 2003; ZOUNEMAT-KERMANI; TESHNEHLAB, 2007) among others. So, after choosing for an error metric and an automatic method, the model must be verified over a k-fold cross validation or repeatedly trained until a suitable model emerges. And all this excluding the fact that, even after all that steps, the model must be retrained after some period, as the series may have changed.

These issues persist even on newer approaches, as the hybrid models, that use some of the classical ARIMA capabilities, as we can see in (KHASHEI; BIJARI, 2010, 2014; ZHANG, 2003). These works present hybrid models using the classical ARIMA to model the linear patterns and the ANN to model the nonlinear ones. The work of (KHASHEI; BIJARI, 2010) uses a different approach than the work of (ZHANG, 2003), and just the results are compared, but both use already tested configurations or trial-and-error methods, for different presented series. In (KHASHEI; BIJARI, 2014) they use a fuzzy artificial neural network hybrid model, with mostly trial-and-error methods.

Finally, an issue that all the nonparametric methods have, and often not mentioned, is the issue of overfitting. The ANNs also suffer from this, but some algorithm modifications, as shown on (COSTA; BRAGA; MENEZES, 2007) and (WANG; LU, 2006), and software tools, as the MATLAB Neural Networks Toolbox, can prevent it. The work of (COSTA; BRAGA; MENEZES, 2007) presents a different control option for the Levenberg-Marquadt algorithm, preventing the model from overfitting. The work of (WANG; LU, 2006) presents the same Levenberg-Marquadt algorithm but adds a stochastic particle swarm optimization (PSO), which can also help to prevent overfitting. The method used in Matlab is more practical: it is called early stop and it stops the training when the forecasting errors (commonly forecasting MSE) begin to increase. Overfitting is not commonly mentioned any longer as nearly all models are adjusted using some automatic method based on forecasting errors, minimizing the risk of an overfitted model.

So, we have a class of models that, while can offer a vast range of uses and presents very small forecasting errors, is also randomly parametrized, heavily dependent on automatic methods to be configured and that must be retrained after a certain period of use. We intend to offer a model that, while maintaining some, if not all, the best characteristics of the ANN models: (i) can be used in a more systematic manner, preventing overfitting, (ii) can offer tools to help the user to achieve a parsimonious model, (iii) be less randomly generated, (iv) be less dependent on automatic methods and (v) can be used online, without the need to rebuild the model. The ARMA-CIGMN model is the result we achieved and will be presented in the next chapter.

3 THE ARMA-CIGMN

3.1 Initial comments

In this chapter we initially present the original IGMN, based on a Bayesian approach, as shown in (HEINEN; ENGEL; PINTO, 2011) and (HEINEN, 2011) with some modifications made after (PINTO, 2011), to work with time series. We also present our previous attempts to better adapt the original IGMN to time series and to work with MA components. This previous attempts were made modifying the original IGMN to a similar ARMA model, that we call the ARMA-like model, and a similar NOE (Nonlinear Output Errors) model, which we call the NOE-like model.

We then present our proposed model, the ARMA-CIGMN, and the reformulation of the original IGMN. The reformulation of the IGMN into a classical statistical approach led to the Classical IGMN (CIGMN), a necessary step to model the MA components. The ARMA-CIGMN model is a classical statistical version of the IGMN algorithm that can work with both autoregressive and moving average components of a time series.

3.2 The original IGMN Bayesian formulation

This section presents the general formulation, based on the works of (HEINEN, 2011) and (HEINEN; ENGEL; PINTO, 2011). The difference between this general presentation and the time series formulation is the specification of the components over a time series.

Let us assume a set of variables as $X_1, X_2, \ldots, X_p$, represented by the vector $\mathbf{X}$, from a population. And let us assume a sample from this population to be presented as $x_1, x_2, \ldots, x_p$, represented by the vector $\mathbf{x}$. A Gaussian Mixture Model (GMM) is, basically, a weighted sum of M Gaussian density components, as shown in 3.1.

$$p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{i=1}^{M} w_i g(\mathbf{X}|\boldsymbol{\mu_i}, \Sigma_i), \tag{3.1}$$

where $\mathbf{X}$ is a p-dimensional continuous-valued data vector, w_i, $i = 1, 2, \ldots, M$, are the mixture weights and $g(\mathbf{X}|\boldsymbol{\mu_i}, \Sigma_i)$, $i = 1, 2, \ldots, M$, are the Gaussian density components. Each component density is a p-variate Gaussian function of the form,

$$g(\mathbf{x}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}_i|^{1/2}} exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu_i})^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu_i})\right\}, \tag{3.2}$$

with mean vector $\boldsymbol{\mu_i}$ and covariance matrix $\boldsymbol{\Sigma}_i$. The mixture weights w_i satisfy the condition of

$$\sum_{i=1}^{M} w_i = 1. \tag{3.3}$$

Using Bayes' Theorem, the *a posteriori* probability for any i component is given by

$$P(i|\mathbf{x}_t, \boldsymbol{\theta}) = \frac{w_i g(\mathbf{x}_t|\boldsymbol{\mu_i}, \Sigma_i)}{\sum_{k=1}^{M} w_k g(\mathbf{x}_t|\boldsymbol{\mu_k}, \Sigma_k)} \tag{3.4}$$

The Gaussian mixture model is completely parametrized by the mean, weight vectors and the covariance matrices, for all components, i.e., for all distributions. The mean, weight vectors and the covariance matrices are represented by $\boldsymbol{\theta}$, where

$$\boldsymbol{\theta}_i = \{w_i, \boldsymbol{\mu_i}, \Sigma_i\}, \forall i = 1, 2, \ldots, M. \tag{3.5}$$

The covariance matrix $\boldsymbol{\Sigma}_j$ can be full rank or constrained to be diagonal. The parameters can also be shared among all components, with some care to the w_i, because of (3.3).

These parameters can be computed by various means, being the most common, the maximum likelihood (ML) estimation. Assuming a sample of size t, a training data, and the independence between the vectors, the ML estimation is based on maximizing

$$p(\mathbf{x}|\theta) = \prod_{i=1}^{t} p(\mathbf{x}_i|\boldsymbol{\theta}) = \prod_{i=1}^{t} \left[\sum_{i=1}^{M} w_i g(\mathbf{x}|\boldsymbol{\mu_i}, \Sigma_i) \right] \tag{3.6}$$

One simple way of understanding the IGMN is as an incremental way to obtain the w_i and the necessary number of components M, as represented on Figure 3.1. One of the problems in Gaussian mixture models is the *a priori* need to define the number of components, i.e., M, should be set before adjusting the parameters. In an online model, this is not convenient, since the full data sample is unavailable at the beginning of the process.

Initially, the IGMN model starts with a single component ($M = 1$), with the mean ($\boldsymbol{\mu}$) set at the same values of the first input data vector, i.e., $\mathbf{x}_1$. The covariance matrix is set at a baseline, as $\boldsymbol{\Sigma}_1 = diag(\boldsymbol{\sigma}_{ini}^2)$, where $diag(\boldsymbol{\sigma}_{ini})$ is a diagonal matrix calculated using an user defined fraction γ of the overall variance. Usually, $\sigma_{ini} = 3\gamma s_x$, where s_x represents the sample standard deviation of the dataset. If a dataset is not available, the constant 3 is chosen, based on the coverage of the Gaussian distribution.

So, in the first input data vector, the $\boldsymbol{\theta}_1$, we have

$$w_1 = 1, \boldsymbol{\mu}_1 = \mathbf{x}_1 \quad \text{and} \quad \boldsymbol{\Sigma}_1 = diag(\boldsymbol{\sigma}_{ini}^2). \tag{3.7}$$

When the second input vector arrives, the algorithm must decide, based on a threshold, if the new vector belongs to the old component ($M = 1$), and that component must be updated, or if the new vector belongs to a new component. This decision is based on

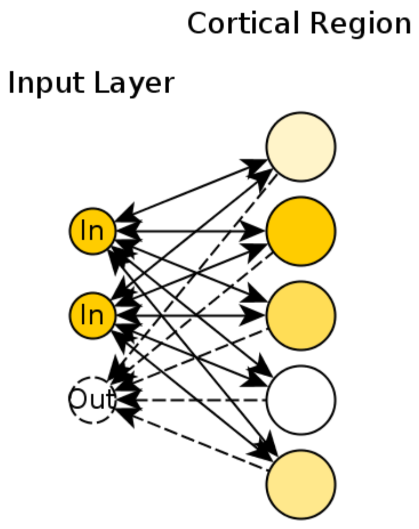


Figure 3.1: An example of IGMN with 3 input nodes and 5 Gaussian components. Two of the input elements were selected for estimating the third one. The different color intensities inside each Gaussian component represent their different likelihoods after seeing data $\mathbf{x}_i$ (only the known elements), and are used to weight the contributions of each component to the final result.

an user defined parameter τ_{min}, usually set around 0.01, for example, using a modified density function, as shown in (3.8).

$$\hat{g}(\mathbf{x}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma}_i) = exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu_i})^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu_i})\right\} \tag{3.8}$$

The value computed with (3.8) is always between 0 and 1, where 1 represents a vector identical to the mean vector, i.e., the data is already represented. This calculation is made for all M components. If, for all M components, the computed $\hat{g}(\mathbf{x}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma}_i)$ is bellow τ_{min}, then a new component is created.

There is another user defined parameter to help prevent model overfitting. For example, if a large number of data vectors is similar, in respect to the mean, the component that better represents these data is updated too many times. This is, however, subjective to each problem and user. The parameter τ_{max} prevents overfitting using a superior limit for the $\hat{g}(\mathbf{x}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma}_i)$ computed with (3.8). Usually, τ_{max} is set up near 1 (0.99, for example) to prevent that very similar data modify the other parameters. This calculation is also made for all M components. In short, after a data vector arrives, using (3.8), there are three options:

1. The value is bellow τ_{min} and a new component is created (as seen above, in (3.7));
2. The value is greater than the τ_{max} and this vector is considered too similar and is no further considered in the update process.
3. The value is between τ_{min} and τ_{max} and no new component is created and the vector is used to update the components;

The focus now is when the value computed with (3.8) is between τ_{min} and τ_{max} and the model is updated. First, the *a posteriori* probabilities are computed for all M components, according to (3.4). The following equations present the other parameters updates according to the algorithm (PINTO, 2011).

$$v_{i,t} = v_{i,t-1} + 1 \tag{3.9}$$

$$sp_{i,t} = sp_{i,t-1} + P(j|\mathbf{x}_t, \boldsymbol{\theta}) \tag{3.10}$$

$$\mathbf{e}_i = \mathbf{x} - \boldsymbol{\mu_{i,t}} \tag{3.11}$$

$$\psi_i = \frac{P(j|\mathbf{x}_i, \boldsymbol{\theta})}{sp_i} \tag{3.12}$$

$$\Delta\boldsymbol{\mu_i} = \psi_i \mathbf{e}_i \tag{3.13}$$

$$\boldsymbol{\mu_{i,t}} = \boldsymbol{\mu_{i,t-1}} + \Delta\boldsymbol{\mu_i} \tag{3.14}$$

$$\boldsymbol{\Sigma}_{i,t} = (1-\psi_i)\boldsymbol{\Sigma}_{i,t-1} - \Delta\boldsymbol{\mu_i}\Delta\boldsymbol{\mu_i}' + \psi_i(\mathbf{e}\mathbf{e}') \tag{3.15}$$

$$w_i = \frac{sp_i}{\sum_{q=1}^{M} sp_q} \tag{3.16}$$

where sp_i and v_i are the accumulator and the age of component i, respectively, and w_i, as defined before, is its prior probability.

The age of the component, v_i, is necessary to evaluate the stability criterium (HEINEN, 2011) for a component. For example, if an outlier is presented to the model, probably a new component will be created. However, this component, also probably, will not be updated again. So, after some time without any updates, a component (based on its age, v_i) is deleted.

The sp_i parameter determines the *a priori* probability. One problem is that the sp_i, as seen on (3.10), is based on the quantity of updates that the component has experienced. To control the possible over estimate, i.e., too many updates, of the *a priori* the τ_{max} parameter is used.

The ψ_i parameter is the contribution over the sp_i of the new component or new data. It is similar to a weight and is used to update the mean (3.14) and the covariance matrix (3.15). If, at any time t a data vector is presented with missing data, then the IGMN enters its recalling method.

The recalling is made estimating the posterior probability using only the provided data. Let us assume that an incomplete data vector is presented, $\mathbf{x}_i$. In this case, the posterior probability is calculated using (3.4) with few modifications. The *a posteriori* probability equation becomes (3.17).

$$P(j|\mathbf{x}_i, \boldsymbol{\theta}) = \frac{w_j g(\mathbf{x_i}|\boldsymbol{\mu_j}, \boldsymbol{\Sigma}_j)}{\sum\limits_{k=1}^{M} w_k g(\mathbf{x}_i|\boldsymbol{\mu_k}, \boldsymbol{\Sigma}_k)} \quad \forall j = 1, 2, \ldots, M \tag{3.17}$$

With all the *a posteriori* probabilities calculated, the missing data, $\hat{\mathbf{x}}_t$, can be reconstructed using (3.18).

$$\hat{\mathbf{x}}_t = \sum_{j=1}^{M} P(j|\mathbf{x}_i, \boldsymbol{\theta})(\boldsymbol{\mu_{j,t}} + \boldsymbol{\Sigma}_{j,ti}\boldsymbol{\Sigma}_{j,i}^{-1}(\mathbf{x}_i - \boldsymbol{\mu_{j,i}})) \tag{3.18}$$

where $\mu_{j,t}$ is the jth's component mean of the missing data, $\mu_{j,i}$ is the jth's component mean of the incomplete data vector, $\mathbf{x}_i$, $\boldsymbol{\Sigma}_{j,ti}$ and $\boldsymbol{\Sigma}_{j,i}$ are the submatrices of $\boldsymbol{\Sigma}_j$, as in (3.19).

$$\boldsymbol{\Sigma}_j = \begin{bmatrix} \boldsymbol{\Sigma}_{j,t} & \boldsymbol{\Sigma}_{j,ti} \\ \boldsymbol{\Sigma}_{j,it} & \boldsymbol{\Sigma}_{j,i} \end{bmatrix} \tag{3.19}$$

For example, let us assume a simple bivariate model in (x, y). If any y_i, is missing, then the recalling function will be as:

$$\hat{y}_i = \sum_{j=1}^{M} P(j|x_i, \boldsymbol{\theta}) \left(\bar{y} + \frac{cov(x,y)}{var(x)}(x_i - \bar{x}) \right) \tag{3.20}$$

Suppose that the model has only one component, i.e., $M = 1$, so (3.20) becomes:

$$\hat{y}_i = \bar{y} + \frac{cov(x,y)}{var(x)}(x_i - \bar{x}) \tag{3.21}$$

The sample covariance (bivariate) is computed with (3.22) and the variance with (3.23).

$$cov(x,y) = \frac{\sum\limits_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n - k} \tag{3.22}$$

$$var(x) = s^2 = \frac{\sum\limits_{i=1}^{n} (x_i - \bar{x})^2}{n - k} \tag{3.23}$$

where k is the number of parameters in the model, which is not relevant, as both variance and covariance use the same k. Replacing both (3.22) and (3.23) in (3.21) and with both denominators $(n - k)$ being the same, then:

$$\hat{y}_i = \bar{y} + \frac{\sum\limits_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum\limits_{i=1}^{n} (x_i - \bar{x})^2}(x_i - \bar{x}) \tag{3.24}$$

Equation (3.24) is very similar to the linear regression model, using ordinary least squares. In the linear regression model, there are two parameters, β_0 and β_1, that are computed as bellow.

$$\beta_0 = \bar{y} + \beta_1\bar{x} \tag{3.25}$$

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{3.26}$$

Using (3.26) to replace the ratio in (3.24) by β_1 results

$$\hat{y}_i = \bar{y} + \beta_1(x_i - \bar{x}) \tag{3.27}$$

Expanding (3.27) even more, we get

$$\hat{y}_i = (\bar{y} - \beta_1\bar{x}) + \beta_1 x_i \tag{3.28}$$

Finally, using the definition of β_0 in (3.25) and replacing in (3.28) we obtain

$$\hat{y}_i = \beta_0 + \beta_1 x_i \tag{3.29}$$

So the recalling model of the IGMN is a probabilistic combination of linear regression models. For the specific case where $M = 1$ and $D = 2$, the recalling model is shown in (3.29). This result opens some other uses for the IGMN. In the particular, but not uncommon, case where we have only one output, a wide range of models is available. For example, econometric models, as the log-log, lin-log or log-lin models can be learned by the algorithm. Classification models, models with dummies variables among others can be adapted, in some cases, without any modifications whatsoever to the algorithm. These variations are not presented in this work, but can be seen on (NETER et al., 1996).

The probabilistic combination of linear models allows the IGMN model to be considered nonlinear. This can also be shown in the general case, for any M and D. The general model class is called a Normal Correlation Model (NETER et al., 1996), or Gaussian Correlation Model. In a correlation model, all variables are random and their combined densities are multivariate Gaussian (or Normal) densities. To demonstrate the similarities, let us assume a bivariate model using Y_1 and Y_2. The density function for the correlation model using the bivariate Gaussian distribution is

$$\begin{aligned} f(Y_1, Y_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho_{12}^2}} \exp\Bigg\{ -\frac{1}{2(1-\rho_{12}^2)} \Bigg[\left(\frac{Y_1 - \mu_1}{\sigma_1}\right)^2 \\ -2\rho_{12}\left(\frac{Y_1 - \mu_1}{\sigma_1}\right)\left(\frac{Y_2 - \mu_2}{\sigma_2}\right) + \left(\frac{Y_2 - \mu_2}{\sigma_2}\right)^2 \Bigg] \Bigg\} \end{aligned} \tag{3.30}$$

which is the same Gaussian distribution presented in (3.2), except that in (3.30) it is the bivariate distribution. Note, however, that in (3.30) there are five different parameters: μ_1, σ_1,μ_2, σ_2 and ρ_{12}. Where μ_1 and σ_1 are the mean and standard deviation of Y_1; μ_2 and σ_2

the mean and standard deviation of Y_2 and ρ_{12} the coefficient of correlation between Y_1 and Y_2, defined by

$$\rho_{12} = \rho\{Y_1, Y_2\} = \frac{\sigma_{12}}{\sigma_1 \sigma_2} \tag{3.31}$$

where σ_{12} denotes de covariance between Y_1 and Y_2.

Now suppose that Y_2 has a known value and we want to estimate Y_1 based on Y_2, i.e., $f(Y_1|Y_2)$, that is defined as

$$f(Y_1|Y_2) = \frac{f(Y_1, Y_2)}{f_2(Y_2)} \tag{3.32}$$

where $f(Y_1, Y_2)$ is the joint density function of Y_1 and Y_2, as in (3.30), and $f_2(Y_2)$ is the marginal density function of Y_2.

Then the conditional probability distribution of Y_1 for any given value of Y_2 is

$$f(Y_1|Y_2) = \frac{1}{\sqrt{2\pi\sigma_{1.2}^2}} \exp\left\{ -\frac{1}{2} \left(\frac{Y_1 - \alpha_{1.2} - \beta_{12} Y_2}{\sigma_{1.2}} \right)^2 \right\} \tag{3.33}$$

where $\alpha_{1.2} + \beta_{12} Y_2$ is the mean and $\sigma_{1.2}$ is the standard deviation.

The parameters $\alpha_{1.2}$, β_{12} and $\sigma_{1.2}$ are functions of the parameters of the joint probability distribution (3.30), as follows:

$$\alpha_{1.2} = \mu_1 - \mu_2 \rho_{12} \frac{\sigma_1}{\sigma_2} \tag{3.34}$$

$$\beta_{12} = \rho_{12} \frac{\sigma_1}{\sigma_2} \tag{3.35}$$

$$\sigma_{1.2}^2 = \sigma_1^2 (1 - \rho_{12}^2) \tag{3.36}$$

As seen on Equation (3.33), the mean for $f(Y_1|Y_2)$, i.e., $E[Y_1|Y_2] = \alpha_{1.2} - \beta_{12} Y_2$ is similar to the regression model presented in (3.29), with $\alpha_{1.2}$ representing the β_0 and β_{12} representing β_1. The notation on α and β is necessary because the constant and slope are different if different Y is given. The parameter $\alpha_{1.2}$ is the constant if Y_1 is unknown and we know Y_2.

For example, in a model with four different variables (Y_1, Y_2, Y_3 and Y_4). If Y_1, Y_3 and Y_4 are known and we want to estimate the value of Y_2, then we have the parameter $\alpha_{2.134}$ and β_{2134}.

3.3 The ARMA-like IGMN

The first modification we made on the original IGMN to better adapt it to time series consists in using similar inputs as the classical ARMA model, i.e., the actual observed p past values $x_{(t-p)}$ and the q past errors $e_{(t-q)}$, defined in Equation (3.37).

$$e_{(t-q)} = x_{(t-q)} - \hat{x}_{(t-q)} \tag{3.37}$$

For the IGMN to use the past errors $e_{(t-q)}$, first they have to be estimated. So, in the first q steps, it is necessary to fill in a vector buffer e for future use. With this first e vector, the IGMN begins estimating its parameters and components, using both the past values and the past errors.

