3

Morphological-Rank-Linear Models for Financial Time Series Forecasting

Ricardo de A. Araújo[1], Gláucio G. de M. Melo[1],
Adriano L. I. de Oliveira[2] and Sergio C. B. Soares[2]
[1]*Information Technology Department, [gm]² Intelligent Systems, Campinas, SP,*
[2]*Systems and Computing Department, University of Pernambuco, Recife, PE,*
Brazil

1. Introduction

The financial time series forecasting is considered a rather difficult problem, due to the many complex features frequently present in such time series (irregularities, volatility, trends and noise). Several approaches have been studied for the development of predictive models able to predict time series, based on its past and present data.

In the attempt to solve the time series prediction problem, a wide number of linear statistical models were proposed. Among them, the popular linear statistical approach based on Auto Regressive Integrated Moving Average (ARIMA) models [1] is one of the most common choices. However, since the ARIMA models are linear and most real world applications involve nonlinear problems, this can introduce an accuracy limitation of the generated forecasting models.

In the attempt to overcome linear statistical models limitations, other nonlinear statistical approaches have been developed, such as the bilinear models [2], the threshold autoregressive models [3], the exponential autoregressive models [4], the general state dependent models [5], amongst others. The drawbacks of those nonlinear statistical models are the high mathematical complexity associated with them (resulting in many situations in similar performances to the linear models) and the need, most of the time, of a problem dependent specialist to validate the predictions generated by the model, limiting the development of an automatic forecast system [6].

Alternately, Artificial Neural Networks (ANNs) based approaches have been applied for nonlinear modeling of time series in the last two decades [7-14]. However, in order to define a solution to a given problem, ANNs require the setting up of a series of system parameters, some of them are not always easy to determine. The ANN topology, the number of processing units, the algorithm for ANN training (and its corresponding variables) are just some of the parameters that require definition. In addition to those, in the particular case of time series forecasting, another crucial element necessary to determine is the relevant time lags to represent the series [15]. In this context, evolutionary approaches for the definition of neural network parameters have produced interesting results [16{20]. Some of these works have focused on the evolution of the network weights whereas others aimed at evolving the network architecture.

Source: New Achievements in Evolutionary Computation, Book edited by: Peter Korosec,
ISBN 978-953-307-053-7, pp. 318, February 2010, INTECH, Croatia, downloaded from SCIYO.COM

In this context, a relevant work was presented by Ferreira [15], consisting of the Time-delay Added Evolutionary Forecasting (TAEF) method definition, which performs a search for the minimum number of necessary dimensions (the past values of the series) to determine the characteristic phase space of the time series. The TAEF method [15] finds the most fitted predictor model for representing a time series, and then performs a behavioral statistical test in order to adjust time phase distortions that may appear in the representation of some series.

Nonlinear filters, on the other hand, have been widely applied to signal processing. An important class of nonlinear systems is based on the framework of Mathematical Morphology (MM) [21, 22]. Many works have focused on the design of morphological systems [21, 23-28]. An interesting work was presented by Salembier [29, 30], which designed Morphological/Rank (MR) filters via gradient-based adaptive optimization. Also, Pessoa and Maragos [31] proposed a new hybrid filter, referred to as Morphological/Rank/Linear (MRL) filter, which consists of a linear combination of an MR filter [29, 30] and a linear Finite Impulse Response (FIR) filter [31]. In the branch of the filters and Artificial Intelligence integration, Pessoa and Maragos [32] also proposed a neural network architecture involving MRL operators at every processing node.

In the morphological systems context, another work was presented by Araújo et al. [33, 34]. It consists of an evolutionary morphological approach for time series prediction, which provides a mechanism to design a predictive model based on increasing and non-increasing translation invariant morphological operators and according to Matheron decomposition [35] and Banon and Barrera decomposition [36].

This work proposes the Morphological-Rank-Linear Time-lag Added Evolutionary Forecasting (MRLTAEF) method in order to overcome the random walk dilemma for financial time series prediction, which performs an evolutionary search for the minimum dimension to determining the characteristic phase space that generates the financial time series phenomenon. The proposed MRLTAEF method is inspired on Takens Theorem [37] and consists of an intelligent hybrid model composed of an MRL filter [31] combined with a Modified Genetic Algorithm (MGA) [16], which searches for the particular time lags capable of a fine tuned characterization of the time series and estimates the initial (sub-optimal) parameters of the MRL filter. Each individual of the MGA population is trained by the averaged Least Mean Squares (LMS) algorithm to further improve the MRL filter parameters supplied by the MGA. After training the model, the MRLTAEF method chooses the most tuned predictive model for the time series representation, and performs a behavioral statistical test [15] and a phase fix procedure [15] to adjust time phase distortions observed in financial time series.

