FORECASTING INFLATION WITH ANN MODELS

Vicente Ríos Ibáñez

Documentos de Trabajo

N° 12

Banco Central del Paraguay

Documentos de Trabajo

Central Bank of Paraguay

Working papers

December 2010

Forecasting Inflation with ANN Models

Vicente Rios Ibáñez*

Msc. Macroeconomic Policy and Financial Markets (BGSE)

*Se agradecen los consejos, opiniones y ayuda prestada por los compañeros Gustavo Biedermann, Victor Ruiz, Dario Rojas, Jazmín Gustale y Carlino Velázquez. Las opiniones y errores que puedan surgir en este documento, son de exclusiva responsabilidad del autor y no compromete la posición institucional del Banco Central del Paraguay.

Forecasting Inflation with ANN Models

Vicente Rios Ibáñez

Departamento de Síntesis Macroeconómica e Investigación

Banco Central del Paraguay

December 2010

Abstract

In this research I investigate the alternative methodology of training artificial neural networks models with the early stopping procedure and I analyze their outcomes in terms of accuracy when forecasting monthly Paraguayan inflation time series. The results show that despite of neural network modelling being a competitive alternative to classical linear modelling it doesn't improve the overall forecast performance of best ARMA specifications selected through common in-sample estimation procedures, in a set of four control subsamples of 24 months each, ranging from 2002:04 to 2010:04. However, it is also a remarkable feature of all the checks performed in this research, that artificial neural network models outperform ARMA specifications in 24-steps-ahead horizon forecasts in all the subsamples of control.

1. Introduction

ANN modelling has gained attention in past years as an attractive technique for estimation and forecasting in economics. The main advantage of the artificial neural networks models is that they are free from the assumption of linearity that is commonly imposed in order to make the traditional methods tractable. Moreover, as Hornik and M-Stinchcombe and H.White (1989) Franses and Van Dijk (2000) show, neural networks are universal approximators and they can fit arbitrarily well any complex function by increasing the number of layers and neurons of the network.

Recent literature in ANN models show the great capability of ANN models in identifying behavior patterns, especially, non-linear ones, which allows the detection of non-linear dynamics and perform high accuracy forecasts. Nakamura (2005) shows that ANN methods outperform results obtained from linear autoregressive models. Moreover, Paul McNelis and Peter McAdamn (2004) show that non-linear Phillip curve specifications based on thick NN models can be competitive with the linear specification and that they perform better in periods of structural change. NN modeling also has the capability, as it has been shown in Tckaz (1999), to recognize and to model non-typical observations such as outlier behaviors or changes in the level, showing notorious advantages from what linear models can do in uncertain environments.

This document investigates alternative methods of forecasting and evaluates their performance in order to improve and complement inflation forecast exercises of Paraguay's Central Bank. The alternative approach purposed here constitutes a powerful alternative in regression standard techniques to model and forecast time series. In this research, I apply neural network-based trained models to: i) perform connections between the past and present values of the inflation time series and ii) to extract the structures and relations driving the information system associated to inflation.

In this research I proceed in the following way:

I first estimate different Autoregressive Moving Average (ARMA) models for different subsamples and I determine the most competitive specifications according to different information and goodness of fit criteria such as the Akaike's Information Criteria(AIC), the Bayesian Information Criteria(BIC), maximum likelihood (ML) and the minimum Root Mean Squared Error (RMSE).

Second, for each subsample I built and train a neural network to learn about each partition of the data with the early stopping procedure documented by Nakamura (2005) that will compete with the best ARMA(p,q) model specifications. I run a double loop in the MATLAB programming environment for lags and layers so that the training takes place accounting for all possible network configurations minimizing the in-sample RMSE for different configurations of lags and layers.

Third, I compute the out of sample root mean squared forecast error (RMSFE) for each subsample of interest for both, the ARMA(p,q) models specifications and the ANNs one. Finally, I analyze the predictive accuracy in each of them and conclude whether ANN models are competitive with ARMA(p,q) models or not.

My main finding is that Artificial Neural Networks (ANN) trained to forecast monthly inflation rate series in Paraguay with the early stopping procedure and with the learning process based on Levenger-Marquardt algorithm are a) competitive with ARMA(p,q) models but b) do not outperform univariate ARMA(P,Q) models for the all set of check subsamples analyzed according to the RMSFE measure. The later, for the case of Paraguay's inflationary rate process contradicts the intuition of superiority provided by in Tckaz (1999), Paul McNelis and Peter McAdamn (2004) Nakamura (2005), Haider and Hanif (2009), Manfred Esquivel (2009) among others, at least, when using macroeconomic Paraguayan data.

2. Artificial Neural Networks Architecture

ANN consists of an interconnected group of artificial neurons that processes information using a connectionist approach. In most cases an ANN is an adaptative system that changes its structure

based on external or internal information that flows through the network during the learning phase. It is basically a non-linear mathematical model or computational model that is inspired by the structure and functional aspects of biological neural networks of the kind of equation (1)

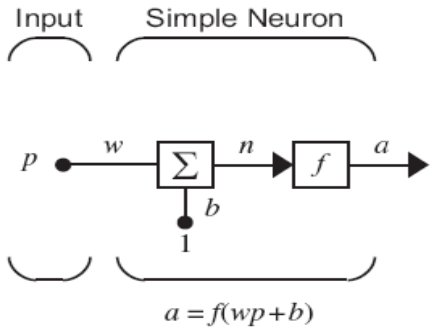
(1) $y = a^L = f^L\left(LW_{L,L-1},....,\left(f^3\left(LW_{3,2}f^2\left(LW_{2,1}f^1\left(IW_{1,1}p+b_1\right)+b_2\right)+b_3\right)\right)...+b_L\right)$ for L=1...H

Where y is the output, f is the transfer function, $LW_{i,j}$ is the layer weight, b_i is the bias and p is the matrix of inputs (for a more detailed explanation below) and H is the number of hidden layers. In this section, I am not focusing on explaining the ANN modeling theory in depth, however, I set up the basic points that will help to understand and clarify the set up of the ANN model I use to forecast inflation. For a detailed explanation in ANN theory and design see Haykin,S. (1998).

Neurons

The artificial networks are similar to the biological networks in the sense that functions are performed collectively and in parallel by the units (neurons). The word *network* in the term ANN refers to the inter-connections between the neurons in the different layers of each system being the fundamental building block for neural networks the single-input neuron, such as the one in the example that appears below:

Figure 1: Simple Neuron



Source: MATLAB NN Toolbox 7.0 User's guide

The neurons operate in the following way:

1. First, the scalar input *p* is multiplied by the scalar weight *w* to form the product *wp* which is also a scalar.
2. Second the weighted input *wp* is added to the scalar bias *b* to form the net input *n*. Bias can be seen as a shifting function f to the left by an amount *b*.
3. Finally, the net input *n* is passed through the transfer function *f*, which produces the scalar output *a*.

The names given to these three processes are: the weight function, the net input function and the transfer function. Note that *w* and *b* are both adjustable scalar parameters of the neuron. The central idea of neural networks is that such parameters can be adjusted so that the network exhibits some desired interesting behaviour. In words of Nakamura (2005): "O*ne can think about these parameters as the equivalent of estimated coefficients in linear regression models."*

Transfer functions

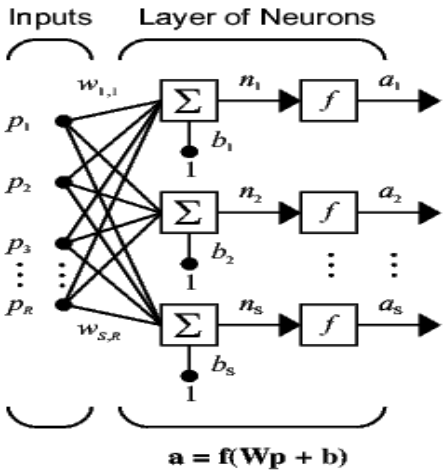
Typical transfer functions used in Neural Network Architecture to generate the output *f* are the tan-sigmoid transfer function, the log-sigmoid transfer function and the lineal transfer function. They differ in the tasks of application. Sigmoid output neurons are often used for patter recognition problems, while linear output neurons are used for function fitting problems. Commonly, *tansigmoid functions* are used as the input processing function and *purely linear functions* as the output function but modellers and researchers usually test the robustness of the results by using the network under different functional forms.

Feed-forward Networks Architectures

Feed forward networks often have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons. Multiple layers of neurons with nonlinear transfer functions allow the network to learn nonlinear relationships between input and output vectors. The linear output layer in the network is often used for function fitting and forecast. The computational advantage of ANN models is that one can pool tones of neurons and use them to work in parallel to solve specific tasks. Parallel computing is a form of computation in which many calculations are

carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel"). Two or more neurons like the one shown earlier can be combined in a layer and a particular network that could also contain one or more such layers of neurons. A one-layer network with *R* input elements and *S* neurons in the layer looks like the Figure 2

Figure 2: One layer neural system



Source: MATLAB NN Toolbox 7.0 User's guide

The input vector elements enter the network through the weight matrix W in the following way:

$$W = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \cdots & \cdots & \cdots & \cdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{pmatrix}$$

Note that the row indices on the elements of matrix *W* indicate the destination neuron of the weight, and the column indices indicate which source is the input for that weight. Thus, the indices in $w_{1,2}$ say that the strength of the signal from the second input element to the first neuron is $w_{1,2}$.

A multilayer feed-forward network is similar to a single-layer one. The main difference is that instead of having a hidden layer pass its calculated values to an output layer, it passes them on to another hidden layer. Both types of networks are typically implemented by fully connecting each

layer's neurons with the preceding layer's neurons. Thus, if Layer A has k neurons and sends its information to Layer B, with n neurons, each neuron in Layer A has n connections for its calculated output, while each neuron in Layer B has k input connections.

3. ANN Modelling Methodology

ANN modelling methodology is divided in four basic steps. The first one consists in pre-processing and scaling the data since it helps to increase the network's efficiency in computing, Haiden and Hanif (2009). The second one consists in dividing the data in different subsets as in Nakamura (2005). The first subset is usually the training set where the Networks learn. The second set is a validation set which is used to implement the early stopping. The third step involves the design of an appropriate architecture (lags and layers) and the use of a balanced algorithm to make computations on the Jacobian matrix of the objective function of the training as in Recep-Duzgun (2010). The fourth and final step is the ANN model selection.

Data pre-processing and scaling

The first steps in ANN modelling are pre-processing and scaling the data. Transformation, normalization and data smoothing are three common ways of pre-processing data. Transformation and normalization makes the statistical distribution of each input and output data roughly uniform. The values are scaled to match the range that the input neurons use. Data normalization methods, which include simple linear scaling and statistical measures of central tendency and variance, remove outliers and spread out the distribution of the data. Data smoothing filters out noise in the data.

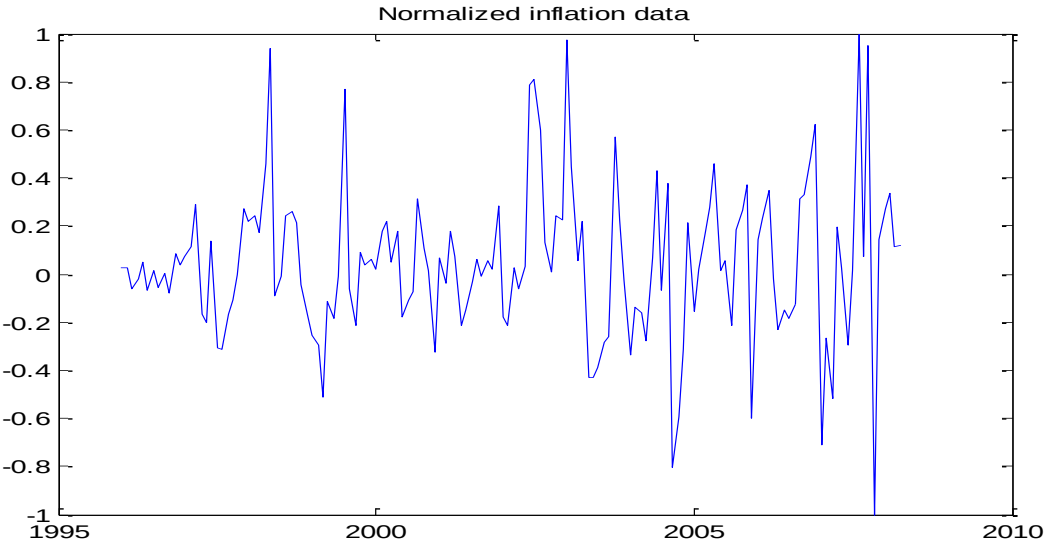
The reason why pre-processing the data is useful is that it makes more efficient the neural network behaviour. For example, in multilayer networks sigmoid transfer functions are generally used in hidden layers. The bad news is that these functions become saturated when the input is greater than three. (i.e, $\exp(-3) \cong 0.05$) and if this happens at the beginning of the training

process problems of speed in computing or of the nature as the ones documented by Judd (1998) can arise. Thus, a solution is to normalize the inputs before applying them to the network. I apply the normalization to both, the input and the output vector using the following linear scaling function:

$$(1)\quad x^{*}_{kt} = 2\frac{x_{kt} - \min(x_k)}{\max(x_k) - \min(x_k)} - 1$$

The results of applying this scaling function ensures the data will fall in the interval [-1,1]. The inflation series derived from this normalization exhibits an almost chaotic pattern and which can be seen in Graph 1.