For example, an ARMA(2,2) model, as described in Equation (3.38) can be mapped to an IGMN structure as shown in Figure 3.2.

$$x_{(t)} = \phi_1 x_{(t-1)} + \phi_2 x_{(t-2)} + \theta_1 e_{(t-1)} + \theta_2 e_{(t-2)} + e_t \tag{3.38}$$

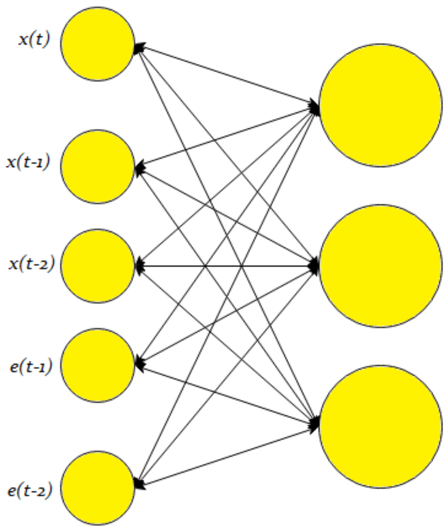


Figure 3.2: An example of IGMN ARMA-like structure with 3 components representing an ARMA-like(2,2) model, where $x_{(t)}$ represents the present value, $x_{(t-1)}$ and $x_{(t-2)}$ the autoregressive past values, AR(2), and $e_{(t-1)}$ and $e_{(t-2)}$ the past errors, similar as in an MA(2) model.

This approach, however, presents two problems: (i) the additional step of recalling within the learning mode, and (ii) the nonlinearity of the estimates.

The first problem occurs when the IGMN, in learning mode, for every new data, must enter the recalling mode to obtain the estimates. Then, with the estimate, it calculates the error, includes this error on the input data and enters learning mode again. This additional step of recalling for each new data makes the IGMN slower when learning.

The second problem is the nonlinearity of its estimates. For example, let us assume a simple ARMA(1,1) model, as described on Equation (3.39).

$$x_{(t)} = \phi x_{(t-1)} + \theta e_{(t-1)} + e_t \tag{3.39}$$

So, the estimates of x can be achieved using

$$\hat{x}_{(t)} = \phi x_{(t-1)} + \theta e_{(t-1)}$$

While $x_{(t-1)}$ is an observable value, $e_{(t-1)}$ must be calculated. The same approach can be used to past values, and so $\hat{x}_{(t-1)}$ is predicted by

$$\hat{x}_{(t-1)} = \phi x_{(t-2)} + \theta e_{(t-2)}$$

and so on, for any $\hat{x}_{(t-r)}$, where $r = 1, 2, \ldots, (t-1)$.

We previously defined e_t, on Equation (3.37). Assuming $q = 1$, for simplicity, and replacing $\hat{x}_{(t-q)}$ we have

$$e_{(t-1)} = x_{(t-1)} - \hat{x}_{(t-1)}$$
$$e_{(t-1)} = x_{(t-1)} - (\phi x_{(t-2)} + \theta e_{(t-2)})$$

We can backtrack the errors, replacing every $e_{(t-q)}$ as in Equation (3.37) and notice that the past errors also depend on θ, and, on this example, also depend on ϕ. The estimates used on the original IGMN are based on a OLS estimator, which can not be applied in a nonlinear system, as discussed on (GUJARATI, 2006; HAMILTON, 1994; NETER et al., 1996). The limitations of this approach will be further discussed in Chapter 4.

3.4 The NOE-like IGMN

With this version we intend that the IGMN should model the behavior of the past errors $e_{(t-r)}$ implicitly and not explicitly, as with the classical ARMA model. For example, an ARMA(2,2) model can be represented on a IGMN structure as shown on Figure 3.3 and Equation (3.40)

$$x_{(t)} = \phi_1 x_{(t-1)} + \phi_2 x_{(t-2)} + \theta_1 \hat{x}_{(t-1)} + \theta_2 \hat{x}_{(t-2)} + e_t \tag{3.40}$$

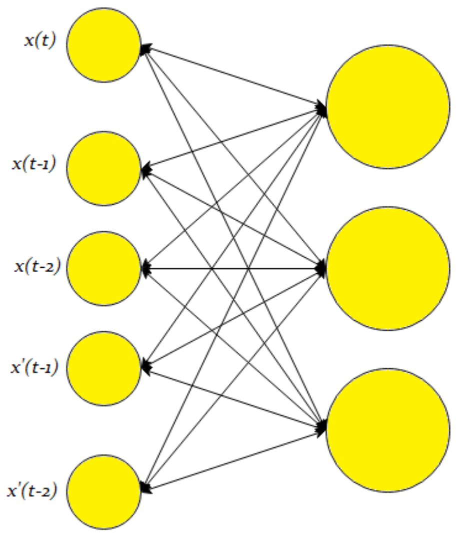


Figure 3.3: An example of the IGMN NOE-like structure with 3 components representing a NOE-like(2,2) model, where $x_{(t)}$ represents the present value, $x_{(t-1)}$ and $x_{(t-2)}$ the autoregressive past values and $x'_{(t-1)}$ and $x'_{(t-2)}$ the past estimated values.

This version uses the past predicted values (or past forecasts) information as output errors, similar to the well known NOE model. Although presenting some good results, as we can see later in this work, it also presents two problems, similar as with the ARMA-like version: (i) the additional step for forecasting and (ii) the nonlinearity of its estimates. To better present this, we use a simple ARMA(1,1) model which, with this modification, can be presented as shown on Equation (3.41).

$$x_{(t)} = \phi x_{(t-1)} + \theta \hat{x}_{(t-1)} + e_t \tag{3.41}$$

where $\hat{x}_{(t-1)}$ is computed using Equation (3.42).

$$\hat{x}_{(t-1)} = \phi x_{(t-2)} + \theta \hat{x}_{(t-2)} \tag{3.42}$$

Here, as with the ARMA-like version, we can see that $\hat{x}_{(t-1)}$ depends on two different parameters, ϕ and θ, which is practically the same problem addressed in the ARMA-like version, plus the extra recalling step to obtain the predicted $\hat{x}_{(t-r)}$ values. There are some alternatives to solve this problem, but most of them are not able to run online or incrementally (NETER et al., 1996; MORETTIN; TOLOI, 2006).

Also, there is another issue in both versions when working with a time series that presents only the MA process, which is called a pure MA process. It is not very common to work on a pure MA process time series, but for any pure MA process, an MA(q) model more generally, both versions need, at least one AR component. In the NOE-like version, and in the ARMA-like version, the past predicted values $\hat{x}_{(t-r)}$, and so the past errors $e_{(t-r)}$ can only be computed with an AR(1) component, at least. For example, suppose an MA(1) process, as described on Equation (3.43)

$$x_{(t)} = \theta e_{(t-1)} + e_t \tag{3.43}$$

For both versions we may start the process with $e_{(0)} = 0$ or $\hat{x}_{(1)} = x_{(1)}$ and work from there

$$\begin{aligned}
\hat{x}_{(1)} &= \theta e_{(0)} = 0 \\
\hat{x}_{(2)} &= \theta e_{(1)} = \theta(x_{(1)} - \hat{x}_{(1)}) = \theta x_{(1)} \\
\hat{x}_{(3)} &= \theta e_{(2)} = \theta(x_{(2)} - \hat{x}_{(2)}) = \theta(x_{(2)} - \theta x_{(1)})
\end{aligned}$$

and so on, which is very similar as an AR process. The initialization of $\hat{x}_{(1)} = x_{(1)}$ or $e_{(0)} = 0$ can be modified but with little or no practical effect. In the next section, we present our proposal to solve the MA components estimation and, in Chapter 4 we present the results for this approach.

3.5 The ARMA-CIGMN

In this section we reformulate the original IGMN. Some minor modifications where made in the overall algorithm, but a major one in the approach, to take into account the moving average (MA) component. While the original formulation uses a Bayesian approach, this one uses a classical statistical approach. This is made to further present the IGMN as a combination of correlation models, which is similar to a combination of regression models, and to adapt the MA parameters, as the method present here is only possible under a classical statistical approach. In fact, we begin the presentation with the correlation model and its similarities to the classical regression model. We reformulate the IGMN algorithm in the classical statistical approach resulting in the Classical IGMN (CIGMN). Thereafter, we present the modifications made in the CIGMN algorithm to take into account the moving average process (MA(q) components) resulting in the ARMA-CIGMN model.

As seen in past works of (FLORES; PINTO; ENGEL, 2012) and (PINTO; ENGEL; HEINEN, 2011), the vast majority of ANN models fail to incorporate the moving average (MA) component in time series analysis. As previously shown, a wider autoregressive (AR) component window can be used to reduce the MA component effect. However, to best use the information and to show a more theoretical and practical modeling, we added a MA structure to the IGMN model. Although this MA structure is simple, it increases the dimensionality of the data and, so, the model becomes more complex.

3.5.1 Correlation Model

Let us suppose a data frame described by a k-by-n matrix $\mathbf{X}$, where k is the number of different random variables $X_1, X_2, \dots, X_k$ and n it is the sample size of such variables.

The covariance between any pair of variables is defined as

$$Cov(X_i, X_j) = \sigma(X_i, X_j) = E[(X_i - E[X_i])(X_j - E[X_j])] \tag{3.44}$$

where $i \neq j$, and $i = j = 1, 2, 3, \dots, k$. In the case where $i = j$, we have

$$\begin{aligned} Cov(X_i, X_i) = \sigma(X_i, X_i) = E[(X_i - E[X_i])(X_i - E[X_i])] = \\ E[(X_i - E[X_i])^2] = E[X_i^2] - (E[X_i])^2 = VAR(X_i) = \sigma^2(X_i) \end{aligned} \tag{3.45}$$

which is the variance of the variable X_i.

In a k-dimensional, multivariate approach, the variances and covariances of each single pair (X_i, X_j) are grouped in a k-by-k matrix, the covariance matrix $\mathbf{\Sigma}$.

Suppose we have a k-dimensional realization vector of $\mathbf{X}$, $\mathbf{x}_i$, for $i = 1, 2, 3, \dots, n$. To estimate a covariance matrix from a known sample, we use

$$\mathbf{S} = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \tag{3.46}$$

where $\bar{\mathbf{x}}$ is the mean vector calculated over the sample, and defined as

$$\bar{\mathbf{x}} = \begin{bmatrix} \bar{\mathbf{x}}_1 \\ \bar{\mathbf{x}}_2 \\ \vdots \\ \bar{\mathbf{x}}_k \end{bmatrix} = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i) \tag{3.47}$$

Using Equation 3.46 and rewriting it, we have a more computational, and easy way to adapt it to an online learning environment.

$$
\begin{aligned}
&\sum_{i=1}^{n}(\mathbf{x}_i-\bar{\mathbf{x}})(\mathbf{x}_i-\bar{\mathbf{x}})^T=\sum_{i=1}^{n}(\mathbf{x}_i-\bar{\mathbf{x}})(\mathbf{x}_i^T-\bar{\mathbf{x}}^T)\\
&=\sum_{i=1}^{n}(\mathbf{x}_i\mathbf{x}_i^T-\mathbf{x}_i\bar{\mathbf{x}}^T-\bar{\mathbf{x}}\mathbf{x}_i^T+\bar{\mathbf{x}}\bar{\mathbf{x}}^T)\\
&=\sum_{i=1}^{n}(\mathbf{x}_i\mathbf{x}_i^T)-\sum_{i=1}^{n}(\mathbf{x}_i\bar{\mathbf{x}}^T)-\sum_{i=1}^{n}(\bar{\mathbf{x}}\mathbf{x}_i^T)+n\bar{\mathbf{x}}\bar{\mathbf{x}}^T\\
&=\sum_{i=1}^{n}(\mathbf{x}_i\mathbf{x}_i^T)-\left(\sum_{i=1}^{n}\mathbf{x}_i\right)\bar{\mathbf{x}}^T-\bar{\mathbf{x}}\sum_{i=1}^{n}\mathbf{x}_i^T+n\bar{\mathbf{x}}\bar{\mathbf{x}}^T\\
&=\sum_{i=1}^{n}(\mathbf{x}_i\mathbf{x}_i^T)-\frac{1}{n}\left(\sum_{i=1}^{n}\mathbf{x}_i\right)\left(\sum_{i=1}^{n}\mathbf{x}_i^T\right)-\frac{1}{n}\left(\sum_{i=1}^{n}\mathbf{x}_i\right)\left(\sum_{i=1}^{n}\mathbf{x}_i^T\right)+n\bar{\mathbf{x}}\bar{\mathbf{x}}^T\\
&=\sum_{i=1}^{n}(\mathbf{x}_i\mathbf{x}_i^T)-2\bar{\mathbf{x}}\sum_{i=1}^{n}\mathbf{x}_i^T+n\bar{\mathbf{x}}\bar{\mathbf{x}}^T\\
&=\sum_{i=1}^{n}(\mathbf{x}_i\mathbf{x}_i^T)-2\bar{\mathbf{x}}\sum_{i=1}^{n}\mathbf{x}_i^T+\bar{\mathbf{x}}\sum_{i=1}^{n}\mathbf{x}_i^T\\
&=\sum_{i=1}^{n}(\mathbf{x}_i\mathbf{x}_i^T)-\bar{\mathbf{x}}\sum_{i=1}^{n}\mathbf{x}_i^T\\
&=\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T-n\bar{\mathbf{x}}\bar{\mathbf{x}}^T
\end{aligned}
$$

which results in

$$
\mathbf{S}=\frac{1}{n-1}\left(\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T-n\bar{\mathbf{x}}\bar{\mathbf{x}}^T\right) \tag{3.48}
$$

Observing Equation (3.48), it is noted that we only need to store two sets of values to obtain the covariance matrix $\mathbf{\Sigma}$:(i) the outer product summation of the values, $x_ix_i^T$ and (ii) the summation vector $\sum x_i$. This can also be demonstrated in Equation (3.49).

$$
\mathbf{S_n}=\frac{1}{n-1}\left(s_n^2-\frac{1}{n}s_ns_n^T\right) \tag{3.49}
$$

where

$$
\begin{aligned}
s_n^2&=\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T\\
s_n&=\sum_{i=1}^{n}\mathbf{x}_i
\end{aligned}
$$

The estimate $\mathbf{S_n}$ and the vector $\bar{\mathbf{x}}$ are considered best linear unbiased estimator (BLUE) for the parameters $\mathbf{\Sigma}$ and $\boldsymbol{\mu}$, respectively. With the covariance matrix and the mean vector, a multivariate Gaussian distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is completely defined, as seen on Equation (3.50).

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \tag{3.50}$$

Now suppose that any subset of the k variables is missing. It is possible to estimate it, based on the vector $\bar{x}$ and the matrix $\mathbf{S_n}$. The process is similar to a regression model, however, in this scenario, any of the k variables can be the output (or response) variable, even more than one. This process in known as a correlation model (NETER et al., 1996), where all the variables are considered random and any subset can be used for making inferences about the remaining variables. This, however, does not change how these inferences are made.

For now, suppose a set of Y_l missing variables, where $l < k$ and $l \geq 1$, and X_{k-l} known variables. For simplicity, we assume these variables are grouped together. In a regression model, the estimation of the Y_l variables are made using Equation (3.51).

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}} \tag{3.51}$$

where $\hat{\boldsymbol{\beta}}$ is the estimate of $\boldsymbol{\beta}$ and $\hat{\mathbf{Y}}$ is the estimate of the missing Y_l variables.

These $\boldsymbol{\beta}$ are known as regression parameters and represent how the unknown variables can be explained by the known variables, i.e., $\hat{\mathbf{Y}}|\mathbf{X}$. Using the similarities between the correlation and the regression model, an estimate of $\boldsymbol{\beta}$ can be expressed as in Equation 3.52.

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{Y}) \tag{3.52}$$

Equation (3.52) represents an estimate of $\boldsymbol{\beta}$ using the ordinary least squares (OLS), which is a BLUE estimator (NETER et al., 1996) under some assumptions. It is important to note that even the $\hat{\boldsymbol{\beta}}$ can be calculated using the same set of values used to obtain $\bar{\mathbf{x}}$ and $\mathbf{S_n}$. To show this, assume that the outer product is computed using $\sum \mathbf{x}_i\mathbf{x}_i^T$, where all variables are known and are represented by M. Suppose that some of the variables are missing, i.e., $\mathbf{Y}$. So, M can be represented as in Equation (3.53).

$$\mathrm{M} = \begin{bmatrix} X^TX & X^TY \\ (X^TY)^T & Y^TY \end{bmatrix} \tag{3.53}$$

Observing Equation (3.53), we can recognize the first and second part of the Equation (3.52).

All these parameters can be computed at any given time, even after a short sample of n elements, where $n > k$. The model presented here is, however, a linear model. It has almost the same restrictions as an OLS regression model.

3.5.2 The Classical Incremental Gaussian Mixture Network - CIGMN

The correlation model is not suitable for nonlinear systems. However, a combination of correlation models can be. The original IGMN has this characteristic, but uses a Bayesian approach. To our needs, the IGMN must be reformulated using a classical statistical approach, which we call the Classical Incremental Gaussian Mixture Network, CIGMN. We use almost the same principles as in the correlation model and the original IGMN, but using a classical statistical Gaussian Mixture Model instead of a simple Multivariate Gaussian.

The probability density function of a Gaussian Mixture Model can be described as in Equation (3.54).

$$g(\mathbf{x}) = \sum_{i=1}^{m} \pi_i f(\mathbf{x}) \tag{3.54}$$

where π_i is the mixture weight, $\sum_{i=1}^{m} \pi_i = 1$, and m components. Each of the m components is a Gaussian distribution or a Multivariate Gaussian distribution, also known as Multivariate Normal distribution or simply MVN, as shown in Equation (3.50). The same occurs with the CIGMN, where each of its components is a MVN, completely defined with the estimates of the parameters $\boldsymbol{\mu}$ and Σ, i.e., $\bar{x}$ and $\mathbf{S_n}$.

One of the differences between the correlation model and the mixture model is that the estimation of the parameters of the former is linear, while in the last one is not. One of the most used methods to estimate π, $\boldsymbol{\mu}$ and Σ is the EM algorithm (MCLACHLAN; PEEL, 2000; SHUMWAY; STOFFER, 2000). However, in our approach, the use of the EM is not possible, mainly because: (i) we do not know, in advance, how many components are necessary and (ii) we do not have the entire sample available. To be an one-shot online incremental model, the CIGMN can not depend on the EM algorithm to estimate its parameters. Besides, our model is not an attempt to reproduce the exact Gaussian Mixture, but a way of representing a complex system. The CIGMN, therefore, does not aim to be a good estimate for π, $\boldsymbol{\mu}$ and Σ, but uses these parameters to better describe a nonlinear system.

Suppose we have a continuous data flow, represented here by $\mathbf{x}$. Our presentation of time series, assumes that the user defines a window of k observations for $\mathbf{x}$, transforming this data flow in vectors of size k. These vectors then form a matrix, as shown in Equation (3.55).

$$\begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,(k-1)} & x_{1,k} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,k} & x_{2,(k+1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,(k+n-2)} & x_{n,(k+n-1)} \end{bmatrix} \tag{3.55}$$

The CIGMN uses only one line of the matrix presented in Equation (3.55) at a time step and can be summarized in two different modes: (i) learning mode and (ii) recalling mode.

3.5.2.1 Learning Mode

The CIGMN learning mode can be performed perpetually, without a defined training phase. Wherever a complete pattern is presented, the CIGMN learning mode is applied. In this work, this means that a complete time window of size k is presented. In learning mode, CIGMN uses the data pattern to one of two possibilities: (i) create a new component or (ii) updating the existing ones. To decide between create or update, the IGMN uses the quadratic Mahalanobis' distance as a test statistic using the following hypothesis:

$$\begin{cases} H_0 : \mathbf{X}_t \in N(\mu_i, \Sigma_i), & \text{updates the existing components} \\ H_1 : \mathbf{X}_t \notin N(\mu_i, \Sigma_i), & \text{creates a new component} \end{cases}$$

where $\mathbf{X}_t$ is a random vector of time t, μ_i is the mean of the i-th component and Σ_i is the covariance matrix of the i-th component. The test is made for all i components and is a unilateral test based on the Mahalanobis' distance, the Qui-Square distribution and an user defined α, which is a type I error measure. If, for any i component, $M^2 \leq \chi^2_{(k,\alpha)}$, then it updates the components. If, for all i components, $M^2 > \chi^2_{(k,\alpha)}$ then it creates a new component. Where M^2 is the quadratic Mahalanobis' distance, as seen on Equation (3.56) and $\chi^2_{(k,\alpha)}$ is a limit score based on the χ^2 distribution with k degrees of freedom and α probability on the right side.

$$M_i^2 = (\mathbf{x} - \bar{\mathbf{x}}_i)^T \mathbf{S_{n,i}}^{-1} (\mathbf{x} - \bar{\mathbf{x}}_i), \quad \forall i = 1, 2, 3, \ldots, m \tag{3.56}$$

As seen on Equation (3.54), every component has its own set of parameters $\bar{\mathbf{x}}$ and $\mathbf{S_n}$. To update its parameters, first the contribution of all components is computed and then, weighted. The contribution of each of its components is based on the likelihood of the data, as seen on Equation (3.57) and the weights on Equation (3.58).

$$L_i = \frac{1}{\sqrt{|\mathbf{S_{n,i}}|}} exp\left(-\frac{1}{2} M_i^2\right), \forall i = 1, 2, 3, \ldots, m \tag{3.57}$$

$$w_i = \frac{L_i}{\sum_{i=1}^{m} L_i} \tag{3.58}$$

where w_i is used to weight the updates across the significant components.