Furthermore, an experimental analysis is conducted with the proposed MRLTAEF method using six real world financial time series. Five well-known performance metrics are used to assess the performance of the proposed method and the obtained results are compared with the previously presented methods in literature.

This work is organized as follows. In Section 2, the fundamentals and theoretical concepts necessary for the comprehension of the proposed method are presented, such as the time series prediction problem, the random walk dilemma for financial time series prediction, linear and nonlinear statistical models, neural network models, genetic algorithms (standard and modified), intelligent hybrid models (in particular the TAEF method). Section 3 shows the fundamentals and theoretical concepts of mathematical morphology and the MRL filter

definition and its training algorithm. Section 4 describes the proposed MRLTAEF method. Section 5 presents the performance metrics which are used to assess the performance of the proposed method. Section 6 shows the simulations and the experimental results attained by the MRLTAEF method using six real world financial time series, as well as a comparison between the results achieved here and those given by standard MLP networks, MRL filters and the TAEF method [15]. Section 7 presents, to conclude, the final remarks of this work.

2. Fundamentals

In this section, the fundamentals and theoretical concepts necessary for the comprehension of the proposed method will be presented.

2.1 Time series forecasting problem

A time series is a sequence of observations about a given phenomenon, where it is observed in discrete or continuous space. In this work all time series will be considered time discrete and equidistant.

Usually, a time series can be defined by

$$X_t = \{x_t \in \mathbb{R} \mid t = 1, 2, \ldots, N\}, \tag{1}$$

where t is the temporal index and N is the number of observations. The term X_t will be seen as a set of temporal observations of a given phenomenon, orderly sequenced and equally spaced.

The aim of prediction techniques applied to a given time series (X_t) are to provide a mechanism that allows, with certain accuracy, the prediction of the future values of X_t, given by X_{t+k}, k = 1, 2, …, where k represents the prediction horizon. These prediction techniques will try to identify certain regular patterns present in the data set, creating a model capable of generating the next temporal patterns, where, in this context, a most relevant factor for an accurate prediction performance is the correct choice of the past window, or the time lags, considered for the representation of a given time series.

Box & Jenkins [1] shown that when there is a clear linear relationship among the historical data of a given time series, the functions of auto-correlation and partial auto-correlation are capable of identifying the relevant time lags to represent a time series, and such procedure is usually applied in linear models. However, when it uses a real world time series, or more specifically, a complex time series with all their dependencies on exogenous and uncontrollable variables, the relationship that involves the time series historical data is generally nonlinear, which makes the Box & Jenkins' analysis procedure of the time lags only a crude estimate.

In a mathematical sense, such a relationship involving time series historical data defines a d-dimensional phase space, where d is the minimum dimension capable of representing such relationship. Therefore, a d- dimensional phase space can be built so that it is possible to unfold its corresponding time series. Takens [37] proved that if d is sufficiently large, such phase space is homeomorphic to the phase space that generated the time series. Takens' Theorem [37] is the theoretical justification that it is possible to build a state space using the correct time lags, and if this space is correctly rebuilt, Takens' Theorem [37] also guarantees that the dynamics of this space is topologically identical to the dynamics of the real system state space.

The main problem in reconstructing the original state space is naturally the correct choice of the variable d, or more specifically, the correct choice of the important time lags necessary for the characterization of the system dynamics. Many proposed methods can be found in the literature for the definition of the lags [38-40]. Such methods are based on measures of conditional probabilities, which consider,

$$X_t = f(x_{t-1}, x_{t-2}, \ldots, x_{t-d}) + r_t, \tag{2}$$

where $f(x_{t-1}, x_{t-2}, \ldots, x_{t-d})$ is a possible mapping of the past values to the facts of the future (where x_{t-1} is the lag 1, x_{t-2} is the lag 2,…, x_{t-d} is the lag d) and r_t is a noise term.
However, in general, these tests found in the literature are based on the primary dependence among the variables and do not consider any possible induced dependencies. For example, if

$$f(x_{t-1}) = f(f(x_{t-2})), \tag{3}$$

it is said that x_{t-1} is the primary dependence, and the dependence induced on x_{t-2} is not considered (any variable that is not a primary dependence is denoted as irrelevant).
The method proposed in this work, conversely, does not make any prior assumption about the dependencies between the variables. In other words, it does not discard any possible correlation that can exist among the time series parameters, even higher order correlations, since it carries out an iterative automatic search for solving the problem of finding the relevant time lags.