Graph 1: Inflation normalized series



Source: Own elaboration with BCP data

Dividing the data

The second step when training multilayer networks consists in dividing the data into two subsets, training and validation set. The first subset is the training set, which is used for computing the gradient and updating the network weight and biases. In this research I adjust the quantity of observations that each network reads from the training set to the 80% of available data.

The error on the validation set is monitored during the training process. The validation error normally decreases during the initial phases of the training process as it does the training set error. However, when the network begins to over-fit the data, the error on the validation set typically begins to rise and the network weights and biases are saved at the minimum of the validation set error. This practice is also known as "early-stopping" and its benefits in avoiding over-parameterization are well explained in Nakamura (2005).

For practicing early stopping, the researcher has to be careful not to use algorithms that converge too rapidly to the goal of the training. If one is using a fast algorithm like the LM, the training parameters have to be set such that the convergence is relatively slow. In this sense, LM algorithm has been documented by the literature to be one of the most balanced ones in terms of computational burden and accuracy of the results (see for example the NN MATLAB toolbox for a great analysis between the time-accuracy trade-off that researchers face when choosing algorithms for the learning process).

Learning

Learning in ANN models consists in updating the connections between neurons and layers. The flow of information in a network can be represented in a general form as:

(2) $Wa=b$

Where W is an n x k matrix. The learning process implies modifications of each w_{ji} in W. A similar mathematical analogy applies to multilayer feed-forward networks, but in that case, there is a W for every layer and ''b'' is used as the value for ''a'' when moving to subsequent layers. Several algorithms can be used to compute the gradient performance function to determine how to adjust the weights to optimize performance.

The simplest implementation of back-propagation (or gradient descent) techniques for learning, updates the network weights and biases in the direction in which the performance function

decreases most rapidly, the negative of the gradient. The way the matrix of coefficients evolves through the learning phase can be characterized by the following equation:

(3) $W_{k+1} = W_k - \alpha_k B_k$

In this research I use an algorithm that mutes from Newton's method to steepest descent to adjust the weights of the matrix W minimizing the RMSE. The feed forward multi layer network algorithm that trains my net is the Levenberg-Marcquardt (1946). I use this technique due to that it is considered more powerful and faster than the conventional gradient descent technique (Hagan and Menhaj, (1994); Kisi (2007)). It is a high-performance algorithm that can converge from ten to one hundred times faster than the algorithms discussed previously. The way the algorithm concretely works appears in equation (10) below. If I want to minimize a function f(θ) with respect the parameter vector θ, the Newton's method would be:

(4) $\Delta\theta = -[\nabla^2 f(\theta)]^{-1} \nabla f(\theta)$

The Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feedforward networks), then the Hessian matrix can be approximated. Let $\nabla^2 f(\theta)$ is the Hessian matrix and $\nabla f(\theta)$ the gradient. Now, assume that $f(\theta)$ is the sum of squares functions:

(5) $f(\theta) = \sum_{i=1}^{N} e_i^2(\theta)$, then:

(6) $\nabla f(\theta) = J^T(\theta) e(\theta)$ and

(7) $\nabla^2 f(\theta) = J^T(\theta) J(\theta) + S(\theta)$ with

(8) $S(\theta) = \sum_{i=1}^{N} e_i \nabla^2(\theta) e_i(\theta)$ and J the jacobian matrix.

The main difference between Gauss-Newton (GN) method and Levenberg-Marquardt(1946) is that it in GN is assumed $S(\theta) \approx 0$ with:

(9) $\nabla(\theta) = [J^T(\theta)J(\theta)]^{-1} J^T(\theta)e(\theta)$

while Levenberg-Marquardt (1946) modification to the GN method is:

(10) $\nabla(\theta) = [J^T(\theta)J(\theta) + \mu]^{-1} J^T(\theta)e(\theta)$ $\nabla(\theta) = [J^T(\theta)J(\theta) + \mu]^{-1} J^T(\theta)e(\theta)$.

If μ is big, the algorithm becomes steepest descent and if it is small it becomes Gauss method. When μ is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift toward Newton's method as quickly as possible. The term μ can therefore be controlled to ensure descent even when second-order terms, which restrict the efficiency of the Gauss-Newton method, are encountered. Therefore, Levenberg-Marquardt method uses a search direction that is a cross between the Gauss-Newton direction and the steepest descent direction.

Training and early stopping procedure

ANN specification and training involves the specification of the following parameters (as in the MATLAB NN Toolbox 7.0): a) the number of hidden layers, b) the maximum lag, c) the training set, d) the forecast period, e) the learning rate of the network, f) the learning increment, g) the learning decrement, h) the number of training parameters epochs and i) the target RMSE. Table I shows the ANN selection of parameter values to configure the network.

One of the main issues needed to comment is related to the learning rate of the network. Picking the learning rate for a nonlinear network is a challenge. As with linear networks, a learning rate that is too large leads to unstable learning. If the learning rate is set too high, the algorithm can oscillate and become unstable. If the learning rate is too small, the algorithm takes too long to converge. It is not practical to determine the optimal setting for the learning rate before training, and, in fact, the optimal learning rate changes during the training process, as the algorithm moves across the performance surface. The value used for all Network configurations is 0.95.

Table 1: Multilayer Feed-Forward Network configuration

Parameter	Value
Hidden layers	from 1 to 12
Max lag	from 1 to 12
Training set	0.8
Forecast period	24
Learning rate	0.95
Learning increment	0.8
Learning decrement	0.4
Training parameter epochs	1000
Target RMSE	0.005

The idea now is to determine the number of layers and lags of the ANN system optimally since that will deliver the final configuration of the model to run forecasts. In the early stopping approach validation, vectors are used to stop training early if the network performance on the validation vectors fails to improve or remains the same for a certain number of epochs in a row and test vectors are used as a further check that the network is generalizing well but do not have any effect on training.

The procedure goes as follows: First, input and ouput vectors are read into the model, then, after randomly selecting a set of parameters (I use the pseudo-random number generator of MATLAB), the network processes the inputs and generates a predicted output vector. After that, a mean square error (RMSE) is generated by comparing such output vector with the time series of actual data through the different sub-divisions of the data as it can be seen in graph xx in NN Appendix.

Then the network adjusts the initial set of parameters in the direction of the negative gradient of the RMSE, produces a new output vector, calculates a new RMSE, adjust the parameters and so on. (in next lines ending criteria in NN training are defined). The adaptive value μ is increased by $\Delta\mu$ or more concretely 0.8 until the change above results in a reduced performance value. The change is then made in the network and mu is decreased by $\nabla\mu$ meaning 0.4. Since I'm using a fast algorithm, I set the training parameters so that the convergence is relatively slow. For example, I set mu to a relatively large value, such as 0.95 or 1, and set mu_dec and mu_inc to

values close to 1, such as 0.7 and 1.05, respectively. Training occurs according to training parameters with the following values:

Table 2: Levenberg and Marquardt Algorithmic learning process parameterization

Parameter	Value
Max epochs	1000
MSE	0.005
Max fail	5
Min $\nabla(\theta)$	1,00E-10
μ	1
$\nabla\mu$ *	0.7
$\Delta\mu$ *	1.05
Max μ *	1,00E+10
Time (t)	Inf

Training stops when any of these conditions occurs:

1) The maximum number of EPOCHS (repetitions) is reached.

2) The maximum amount of time has been exceeded.

3) Performance has been minimized to th goal.

4) The performance gradient falls below min $\nabla(\theta)$.

5) μ exceeds Max μ *.

6) Validation performance has increased more than max fail time since the last time it decreased (when using validation).

NN selection

The generic model specification I estimate and from which I compute the RMSE reads as:

(11) $$y_t = F(X_t, \Theta)$$

(12) $$F(X_t, \Theta) = \sum_{k=1}^{Q} \beta_k g\left(\gamma_k Z_t\right) + \varepsilon_t$$

With $Z_t \subseteq X_t, X_t = \{y_{t-1},...,y,w_1,....,w_m\}$ with $y_{t-j}, j=1,...,p$ lags of the dependent variable, $w_j, j=1,...,m$ exogenous variables and tanh (u) the transfer function

$$\tanh(u) = \frac{e^{2u}-1}{e^{2u}+1} \quad (13)$$

In the learning process of each of the 12 lags x 12 layers configurations of NN I train the network until it satisfies one of the former conditions. After that, I use the estimated coefficients of the trained network to simulate data corresponding to the validation and test periods and I compute the root mean squared error.

For each simulation I store the RMSE and then I compare all of them to select the one that performed better, with the lowest RMSE value. The training process produces the different optimal configurations for the different sets as it is shown in Table XI below.

Table 3: Selected NN Networks

Subsample of forecast	Selected Competing Networks	RMSE
2002:04-2004:04	(3 , 5)	0,156
2004:04-2006:04	(4 , 4)	0,22
2006:04:2008:04	(4 , 12)	0,10
2008:04-2010:04	(3 , 8)	0,135

The optimal configurations of layers and lags for the ANN system I found for the different subsamples show that most of estimated optimal ANN systems usually have three or four layers with a varying number of lags.

4. ARMA methodology

Data analysis

A lengthly time series of data is required for univariate time series forecasting. It is usually recommended that at least 50 observations to be available and this is the case here since our data set consists on a vector of 172x1. As it can be seen in Table I the monthly inflation mean for the

whole sample period under study is 0.659%. We make a partition of the sample set in order to check whether radical changes in the behaviour of the series have occurred or not. Mean and standard deviation statistics for 1995-2003 and 2003-2010 show that while the period between 1995-2003 was more inflationary than 2003-2010 the latter has been more volatile. However, further statistical tests (i.e Chow test or Andrews) should be taken to determine the existence of breaks, I avoid them here. In the trade-off an econometrician has to face between quantity of observations and regime stability, I chose a higher quantity of observations and I use all of them for estimation.

Table 4: Inflationary process stats

Period	1995-2010	1995-2003	2003-2010
Mean	0,00659117	0,00796097	0,00625369
Standard Deviation	0,00954361	0,00845609	0,00979184

Testing for stationarity

In order to estimate an ARIMA model is first necessary to test the stationarity of the time series and to check if differencing is required. To proceed in this direction one can run Augmented Dickey Fuller test. The testing procedure for the ADF test is the same as for the Dickey–Fuller test but it is applied to the model:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \cdots + \delta_p \Delta y_{t-p} + \varepsilon_t, \quad (14)$$

where α is a constant, β the coefficient on a time trend and *p* the lag order of the autoregressive process. Imposing the constraints α = 0 and β = 0, corresponds to modelling a random walk and using the constraint β = 0, corresponds to modelling a random walk with a drift. Consequently, there are three main versions of the test. For all the subsamples of data I found that in general, inflation is statistically considered an stationary process.

Model estimation: Maximum Likelihood estimation

Maximum likelihood estimation is a cornerstone of modern inferential procedures. To give a foundation of the estimation procedure we start forming a brief reminder on the basics of ML estimation. Given a sample size T, it is possible to define the density function. for the whole sample, namely the joint distribution of all the observations $f(Y;\theta)$, where $Y=\{y_1,...,y_T\}$ It's shape is determined by a k-vector of unkown parameters θ which we assume is contained in ta set Θ and which can be used to evaluate the probability of observing a sample with any given characteristics.

After observing the data, the values Y are given, and this function can be evaluated for any legitimate value of θ. In standard cases, this function has a unique maximum. The location of the maximum is unaffected if we consdir the logarithm of the likelihood or log-likelihood: fhis function will be denoted as: $l(\theta)=\log f(Y;\theta)$. The functions lt($\theta$) are called log-likelihood contributions: $l(\theta)=\sum_{t=1}^{T} l_t(\theta)$. Moreover, the location of the maximum is obviously determined by the data. This means that the value: $\hat{\theta}(Y)=ArgMax_{\theta\in\Theta} l(\theta)$ is some function of the observed data, which has the property, under mild conditions, of being a consistent, asymptotically normal and asimpottitcally efficient estimator of θ.