With all the w_i computed, we move on to first update the sufficient statistics, i.e., s_n and s_n^2. This is made using Equations (3.59) and (3.60), respectively.

$$s_{(n+1),i} = \sum_{j=1}^{n} \mathbf{x}_j + w_i \mathbf{x}_{(j+1)} = s_{n,i} + w_i \mathbf{x}_{(j+1)}, \tag{3.59}$$

$$s^2_{(n+1),i} = \sum_{j=1}^{n} \mathbf{x}_j \mathbf{x}_j^T + w_i \mathbf{x}_{(j+1)} \mathbf{x}^T_{(j+1)} = s^2_{n,i} + w_i \mathbf{x}_{(j+1)} \mathbf{x}^T_{(j+1)}, \ \forall i = 1, 2, 3, \ldots, m \tag{3.60}$$

where $\mathbf{x}_{(j+1)}$ represents the new data, with s_n and s_n^2 calculated over all components.

The next step is to update the remaining parameters, shown in Equations from (3.61) through (3.64).

$$n_{(n+1),i} = n_{n,i} + w_i, \tag{3.61}$$

$$p_{(n+1),i} = \frac{n_{(n+1),i}}{\sum_{i=1}^{k} n_{(n+1),i}} \tag{3.62}$$

$$\bar{x}_{(n+1),i} = \frac{s_{(n+1),i}}{n_{(n+1),i}}, \tag{3.63}$$

$$\mathbf{S_{(n+1),i}} = \frac{1}{n_{(n+1),i} - 1} \left(s^2_{(n+1),i} - \frac{1}{n_{(n+1),i}} s_{(n+1),i} s^T_{(n+1),i} \right) \tag{3.64}$$

where $p_{(n+1),i}$ is an estimate for the mixture weight π_i.

If, when presenting a data vector, it is necessary to create a new component, then it means that this data vector is not statistically significant to all m components. So, a new component $m+1$ is created and the likelihood of the new data x_n will be $L_{m+1} = 1$ and, by definition, $L_i = 0,\ \forall i = 1, 2, \ldots, m$. So, adapting Equation (3.58), the result is shown on Equation (3.65).

$$w_{m+1} = \frac{L_{m+1}}{\sum_{i=1}^{m+1} L_i} = \frac{L_{m+1}}{L_1 + \ldots + L_m + L_{m+1}} = \frac{L_{m+1}}{0 + \ldots + 0 + L_{m+1}} = 1 \quad (3.65)$$

This is the learning mode of the CIGMN and, as said before, can be done perpetually. At some point, it is necessary to address the problem of creating too many components. This can be done with a multivariate Gaussian distribution test. It is, however, not presented here because in the used series it is not an issue. For this test, the CIGMN must not be in learning mode and neither in the recalling mode, but which we call an stand-by mode, as the process for the multivariate Gaussian test is computationally costly. Nevertheless, with the s_n and s_n^2 stored for each component, even if some components are removed, this information can be reallocated on other components, loosing no information whatsoever.

3.5.2.2 Recalling Mode

CIGMN enters in the recalling mode whenever the new data $\mathbf{x}_n$ is not complete. If any, but not all, elements are missing, the CIGMN initiates the recalling mode to estimate the missing elements. As stated before, the recalling mode can be described as a linear combination of regression models. The CIGMN allows three different linear combinations using the likelihood L_i of the non-missing data, the mixture weight p_i of the components and a combination of both.

Before further presentations, let us assume the following covariance matrix representation which is similar to the one shown on Equation (3.53).

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{k,k} & \mathbf{M}_{k,l} \\ \mathbf{M}_{l,k} & \mathbf{M}_{l,l} \end{bmatrix} \quad (3.66)$$

Equation (3.66) is used as a model, showing four submatrices, $\mathbf{M}_{k,k}$, $\mathbf{M}_{k,l}$, $\mathbf{M}_{l,k}$ and $\mathbf{M}_{l,l}$. The $\mathbf{M}_{k,k}$ is a submatrix about the known k elements. The matrices $\mathbf{M}_{k,l}$ and $\mathbf{M}_{l,k}$ are submatrices about the interaction between the k known elements and the l missing elements. Finally, the $\mathbf{M}_{l,l}$ is about the l missing elements. This four submatrices will be used to better show how to obtain the regression $\hat{\boldsymbol{\beta}}$ and the recalling likelihood, using a reduced $\mathbf{S}$.

So, assuming that $\mathbf{Y}_l$ elements are missing, $l < k$, the first step is to obtain all the $k - l$ estimated regression $\hat{\boldsymbol{\beta}}$, as shown on Equation (3.67).

$$\hat{\boldsymbol{\beta}}_i = (\mathbf{X}_i^T \mathbf{X}_i)^{-1} (\mathbf{X}_i^T \mathbf{Y}_i), \quad \forall i = 1, 2, 3, \ldots, m. \quad (3.67)$$

where $(\mathbf{X}_i^T \mathbf{X}_i)$ represents the $\mathbf{M}_{k,k}$ submatrix of the outer product matrix $\mathbf{M}$ for each component and $(\mathbf{X}_i^T \mathbf{Y}_i)$ represents the $\mathbf{M}_{k,l}$ submatrix of the outer product matrix $\mathbf{M}$. As shown before, on Equation (3.53), every component stores the outer product and, when one or more values are missing, it is used to obtain the regression $\hat{\boldsymbol{\beta}}$.

The second step is to compute the recalling likelihood Lr for each component, as shown on Equation (3.68).

$$Lr_i = \frac{1}{\sqrt{|\mathbf{S}^*_{\mathbf{n,i}}|}} exp\left(-\frac{1}{2}M_i^{*2}\right), \forall i = 1, 2, 3, \ldots, m \tag{3.68}$$

where $\mathbf{S}^*_{\mathbf{n,i}}$ is the $\mathbf{M}_{k,k}$ submatrix for each one of the $\mathbf{S}_{\mathbf{n,i}}$ matrices and M_i^{*2} is the quadratic Mahalanobis' distance for each one of the components, as shown on Equation (3.56), using only the known elements $\bar{x}$ and the $\mathbf{S}_{\mathbf{n,i}}$, as shown on Equation (3.69).

$$M_i^2 = (\mathbf{x}_k - \bar{\mathbf{x}}_{i,k})^T {\mathbf{S}^*_{\mathbf{n,i}}}^{-1} (\mathbf{x}_k - \bar{\mathbf{x}}_{i,k}), \quad \forall i = 1, 2, 3, \ldots, m \tag{3.69}$$

where $\mathbf{x}_k$ represent the known elements and $\bar{\mathbf{x}}_{i,k}$ the mean of the known elements.

With all the regression $\hat{\boldsymbol{\beta}}$ estimated, the estimate for the missing values can be made in many different ways. In this work, we use three approaches: the weighted likelihood, Lr_i^*, the mixture weights p_i and a weighted combination of the recalling likelihood and the mixture weights, U_i. To use the weighted likelihood approach, first we need to compute the weighted recalling likelihood, as shown on Equation (3.70) and then using Lr_i^* to weight all the regressions, as shown on Equation (3.71). The first step is made to guarantee that $\sum_{i=1}^{m} Lr_i^* = 1$. However this approach does not use the information on the mixture weights p_i and it is suitable for systems with small $n_{n,i}$, but significant, components. Some of the used series may fall in this category.

$$Lr_i^* = \frac{Lr_i}{\sum_{i=1}^{m} Lr_i} \tag{3.70}$$

$$\hat{\mathbf{Y}} = \sum_{i=1}^{m} Lr_i^* \mathbf{X} \boldsymbol{\beta}_i \tag{3.71}$$

The simplistic approach is to use only the mixture weights computed using Equation (3.62). This approach is more appropriate on components with small $n_{n,i}$ or components that are largely distinct, and is shown on Equation (3.72)

$$\hat{\mathbf{Y}} = \sum_{i=1}^{m} p_i \mathbf{X} \boldsymbol{\beta}_i \tag{3.72}$$

Another approach is to combine the information on the mixture weights p_i with the recalling likelihood Lr_i. In this approach we first need to compute the weighted combination between the mixture weights p_i and the recalling likelihood Lr_i, which we call U_i, as shown on Equation (3.73). And then, as was done before, weight the estimates for all the components, as shown on Equation (3.74). It is important to notice that these approaches have, for the most cases, very similar results, but only the best results are presented in the Experiments section.

$$U_i = \frac{Lr_i p_i}{\sum_{i=1}^{m} Lr_i p_i} \tag{3.73}$$

$$\hat{\mathbf{Y}} = \sum_{i=1}^{m} U_i \mathbf{X} \boldsymbol{\beta}_i \tag{3.74}$$

Algorithm (1) shows the learning mode of the CIGMN. Algorithm (2) shows how the CIGMN creates a new component, when necessary. Algorithm (3) shows how the CIGMN updates its components and finally, Algorithm (4) shows the CIGMN recalling mode. And Table 3.1 presents a brief comparison between the original IGMN and the CIGMN.

```
Data: k-dimensional vector $x$
Result: CIGMN Model
initialization;
for $i = 1$ to $m$ do
    Obtain the Quadradtic Mahalanobis' Distance $M_i^2$ - Equation (3.56) ;
    Obtain the likelihood $L_i$ - Equation (3.57) ;
end
if $M_i^2 > \chi_k^2 \quad \forall i = 1 : m$ then
    Create a new component - Algorithm 2;
else
    Update the existing components - Algorithm 3;
end
```

Algorithm 1: Pseudo-code for the CIGMN Learning mode

```
Data: k-dimensional vector $x$ and $L_i$
Result: CIGMN Model with $m + 1$ components
begin
    CIGMN$sumx2_{(m+1)} \leftarrow x'x$ ;
    CIGMN$sumx_{(m+1)} \leftarrow x$ ;
    CIGMN$n_{(m+1)} \leftarrow 1$ ;
    CIGMN$mean_{(m+1)} \leftarrow x$ ;
    CIGMN$varcov_{(m+1)} \leftarrow diag(x'x)$ ;
end
```

Algorithm 2: Pseudo-code to create a new component on CIGMN Learning mode

Differences between IGMN and CIGMN		
	Original IGMN	Classic IGMN
Updates	Uses bayesian approach to update the necessary values, based on prior and posterior distributions.	Uses a classical approach to update the mean vector, covariance matrix and the components weights.
Creation of a new component	Based on values τ_{min} and τ_{max} to reduce overfitting.	Uses the Mahalanobis distance with a χ^2-test.
Inference	Also made using bayesian approach, with priors and posteriors well defined.	Based on the OLS estimation, combined with the components weights.
Parameter estimation	Uses the EM algorithm.	Based on maximum likelihood estimation and OLS estimation.

Table 3.1: A comparative table for some different characteristics between the original IGMN and the Classical IGMN.

```
Data: k-dimensional vector $x$ and $L_i$
Result: CIGMN Model with updated components
initialization ;
Obtain all $w_i$ - Equation (3.58) ;
for $i = 1$ to $m$ do
    Update the outer product $s_i^2$ - Equation (3.60) ;
    Update the summation $s_i$ - Equation (3.59) ;
    Update the $n_i$ - Equation (3.61) ;
    Update the mixture weight $p_i$ - Equation (3.62) ;
    Update the mean $\bar{x}_i$ - Equation (3.63) ;
    Update the variance-covariance matrix $\mathbf{S}_i$ - Equation (3.64) ;
end
```

Algorithm 3: Pseudo-code to update existing components on CIGMN Learning mode

```
Data: k-dimensional vector $x$ with missing data
Result: Estimate of the missing data, $\hat{Y}$
initialization ;
Obtain all $\hat{\beta}_i$ - Equation (3.67) ;
Obtain all $Lr_i$ - Equation (3.68) ;
Obtain all $U_i$ - Equation (3.73) ;
Estimate $\hat{Y}$ accordingly - Equation (3.71) or (3.74) ;
```

Algorithm 4: Pseudo-code to estimate missing data using an CIGMN model

3.5.3 Modeling the Moving Average (MA) component

First, assume that a time series presents the moving average (MA) component where, as shown before, $\boldsymbol{\theta}$ represents the MA parameters, $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_q\}$. The actual and future errors are unknown and random errors, but the past errors can be estimated. As shown in works of (ENDERS, 1995) and (HAMILTON, 1994), the $\boldsymbol{\theta}$ can be estimated via a sequential method. First, by definition, we assume that

$$\epsilon_0 = 0$$

and follows that

$$\begin{aligned}
\epsilon_1 &= y_1 \\
\epsilon_2 &= y_2 - \theta_1\epsilon_1 = y_2 - \theta_1 y_1 \\
\epsilon_3 &= y_3 - \theta_1\epsilon_1 - \theta_2\epsilon_2 = y_3 - \theta_2 y_2 - \theta_1 y_1
\end{aligned}$$

and so on. In general, we have

$$\epsilon_t = y_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - \ldots - \theta_q\epsilon_{t-q} = y_t - \sum_{i=1}^{q}\theta_i\epsilon_{t-i} \tag{3.75}$$

where q is the q-th MA order, i.e., MA(q).

The procedure presented in Equation (3.75) is done sequentially, with the initial value for $\boldsymbol{\theta}$ defined by the user, as a $\hat{\boldsymbol{\theta}}_{init}$. The sequential procedure used to estimate $\boldsymbol{\theta}$, or simply $\hat{\boldsymbol{\theta}}$ corresponds to the same method we use to estimate the recalling $\hat{\boldsymbol{\beta}}$, as seen on Equation (3.52). However, with the MA modification, the error vector $\boldsymbol{\epsilon}$ must be reconstructed for every new data, i.e., for each step. Assuming that, by definition, the first estimate $\hat{\epsilon}_0 = 0$ or simply $e_0 = 0$ and that $e_1 = x_1$, then, all the others q elements are also, sequentially computed, as shown on Equation (3.76).

$$e_n = x_n - \sum_{i=1}^{q}\hat{\theta}_{i,(n-1)}e_{i,(n-1)} \tag{3.76}$$

Being e_n dependent on observed data x_n and also $\hat{\theta}_i$, for every new data, the error vector $\mathbf{e}$ must be reconstructed with the actual $\hat{\boldsymbol{\theta}}$, estimated in the same way the $\hat{\boldsymbol{\beta}}$ were, as in Equation (3.52). So, for now, we have $\hat{\theta}_{(i,n)}$ where n is the sample size or, in this time series scenario, the available time patterns. For simplicity we use a common e for all components, using the last L_i as a weighted mean, as shown on Equation (3.77).

$$e_n = \sum_{i=1}^{m} L^*_{n,i}e_{n,i} \tag{3.77}$$

where $L^*_{i,n}$ is the weighted $L_{n,i}$ as shown on Equation (3.78).

$$L^*_{n,i} = \frac{L_{n,i}}{\sum_{i=1}^{m} L_{n,i}} \tag{3.78}$$

For example, let us assume an MA(1) process. When the first data vector arrives, the CIGMN uses the above definition and sets the $\hat{\boldsymbol{\epsilon}}$ or simply e at zero (0). So, the first input data that the CIGMN model receives is

$$[e_0 \quad x_1] = [0 \quad x_1]$$

instead of

$$[x_1 \quad x_2]$$

Using the estimates e, we reconstruct the time series and improve our $\hat{\boldsymbol{\theta}}$ estimate,

$$\begin{aligned}
&\epsilon_2 = y_2 - \hat{\theta}_{1,1}\epsilon_1 = y_2 - \hat{\theta}_{1,1}y_1 \\
&\epsilon_3 = y_3 - \hat{\theta}_{1,2}\epsilon_2 \\
&\epsilon_4 = y_4 - \hat{\theta}_{1,3}\epsilon_3 \\
&\ldots \\
&\epsilon_t = y_t - \hat{\theta}_{1,(t-1)}\epsilon_{t-1}
\end{aligned}$$

where $\hat{\theta}_{1,(t-1)}$ represents the $\hat{\theta}_1$ estimate at t-th step. The difference in the MA process is that the estimates are used at every new data, instead of using only when there are missing values. This, however, introduces a new step in the used algorithms. At the initial step of the process, the steps described in Algorithm 5 are used.

```
Data: $k$-dimensional vector $x$ and $\beta_0$
Result: $(k + q)$-dimensional vector
begin
    Set $q$-dimensional $e$ vector with zeros $0$ ;
    Use $\beta_0$ to obtain the first $min(q, k)$ errors ;
    Bind the $e$ vector with the $x$ vector ;
    Uses the new $x^*$ as the input vector ;
end
```

Algorithm 5: Pseudo-code of the initial steps necessary for the MA parameters estimate

After the initial step (used only with the first input data), the steps on Algorithm 6 shows how the MA $\hat{\boldsymbol{\theta}}$ updates.

```
Data: $k$-dimensional vector $x$ and $\hat{\theta}_{T-1}$
Result: $(k + q)$-dimensional vector and $\hat{\theta}_T$
begin
    Recover $q$-dimensional $e_{T-1}$ vector ;
    Updates $\theta_{T-1}$ using the $x^*$ vector ;
    Updates $e_{T-1}$ ;
end
```

Algorithm 6: Pseudo-code of the initial steps necessary for the MA parameters estimate

With these definitions, the CIGMN model with the MA process capabilities is called an ARMA-CIGMN(p,q) model, where p represents the AR order and q the MA order of the model.

4 EXPERIMENTS AND RESULTS

4.1 Initial Comments

In this chapter we propose, design and analyze four different types of experiments, based on three different real time series and three simulated models. The models used for comparisons are (i) the ARIMA family, (ii) the original IGMN, (iii) the ARMA-like, (iv) the NOE-like and (v) the ARMA-CIGMN models. We used well known time series, chosen for their characteristics: (i) the air passengers data, (ii) the monthly sunspots and (iii) the Canadian lynx data. The simulated series were an MA(1) model, a pure MA process of order 1, with different parameters, an ARIMA(1,1,1) and an ARMA(1,1), using the same parametrization as the simulated MA(1) series.

In the first experiment we use the air passengers data and the sunspot data to show the advantages of using acf and pacf graphs, the issue with MA components and compare the original IGMN models, the ARIMA model and the ARMA-CIGMN model. With the exception of the ARIMA model, the other models are compared with different configurations using these two data series.

The second experiment compares the ARMA-like, the NOE-like model, the original IGMN and the ARMA-CIGMN models, using the air passengers data. This experiment was designed to show how the different models compare with a nonstationary series that presents MA components.

The third experiment is more focused on the MA components and the integration coefficient. In this case, MA(1), ARIMA(1,1,1) and ARMA(1,1) models were simulated. The models used for comparison for the MA(1) series were: a classical MA(1), an ARMA-like, an original IGMN using bias and the ARMA-CIGMN. For the ARIMA(1,1,1) and the ARMA(1,1) series, we only used the classical and the ARMA-CIGMN models. In this experiment, one-step forecasts were made, to better verify the capabilities of each model for all the simulated models.

At last, we use the Canadian lynx series to demonstrate that the ARMA-CIGMN models the MA components and, also, to show its forecasting capabilities. In this experiment, we choose some of the smallest forecasting errors models to compare with the ARMA-CIGMN model.

4.2 Presentation of the real time series data

Here we present the real time series data. These three series are widely used because some of their characteristics like nonstationarity, nonlinearity among others.

4.2.1 The air passengers data

The air passengers time series represents a total of 144 observations about the monthly total of international airline passengers between 1949 and 1960, in thousands, as seen in Figure 4.1.