Sometimes it is possible to write down explictly the function $\hat{\theta}(Y)$ but in general it need not be since in these circumstances, the maximum can be found by means of numerical techniques. These often rely on the fact that the log-likelihood is a smoth function of θ, and therefore on the maximum its partial derivatives should all be 0. The gradient vector, is a k-vector $g(\theta)$ with typical element:

(15) $$g(\theta_i)=\frac{\partial l(\theta)}{\partial \theta_i}=\sum_{t=1}^{T}\frac{\partial l_t(\theta)}{\partial \theta_i}$$

The gradient-based algorithm used to estimate the likelihood function for different combinations of the parameters can be shortly illustrated as follows:

1. Pick a point θ_0
2. Evalaute $g(\theta_0)$
3. If $g(\theta_0)$ is samll, stop. Otherwise, compute a direction vector $d(g(\theta_0))$
4. Evalaute $\theta_1 = \theta_0 + d(g(\theta_0))$
5. Subsistute θ_0 with θ_1;
6. Restart from 2.

Model selection: Penalty functions

Because of the highly subjective nature of the traditional Box-Jenkins methodology, time series analysts have sought alternative objective methods for identifying ARIMA models. Objective penalty criteria have been used by some authors instead o the traditional Box-Jenkins (1976) procedure. For examples of the use of objective penalty functions criteria see Gómez and Maravall (1998) or Meyler, Kenny and Quinn (1998).

Penalty function statistics such as Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC) among others have been used to assist time series analysts in reconciling the need to minimize errors with the conflicting desire for model parsimony. These statistics all take the form of minimizing the sum of the residuals sum of squares plus a 'penalty' term which incorporates the number of estimated parameters coefficients to factor in model parsimony.

My approach is to test the ability of several combinations of AR and MA parameters in capturing the main features of the data for information criteria such as the Akaike Information and the Bayesian information one.

For the ARMA(p,q) process, I specify 100 models, that combine AR and MA parameters for p=1,2,3,…,10 and q=1,2,3…,10 and compute the values for these information criteria of model selection that are specified below for each subsample of interest.

$$BIC(p) = \ln|\Sigma(p)| + \frac{\ln T}{T} pn^2$$

$$AIC(p) = \ln|\Sigma(p)| + \frac{2}{T} pn^2$$

I also run the test for pure AR(p) and MA(q) processes, however forecasting results are poor relative to ARMA in almost all subsamples of interest and I will take them out of the analysis. In the following part of this section I will proceed by reporting the results obtained in the model selection for each subsample according to information criteria commented before (For a more detailed explanation see ARMA estimation Appendix).

Table 5: Selected ARMA (P,Q) models

Subsample of forecast	Selected competing ARMA (P,Q) models
2002:04-2004:04	(7,9) (9,5) (1,1) (10,9)
2004:04-2006:04	(7,6) , (1,1) (8,7) (10,9)
2006:04:2008:04	(8,8) (2,1) (9,5)
2008:04-2010:04	(10,9) (2,2) (8,7)

5. Forecast results

Two program codes are written in MATLAB language for the design and direct forecast computing of the Feed-Forward network and ARMA approach to time series modelling. I built 12-lags*10-layers networks and I train them with the LM algorithm for different subsamples of analysis. The idea is to replicate the situation that policy makers faced at the moment of forecasting with ARMA models and to check whether forecasts based on NN would have delivered more accurate results. The way to proceed is straightforward. I first compute the optimal network configuration through the in sample RMSE measure and second I use that

configuration to perform forecasts computing the RMSFE for the periods of out-of-sample test for the NN and ARMA models selected before. The RMSFE is defined as:

$$RMSFE(n) = \sqrt{(T)^{-1}\sum_{i}^{T}\left(y_i^a - y_i^f(n)\right)^2} \quad (16)$$

Where y^a is the actual value of inflation, y^f the forecast valued of inflation and T are the number of periods.

In the remaining of this section I will proceed providing the results of each process of neural network configuration and training together with a comparison between the more in-sample accurate ARMA(p,q) models selected in section 4 and ANN models selected in section

First Subsample: 2002:04-2004:04

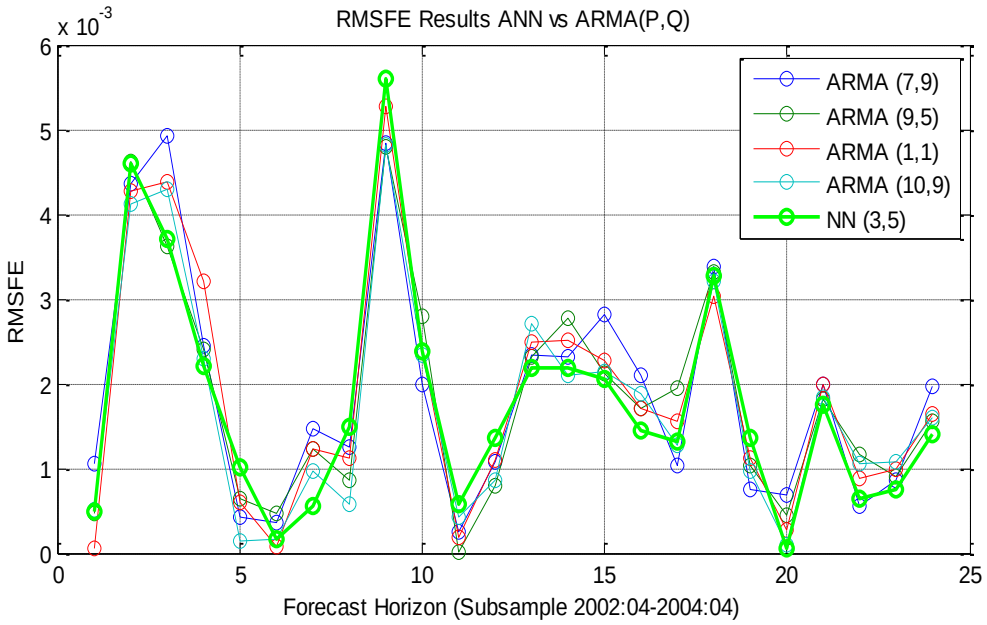
The best model forecasting out of sample in this subsample of forecast 1 is the ARMA (10, 9). However the forecasts of all the models tend to deliver the same errors as it can be seen in Graph 2, where the evolution of RMSFE along the sample of control of all models is reported

Table 6: RMSFE results comparison.

RMSFE	ARMA(7,9)	ARMA(9,5)	ARMA(1,1)	ARMA(10,9)	NN(3,5)
horizon 1	0,0010523	0,000474435	**5,29E-05**	0,00050048	0,00048064
horizon 4	0,0024547	0,002400895	0,00321557	0,002283409	**0,00221386**
horizon 12	0,0010845	**0,000802928**	0,00109589	0,000855934	0,00135465
horizon 24	0,0019573	0,00156342	0,00164656	0,001598565	**0,00139734**
Average	0,0018829	0,001824925	0,00184996	0,001724682	0,00177128

In the first subsample the NN (3,5) is the second best model among the set of competing models with an average RMSFE of 0.00177 and it produces the most accurate forecasts for the 4months and 24 months horizons. Meaning even with mistakes the network is accurate generalizing the long term pattern of the data.

Graph 2: First subsample RMSFE ANN vs ARMA(P,Q)



Source: Own elaboration with BCP data

For the first horizon the best model in terms of accuracy is a parsimonious ARMA(1,1) specification chosen by the BIC criteria while for the a horizon of 12 periods the best model is the ARMA(9,5) chosen by the AIC criteria.

Second Subsample 2: 2004:04-2006:04

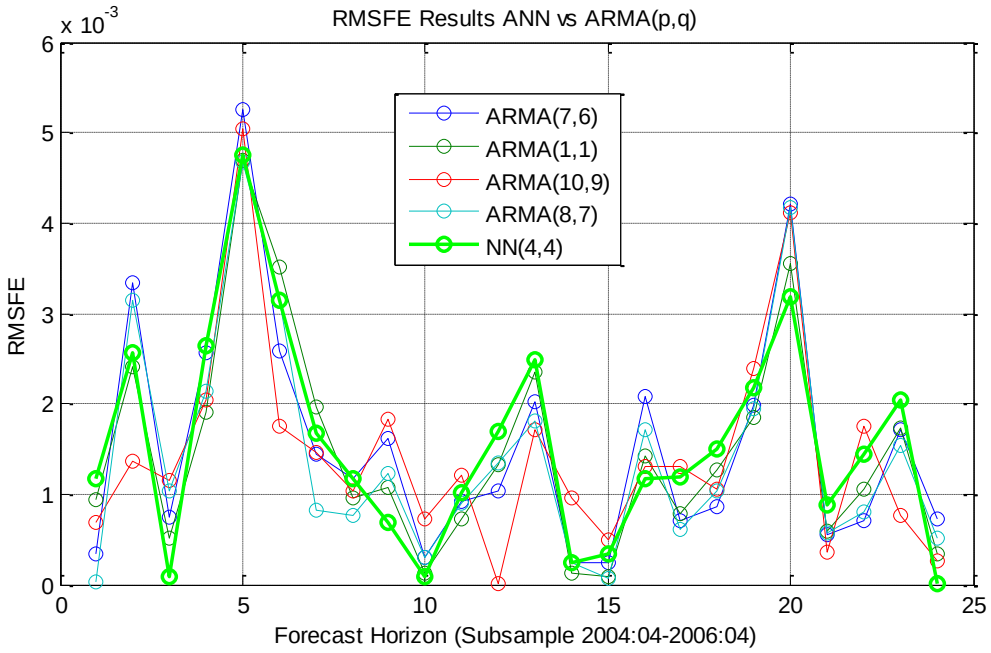
The best model forecasting out of sample in this subsample of forecast 2 is the ARMA (10, 9). However the forecasts of all the models tend to deliver the same errors as it can be seen in Table 7 below. In this subsample the neural network delivers the worst forecast of all of the competing modes.

Table 7: RMSFE results in the second subsample

RMSFE	ARMA(7,9)	ARMA(9,5)	ARMA(1,1)	ARMA(10,9)	NN(3,5)
horizon 1	0,0003477	0,000931	0,0006833	**3,12E-05**	0,00116
horizon 4	0,0025675	**0,0019122**	0,0020474	0,00214343	0,00264
horizon 12	0,0010352	0,0013312	**3,80E-06**	0,00134362	0,0017
horizon 24	0,000717	0,0003477	0,0002662	0,00051575	**2,50E-06**
Average	0,0015563	0,0014706	0,0014488	**0,00143897**	0,00156

For the first horizon the best model is the ARMA(10,9) . For the horizon of four periods the best model is the ARMA (9,5) while for the 12 month horizon the best model is an ARMA(1,1). Again, for the 2 years ahead forecast NN gets the best result.

Graph 3: RMSFE Results ANN vs ARMA(P,Q)



Source: Own elaboration with BCP data

Forecast Subsample 3: 2006:04-2008:04

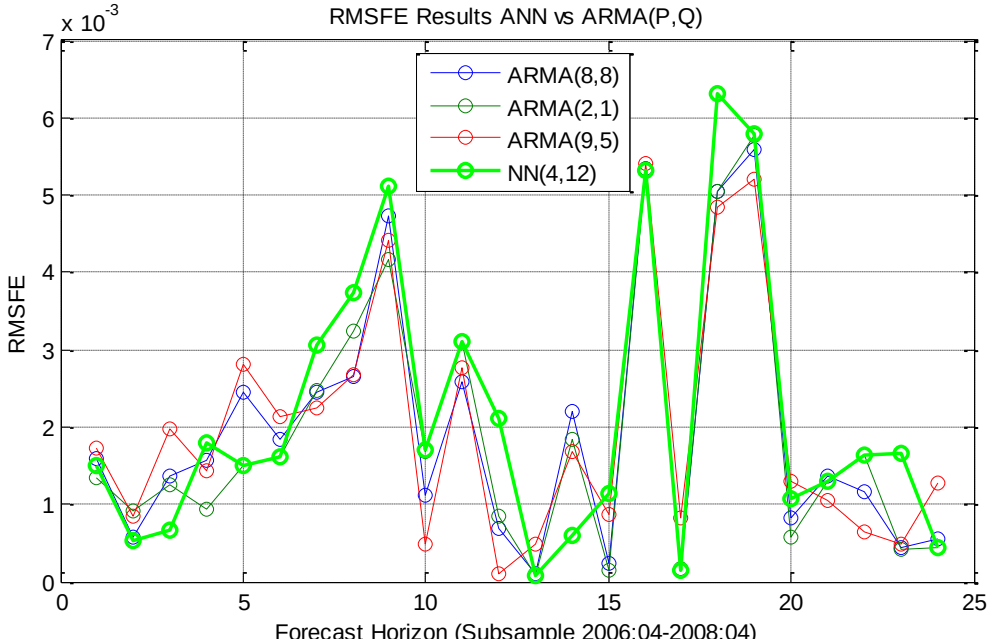
In the third subsample the winner model is the ARMA(2,1) chosen by the BIC basically because of the low errors forecasting the first periods. Concretely is the best model when forecasting 1 and 4 horizons ahead.

Table: 8: RMSFE results in the third subsample

RMSFE	ARMA(8,8)	ARMA(2,1)	ARMA(9,5)	NN(4,12)
horizon 1	0,00158198	**0,00133942**	0,00172615	0,00151202
horizon 4	0,00157054	**0,00092989**	0,00143437	0,00179896
horizon 12	**0,00067808**	0,00085514	0,00010733	0,00210491
horizon 24	0,00055003	0,00044976	0,00127978	**0,00044848**
Average	0,00194081	**0,00190952**	0,00198767	0,00216457

The ANN(4,12) model performs 8.8% worse than the ARMA(2,1) model on average but stills delivering the best forecast when forecasting 24 months. On the other hand ARMA(8,5) was the best model to forecast inflation 12 months ahead.

Graph 4: RMSFE ANN VS ARMA (P,Q)



Source: Own elaboration with BCP data

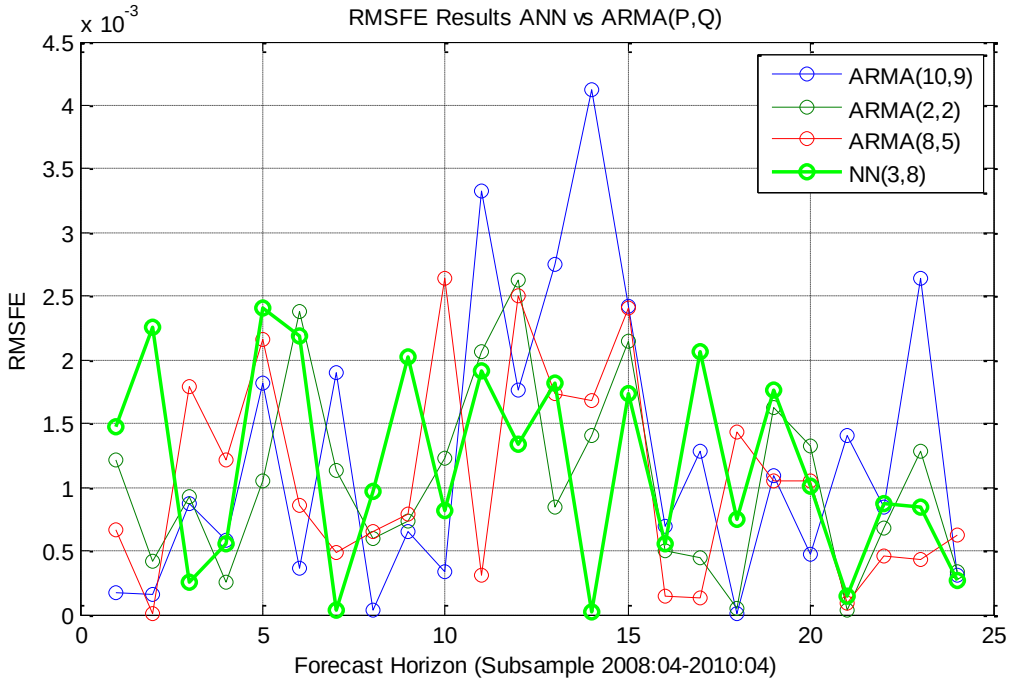
Forecast Subsample 4: 2008:04-2010:04

In the fourth subsample of test I found that the best model forecasting is the ARMA(8,7) selected by the RMSE criteria. The NN(3,8) gets the third place with an average RMSFE OF 0.00117. As in all the other subsamples they outperform ARMA(P,Q) specifications in the long run step ahead forecast.

Table 9: RMSFE results in the fourth subsample

RMSFE	ARMA(10,9)	ARMA(2,2)	ARMA(8,7)	NN(3,8)
horizon 1	0,00016826	0,00121882	0,00067017	0,00147262
horizon 4	0,00058856	0,00025881	0,00120862	0,0005503
horizon 12	0,00176785	0,00262773	0,00250007	0,00134093
horizon 24	0,00030947	0,00034226	0,00062254	0,00026343
Average	0,00125169	0,00105433	0,00105401	0,00117024

Graph 5: RMSFE Results ANN vs ARMA(P,Q)



Source: Own elaboration with BCP data

6. Conclusion

In this research I have studied the ability in forecasting monthly inflation of NN models that have been trained implementing the early stopping procedure and that have learned through the LM algorithm, finding that this common NN approach outperforms ARMA models in 24 step ahead out of sample direct forecast. However, for all the subsamples of test the RMSFE of optimal NN configurations tends to be higher on average. Despite being a competitive alternative to ARMA(P,Q) models chosen under procedures such as the AIC, BIC, Likelihood or RMSE, the results shown here contradict the intuition of superiority in out-of-sample forecasts provided

by in Tckaz (1999), Paul McNelis and Peter McAdamn (2004) Nakamura (2005), Haider and Hanif (2009), Manfred Esquivel (2009) among others. Further research using abundant exchange rate data as target and with fuzzy logic as inference engine for the network, is recommended to explore the full potential of this method, given that for monthly frequency ANN classical modeling does not deliver an improvement in forecasts accuracy.

7. References

Álvaro Solera Ramírez (2005) *Pronostico de la Inflación en Costa Rica: Una estimación con redes neuronales artificiales*

Emi Nakamura (2005). *"Inflation Forecasting using a Neural Network"*

Franses and Van Dijk (2000). *"Non linear time series models in empirical finance"*

Gomez, Victor & Maravall, Agustin & Pena, Daniel, (1998). *"Missing observations in ARIMA models: Skipping approach versus additive outlier approach"*

Hagan and Menhaj, (1994); *"Training Feed Forward Networks with the Marquardt Algorithm"*

Haider, Adnan and Hanif, Muhammad Nadeem (2009). "*Inflation forecasting in Pakistan using Artificial Networks"*

Haykin, S., 1998. Neural Networks - *A Comprehensive Foundation*

Hornik, K., M., Stinchcombe, and H., White, (1989). *"Multilayer Feed Forward Networks are Universal Approximators"*

Kenny and Quinn (1998). "*Forecasting Irish Inflation Using ARIMA Models"*

Kisi, Ö., 2007. *"Streamflow Forecasting using Different Artificial Neural Network Algorithms*

Manfred Esquive (2009) *"Performance of ANN in forecasting Costa Rica Inflation"*

Paul McNelis & Peter McAdam, *(*2004). *"Forecasting inflation with thick models and neural networks,"*

Recep Düzgün (2010) "*Generalized Regression Neural Networks for Inflation Forecasting"*

Serju, P., *2002. "Monetary Conditions & Core Inflation: An Application of Neural Networks",*

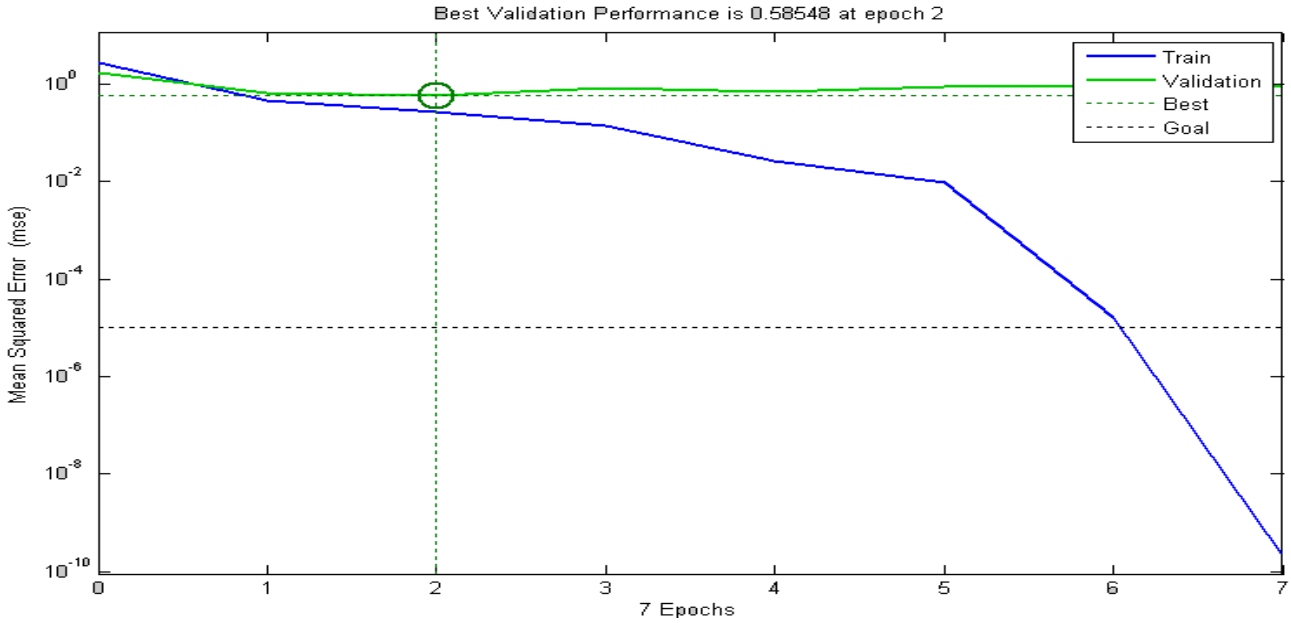
Stock, J. H., and M. W., Watson, (1999). *"A Comparison of Linear and Non-Linear Univariate Models for Forecasting Macroeconomic Time Series"*

Tkacz, G., 2001. "*Neural Network Forecasting of Canadian GDP Growth*"

8. NN APPENDIX

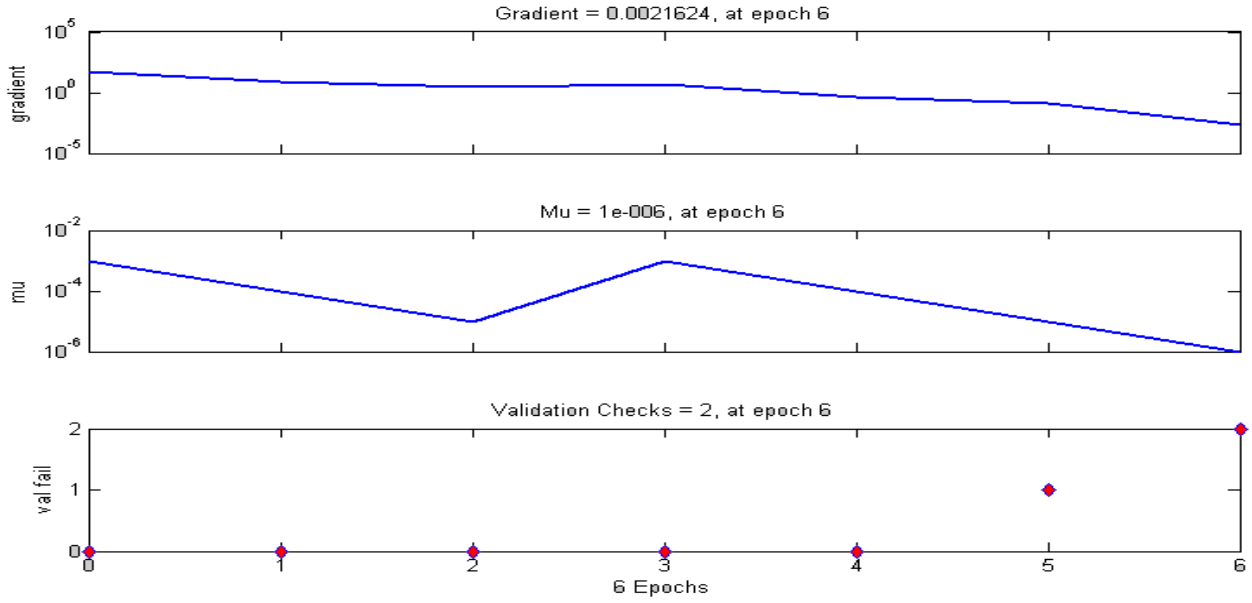
8.1 TRAINING AND LEARNIN

Graph 6: Training process



Source: Own elaboration with BCP data

Graph 7: Training state



Source: Own elaboration with BCP data

8.2 NN Model selection

Table 10: RMSE First subsample 1996-2002:04

RMSE Matrix	1 Lag	2 Lags	3 Lags	4 Lags	5 Lags	6 Lags	7 Lags	8 Lags	9 Lags	10 Lags	11 Lags	12 Lags
1 Layer	0,175	0,167	0,17	0,174	0,172	0,181	0,168	0,169	0,182	0,1686	0,159	0,177
2 Layers	0,1722	0,168	0,166	0,18	0,173	0,172	0,166	0,172	0,169	0,2178	0,178	0,162
3 Layers	0,1761	0,164	0,179	0,165	**0,156**	0,198	0,172	0,194	0,185	0,1692	0,169	0,161
4 Layers	0,1811	0,204	0,171	0,177	0,181	0,18	0,175	0,178	0,167	0,1788	0,204	0,207
5 Layer	0,1703	0,183	0,177	0,174	0,178	0,177	0,18	0,178	0,191	0,2147	0,192	0,241
6 Layers	0,1724	0,173	0,179	0,175	0,207	0,178	0,174	0,187	0,187	0,1963	0,17	0,186
7 Layer	0,2173	0,188	0,189	0,203	0,233	0,18	0,192	0,187	0,184	0,2395	0,241	0,168
8 Layers	0,2079	0,19	0,179	0,173	0,179	0,178	0,203	0,209	0,226	0,2233	0,19	0,199
9 Layer	0,2767	0,206	0,188	0,208	0,179	0,199	0,211	0,188	0,193	0,2088	0,169	0,202
10 Layers	0,1785	0,18	0,203	0,22	0,176	0,225	0,206	0,213	0,187	0,2441	0,187	0,229