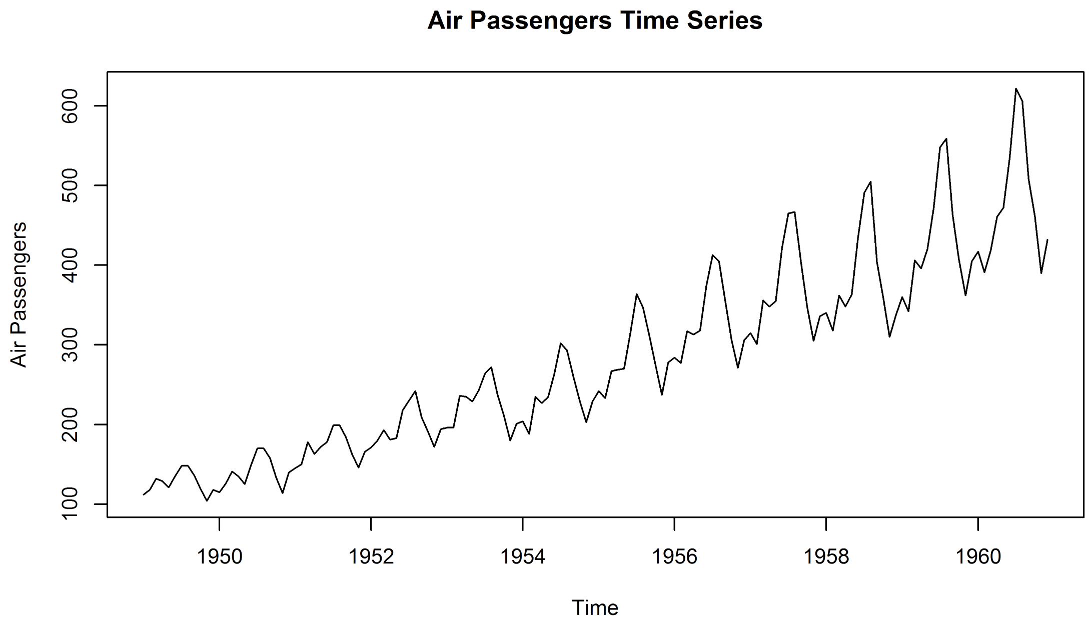


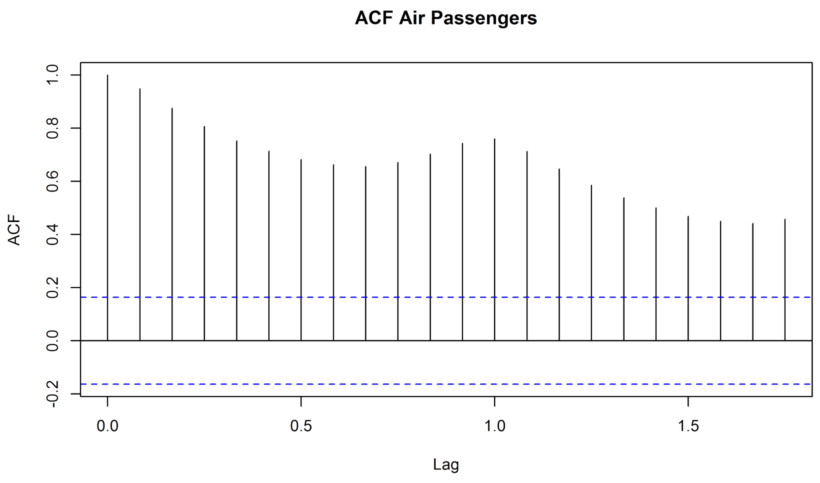
Figure 4.1: The air passengers time series data

This particular series can be modelled in the classical statistical approach with a reasonably simple model, a SARIMA(1,0,1)x(0,1,1)$_{12}$, even being considered a nonlinear time series. The majority of ANN models applied to this series uses a 12-lag window or 24-lag window (PINTO, 2011). Figure 4.2 shows the acf and pacf of the series. Observing the pacf graph, one can notice the statistically significant lags that will be used in the IGMN models. The acf graph shows only a non stationary time series, this is why the integration coefficients in the SARIMA model are commonly used. This is also characteristic for a nonlinear time series.

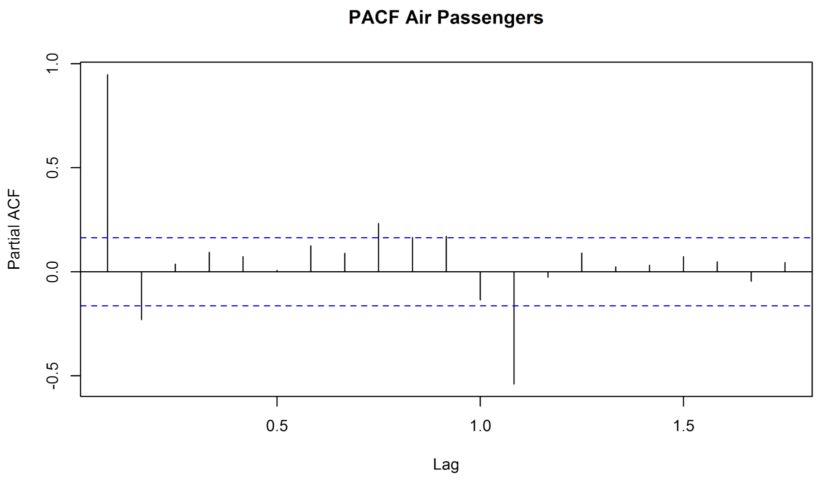
4.2.2 The sunspots data

The sunspots time series represents the monthly mean relative sunspot numbers from 1749 to 1983. This series is commonly used to verify the capabilities of different models, as it is considered a nonlinear time series (MAKRIDAKIS; WHEELWRIGHT; HYNDMAN, 1998).

The classical statistical models to this series range from an AR(9) to more complex models. Usually, the ANN models applied to this series have a 12-lag window or even a wider one (PINTO, 2011). Observing the graphs on Figure 4.4 one can see that the acf graph is similar to the air passengers data, but the pacf shows that the statistically significant lags are on the beginning of the series and after a long period.



(a)



(b)

Figure 4.2: The acf(a) and pacf(b) of the air passengers data

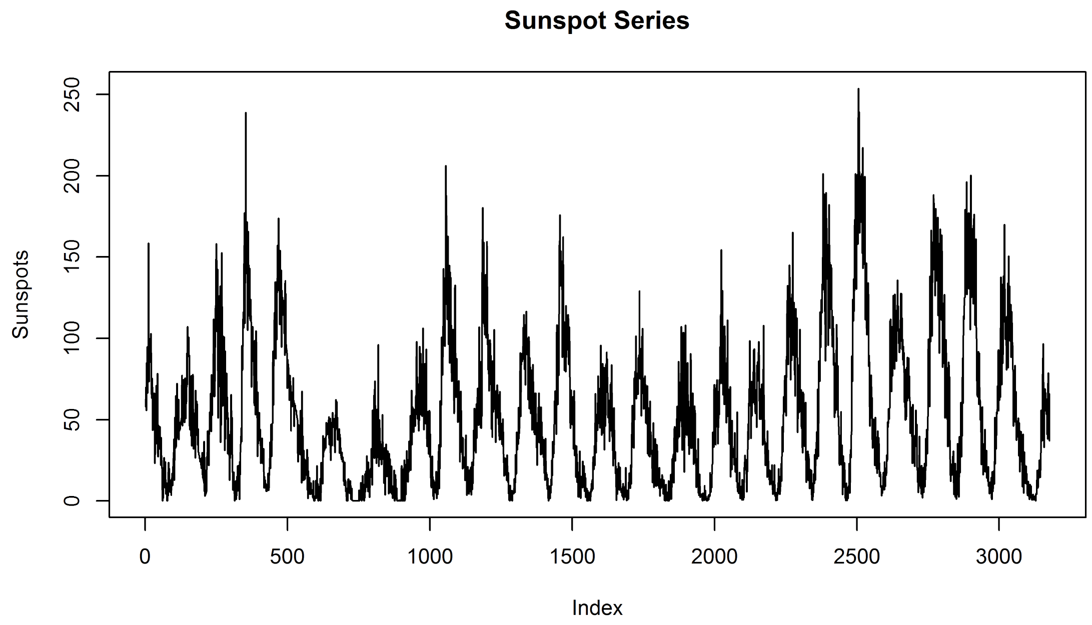


Figure 4.3: The monthly sunspots numbers time series data

4.2.3 The Canadian lynx series

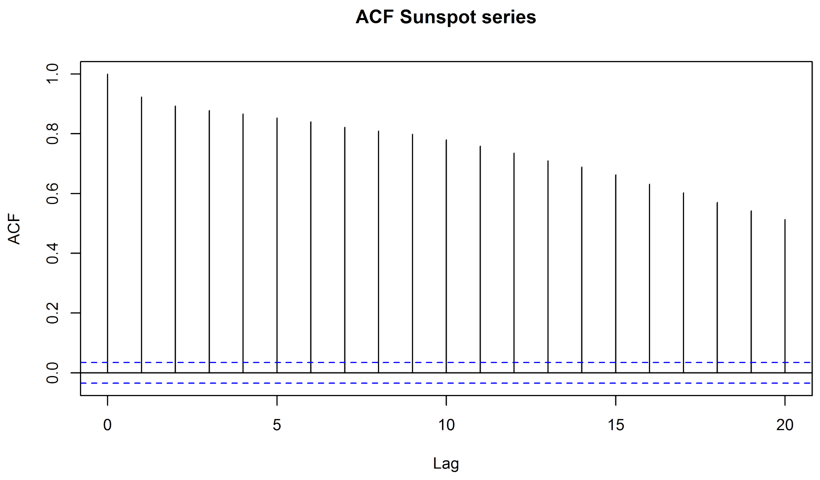
This is an annual record of Canadian lynx fur sold from the London archives and represents the amount of lynx trapped for each year in Northern Canada. The series has 114 values beginning in 1821 and ending in 1934. Figure 4.5 presents the series.

This series was treated as a log_{10} of the data, based on (KHASHEI; BIJARI, 2010) and, also, for comparison with other models. Figure 4.6 presents the acf and pacf graphs of this series, which uses a classical AR(12) as a reference classical model. The series presents a nonlinear behavior and an approximately 10 year periodicity.

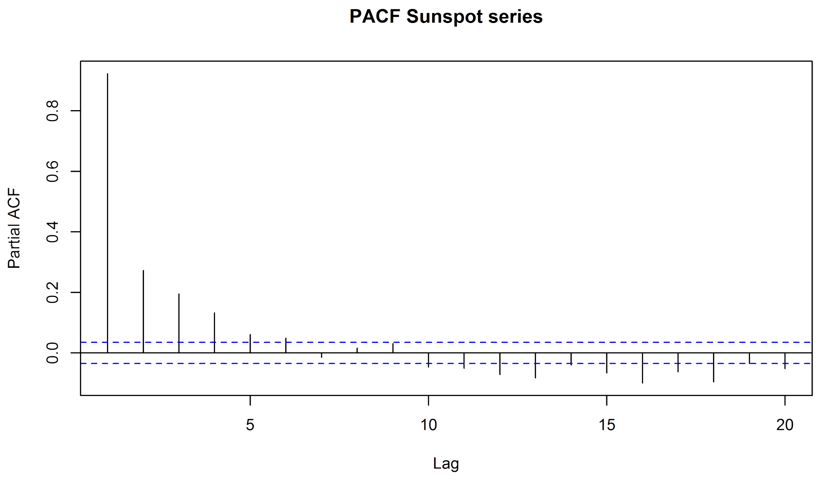
Observing the graphs on Figure 4.6 one can see that the acf graph presents a variable seasonality, as the peak values occur on lags 5, 14, 24 and so on. The pacf only indicates that the model should present an MA component.

4.3 Modeling using acf and pacf

The acf and pacf functions are used in two different moments. In the beginning of the modeling, to compute the input representative lag window, or the representative time-delays, and after the modeling to analyze the residuals. The residuals of any given model (linear or nonlinear) must be uncorrelated. This is the same method used in the classical statistical modeling. The only remark is that the original air passengers data are used, not the differential, integrated or log data. For the purpose of these experiments, we try to use nonlinear time series and that is one of the reasons to use the original data, namely to test the capabilities of the acf and pacf function as tools to improve the performance of the IGMN model. The other major reason is to be able to compare both models. The same is done with the sunspot series. After modeling using the acf and pacf functions, IGMN models are generated, as the traditional method to compare with, using the whole lag window, or time-delays. We also used the classical SARIMA model to present the results with an observation: the SARIMA model uses the differentiated series, as presented



(a)



(b)

Figure 4.4: The acf(a) and pacf(b) of the sunspots data

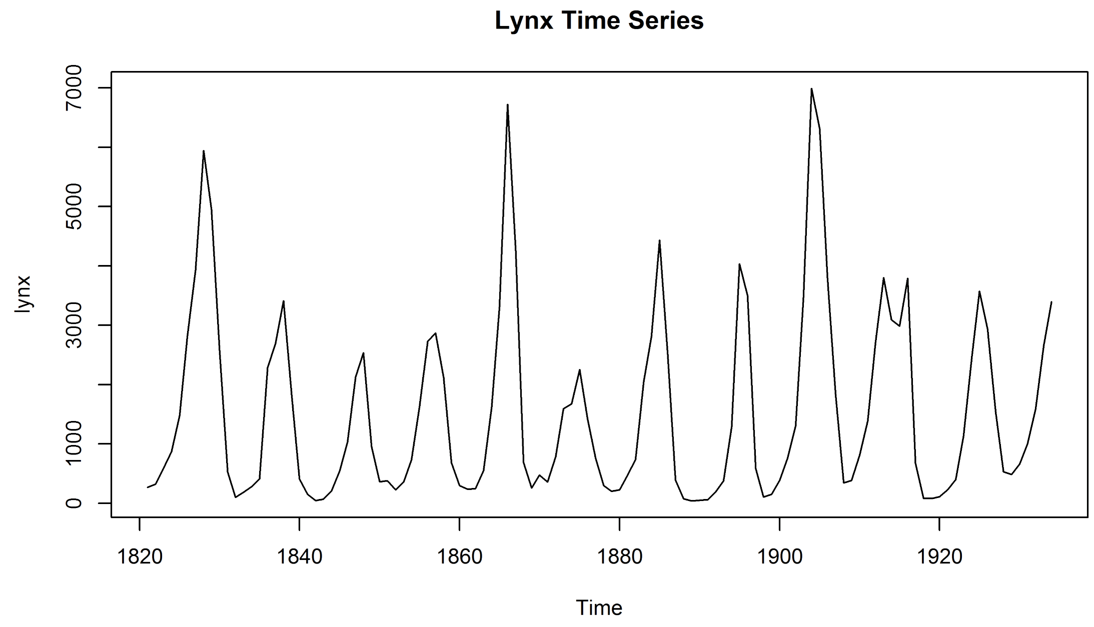


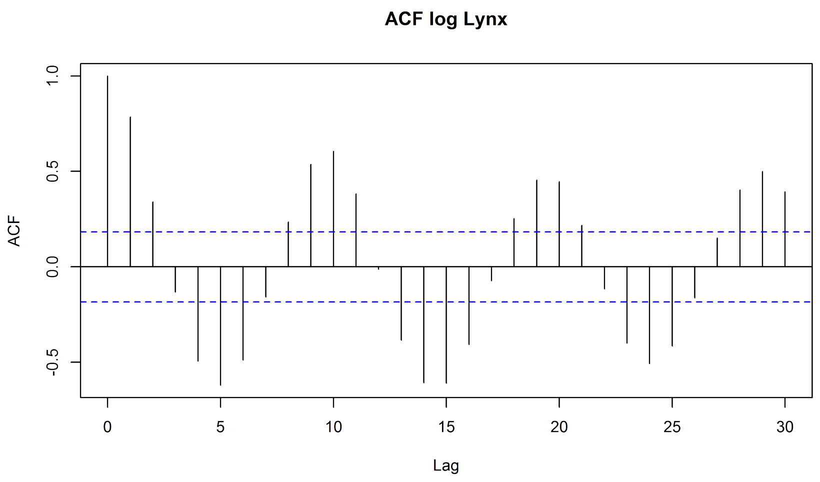
Figure 4.5: The lynx time series

before. And finally the proposed ARMA-CIGMN model.

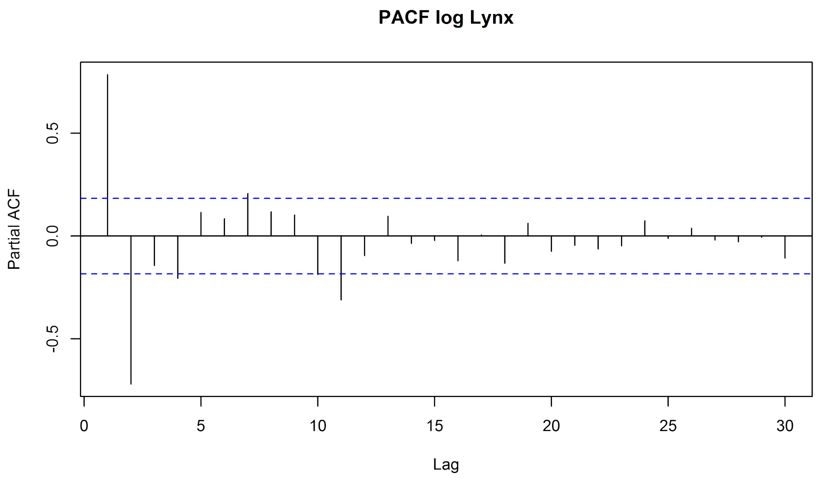
The lag window or time-delay representation used in this work is as follows: a lag window of 1 is the input of $x_{(t)}$ to forecast $x_{(t+1)}$, a lag window of 2 is the input of $x_{(t-1)}$ and $x_{(t)}$ to forecast $x_{(t+1)}$ and so on. Using the acf and pacf we apply the term representative lag window because the model does not use the whole lag window, but just some elements of it, the representative lags. A model can be described as using the $x_{(t-6)}$, $x_{(t-1)}$ and $x_{(t)}$ to forecast $x_{(t+1)}$. This is not a lag window of 7, which is described by $x_{(t-6)}$, $x_{(t-5)}$, ..., $x_{(t-1)}$ and $x_{(t)}$ to forecast $x_{(t+1)}$. To assess the contribution of the acf and pacf functions to produce more compact models, two models with distinct input-layers were presented for the IGMN. One of them has a compact input-layer corresponding to the representative lag window with the delays determined using the acf and pacf functions. The other one has an input-layer corresponding to the proper (whole) lag window, as seen above.

The results for the trained models will be presented with the following measures: root mean square error (RMSE) as in (4.1) and the normalized mean square error with the trivial solution (NMSET) as in (4.2). The trivial solution for forecasts represents the use of the last value x_t as a forecast for the future value x_{t+1}, or simply that the last observed value is the best predictor for the future value. These results are computed using only the forecasts, not the whole series. In both series the last observations will be removed for forecasting and to compute the RMSE and NMSET. The forecast corresponds to an one-step horizon.

$$RMSE = \sqrt{\sum_{t=1}^{n} \frac{(x_{(t)} - \hat{x}_{(t)})^2}{n}} \tag{4.1}$$



(a)



(b)

Figure 4.6: The acf(a) and pacf(b) of the lynx series

$$NMSET = \frac{\sum_{t=2}^{n}(x_{(t)} - \hat{x}_{(t)})^2}{\sum_{t=2}^{n}(x_{(t-1)} - x_{(t)})^2} \tag{4.2}$$

where $x_{(t)}$ is the real observation at time t and $\hat{x}_{(t)}$ is the predicted value at time t.

The first series to be modeled corresponds to the air passengers data. Two different original IGMN models were computed: a model with a lag window of 14 observations, i.e., $x_{(t)}, x_{(t-1)}, \ldots, x_{(t-13)}, x_{(t-14)}$ and a model with only the lags $x_{(t)}, x_{(t-12)}$ and $x_{(t-14)}$. We also used the classical SARIMA model and three different configurations of the ARMA-CIGMN. The first one, using the same lag-window as the original IGMN with significant lags, except for an MA component. The second one using the same lag-window of 14 values as the original IGMN, with additional 12 MA components. At last, just to present the MA improvement, the same model with a lag-window of 14 values but without the MA components. The last 30 observations were used to verify the RMSE and NMSET. The results are shown in Table 4.1.

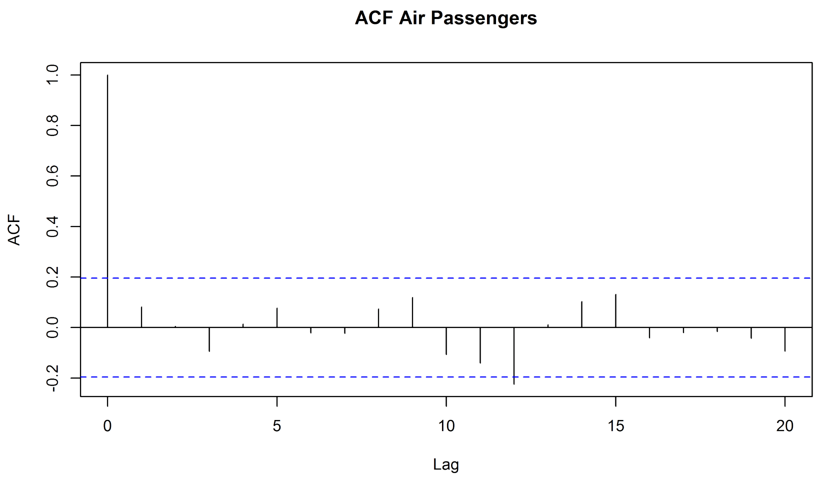
Table 4.1: The air passengers forecasts results. * The SARIMA model does not present clusters.