Table 11: Second Subsample 1996:2004:04

RMSE Matrix	1 Lag	2 Lags	3 Lags	4 Lags	5 Lags	6 Lags	7 Lags	8 Lags	9 Lags	10 Lags
1 Layer	0,243	0,25	0,26	0,253	0,254	0,266	0,284	0,292	0,282	0,292
2 Layers	0,248	0,252	0,268	0,26	0,267	0,258	0,264	0,309	0,271	0,267
3 Layers	0,273	0,248	0,268	0,246	0,297	0,255	0,266	0,297	0,252	0,276
4 Layers	0,275	0,277	0,284	**0,227**	0,251	0,268	0,288	0,257	0,369	0,387
5 Layer	0,273	0,253	0,27	0,282	0,247	0,253	0,271	0,259	0,325	0,329
6 Layers	0,328	0,491	0,283	0,273	0,276	0,332	0,284	0,317	0,326	0,292
7 Layer	0,348	0,268	0,389	0,264	0,382	0,273	0,3	0,294	0,337	0,326
8 Layers	0,268	0,358	0,343	0,353	0,264	0,332	0,363	0,336	0,552	0,33
9 Layer	0,331	0,287	0,384	0,267	0,302	0,335	0,335	0,331	0,402	0,335
10 Layers	0,364	0,314	0,296	0,361	0,317	0,308	0,676	0,333	0,329	0,299
11 Layers	0,315	0,414	0,378	0,267	0,367	0,356	0,305	0,367	0,558	0,527
12 Layers	0,314	0,275	0,357	0,474	0,294	0,351	0,415	0,523	0,364	0,367

Table 12: RMSE Third subsample, 1996-2006:04

RMSE	1 Lag	2 Lags	3 Lags	4 Lags	5 Lags	6 Lags	7 Lags	8 Lags	9 Lags	10 Lags	11 Lags	12 Lags
1 Layer	0,123	0,128	0,127	0,123	0,13	0,123	0,124	0,13302	0,128	0,1167	0,1326	0,12
2 Layers	0,125	0,129	0,127	0,141	0,125	0,133	0,127	0,13488	0,122	0,1414	0,1311	0,1238
3 Layers	0,121	0,122	0,127	0,144	0,127	0,122	0,137	0,12901	0,131	0,1277	0,1281	0,1303
4 Layers	0,345	0,12	0,125	0,134	0,145	0,117	0,139	0,12732	0,13	0,1284	0,1393	**0,1062**
5 Layer	0,125	0,133	0,147	0,126	0,129	0,135	0,128	0,17584	0,147	0,1263	0,1295	0,1315
6 Layers	0,135	0,127	0,117	0,136	0,133	0,154	0,143	0,17356	0,156	0,167	0,1361	0,1063
7 Layer	0,123	0,224	0,128	0,147	0,155	0,167	0,128	0,16466	0,159	0,3382	0,1678	0,1227
8 Layers	0,122	0,118	0,227	0,254	0,152	0,153	0,138	0,15104	0,184	0,155	0,1774	0,1386
9 Layer	0,132	0,161	0,209	0,136	0,153	0,183	0,17	0,13585	0,213	0,136	0,2003	0,1379
10 Layers	0,119	0,162	0,153	0,178	0,22	0,186	0,249	0,13589	0,149	0,1612	0,2065	0,1689
11 Layers	0,146	0,156	0,16	0,129	0,178	0,209	0,149	0,13238	0,218	0,1954	0,1631	0,1417
12 Layers	0,346	0,294	0,165	0,161	0,149	0,127	0,211	0,16641	0,169	0,1735	0,1847	0,1769

Table 13: RMSE Fourth subsample, 1996-2008:04

RMSE	1 Lag	2 Lags	3 Lags	4 Lags	5 Lags	6 Lags	7 Lags	8 Lags	9 Lags	10 Lags	11 Lags	12 Lags
1 Layer	0,195	0,193	0,179	0,18	0,18	0,186	0,187	0,17943	0,184	0,181	0,18	0,1771
2 Layers	0,198	0,188	0,187	0,183	0,193	0,188	0,177	0,18337	0,172	0,1699	0,1788	0,1703
3 Layers	0,213	0,201	0,38	0,181	0,176	0,176	0,164	**0,1351**	0,163	0,2124	0,1873	0,1628
4 Layers	0,187	0,191	0,177	0,183	0,182	0,181	0,177	0,23122	0,156	0,1906	0,1672	0,1845
5 Layer	0,364	0,191	0,267	0,18	0,203	0,185	0,175	0,17344	0,17	0,1843	0,1835	0,1766
6 Layers	0,273	0,227	0,189	0,183	0,198	0,201	0,161	0,16269	0,177	0,1711	0,1714	0,1684
7 Layer	0,328	0,219	0,161	0,188	0,205	0,192	0,19	0,20077	0,169	0,1736	0,184	0,1907
8 Layers	0,49	0,19	0,22	0,192	0,188	0,215	0,185	0,21274	0,175	0,1881	0,1518	0,206
9 Layer	0,177	0,222	0,215	0,178	0,193	0,179	0,187	0,21627	0,181	0,1788	0,1954	0,1611
10 Layers	0,297	0,193	0,266	0,194	0,222	0,189	0,199	0,18111	0,193	0,1736	0,2115	0,1747
11 Layers	0,25	0,266	0,222	0,181	0,265	0,168	0,2	0,17715	0,226	0,2174	0,1859	0,1533
12 Layers	0,245	0,257	0,176	0,151	0,435	0,191	0,198	0,21308	0,2	0,2274	0,2554	0,1668

8 ARIMA ESTIMATION APPENDIX

9.1 First subsample, 1996-2002:04

Table 14: Augmented Dickey Fuller tests results

Test: ADF AR model	1 Lag	2 Lags	3 Lags	4 Lags	5 Lags	6 Lags	7 Lags	8 Lags	9 Lags	10 Lags	11 Lags	12 Lags
Critical Value	-1,94	-1,94	-1,94	-1,94	-1,94	-1,94	-1,94	-1,94	-1,94	-1,94	-1,94	-1,94
T-Statistic	-4,93	-4,58	-3,62	-2,95	-2,45	-2,44	**-1,84**	**-1,67**	**-1,48**	**-1,64**	-2,01	**-1,64**
ADF AR with drift	1 Lag	2 Lags	3 Lags	4 Lags	5 Lags	6 Lags	7 Lags	8 Lags	9 Lags	10 Lags	11 Lags	12 Lags
Critical Value	-2,88	-2,88	-2,88	-2,88	-2,88	-2,88	-2,88	-2,88	-2,88	-2,88	-2,88	-2,88
T-Statistic	-7,45	-7,59	-6,47	-5,59	-4,92	-5,18	-4,06	-3,8	-3,5	-3,99	-5,28	-4,44
ADF trend stationary	1 Lag	2 Lags	3 Lags	4 Lags	5 Lags	6 Lags	7 Lags	8 Lags	9 Lags	10 Lags	11 Lags	12 Lags
Critical Value	-3,44	-3,44	-3,44	-3,44	-3,44	-3,44	-3,44	-3,44	-3,44	-3,44	-3,44	-3,44
T-Statistic	-7,46	-7,62	-6,51	-5,64	-4,97	-5,25	-4,13	-3,88	-3,6	-4,09	-5,41	-4,57

After running the tests, what I find is that for different specifications and for different lag lengths inflation series is a stationary process since for a 5% significance level we have to reject the null H_0 of inflation having a unit root until a specification of 9 lags of autoregressive coefficients for the ADF trend-stationary and AR with drift test modalities. However when I run just the ADF AR test I have to reject the null at the lag order of 7 as it is shown above.

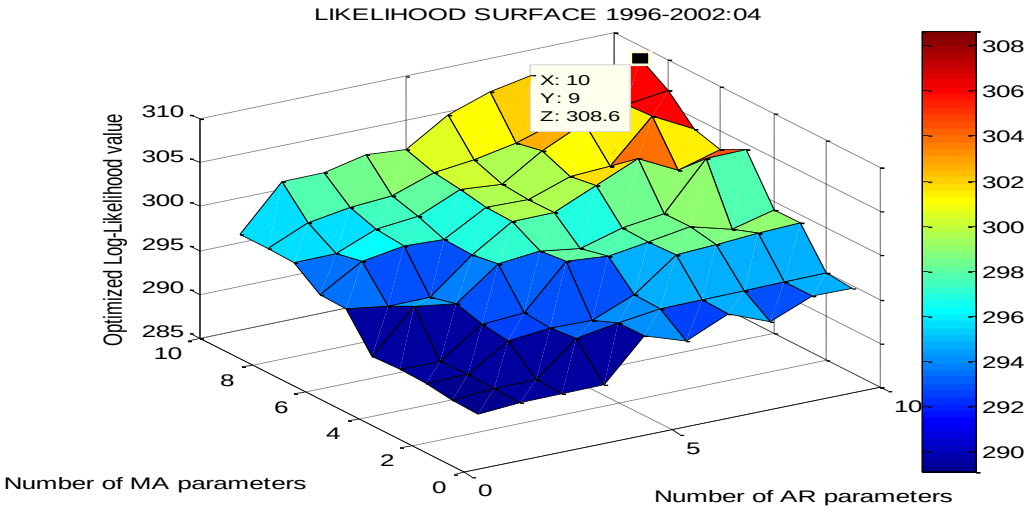
Model selection tests

The optimized log-likelihood estimation suggests a specification consisting of an ARMA(10,9) model with a value of 308.6 as it shows graph 8. In general this information criteria tends to reject ARMA model specifications with few parameters as it does the AIC criteria. AIC information criteria reach its minimum in the fitted ARMA(9,5) with value for the AIC function of -576.7. Bayesian information surface suggests an ARMA(1,1) with a minimum in -560 and attending to RMSE the model to use is an ARMA (7,9).

Tale 15: Competing ARMA (p,q) models

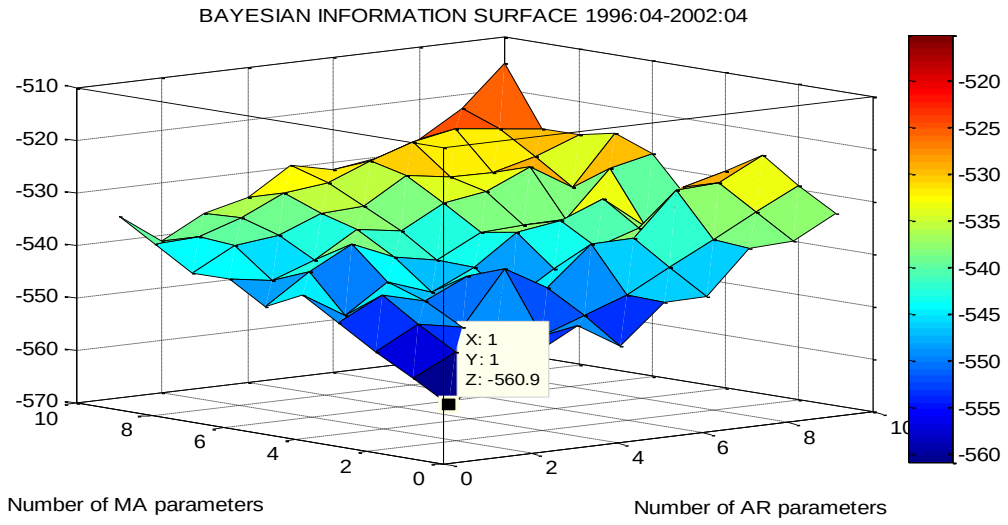
Competing Models	Criteria
ARMA(7,9)	RMSE
ARMA(9,5)	AIC
ARMA(1,1)	BIC
ARMA(10,9)	Likelihood

Graph 8: Likelihood surface



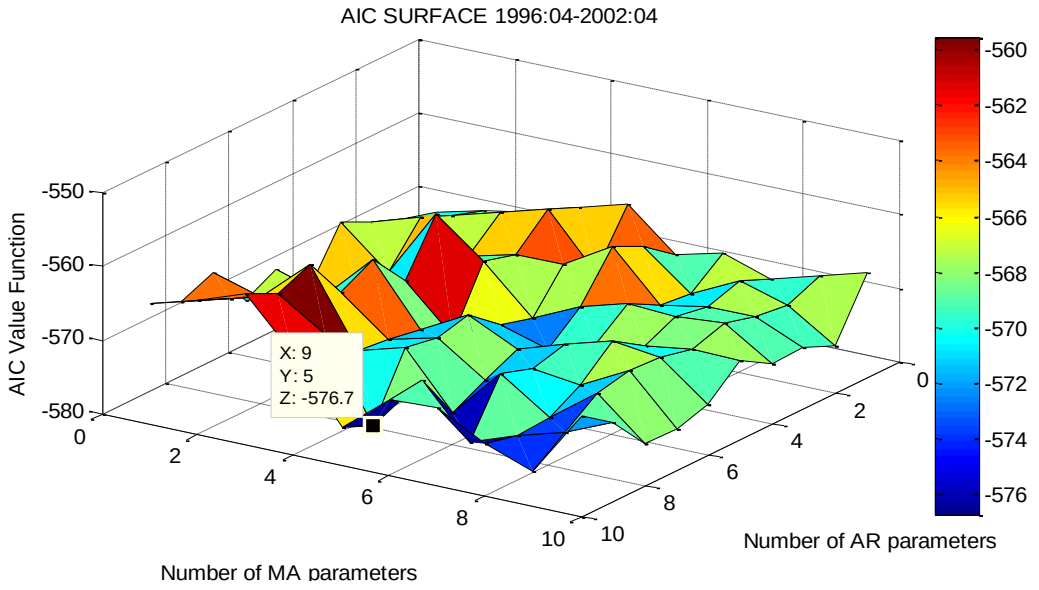
Source: Own elaboration with BCP data

Graph 9: Bayesian information surface



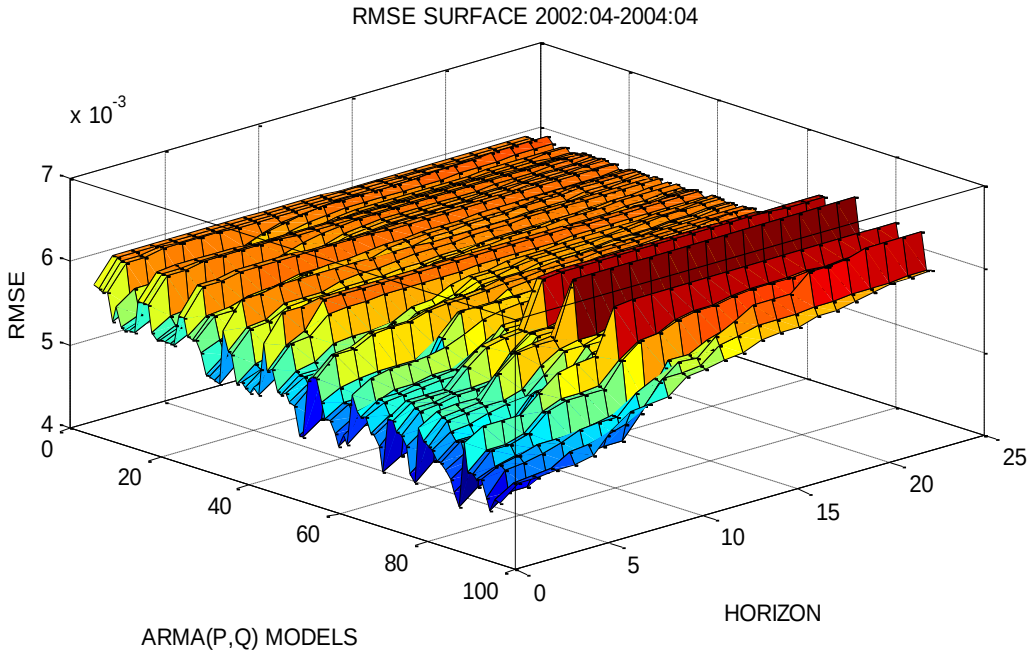
Source: Own elaboration with BCP data

Graph 10: Akaike information surface



Source: Own elaboration with BCP data

Graph 11: RMSE surface



Source: Own elaboration with BCP data

Second Subsample 1996-2004:04

Table 16: Augmented Dickey Fuller tests results

ADF TESTS	1 Lag	2 Lags	3 Lags	4 Lags	5 Lags	6 Lags	7 Lags	8 Lags	9 Lags	10 Lags	11 Lags	12 Lags
Stat 1--> ADF AR model	-3,46	-2,782	-2,618	-2,16	-1,771	-1,56	-1,438	-1,38	-1,341	-1,383	-1,607	-1,254
Cvalue1 -->ADF AR model	-1,94	-1,944	-1,944	-1,94	-1,944	-1,94	-1,944	-1,94	-1,944	-1,944	-1,944	-1,9445
Stat 2--> ADF AR with drift	-5,36	-4,589	-4,651	-3,99	-3,438	-3,12	-3,036	-2,98	-3,149	-3,378	-4,784	-3,8364
Cvalue2,-->ADF AR with drift	-2,89	-2,891	-2,891	-2,89	-2,892	-2,89	-2,893	-2,89	-2,894	-2,894	-2,895	-2,8952
Stat 3--> ADF trend stationary	-5,32	-4,55	-4,611	-3,95	-3,384	-3,07	-2,976	-2,92	-3,081	-3,349	-4,925	-3,9784
Cvalue3-->ADF trend stationary	-3,46	-3,457	-3,458	-3,46	-3,459	-3,46	-3,46	-3,46	-3,462	-3,462	-3,463	-3,4634

The process of inflation time series until 2004 is a stationary process for the drift specification while for the simple AR and trend stationary model setting the results vary across widely different lag specifications defining a U-pattern, meaning for middle lag lengths (i.e ,6-10 lags) the null of inflation having a unit root has to be accepted.

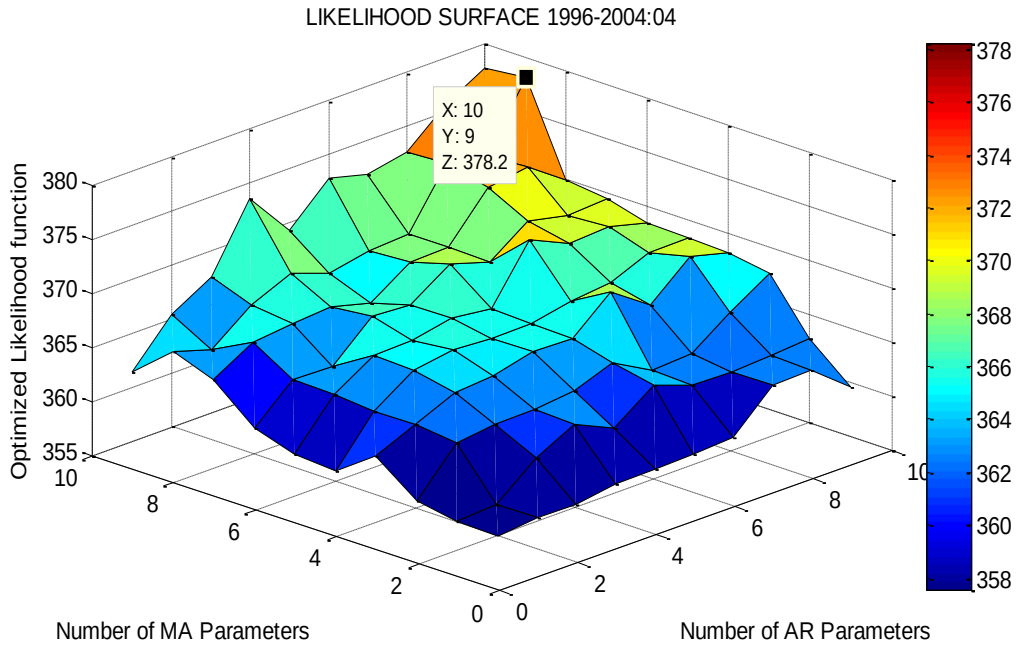
Model selection tests

The optimized log-likelihood estimation suggests a specification consisting of an ARMA(10,9) model with a value of 378.2 as it shows graph 12. In general this information criteria tends to reject ARMA model specifications with few parameters as it does the AIC criteria. AIC information criteria reach its minimum in the fitted ARMA(7,6) with value for the AIC function of -711.7. Bayesian information surface suggests an ARMA(1,1) with a minimum in -696 and attending to RMSE the model to use is an ARMA (10,9).

Table 17: Competing ARMA(p,q) models

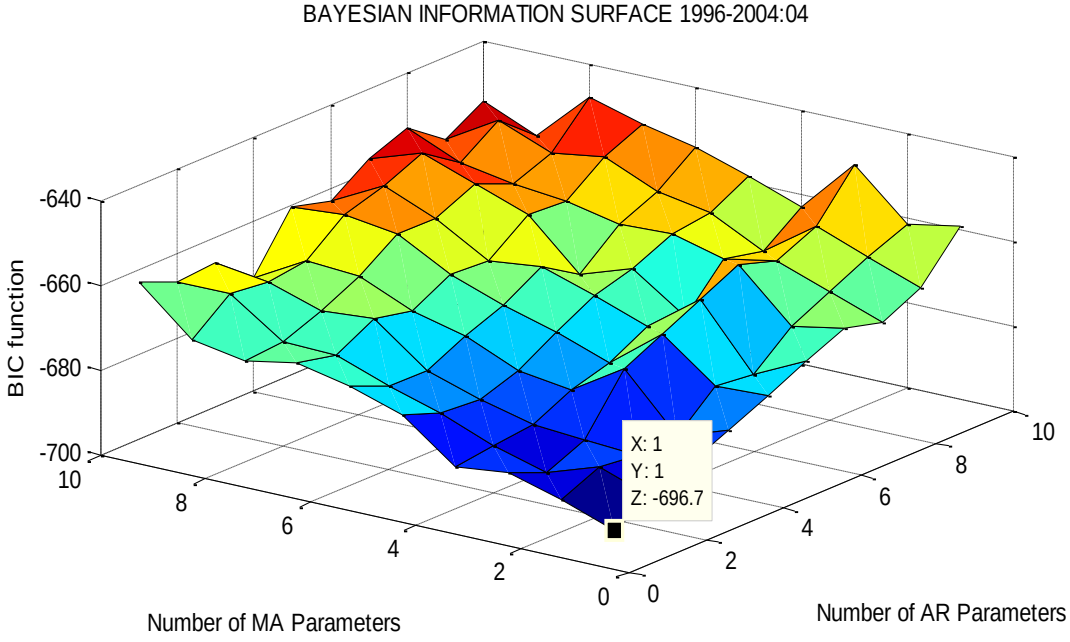
Competing Models	Criteria
ARMA(10,9)	Likelihood
ARMA(7,6)	AIC
ARMA(1,1)	BIC
ARMA(7,9)	RMSE