Models	Clusters	RMSE	NMSET
IGMN $x_{(t)}, x_{(t-12)}$ and $x_{(t-14)}$	1	18.2622	0.1205
IGMN $x_{(t)}, \ldots, x_{(t-14)}$	9	47.2595	0.8069
ARMA-CIGMN(3,1)	4	18.3052	0.1258
ARMA-CIGMN(14,12)	1	14.2494	0.0762
ARMA-CIGMN(14,0)	1	15.9670	0.0957
SARIMA(1,0,1)x(0,1,1)$_{12}$	*	17.3773	0.1134

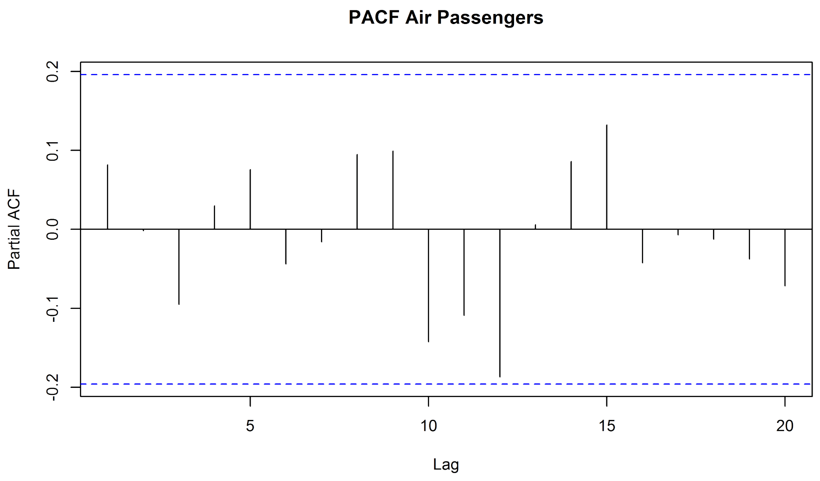
Table 4.1 presents a better performance of the IGMN model that uses the information of the acf and pacf graphs, comparing with the IGMN that uses the entire lag-window. The IGMN model with the significant lags, $x_{(t)}, x_{(t-12)}$ and $x_{(t-14)}$ has 1 cluster instead of 9 and a RMSE of 18.3 and a NMSET of 0.12. The ARMA-CIGMN model using only significant lags achieves a similar, yet slightly greater RMSE and NMSET results, than the one without the MA component. This can be caused by the inclusion of the MA component in the modeling process that may alter the significant lags. The main comparison is made between the classical SARIMA and the 14 lag-window ARMA-CIGMN models with and without the MA component. The classical SARIMA model surpasses the results obtained by all the original IGMN models, and also the ARMA-CIGMN(3,1) with an NMSET of 0.11 and a RMSE of 17.38. The ARMA-CIGMN(14,12) obtained a RMSE of 14.25 and a NMSET of 0.08. The same model without the MA component has slightly greater NMSET and RMSE.

Figure 4.7 presents the final acf and pacf, using the residuals of the model with the lowest NMSET and RMSE in Table 4.1, the ARMA-CIGMN(14,12).

Observing the acf and pacf in Figure 4.7, one can easily see that only on the acf graph, in the lag 12, there is a value over the limit by a small amount. To better show how the MA components of the ARMA-CIGMN may solve the moving average problem, Figure 4.8 shows the acf and pacf for the ARMA-CIGMN model with the same lag-window but without the MA components. In this case, the value at lag 12 for both graphs are significant, indicating an MA(12) component.

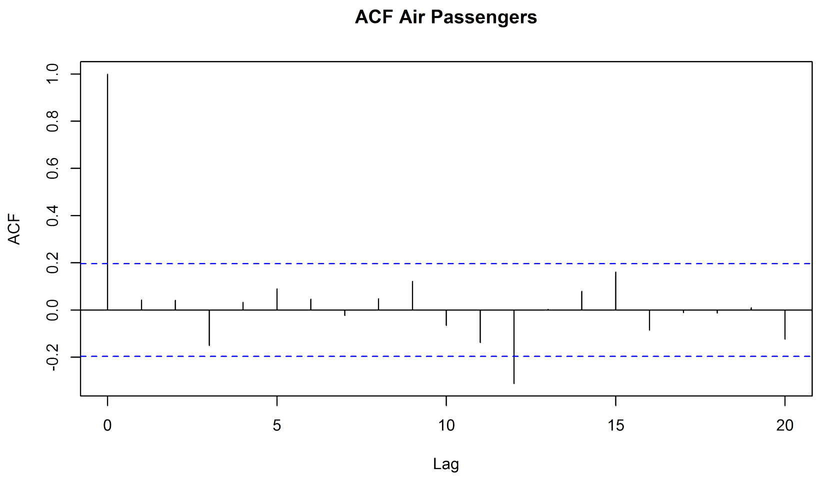


(a)

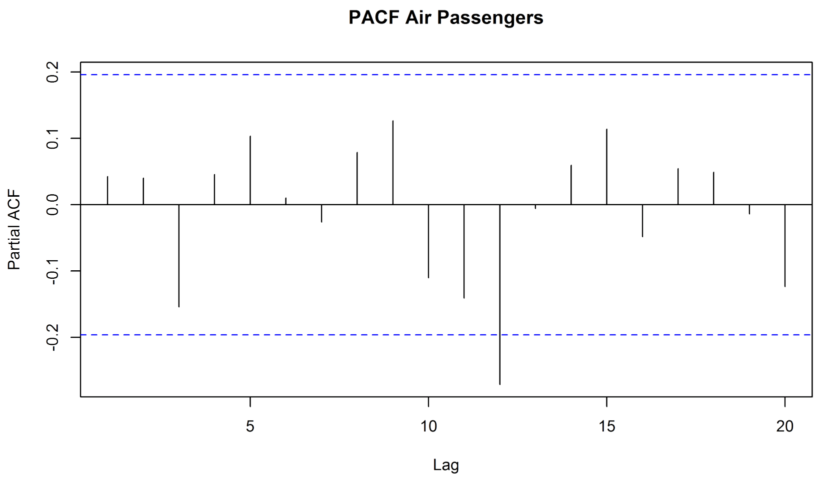


(b)

Figure 4.7: The final acf(a) and pacf(b) of the air passengers data



(a)



(b)

Figure 4.8: The acf(a) and pacf(b) of the air passengers data with the ARMA-CIGMN model without the MA components

The other series analyzed corresponds to the sunspots data. The same steps that were followed in the air passengers data are repeated here. The used models are an IGMN with a lag window of 6 steps, i.e., $x_{(t)}, x_{(t-1)}, \dots, x_{(t-5)}, x_{(t-6)}$, an IGMN with only the significant lags $x_{(t)}, x_{(t-3)}$ and $x_{(t-6)}$, the ARMA-CIGMN using only the significant lags and 2 MA components (ARMA-CIGMN(3,2)), the ARMA-CIGMN using the 6 steps lag-window and also 2 MA components (IGMN-ARMA(6,2)) and the ARMA-CIGMN with the 6 steps lag-windows and without any MA component (ARMA-CIGMN(6,0)). The last 93 observations were used to verify the RMSE and NMSET. The results are shown in Table 4.2.

Table 4.2: The monthly sunspots forecasts results

Models	Clusters	RMSE	NMSET
IGMN $x_{(t)}, x_{(t-3)}$ and $x_{(t-6)}$	11	20.0275	0.9169
IGMN $x_{(t)}, \dots, x_{(t-6)}$	12	22.4864	1.1558
ARMA-CIGMN(3,2)	8	10.1612	0.9322
ARMA-CIGMN(6,2)	10	10.3287	0.9632
ARMA-CIGMN(6,0)	17	10.6282	1.0198

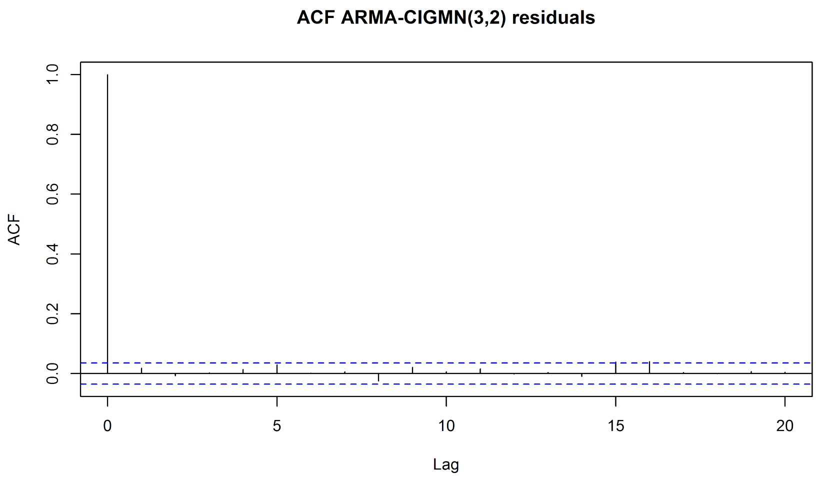
The IGMN model with the significant lags produced smaller NMSET errors than any other model, but in respect to the RMSE, all the ARMA-CIGMN models presented better results, as one can see in Table 4.2. It is important to note that, again, the ARMA-CIGMN model without the MA components has worst results than the model with the MA components. In the sunspot series, the model using only significant lags has the lowest RMSE, 10.1612, and the second best NMSET of 0.9322. Figure 4.9 shows the acf and pacf graphs for the ARMA-CIGMN(3,2) model with the significant lags.

Both the acf an the pacf graphs in Figure 4.9 seem well adjusted, with the exception of two lags on the pacf graph, around lag 15. This reinforces that the ARMA-CIGMN can model a moving average process.

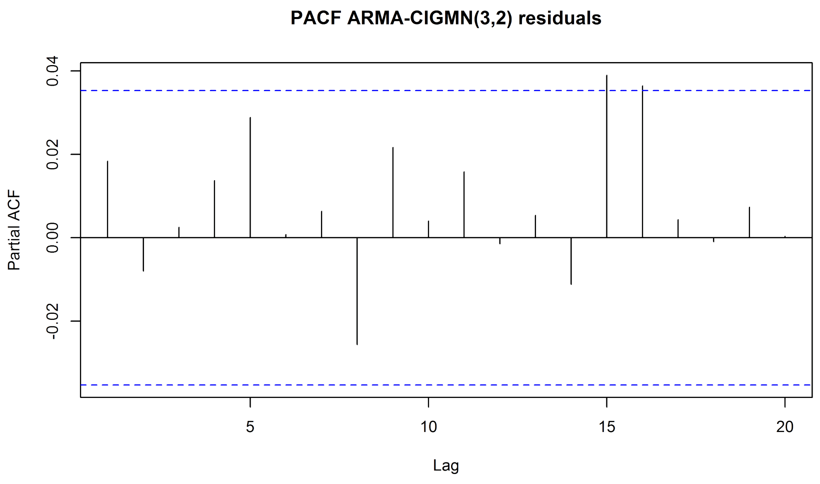
4.4 Comparison among different configurations for the IGMN

As presented before, the modeling starts with the initial acf and pacf of the air passengers series. Figure 4.2 presents the initial acf and pacf of the data. From here on we take different approaches, according to the used configuration. The AR configuration uses the same input configuration as (PINTO, 2011), for comparison. In this model, no other analysis was made. It uses a lag-window of 17 elements. In the significant lags configuration, we used the same model as seen in (FLORES; PINTO; ENGEL, 2012). This configuration, after the acf and pacf analysis, uses the $x_{(t)}, x_{(t-12)}$ and $x_{(t-14)}$ to forecast $x_{(t+1)}$. The ARMA-like model uses the past errors to improve the forecasts and it is based on almost the same structure as the relevant lags model.

In the ARMA-like model, the data used was the $x_{(t)}, x_{(t-12)}, x_{(t-14)}$ and also the $\epsilon_{(t-3)}, \epsilon_{(t-6)}, \epsilon_{(t-10)}$, computed over the forecasts and the real values. It is important to note, however, that this is only possible using an initial buffer on the input of the IGMN. This same method was used in the fourth different configuration of the input layer, using the forecasts values as input, the NOE-like model. It uses the same time lags as the ARMA-like model configuration, but using the past predicted values instead of the past



(a)



(b)

Figure 4.9: The final acf(a) and pacf(b) of the sunspots data using the ARMA-CIGMN(3,2) model

errors, i.e., $x_{(t)}, x_{(t-12)}, x_{(t-14)}$ and also the $\hat{x}_{(t-3)}, \hat{x}_{(t-6)}, \hat{x}_{(t-10)}$ to predict $x_{(t+1)}$.

The last presented models are based on the ARMA-CIGMN, and we use the same configurations as before. The first one, an ARMA-CIGMN(3,1), uses $x_{(t)}, x_{(t-12)}, x_{(t-14)}$ and a MA component e_t to forecast $x_{(t+1)}$. The second one uses the complete lag-window, going from $x_{(t)}$, $x_{(t-1)}$, . . ., $x_{(t-14)}$ to forecast $x_{(t+1)}$, which is similar to the one presented in (PINTO, 2011), but 3 lags shorter. And finally, the third one has the same lag-window as the second one plus the 12 lag-window MA component, e_t, $e_{(t-1)}$, . . ., $e_{(t-12)}$.

The generated models will be compared separately using the same parameters, with the exception of the number and content of the input layer. The only remark is that the original air passengers data are used, not the differential, integrated or log data. As stated before, the time series should be nonlinear and that is one of the reasons to use the original data, namely to test the capabilities of each model.

The results for the trained models will be presented with the following measures: root mean square error (RMSE) as in (4.1) and the normalized mean square error with the trivial solution (NMSET) as in (4.2). These results are computed using only the forecasts, not the whole series. As before, the last 30 observations will be removed to forecasting and compute the RMSE and NMSET. The forecast corresponds to an one-step horizon.

After modeling and forecasting the series, Table 4.3 presents the RMSE and NMSET of the different used models.

Table 4.3: Results of the different IGMN models on the air passengers data

Models	Clusters	RMSE	NMSET
IGMN-AR(17)	1	16.8230	0.1064
IGMN-$x_{(t)}, x_{(t-12)}, x_{(t-14)}$	1	18.2622	0.1205
IGMN-ARMA-like(3,3)	8	24.3729	0.2190
IGMN-NOE-like(3,3)	4	20.3165	0.1583
ARMA-CIGMN(3,1)	4	18.3052	0.1258
ARMA-CIGMN(14,0)	1	15.9670	0.0957
ARMA-CIGMN(14,12)	1	14.2494	0.0762

Observing Table 4.3 we can see that the model with the lowest RMSE (14.2494) and NMSET (0.0762) is the one with the most input lags, using 14 lags and 12 MA components (ARMA-CIGMN(14,12)). The ARMA-CIGMN(14,0), using the inputs similar to an AR model, has come in second with RMSE (15.9670) and NMSET (0.0957). The difference between the NOE and the significant lags model is not relevant. With the exception of the ARMA-CIGMN(3,1), the other two ARMA-CIGMN models have smaller RMSE and NMSET. If we consider the modifications, as the ARMA-like and the NOE-like, the ARMA-CIGMN(3,1) still has better results. Next, we present the acf and pacf graphs of each model. The graphs for the original IGMN model and its modifications were made using the Statistical Toolbox of MATLAB. The graphs for the ARMA-CIGMN were made using R statistical package. Figure 4.10 presents the acf and pacf of the AR configuration, Figure 4.11 the AR with significant lags, Figure 4.12 the ARMA-like, Figure 4.13 the NOE-like, Figure 4.14 the ARMA-CIGMN(3,1), Figure 4.16 the ARMA-CIGMN(14,12) and Figure 4.15 the ARMA-CIGMN(14,0) models, respectively.

Figure 4.10 shows the results for the IGMN-AR(17) and we can see, on both graphs, that all values are inside the limits. This may be an example of using a large AR compo-

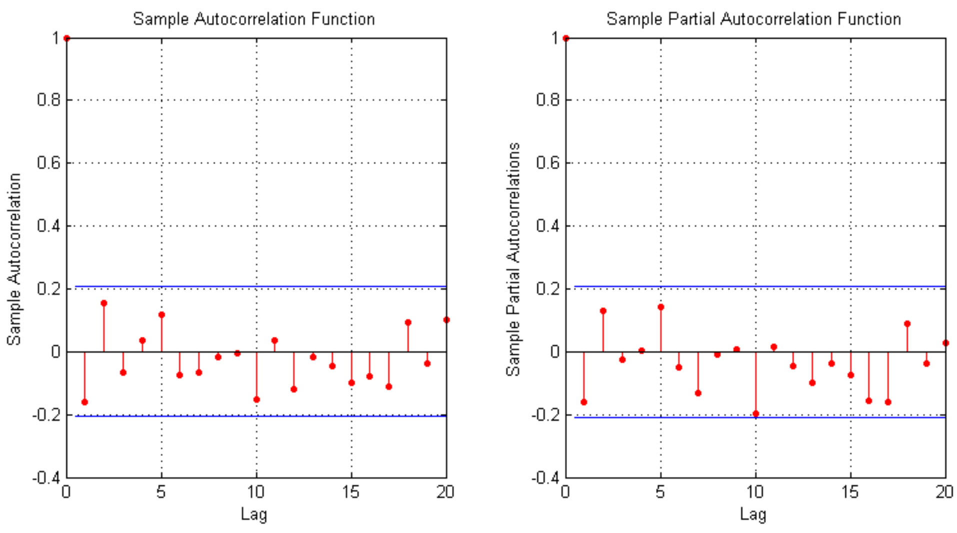


Figure 4.10: The acf and pacf of the IGMN-AR(17) model on the air passengers data

nent, $p = 17$, to solve the underlined MA process.

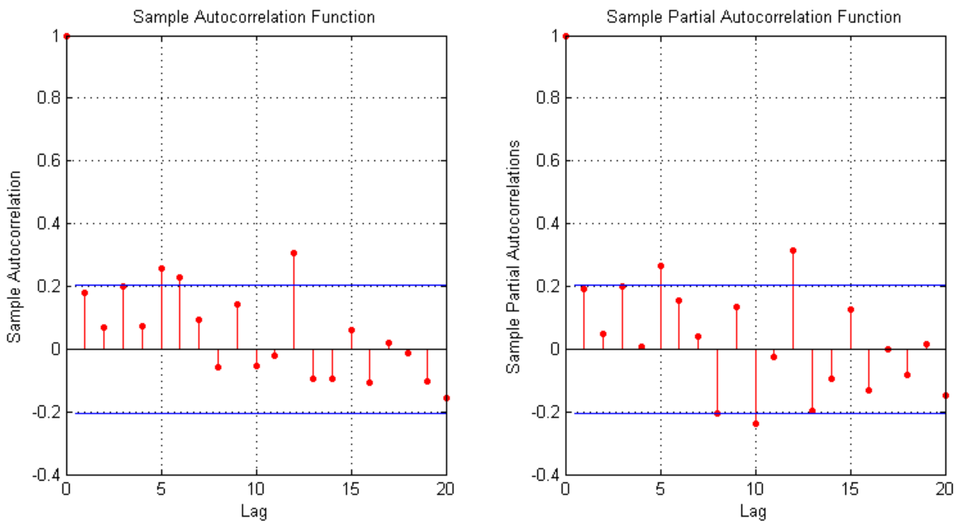


Figure 4.11: The acf and pacf of the significant lags model on the air passengers data

Figure 4.11 shows the results for the IGMN model with significant lags and we can notice that, in both graphs, some values are outside the limits, but not by much and, even with this values, the model is still well adjusted.

Figure 4.12 shows the results for the ARMA-like model and we can notice that all the values are inside the limits, for both graphs. The ARMA-like model seems well adjusted, including the possible MA process.

Figure 4.13 shows the results for the NOE-like model and even with smaller RMSE and NMSET than the ARMA-like model, the model presents too many values outside the limits, specifically on the acf graph. The model clearly fails to address some AR and/or MA components.

Figure 4.14 shows the results for a simple configuration ARMA-CIGMN model, sim-

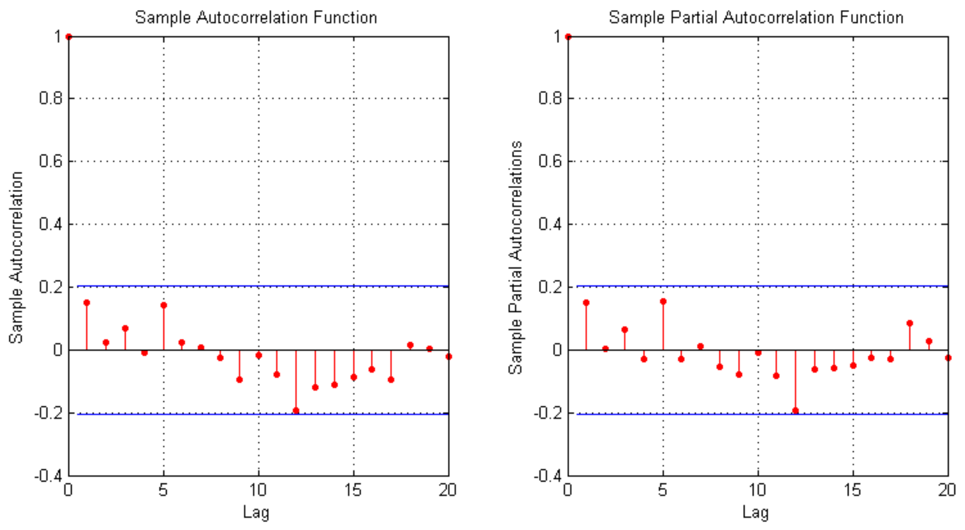


Figure 4.12: The acf and pacf of the ARMA-like model on the air passengers data

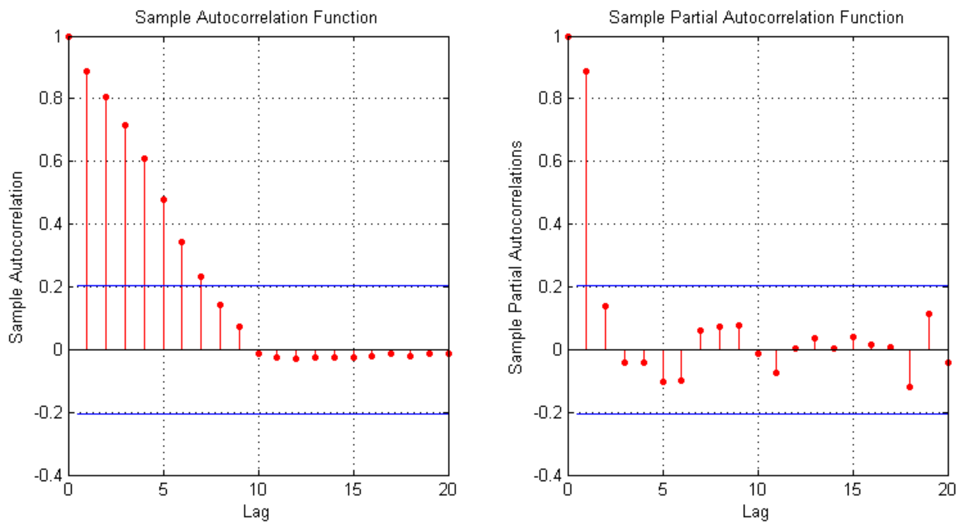
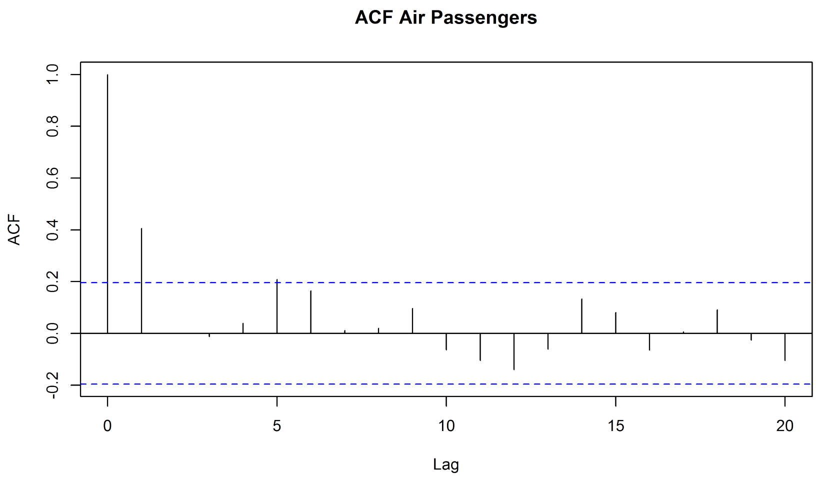
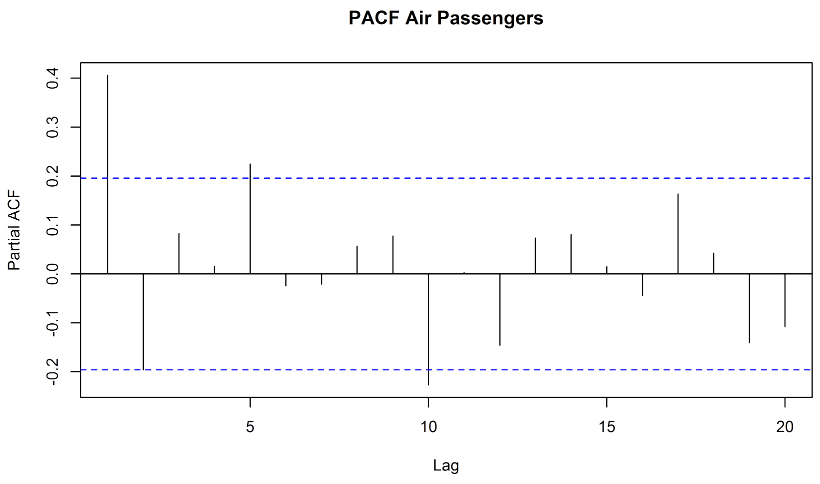


Figure 4.13: The acf and pacf of the NOE-like model on the air passengers data



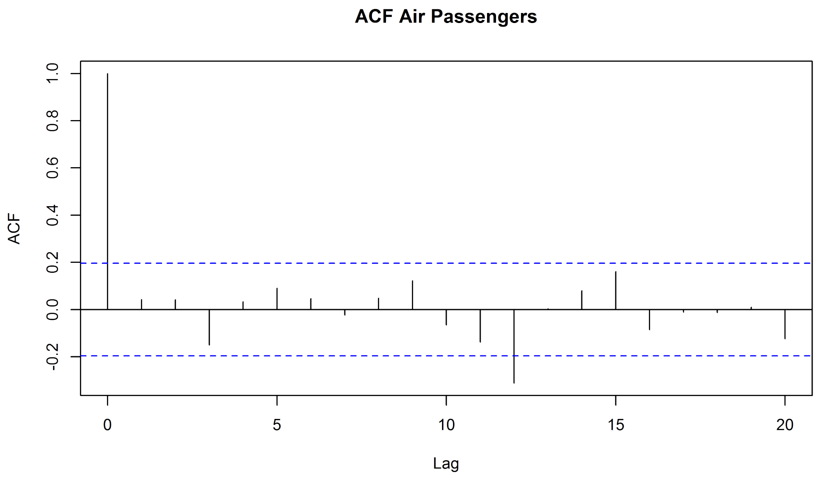
(a)



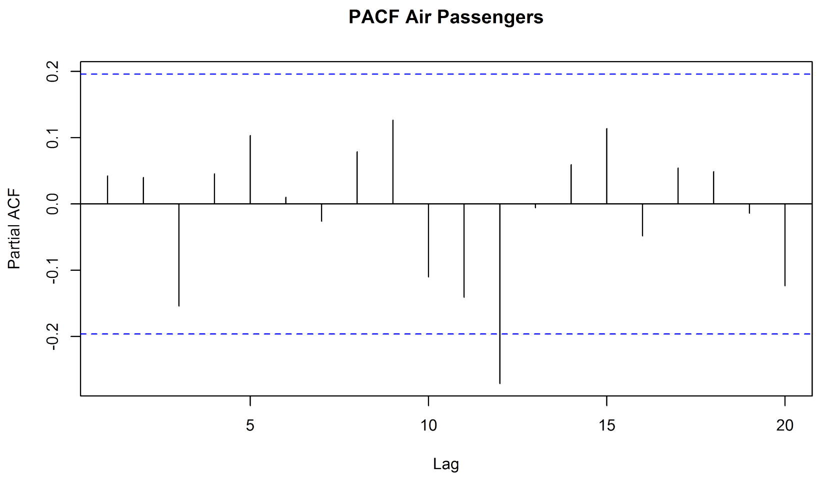
(b)

Figure 4.14: The acf(a) and pacf(b) of the air passengers data with the ARMA-CIGMN(3,1) model

ilar as the ARMA-like, but both graphs[1] shows values outside the limits. Similar to what happens with the significant lags model with exception to the first pacf value.



(a)



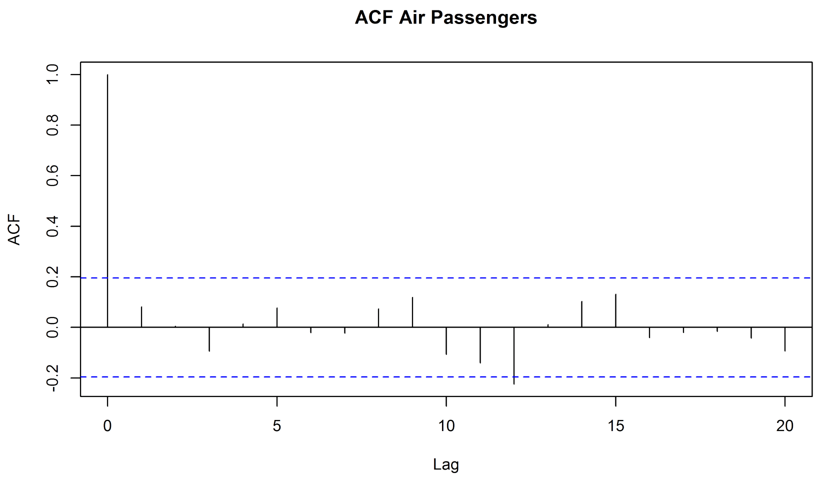
(b)

Figure 4.15: The acf(a) and pacf(b) of the air passengers data with the ARMA-CIGMN(14,0) model

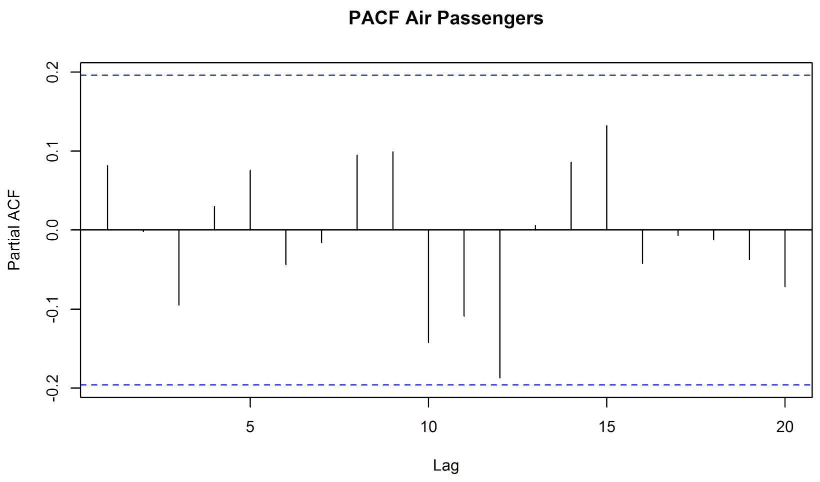
Figure 4.15 shows the results of the ARMA-CIGMN model similar as the ARMA-CIGMN model before, but using only the AR components.

And finally, Figure 4.16 shows the results for the more complex ARMA-CIGMN model using an AR(14) and an MA(12) components. In comparison with the graphs in Figure 4.15, we can clearly see the effect of the MA components, specially in respect to the value at lag 12, corresponding to the order of the MA component in the previous model, the ARMA-CIGMN(14,0). Almost all values are inside the limits for both graphs, with some exception made on the acf graph at lag 12, but even this value may be not relevant. Which reinforces that the ARMA-CIGMN can model MA components and present the smallest NMSET and RMSE of the configurations tested.

[1]The acf graphs that were made using the R statistical software always present the lag 0 correlation by default and it is always equal to 1. This is made for some verifications and it is not common on other software as, for example, the MATLAB Statistical Toolbox, also used.



(a)



(b)

Figure 4.16: The acf(a) and pacf(b) of the air passengers data with the ARMA-CIGMN(14,12) model

4.5 Simulated series

As shown before, the original IGMN model seems not to fit well on series with MA components. The goal here is to simulate different MA(1), an ARIMA(1,1,1) and an ARMA(1,1) models and use different configurations of the IGMN, the classical MA model and the ARMA-CIGMN model for comparisons. In the MA(1) simulated series, we present three different configurations of the original IGMN, the ARMA-CIGMN and the classical model to show the effect on the acf and pacf graphics. In the ARIMA(1,1,1) and the ARMA(1,1) simulated series we compare only the classical model and the ARMA-CIGMN model. Later, the same models are presented with the forecasting errors, for all the simulated series. We choose to present this way to better show the capabilities and limitations of these different models in comparison with the classic model.

4.5.1 Description of the experiment

First, we choose the most simple moving average model to be simulated, an MA(1) model, as in (4.3), which is a simplified presentation of (2.14).

$$x_{(t+1)} = \theta\epsilon_{(t)} + \epsilon_{(t+1)} \tag{4.3}$$

As seen on Chapter 2, $\epsilon_{(t+1)}$ is a white noise, and so, unknown and unpredictable. However, $\epsilon_{(t)}$ is known, computed as seen on Equation (2.2).

Later, we use a complete ARIMA model, using an AR(1) and a MA(1) model, with the integration coefficient 1, resulting in an ARIMA(1,1,1), as described in Equation (4.4).

$$x^*_{(t+1)} = \phi x^*_t - \theta\epsilon_{(t)} + \epsilon_{(t+1)} \tag{4.4}$$

where x^*_t is the integrated variable x_t, i.e., $x^*_t = x_t - x_{t-1}$.

Finally, we simulate an ARMA(1,1), as described in Equation (4.5), to better show the effects caused by the integration coefficient.

$$x_{(t+1)} = \phi x_t - \theta\epsilon_{(t)} + \epsilon_{(t+1)} \tag{4.5}$$

We generated 100 samples of size 1000 each for 9 different ϕ and θ. The last 100 values were separated to verify the one-step horizon forecasting errors. For each one of these samples, IGMN and/or ARMA-CIGMN models were adjusted and the pacf and acf computed. A classical statistical ARIMA(1,1,1) and a MA(1) model were also adjusted, just for comparison purposes. These 9 different parameters come from an increasing sequence with 0.1 increments, being the first ϕ and θ equal to 0.1 and the last equal to 0.9, all positive values, except in the ARIMA and ARMA models, as the θ presents the same value as the ϕ, but negative. Figure 4.17, Figure 4.18 and Figure 4.19 present an example of a simulated MA(1), a simulated ARIMA(1,1,1) and a simulated ARMA(1,1) models, respectively.

The metric used for comparison is the amount of values that are beyond the limits of the acf and pacf, using a lag of 12 steps. It is expected that the different IGMN configurations and the ARMA-CIGMN achieve similar, if not greater, values beyond the limits than a classical model would, that is why the classical models are also used. We also expect that the ARMA-CIGMN get results very similar as the classical model, at least with the MA(1) and ARMA(1,1) simulated series.

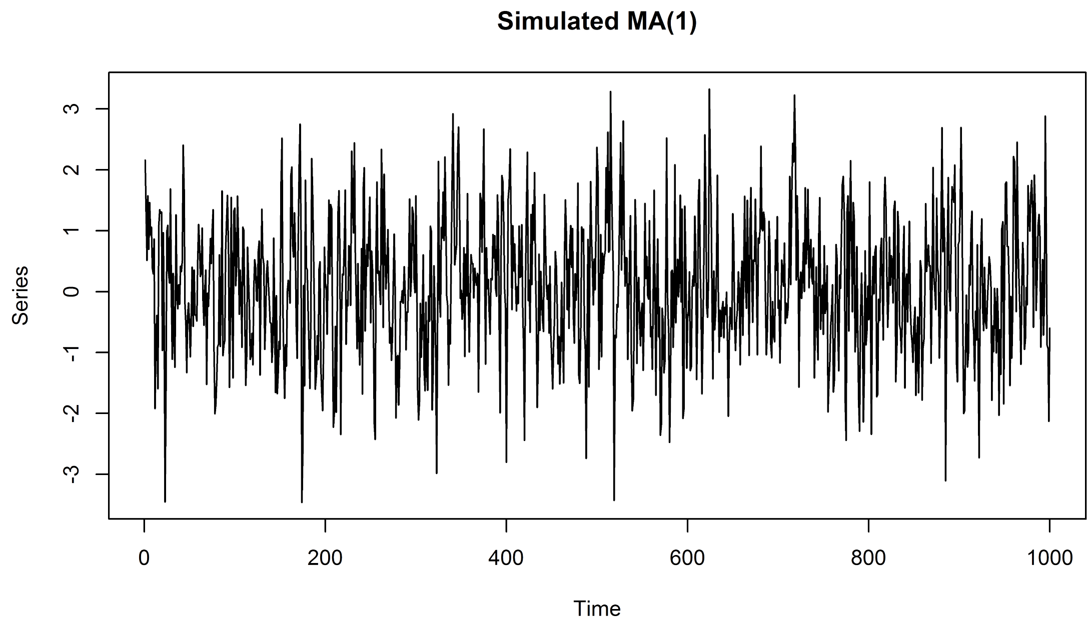


Figure 4.17: An example of a simulated MA(1) series using $\theta = 0.5$.

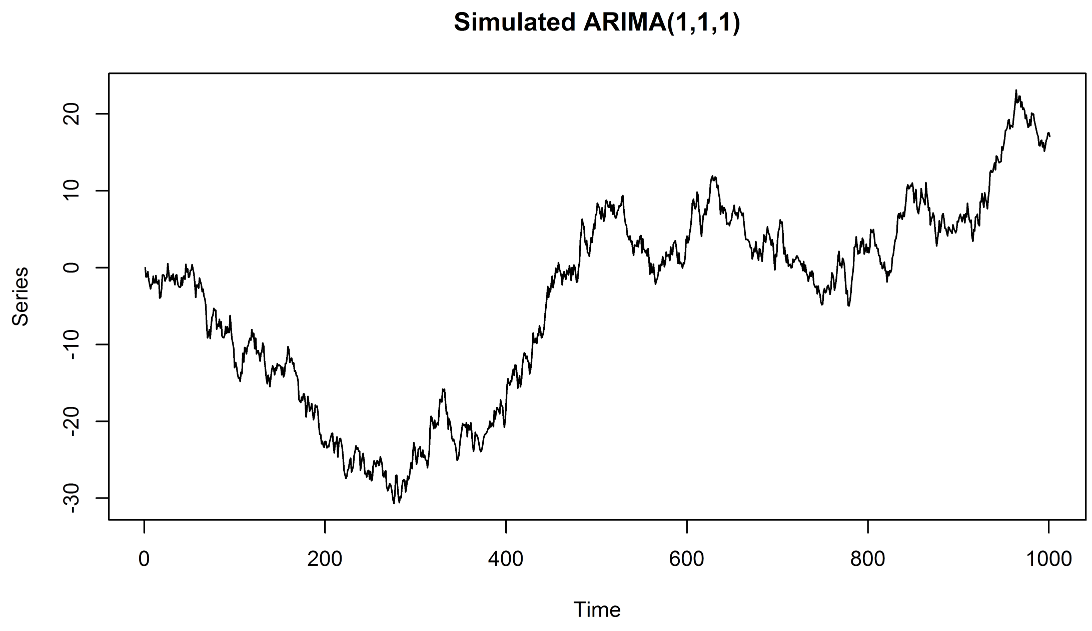


Figure 4.18: An example of a simulated ARIMA(1,1,1) series using $\phi = 0.5$ and $\theta = 0.5$.

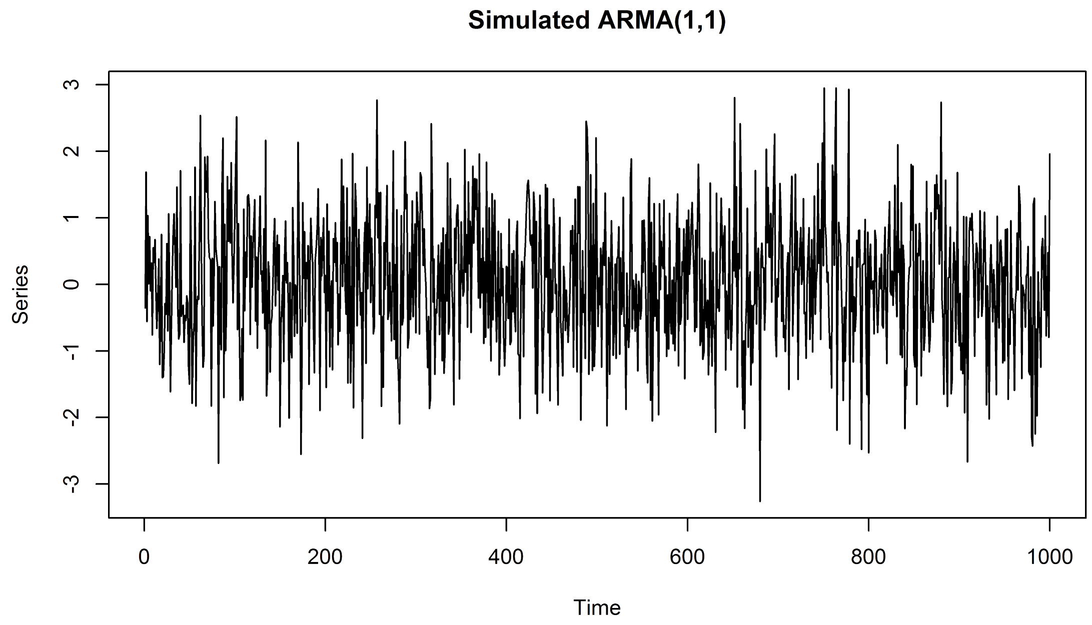


Figure 4.19: An example of a simulated ARMA(1,1) series using $\phi = 0.5$ and $\theta = 0.5$.

4.5.2 The simulated MA(1) series

In the original IGMN, 3 different configurations were used: (i) an ARMA-like configuration, (ii) a simple MA configuration and (iii) a bias configuration. The ARMA-like configuration uses, besides the $\epsilon_{(t)}$, an $x_{(t)}$, which is similar to an ARMA(1,1) model. The simple MA configuration uses only the $\epsilon_{(t)}$. And the bias configuration uses the $\epsilon_{(t)}$ and a bias (values equal to one). This last configuration was used as the IGMN is shown to be similar to a regression model and with this approach we intend to mimic the behavior of a constant in the regression model, commonly represented with β_0 in regression analysis. The ARMA-like configuration was used because the way the algorithm of the IGMN works and this approach is similar to an algorithm for classic MA models presented in (MORETTIN; TOLOI, 2006). Tables from (4.4) to (4.11) show the results based on the model adjustment for these configurations.