Graph 12: Likelihood surface 2



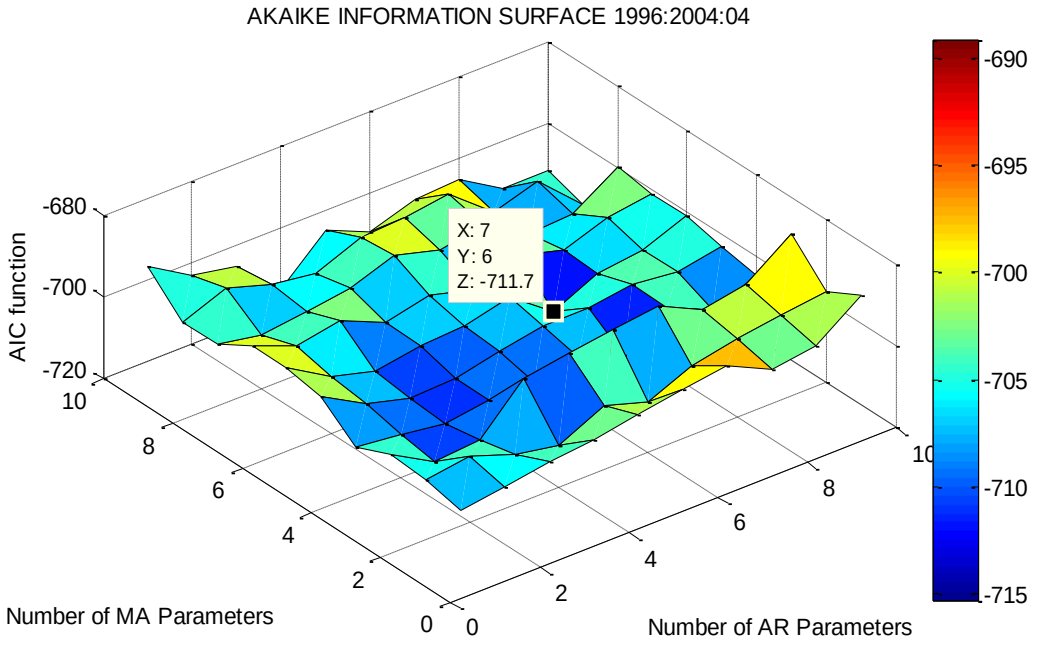
Source: Own elaboration with BCP data

Graph 13: Bayesian information surface 2



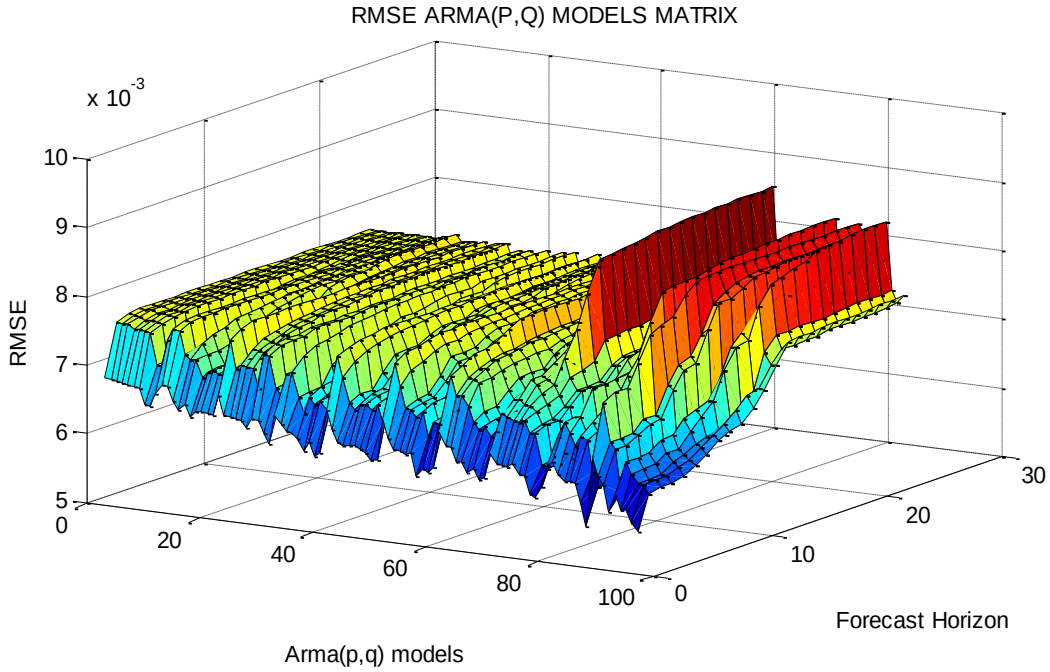
Source: Own elaboration with BCP data

Graph 14: Akaike Information surface 2



Source: Own elaboration with BCP data

Graph 15: RMSE 2 ARMA(P,Q) models matrix



Source: Own elaboration with BCP data

Forecast subsample 3: 2006:04-2008:04

Table 18: Augmented Dickey Fuller tests results

ADF	1 Lag	2 Lags	3 Lags	4 Lags	5 Lags	6 Lags	7 Lags	8 Lags	9 Lags	10 Lags	11 Lags	12 Lags
Stat 1--> ADF AR model	-4,008	-3,438	-2,79	-2,43	-1,9	-1,739	-1,47	-1,304	-1,26573	-1,401	-1,7385	-1,303
Cvalue1 -->ADF AR model	-1,944	-1,944	-1,94	-1,944	-1,94	-1,944	-1,94	-1,944	-1,94387	-1,944	-1,9439	-1,944
Stat 2--> ADF AR with drift	-6,328	-5,942	-5,21	-4,814	-4,05	-3,904	-3,53	-3,269	-3,36067	-3,794	-5,02	-4,143
Cvalue2,-->ADF AR with drift	-2,891	-2,891	-2,89	-2,892	-2,89	-2,893	-2,89	-2,893	-2,8939	-2,894	-2,8947	-2,895
Stat 3--> ADF trend stationary	-6,303	-5,917	-5,19	-4,793	-4,03	-3,886	-3,51	-3,254	-3,34468	-3,775	-4,9968	-4,123
Cvalue3-->ADF trend stationary	-3,449	-3,449	-3,45	-3,45	-3,45	-3,45	-3,45	-3,451	-3,45136	-3,452	-3,4519	-3,452

When running the set of Augmented Dickey Fuller tests I found that inflation can be considered a stationary process and then be modelled with ARIMAspecifications. Running THE ADF AR test I find that only up to a four lag inflation series is stationary and that for all the other specifications, with drift and with trend stationarity far all lag specifications process is stationary.

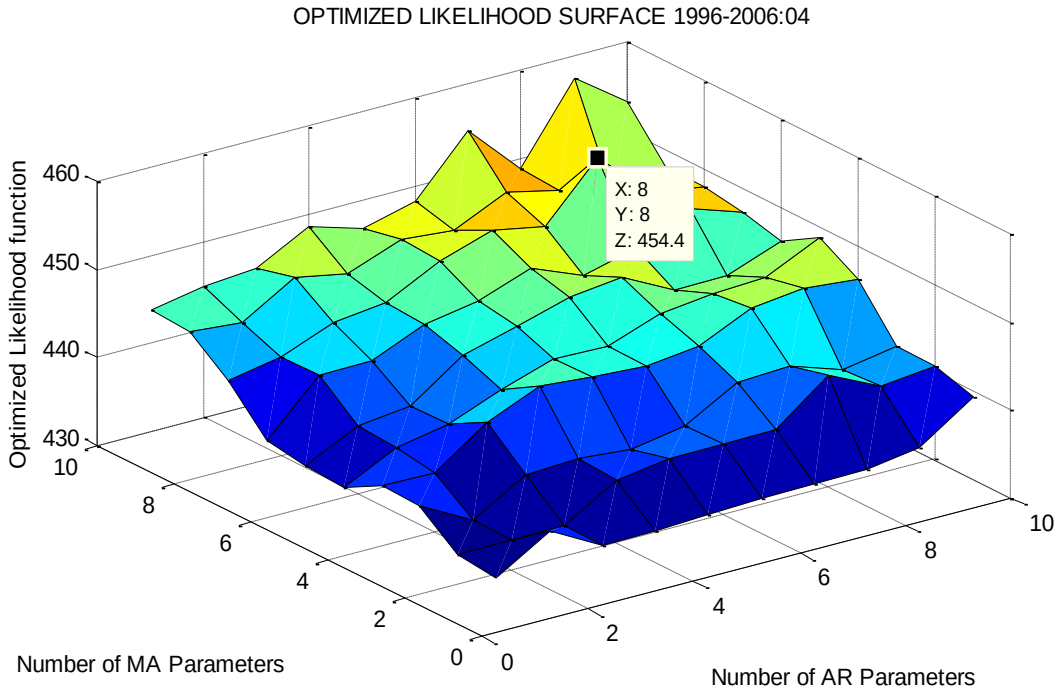
Model selection tests

The optimized log-likelihood estimation suggests a specification consisting of an ARMA(8,8) model with a value of 454.4 as it shows graph JJ. In general this information criteria tends to reject ARMA model specifications with few parameters as it does the AIC criteria. AIC information criteria reach its minimum in the fitted ARMA(8,8) with value for the AIC function of -711.7. Bayesian information surface suggests an ARMA(2,1) with a minimum in -696 and attending to RMSE the model to use is an ARMA (9,5).

Table 19: Selected ARMA models by criteria

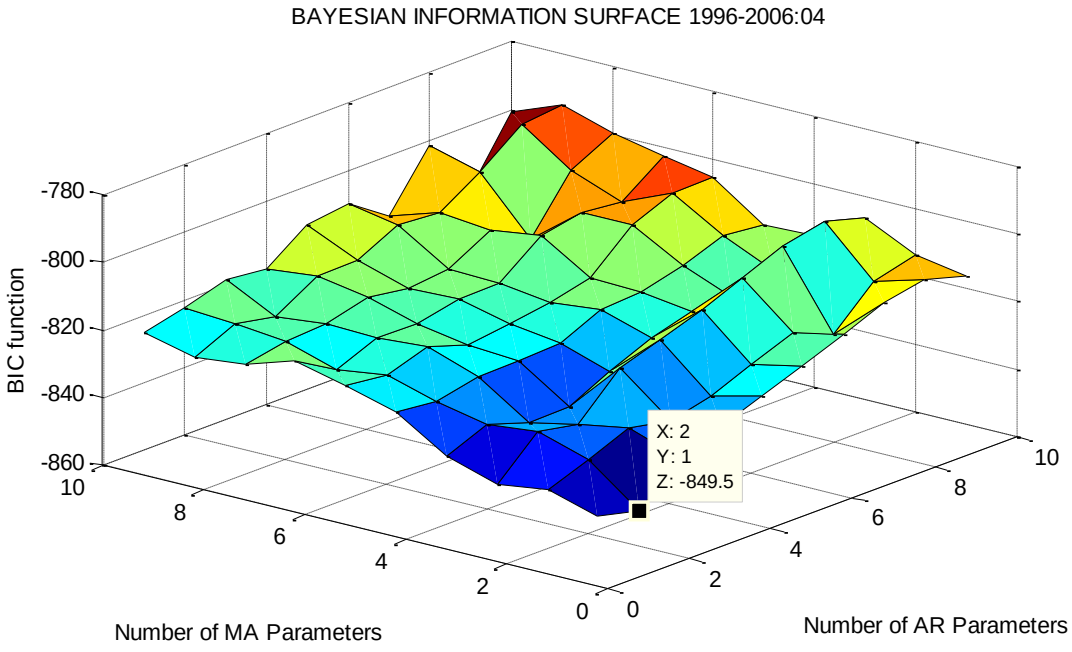
Competing Models	**Criteria**
ARMA(2,1)	BIC
ARMA(8,8)	AIC
ARMA(8,8)	Likelihood
ARMA(9,5)	RMSE