The ARMA-CIGMN model was a pure MA model, i.e., ARMA-CIGMN(0,1). For this presentation, we decided to force the ARMA-CIGMN model to create and update only one component. This decision was made to better verify the MA capabilities of our proposed model. If we do not force the one component model, the combination of different components on a large series can mislead the conclusions. This also simplifies the simulation process, as the model was simply adjusted for every different MA parameter. Tables 4.12 and 4.13 present the results for the ARMA-CIGMN models.

Observing Tables (4.4) and (4.5) one can see how the classic MA model behaves. The percentage of models considered well adjusted, i.e., models with none or one value outside the limits, are no less than 90%, with only two exceptions seen on Table (4.4) with θ equals to 0.1, that has 88% and θ equals to 0.5, that also has 88%. We use these tables for comparison with the different configurations of the IGMN models.

The IGMN model using only the differences shows problems with larger θ in the acf graphics, as seen on Table (4.6). In fact, with $\theta = 0.2$ the IGMN already presents an 82% of models not well adjusted. The same can be seen on Table (4.7).

	Parameters of the MA models - θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	88%	94%	94%	90%	88%	94%	90%	95%	94%
2 or 3 outside	11%	6%	5%	10%	12%	6%	10%	5%	6%
4 or more outside	1%	0%	1%	0%	0%	0%	0%	0%	0%

Table 4.4: Values that lie outside the limits boundaries on the acf graphic for the classical model in the MA(1) simulated series - in %

	Parameters of the MA models - θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	95%	93%	92%	93%	91%	92%	92%	91%	94%
2 or 3 outside	5%	7%	8%	7%	9%	8%	8%	9%	6%
4 or more outside	0%	0%	0%	0%	0%	0%	0%	0%	0%

Table 4.5: Values that lie outside the limits boundaries on the pacf graphic for the classical model in the MA(1) simulated series - in %

	Parameters of the MA models - θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	72%	18%	4%	0%	0%	0%	0%	0%	0%
2 or 3 outside	27%	76%	84%	77%	71%	67%	60%	64%	74%
4 or more outside	1%	6%	12%	23%	29%	33%	40%	36%	26%

Table 4.6: Values that lie outside the limits boundaries on the acf graphic for the IGMN model using only past differences $\epsilon_{(t-1)}$ in the MA(1) simulated series - in %

	Parameters of the MA models - θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	73%	16%	3%	0%	0%	0%	0%	0%	0%
2 or 3 outside	27%	79%	88%	77%	26%	7%	5%	1%	0%
4 or more outside	0%	5%	9%	23%	74%	93%	95%	99%	100%

Table 4.7: Values that lie outside the limits boundaries on the pacf graphic for the IGMN model using only past differences $\epsilon_{(t-1)}$ in the MA(1) simulated series - in %

As in Table (4.6), Table (4.7) also shows problems with this configuration. However, in respect to the pacf graphics, the number of well adjusted models is even lower. With $\theta = 0.6$ there are more than 90% of the models poorly adjusted, reaching 100% (all models) with $\theta = 0.9$.

	Parameters of the MA models - θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	70%	25%	5%	0%	0%	0%	0%	0%	0%
2 or 3 outside	29%	70%	81%	75%	70%	68%	58%	65%	76%
4 or more outside	1%	5%	14%	25%	30%	32%	42%	35%	24%

Table 4.8: Values that lie outside the limits boundaries on the acf graphic for the IGMN model using past differences $\epsilon_{(t-1)}$ and bias in the MA(1) simulated series - in %

	Parameters of the MA models - θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	73%	25%	4%	1%	0%	0%	0%	0%	0%
2 or 3 outside	27%	69%	86%	78%	28%	9%	4%	1%	0%
4 or more outside	0%	6%	10%	21%	72%	91%	96%	99%	100%

Table 4.9: Values that lie outside the limits boundaries on the pacf graphic for the IGMN model using past differences $\epsilon_{(t-1)}$ and bias in the MA(1) simulated series - in %

The IGMN model that uses differences and bias configuration has similar results as the IGMN model using only differences, both in the acf graphics, shown on Table (4.8), as in the pacf graphics, shown on Table (4.9). These two configurations seem to work better on models with lower θ. This could also happen because of the great impact that some of the IGMN parameters seem to have on the final model. The fact is that even this configuration fails to model the MA component, at least for $\theta > 0.2$.

	Parameters of the MA models - θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	97%	94%	92%	91%	62%	16%	0%	0%	0%
2 or 3 outside	3%	6%	8%	9%	38%	77%	85%	76%	68%
4 or more outside	0%	0%	0%	0%	0%	7%	15%	24%	32%

Table 4.10: Values that lie outside the limits boundaries on the acf graphic for the ARMA-like(1,1) model in the MA(1) simulated series - in %

The ARMA-like configuration seems to behave better than the other two on the acf graphics, as shown on Tables (4.10) and (4.11). However, even this configuration fails to adapt with larger θ. The configuration works seemingly well until $\theta = 0.5$ for both graphics. For greater θ, the configuration becomes worst, with 100% models poorly adjusted for pacf graphics, in Table (4.11).

Observing Tables 4.12 and Table 4.13, we note that, compared to the different configurations of the original IGMN model, the ARMA-CIGMN achieves better results. At

	Parameters of the MA models - θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	96%	93%	91%	90%	55%	9%	0%	0%	0%
2 or 3 outside	4%	7%	9%	10%	45%	82%	59%	3%	0%
4 or more outside	0%	0%	0%	0%	0%	9%	41%	97%	100%

Table 4.11: Values that lie outside the limits boundaries on the pacf graphic for the ARMA-like(1,1) model in the MA(1) simulated series - in %

	Parameters of the MA models - θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	91%	91%	91%	89%	92%	88%	83%	82%	58%
2 or 3 outside	9%	9%	9%	10%	8%	12%	17%	17%	38%
4 or more outside	0%	0%	0%	1%	0%	0%	0%	1%	4%

Table 4.12: Values that lie outside the limits boundaries on the acf graphic for the ARMA-CIGMN(0,1) model in the MA(1) simulated series - in %

	Parameters of the MA models - θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	93%	92%	92%	87%	93%	88%	83%	81%	62%
2 or 3 outside	7%	8%	8%	13%	7%	12%	17%	18%	32%
4 or more outside	0%	0%	0%	0%	0%	0%	0%	1%	6%

Table 4.13: Values that lie outside the limits boundaries on the pacf graphic for the ARMA-CIGMN model(0,1) in the MA(1) simulated series - in %

4 or more values outside the limits, the percentage is, at maximum, 6% for $\theta = 0.9$. In the pacf graphs, for seven different parameters, the percentage of 4 or more values outside the limits is zero (0%). However, when compared with the results of the classical MA(1) model, some results are yet to be matched. With lower θ, specifically less than 0.8 the results are very promising. With higher θ, even with the good results, the ARMA-CIGMN falls behind the classical MA(1). But, even with the problems on higher θ, the ARMA-CIGMN shows some interesting results with a pure, simulated, MA(1) process, even surpassing the classical model on some cases.

The next step is to show how these different configurations behave on forecasting in comparison with the classic model. Table (4.14) presents the forecasting mean squared errors (MSE) for each of the tested models and configurations.

Observing Table (4.14), the forecasting errors are similar, with the exception of the maximum error for the configuration with differences only and the configuration with differences and bias. The classical model presents the smallest forecasting errors (minimum of 0.5809), but the difference to the other four models is not relevant. If, however, we consider the mean and median, the model with the lowest mean MSE and median MSE is the ARMA-CIGMN. While the ARMA-CIGMN model stays somewhat behind the classical model in respect with the acf and pacf graphs, in forecasting errors the model surpasses even the model used to generate the series. This is maybe better represented with Figure (4.20), that shows the same information of Table (4.14) using a Box-plot graphic.

		IGMN Models			
Resume	Classic model	Differences only	Differences and bias	ARMA-like(1,1)	ARMA-CIGMN(0,1)
Minimum	0.5809	0.6813	0.6813	0.5899	0.6165
1st Quartile	0.9097	0.9824	0.9814	0.9379	0.9044
Median	1.0092	1.1165	1.1161	1.0469	1.0016
Mean	1.0147	1.1378	1.1367	1.0638	1.0105
3rd Quartile	1.1084	1.2649	1.2655	1.1681	1.1120
Maximum	1.5305	2.1397	2.1113	1.8851	1.5243

Table 4.14: Forecasting mean squared errors for the different models, for all θ used, in the MA(1) simulated series.

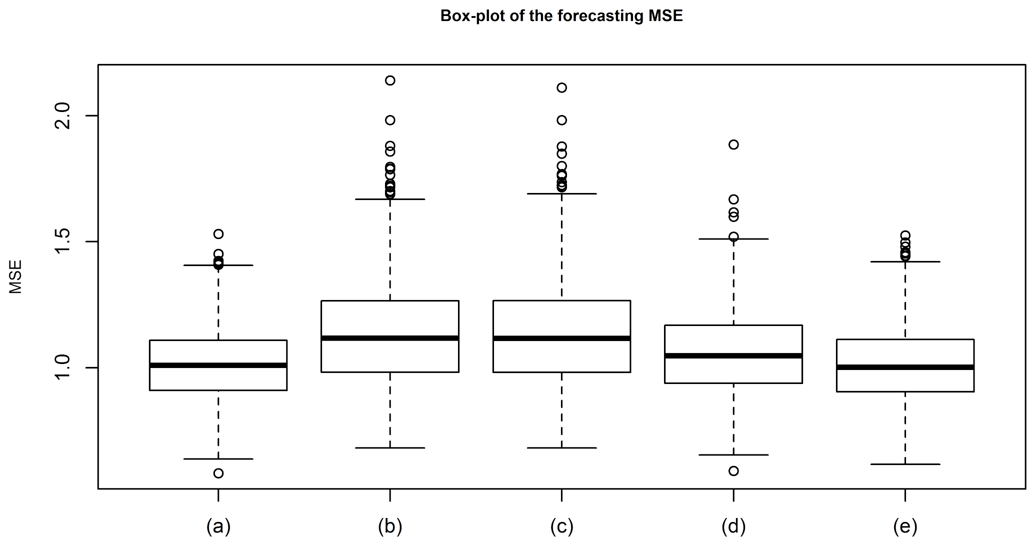


Figure 4.20: Box-plot of the forecasting mean squared errors where: (a) represents the classic model, (b) is the IGMN using only differences, (c) is the IGMN using differences and bias, (d) is the ARMA-like model and (e) the ARMA-CIGMN model, in the MA(1) simulated series.

The Box-plot shown on Figure (4.20) shows that, the configurations that were poorly adjusted using acf and pacf graphics presents more heterogeneity, and more outliers. The ARMA-CIGMN shows a behavior similar to the classical model, as the IGMN configuration using differences and autoregressive component, besides the magnitude of its outliers.

4.5.3 The simulated ARIMA(1,1,1)

In this scenario, we compare only the classical model and the ARMA-CIGMN model. Because of the integration component, we decided to use an ARMA-CIGMN(2,1), i.e., two AR components and one MA component, instead of an ARMA-CIGMN(1,1). The second AR component should model the integration.

Another important difference is that, in the simulated ARIMA(1,1,1), the ARMA-CIGMN models were not forced to use only one component. This was also because of the integration coefficient, as a linear model may not be able to model an integrated series very well. So, in this simulated series, the ARMA-CIGMN may create more than one component. Tables 4.15 and 4.16 present the results for the classical model, while Tables 4.17 and 4.18 present the results for the ARMA-CIGMN models.

	Parameters of the ARIMA models - ϕ and θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	93%	90%	89%	96%	93%	95%	95%	91%	92%
2 or 3 outside	7%	10%	11%	4%	7%	5%	5%	9%	8%
4 or more outside	0%	0%	0%	0%	0%	0%	0%	0%	0%

Table 4.15: Values that lie outside the limits boundaries on the acf graphic for the classical model, in the ARIMA(1,1,1) simulated series - in %

	Parameters of the ARIMA models - ϕ and θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	93%	90%	89%	96%	93%	95%	95%	91%	92%
2 or 3 outside	7%	10%	11%	4%	7%	5%	5%	9%	8%
4 or more outside	0%	0%	0%	0%	0%	0%	0%	0%	0%

Table 4.16: Values that lie outside the limits boundaries on the pacf graphic for the classical model, in the ARIMA(1,1,1) simulated series - in %

From Tables (4.15) and (4.16) we can observe how the classical model behaves similar as in the MA(1) simulated series. Being a confidence interval of 95%, the values were expected, as for most of the parameters, the models considered well adjusted (none or one value outside the limits) are more or equal than 90%, for both acf and pacf and among all parameters, with one exception, as the pacf and acf using $\phi = 0.3$ and $\theta = 0.3$, on Tables 4.16 and 4.15.

	Parameters of the ARIMA models - ϕ and θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	53%	50%	51%	57%	50%	61%	49%	59%	53%
2 or 3 outside	15%	9%	9%	16%	11%	6%	14%	10%	12%
4 or more outside	32%	41%	40%	27%	39%	33%	37%	31%	35%

Table 4.17: Values that lie outside the limits boundaries on the acf graphic for the ARMA-CIGMN(2,1) model, in the ARIMA(1,1,1) simulated series - in %

From Table 4.17 and Table 4.18, we note that, compared to the classical model, the ARMA-CIGMN achieves poor results, being almost 50% of the models well adjusted, i.e., equal or less than 1 value outside the limits. Both Tables, 4.18 and 4.17, show how the integration coefficient has a great impact on the residuals analysis. However, as the classical model, the results presented with the ARMA-CIGMN are the same for both Tables 4.18 and 4.17, with the same percentages, which may indicate that the ARMA-CIGMN can work with both components, even with the integration coefficient.

	Parameters of the ARIMA models - ϕ and θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	53%	50%	51%	57%	50%	61%	49%	59%	53%
2 or 3 outside	15%	9%	9%	16%	11%	6%	14%	10%	12%
4 or more outside	32%	41%	40%	27%	39%	33%	37%	31%	35%

Table 4.18: Values that lie outside the limits boundaries on the pacf graphic for the ARMA-CIGMN(2,1) model, in the ARIMA(1,1,1) simulated series - in %

The next step is to compare the forecasting errors for both models. Table (4.19) presents the forecasting mean squared errors (MSE) for both models using all the different parameters.

Resume	ARIMA(1,1,1)	ARMA-CIGMN(2,1)
Minimum	0.0007	0.6690
1st Quartile	0.0032	0.9866
Median	0.0072	1.1225
Mean	0.0123	2.7603
3rd Quartile	0.0154	1.4535
Maximum	0.1172	154.22

Table 4.19: Forecasting mean squared errors for the classical and the ARMA-CIGMN models, for all ϕ and θ used in the ARIMA(1,1,1) simulated series.

As Table 4.19 shows, the ARMA-CIGMN model achieve less accurate results. The integration coefficient still offer a challenge for the ARMA-CIGMN model, even with the multiple components. For this specific case, the results are too far apart for a graphical presentation, so we choose not to present the Box-plot of the forecasting errors.

4.5.4 The simulated ARMA(1,1)

After the results shown for the ARMA-CIGMN model using an ARIMA(1,1,1) simulated series, the next step is to show the ARMA-CIGMN model on an ARMA(1,1) simulated series, without the integration coefficient. As with the previous simulated series, Tables 4.20 and 4.21 show the results with the classical model and Tables 4.22 and 4.23 show the results using the ARMA-CIGMN(1,1) model.

	Parameters of the ARMA models - ϕ and θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	95%	95%	92%	95%	95%	89%	92%	95%	92%
2 or 3 outside	5%	5%	8%	5%	5%	11%	8%	5%	8%
4 or more outside	0%	0%	0%	0%	0%	0%	0%	0%	0%

Table 4.20: Values that lie outside the limits boundaries on the acf graphic for the classical model in the ARMA(1,1) simulated series - in %

As with the previous series, the classical model is used as a standard, or a control, for comparison. Observing Tables 4.20 and 4.21, all the results are compatible with the previous classical models used. We then proceed to the ARMA-CIGMN analysis.

	Parameters of the ARMA models - ϕ and θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	95%	95%	92%	95%	95%	89%	92%	95%	92%
2 or 3 outside	5%	5%	8%	5%	5%	11%	8%	5%	8%
4 or more outside	0%	0%	0%	0%	0%	0%	0%	0%	0%

Table 4.21: Values that lie outside the limits boundaries on the pacf graphic for the classical model in the ARMA(1,1) simulated series - in %

	Parameters of the ARMA models - ϕ and θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	92%	86%	86%	89%	92%	89%	92%	94%	93%
2 or 3 outside	8%	13%	13%	11%	8%	11%	8%	6%	7%
4 or more outside	0%	1%	1%	0%	0%	0%	0%	0%	0%

Table 4.22: Values that lie outside the limits boundaries on the acf graphic for the ARMA-CIGMN(1,1) model in the ARMA(1,1) simulated series - in %

Without the integration coefficient, the results are, by far, better. Table 4.22 and Table 4.23 present the same results, as with the ARMA-CIGMN model on the ARIMA(1,1,1) simulated series. These results are close to the results of the classical model. This reinforces that the previous results are not better because of the integration coefficient. Table 4.24 present the results of the forecasting errors for all the parameters and models.

Observing Table 4.24, the differences are much smaller, being almost irrelevant between these two models. The results of the minimum, the maximum and the third quartile MSE error of the ARMA-CIGMN are noteworthy, presenting better performance than the classical model. Figure 4.21 presents the Box-plot of both models.

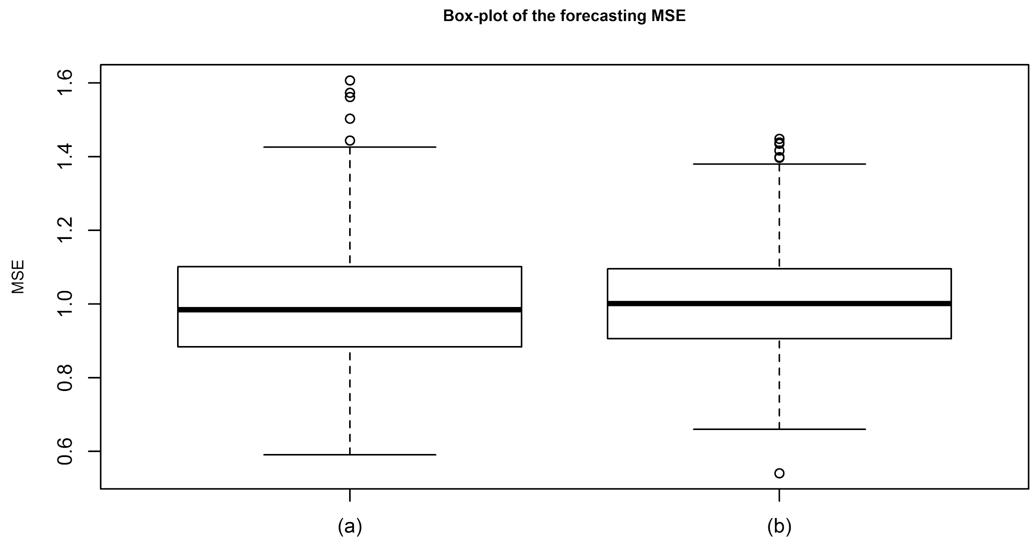


Figure 4.21: Box-plot of the forecasting mean squared errors where: (a) represents the classic model and (b) is the ARMA-CIGMN model in the ARMA(1,1) simulated series.

Figure 4.21 shows that, while the majority of the results are almost the same, the

	Parameters of the ARMA models - ϕ and θ								
Values off limits	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0 or 1 outside	92%	86%	86%	89%	92%	89%	92%	94%	93%
2 or 3 outside	8%	13%	13%	11%	8%	11%	8%	6%	7%
4 or more outside	0%	1%	1%	0%	0%	0%	0%	0%	0%

Table 4.23: Values that lie outside the limits boundaries on the acf graphic for the ARMA-CIGMN(1,1) model in the ARMA(1,1) simulated series - in %

Resume	ARMA(1,1)	ARMA-CIGMN(1,1)
Minimum	0.5909	0.5406
1st Quartile	0.8839	0.9059
Median	0.9848	1.0015
Mean	0.9991	1.0065
3rd Quartile	1.1017	1.0957
Maximum	1.6071	1.4487

Table 4.24: Forecasting mean squared errors for the classical and the ARMA-CIGMN models, for all ϕ and θ used in the ARMA(1,1) simulated series.

classical model presents larger MSE than the ARMA-CIGMN model. This shows that even with simulated series using the classical models, with except for the ARIMA(1,1,1) series, the ARMA-CIGMN can achieve results very similar to the classical models.