Graph 16: Likelihood surface 3



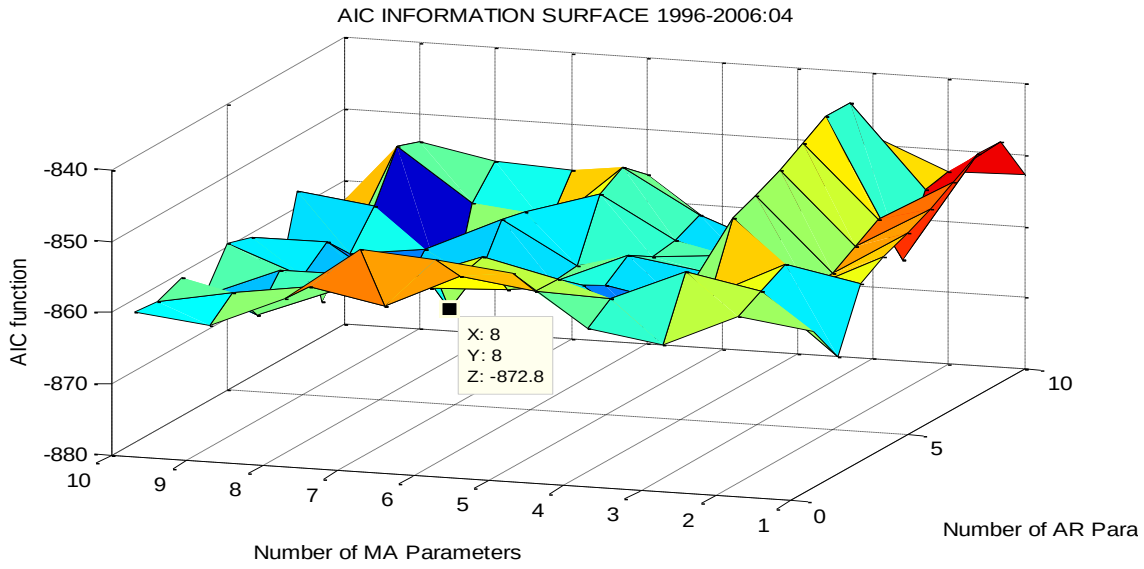
Source: Own elaboration with BCP data

Graph 17: Bayesian information surface 3



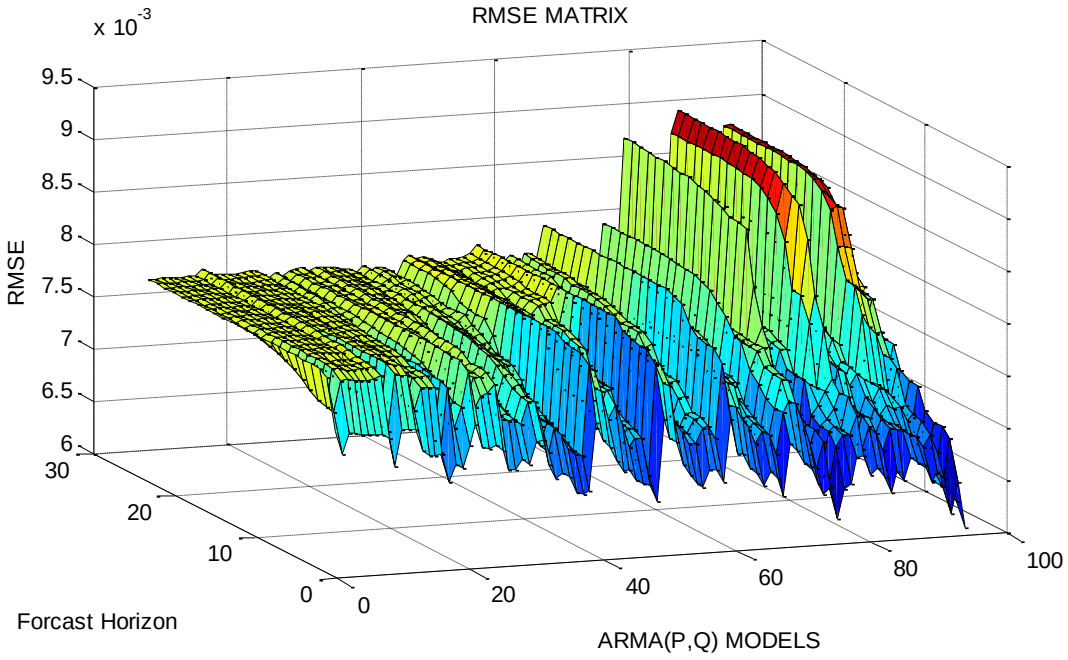
Source: Own elaboration with BCP data

Graph 18: Akaike Information Surface 3



Source: Own elaboration with BCP data

Graph 19: RMSE 3



Source: Own elaboration with BCP data

Forecast subsample 4: 2008:04-2010:04

Table 14: Augmented Dickey Fuller tests results

ADF	1 Lag	2 Lags	3 Lags	4 Lags	5 Lags	6 Lags	7 Lags	8 Lags	9 Lags	10 Lags	11 Lags	12 Lags
Stat 1--> ADF AR model	-4,538	-4,245	-3,31	-2,66	-2,23	-2,149	-1,51	-1,355	-1,14	-1,343	-1,6754	-1,188
Cvalue1 -->ADF AR model	-1,943	-1,943	-1,94	-1,943	-1,94	-1,943	-1,94	-1,943	-1,942	-1,943	-1,943	-1,943
Stat 2--> ADF AR with drift	-7,058	-7,33	-6,29	-5,458	-4,9	-5,062	-3,88	-3,725	-3,44	-4,051	-5,3189	-4,197
Cvalue2,-->ADF AR with drift	-2,882	-2,882	-2,88	-2,882	-2,88	-2,883	-2,88	-2,883	-2,88	-2,883	-2,8836	-2,884
Stat 3--> ADF trend stationary	-7,043	-7,313	-6,27	-5,448	-4,89	-5,051	-3,87	-3,714	-3,43	-4,036	-5,2977	-4,179
Cvalue3-->ADF trend stationary	-3,442	-3,442	-3,44	-3,443	-3,44	-3,443	-3,44	-3,444	-3,444	-3,445	-3,4448	-3,445

Running the ADF AR test I find that only up to a six lags length process, inflation series is stationary and that for all the other specifications we have to accept the null of the existence of unit roots. The ADF with drift and with trend stationarity far all lag specifications process is stationary.

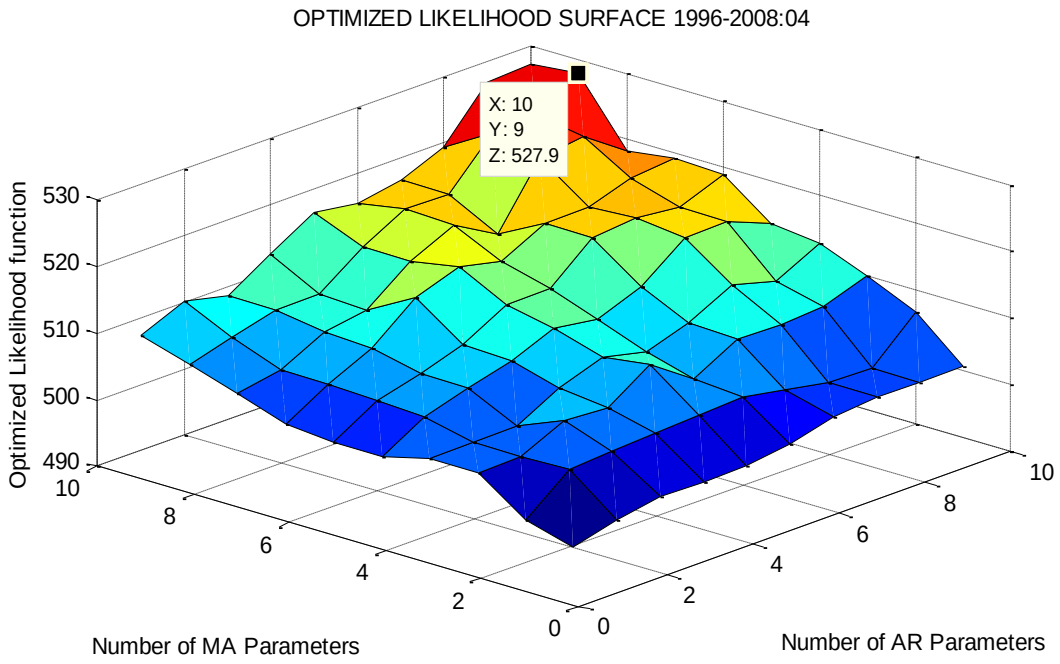
Model selection tests

The optimized log-likelihood estimation suggests a specification consisting of an ARMA(10,9) model with a value function of 527.9 as it shows graph 20. Bayesian Information criteria reach its minimum in the fitted ARMA(2,2) with value for the BIC function of -973.2. Akaike Information Criteria surface suggests an ARMA(10,9) with a minimum in -1014. Finally, attending to RMSE the model to use is an ARMA (8,7)

Table 20: Selected ARMA(P,Q) models by criteria

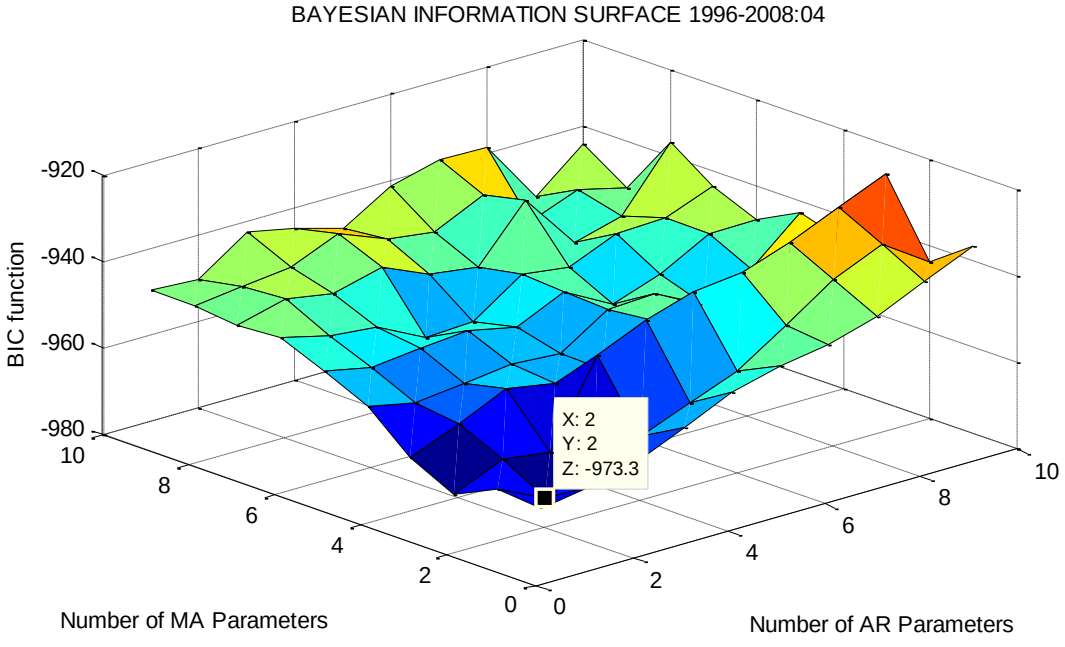
Competing Models	Criteria
ARMA(2,2)	BIC
ARMA(8,7)	RMSE
ARMA(10,9)	Likelihood
ARMA(10,9)	AIC

Graph: 20: Likelihood surface 4



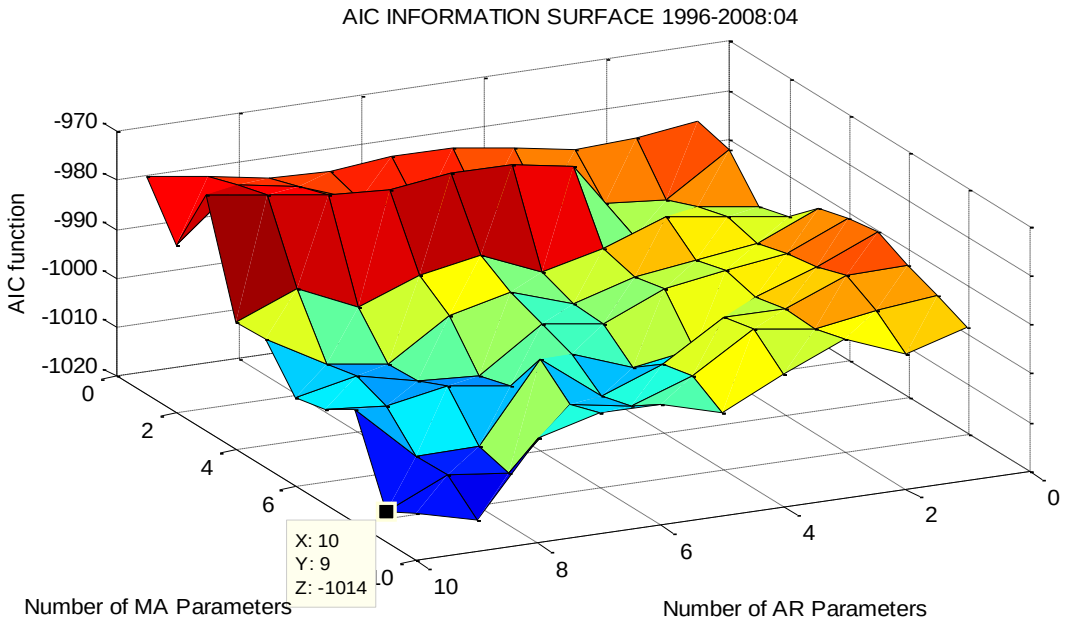
Source: Own elaboration with BCP data

Graph 21: Bayesian information surface 4



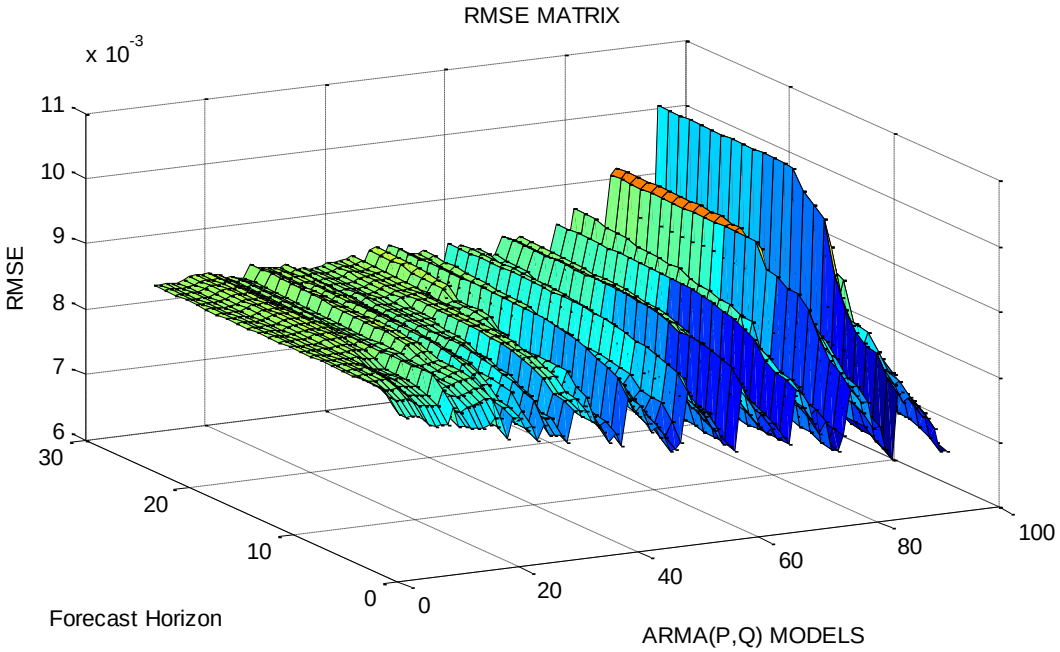
Source: Own elaboration with BCP data

Graph 22: Akaike Information Surface 4



Source: Own elaboration with BCP data

Graph 23: RMSE 4



Source: Own elaboration with BCP data


BANCO CEN
DEL PARAG