4.6 Modeling the lynx series

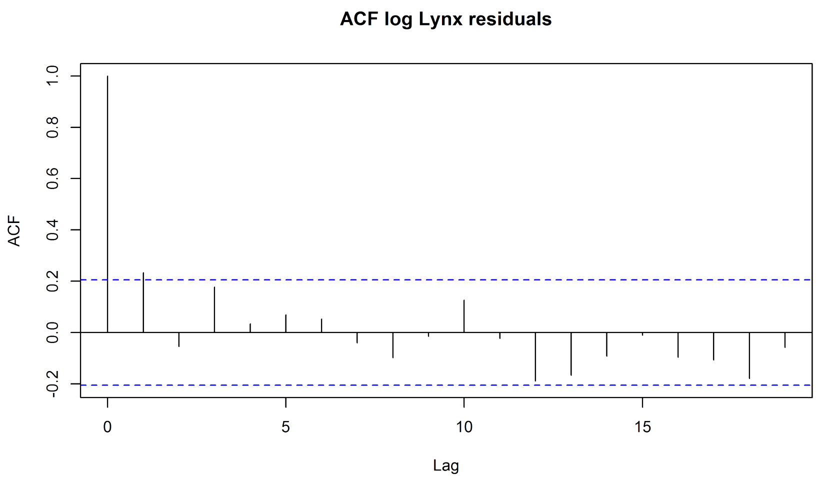
Finally we present the experiment with the Canadian lynx series. This series was chosen to show how the ARMA-CIGMN can be a precise model, even with a small amount of data. The complete series presents 114 values: 100 was used for the ARMA-CIGMN learning and 14 for the recalling, or simply, forecasting. Figure 4.22 presents the acf and pacf graphs after the adjusted ARMA-CIGMN model.

It is possible to notice in Figure 4.22 that, except for a few point outside the limits, the model is well adjusted. We used a single component ARMA-CIGMN model with 5 significant AR components and 2 significant MA components, or simply, an ARMA-CIGMN(5,2). We used a single component IGMN only to compare with the other models presented: an AR(12) classical model, Zhang's hybrid model (ZHANG, 2003) and Khashei-Bijari hybrid model (KHASHEI; BIJARI, 2010). The adjusted model and the log series are shown in Figure 4.23.

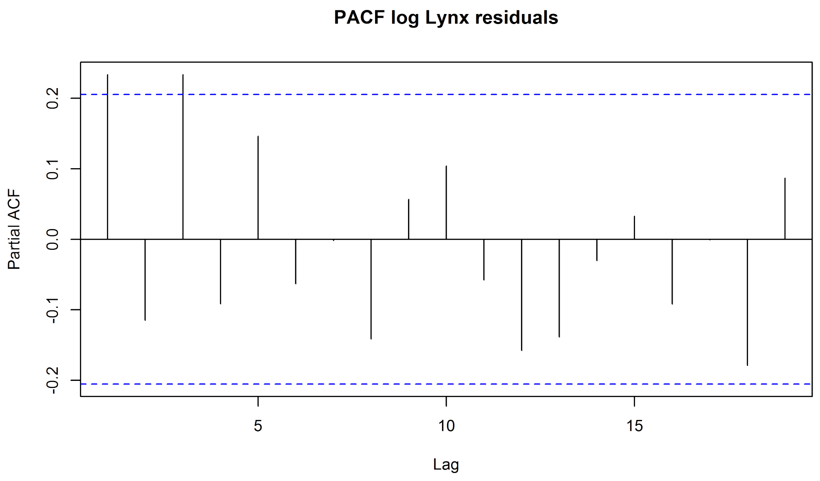
The adjusted model shown in Figure 4.23, besides being a single component model, seems to be well adjusted, specifically near the index 60 values. The model was then tested for forecasting the last 14 values. The real values and the forecasts are shown on Figure 4.24.

The model shows similar results as in the learning data series. Figure 4.24 shows that the model can also present a lower forecasting errors, with exception for the index 8 and 10 values.

Finally, the model was tested against other models using the same metrics used on other works: the mean absolute error (MAE) and the mean squared error (MSE), as seen on Equations (4.6) and (4.7). Table 4.25 presents the comparison results.



(a)



(b)

Figure 4.22: The acf(a) and pacf(b) of the residuals after the ARMA-CIGMN model

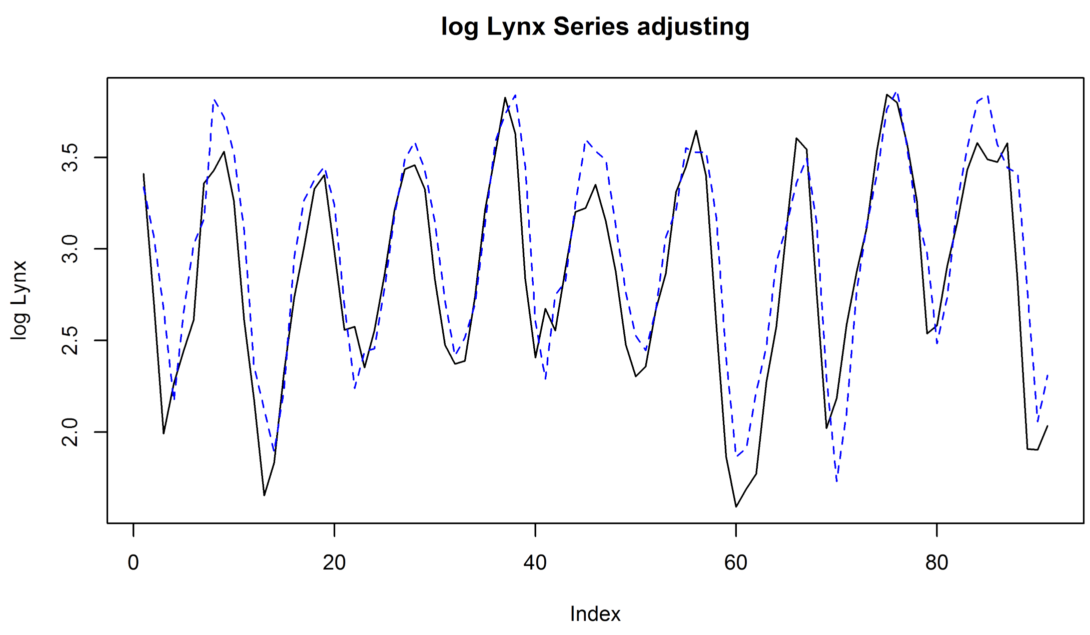


Figure 4.23: The lynx series (line) with the ARMA-CIGMN adjusted model (dotted line)

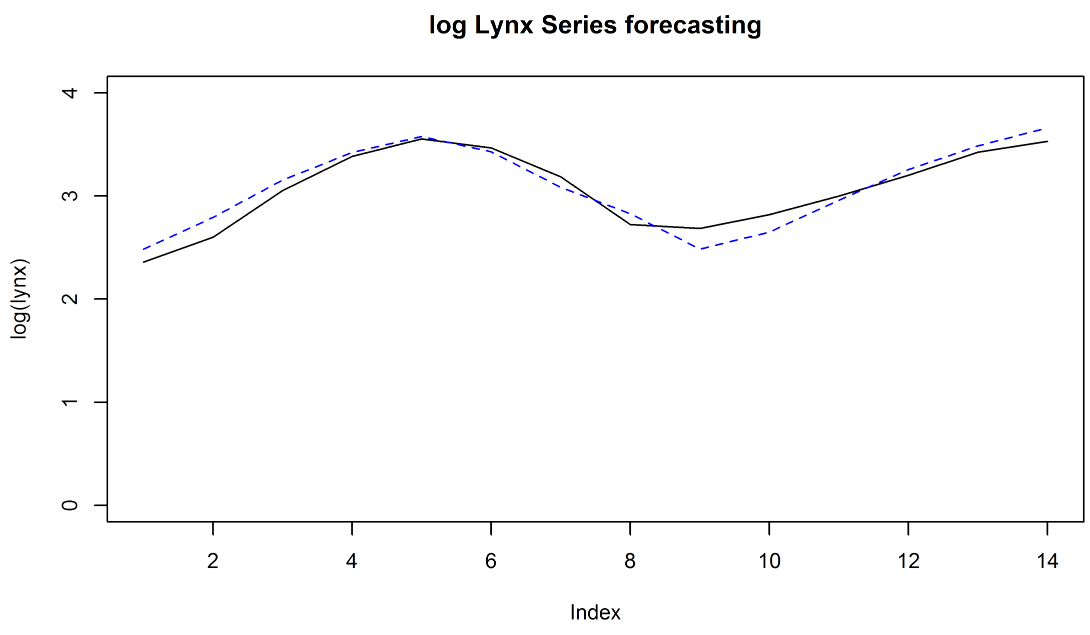


Figure 4.24: The lynx series (line) with the ARMA-CIGMN model forecasts in a one-step-ahead horizon (dotted line)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|e_i| \tag{4.6}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(e_i^2) \tag{4.7}$$

Table 4.25: Comparison between models - Lynx series, 14 values, one-step ahead horizon

Models	MAE	MSE
AR(12)	0.112255	0.020486
Zhang's hybrid model	0.103972	0.017233
Khashei-Bijari hybrid model	0.089625	0.013609
ARMA-CIGMN(5,2)	0.099014	0.013057

The results presented in Table 4.25 show that the ARMA-CIGMN has the smallest MSE error, but the second smallest MAE error. This is probably because the forecasting errors near the 8 and 10 values of the forecasting data series.

5 CONCLUSION AND DISCUSSION

5.1 Discussion

The original IGMN model has shown in past works, (HEINEN, 2011; PINTO, 2011; FLORES; PINTO; ENGEL, 2012), that can model time series data very well. The problem is how to stablish the correct inputs for the network. The first experiment, using the acf and pacf graphs to help the user, shows that the original IGMN fails to detect some behavior of the time series, as the MA component. Even with the ARMA-like and the NOE-like models, the problem with the MA components persists over the different experiments. This problem is more theoretical than practical as the IGMN model, even with simple configurations, can forecast very well, with errors similar, if not lower, to other models. The theoretical problem is that even the IGMN supposes that the input data are independent from each other, which only happens when the resulting acf and pacf graphics for the residuals show all values inside the limits, or at least the deviations are not statistically significant. The other theoretical problem is that the MA components estimation under the Bayesian approach is not online and the method we use is only possible under a classical statistical approach.

The ARMA-CIGMN solves that problem. The ARMA-CIGMN, as the experiments show, can model MA components and, in fact, had improved the forecasting errors over the original IGMN and its different versions. Also, the ARMA-CIGMN works with the acf and pacf graphs and even maintains some of the best characteristics of the original IGMN. This is the main goal of this work and the main contribution: a neural network model that can model the AR component and also the MA component with small forecasting errors.

But we need to highlight the results of the ARMA-CIGMN on two different series: the air passengers and the ARIMA(1,1,1) simulated series. Although the ARMA-CIGMN is well adjusted to the former series, better than the classical model, is has poor results on the last one. Both series present an integration component, but the ARMA-CIGMN works well in only one of them, the air passengers. The fact is that the air passengers series presents an always increasing mean, while the ARIMA(1,1,1), as shown on Figure 4.18, presents a more random behavior of the mean. The ARMA-CIGMN, because of the multiple components, seems to model very well a constant shifting in mean but not when the shifts occur more randomly, as in the simulated ARIMA(1,1,1) series. The simple integration coefficient does not work online, and may present some issues with different components, but it is one possible goal for future works.

Moreover, some new issues arise, while others continue to affect the modeling of a time series. One of these issues is the addition of new parameters that are user-defined. The ARMA-CIGMN needs some inputs from the user, as the initial θ, for example. This parameters can heavily modify the resulting model. All of the models presented in this work used a value of $\theta = 0.1$, because is near, but not equal, to zero (0) and can easily be update for positive or negative values. In fact, the value must be higher than 0 (zero) and less than 1 (one) for statistical purposes, but there is no way to find an optimal value for now. For larger samples, which is the purpose of an online incremental model, it is almost irrelevant, as it tends to converge very fast. But in smaller samples, the value set could have significant influence in the final model.

The other user defined parameter is the α criterium to create new components. If set to a very low value, no new components are created and the IGMN becomes an online linear model, as the incremental part disappears. If it is set too high, too many components are created, the component-wise sample size becomes smaller and the result, and even the model, can diverge. By now, there are no specifications whatsoever for how the parameter must be set, and is still dependent on a trial-and-error method.

Some other minor issues have also arisen during the ARMA-CIGMN model development. Some of them were inherited from the original IGMN, as the option of using diagonal covariance matrices, ignoring the covariances and using only the variances, or simplifying the create-or-update test to work always with a single missing value, very common on univariate time series. But even with these problems, the ARMA-CIGMN model had improve on almost every aspect over the original IGMN and the early versions. The only downside seems to be the increase in computational cost, caused by the increase of its dimensionality due to the MA components. The ARMA-CIGMN achieves lower forecasting errors even when the model has only one component. It surpasses the classical model on the air passengers series, with the classical model working on a differentiate series while the ARMA-CIGMN works on a nonstationary raw data.

We also presented the CIGMN as a combination of linear models, which opens new possibilities and new challenges. As a combination of linear models, component-wise, we have a well known estimator, the OLS. Very few assumptions are made for the OLS, but we can make some other assumptions and have an understanding of each components model, as, for example, assuming Gaussian i.i.d. errors and working with inference and hypothesis tests over the parameters for each component. As a new challenge, we have the problem of possible collinearity inside the components, which can affect the model and the forecasts directly.

Finally, besides the related issues, the ARMA-CIGMN may be considered an all around model, based on the experiments and the results presented. The model performed well on linear series against a classical linear model, non stationary series as the Air passengers, surpassing other similar models and even in non linear time series, as the monthly sunspot series and the Canadian lynx series.

5.2 Future works

As stated before, the ARMA-CIGMN had brought some new issues along some other inheritances from the original IGMN. The new user-defined parameters: α and the initial θ offer some subjectivity and affect the resulting model directly. Relating these values with the series presented is one of our future goals, along the integration issue that arise from the comparison between the results in the air passengers series and the simulated ARIMA(1,1,1) series.

Another area of interest are the IGMN components (also called neurons). Is it possible to assure that all components will have some common area, to avoid gaps when forecasting? Is it possible to define, based on some previous sample, the optimal number of components desired and set this quantity? Can the MA components work independently from the AR components? If so, the model becomes more or less complex? The inclusion of the MA components, although it solves the MA estimation problems, has generated new questions that need to be answered.

REFERENCES

BOX, G.; JENKINS, G. M. **Time series analysis**. 1st.ed. San Francisco: Holden-Day, 1976.

BOX, G.; JENKINS, G. M.; REINSEL, G. **Time series analysis**. 3rd.ed. New York: Prentice Hall, 1994. 592p.

CHIU, D.-Y.; CHEN, P.-J. Dynamically exploring internal mechanism of stock market by fuzzy-based support vector machines with high dimension input space and genetic algorithm. **Expert Systems with Applications**, [S.l.], v.36, p.9, 2009.

CONGDON, P. **Applied bayesian modelling**. 1.ed. London: Wiley, 2003. 457p.

COSTA, M. A.; BRAGA, A.; MENEZES, B. R. de. Improving generalization of MLPs with sliding mode control and the Levenberg-Marquardt algorithm. **Neurocomputing**, [S.l.], v.70, p.1342–1347, 2007.

CRONE, S. F.; HIBON, M.; NIKOLOPOULOS, K. Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. **International Journal of Forecasting**, [S.l.], v.27, p.635–660, 2011.

DELLANA, S. A.; WEST, D. Predictive modeling for wastewater applications: linear and nonlinear approaches. **Environmental Modelling & Software**, [S.l.], v.24, p.96–106, 2009.

DONATE, J. P.; SANCHEZ, G. G.; MIGUEL, A. S. de. Time series forecasting. A comparative study between an evolving artificial neural networks system and statistical methods. **International Journal on Artificial Intelligence Tools**, [S.l.], v.21, n.1, p.26, 2012.

ENDERS, W. **Applied econometric time series**. 1.ed. New York: Wiley, 1995. 433p.

FLORES, J. H. F. **Aplicação de redes neurais artificiais à previsão de vendas de máquinas agrícolas**. Porto Alegre, 2006. 60p.

FLORES, J. H. F.; PINTO, R. C.; ENGEL, P. M. Autocorrelation and partial autocorrelation functions to improve neural networks models on univariate time series forecasting. In: IEEE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, Brisbane (AU). **Proceedings...** [S.l.: s.n.], 2012.

GRANGER, C. W. J.; KING, M. L.; WHITE, H. Comments on testing economic theories and the use of model selection criteria. **Journal of Econometrics**, [S.l.], v.67, p.173–187, 1995.

GUJARATI, D. **Basic Econometrics**. 4th.ed. New York: McGraw-Hill, 2006. 812p.

HAMILTON, J. D. **Time series analysis**. 1.ed. Princeton: Princeton University Press, 1994. 799p.

HAMZAÇEBI, C.; AKAY, D.; KUTAY, F. Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. **Expert Systems with Applications**, [S.l.], v.36, p.3839–3844, 2009.

HAYKIN, S. **Redes neurais**. 2.ed. Porto Alegre: Bookman, 2001. 900p.

HEINEN, M. **A connectionist approach for incremental function approximation and on-line tasks**. 2011. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul - Instituto de Informática.

HEINEN, M.; ENGEL, P.; PINTO, R. IGMN: an incremental gaussian mixture network that learns instantaneously from data flows. In: IX ENIA - BRAZILIAN MEETING ON ARTIFICIAL INTELLIGENCE, Natal (RN). **Proceedings...** [S.l.: s.n.], 2011.

INOUE, A.; KILIAN, L. On the selection of forecasting models. **Journal of Econometrics**, [S.l.], v.130, p.273–306, 2006.

KHASHEI, M.; BIJARI, M. An artificial neural network (p, d, q) model for timeseries forecasting. **Expert Systems with Applications**, [S.l.], n.37, p.479–489, 2010.

KHASHEI, M.; BIJARI, M. Fuzzy artificial neural network (p, d, q) model for incomplete financial time series forecasting. **Journal of Intelligent and Fuzzy Systems**, [S.l.], v.26, n.2, p.14, 2014.

LIN WANG, Y. Z.; CHEN, T. Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. **Expert Systems with Applications**, [S.l.], v.42, n.2, p.8, 2015.

MAKRIDAKIS, S.; WHEELWRIGHT, S. C.; HYNDMAN, R. J. **Forecasting Methods and Applications**. 3st.ed. New York: Wiley, 1998. 642p.

MCLACHLAN, G. J.; PEEL, D. **Finite Mixture Models**. 1st.ed. New York: Wiley, 2000. 438p.

MORETTIN, P. A.; TOLOI, C. M. C. **Análise de séries temporais**. 2.ed. São Paulo: Edgard Blücher, 2006. 538p.

NETER, J. et al. **Applied Linear Statistical Models**. 4th.ed. New York: McGraw-Hill/Irwin, 1996. 1408p.

PINTO, R. C. **Online Incremental One-Shot Learning of Temporal Sequences**. 2011. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul (UFRGS).

PINTO, R.; ENGEL, P.; HEINEN, M. Echo State Incremental Gaussian Mixture Network for Spatio-Temporal Pattern Processing. In: IX ENIA - BRAZILIAN MEETING ON ARTIFICIAL INTELLIGENCE, Natal (RN). **Proceedings...** [S.l.: s.n.], 2011.

QI, M.; ZHANG, G. P. An investigation of model selection criteria for neural network time series forecasting. **European Journal of Operational Research**, [S.l.], v.132, p.666–680, 2001.

RIPLEY, B. D. **Pattern recognition and neural networks**. 1st.ed. Cambridge: Cambridge University Press, 1996. 415p.

ROHATGI, V. K. **Statistical Inference**. 1st.ed. New York: Dover, 2003. 948p.

SHUMWAY, R. H.; STOFFER, D. S. **Time series analysis and its applications**. 1.ed. New York: Springer-Verlag, 2000. 549p.

SIN, C.-Y.; WHITE, H. Information criteria for selecting possibly misspecified models. **Journal of Econometrics**, [S.l.], v.71, p.207–225, 1996.

VOYANT, C. et al. Heterogeneous transfer functionsMultiLayer Perceptron (MLP) for meteorological time series forecasting. **International Journal of Modeling, Simulation, and Scientific Computing**, [S.l.], p.100, Jan. 2015.

WANG, D.; LU, W.-Z. Forecasting of ozone level in time series using MLP model with a novel hybrid training algorithm. **Atmospheric Environment**, [S.l.], v.40, p.913–924, 2006.

YAN, W. Toward automatic time-series forecasting using neural networks. **IEEE Transaction on Neural Networks and Learning Systems**, [S.l.], v.23, n.7, p.1028–1039, 2012.

ZHANG, G. P. An investigation of neural networks for linear time series forecasting. **Computers & Operations Research**, [S.l.], v.28, p.1183–1202, 2001.

ZHANG, G. P. Time series forecasting using a hybrid ARMA and neural network model. **Neurocomputing**, [S.l.], v.50, n.1, p.16, 2003.

ZHANG, G. P.; BERARDI, V. L. Time series forecasting with neural network ensembles: an application for exchange rate prediction. **Journal of Operational Research Society**, [S.l.], v.52, p.652–664, 2001.

ZHANG, G. P.; KLINE, D. M. Quarterly time-series forecasting with neural networks. **IEEE transactions on neural networks**, [S.l.], v.18, n.6, p.1800–1814, 2007.

ZOUNEMAT-KERMANI, M.; TESHNEHLAB, M. Using adaptive neuro-fuzzy inference system for hydrological time series prediction. **Applied Soft Computing**, [S.l.], p.9, 2007.