2.2 The random walk dilemma

A naive prediction strategy is to define the last observation of a time series as the best prediction of its next future value ($X_{t+1} = X_t$). This kind of model is known as the Random Walk (RW) model [41], which is defined by

$$X_t = X_{t-1} + r_t, \tag{4}$$

or

$$\Delta X_t = X_t - X_{t-1} = r_t, \tag{5}$$

where X_t is the current observation, X_{t-1} is the immediate observation before X_t, and r_t is a noise term with a gaussian distribution of zero mean and standard deviation σ ($r_t \approx N(0, \sigma)$). In other words, the rate of time series change (ΔX_t) is a white noise.
The model above clearly implies that, as the information set consists of past time series data, the future data is unpredictable. On average, the value X_t is indeed the best prediction of value X_{t+1}. This behavior is common in the finance market and in the economic theory and its so-called random walk dilemma or random walk hypothesis [41].
The computational cost for time series forecasting using the random walk dilemma is extremely low. Therefore, any other prediction method more costly than a random walk model should have a very superior performance than a random walk model, otherwise its use is not interesting in the practice.
However, if the time series phenomenon is driven by law with strong similarity to a random walk model, any model applied to this time series phenomenon will tend to have the same performance as a random walk model.

Assuming that an accurate prediction model is used to build an estimated value of X_t, denoted by $\widehat{X_t}$, the expected value ($E[\cdot]$) of the difference between $\widehat{X_t}$ and X_t must tend to zero,

$$E[\widehat{X_t} - X_t] \to 0. \tag{6}$$

If the time series generator phenomenon is supposed to have a strong random walk linear component and a very weak nonlinear component (denoted by $g(t)$), and assuming that $E[r_t] = 0$ and $E[r_t r_k] = 0$ ($\forall\ k \neq t$), the expected value of the difference between $\widehat{X_t}$ and X_t will be

$$\begin{aligned} &E[\widehat{X_t} - (X_{t-1} + g(t) + r_t)] \to 0 \\ &E[\widehat{X_t}] - E[X_{t-1}] - E[g(t)] - E[r_t] \to 0 \\ &E[\widehat{X_t}] - E[X_{t-1}] - E[g(t)] \to 0 \\ &E[\widehat{X_t}] \to E[X_{t-1}] + E[g(t)]. \end{aligned}$$

But $E[X_{t-1}] \gg E[g(t)]$, then $E[X_{t-1}] + E[g(t)] \simeq E[X_{t-1}]$ and

$$E[\widehat{X_t}] \to E[X_{t-1}]. \tag{7}$$

Therefore, in these conditions, to escape the random walk dilemma is a hard task. Indications of this behavior (strong linear random walk component and a weak nonlinear component) can be observed from time series lagplot graphics. For example, lagplot graphics where strong linear structures are dominant with respect to nonlinear structures [42], generally observed in the financial and economical time series.

2.3 Linear statistical models

The time series prediction process consists of representing the time series features through a model able to extend such features to the future. According to Box & Jenkins [15], classical statistical models were developed to represent the following kind of information patterns: constants, trends and sazonalities. However, there are some variations that occur in such patterns as irregularities, volatility, noise, amongst other.

In this way, it is possible to verify that the statistical models are based on transcendental or algebraic time functions, which can be represented by

$$X_t = b_1 f_1(t) + b_2 f_2(t) + \cdots + b_k f_k(t) + r_t \tag{8}$$

where b_i and $f_i(t)$ (i = 1, 2,…, k) denotes, respectively, the constant parameters and mathematical functions of t. Term r_t represents a random component or noise.

However, there are other ways for time series modeling, where X_t will be modeled as a temporally ordered random component function, from the present to the past (r_t, r_{t-1}, r_{t-2},…). This kind of representation is kwon as "linear filter models", which is widely applied when the time series observations are highly correlated [1]. In this way, X_t can be defined by

$$Z_t = m + y_0 r_t + y_1 r_{t-1} + y_2 r_{t-2} + \cdots + y_k r_{t-k}, \tag{9}$$

where m and y_i (i = 1, 2,…, k) are constants.

Therefore, the time series prediction process consists of an accurate parameters estimation of the prediction models to build the future behavior of a given phenomenon.

Box & Jenkins Models In the literature, several models were proposed to solve the time series prediction problem. Among these models, it is verified that a wide number of them are linear: Simple Moving Averages (SMA) [43, 44], Simple Exponential Smoothing (SES) [43, 44], Brow's Linear Exponential Smoothing (BLES) [43, 44], Holt's Bi-parametric Exponential Smoothing (HBES) [43, 44], Adaptive Filtering (AF) [43, 44], are some examples of that. However, among these linear models, the Box & Jenkins [1] models receive a special mention, given that, in practice, are the most popular and commonly used to solve the time series prediction problem.

Box and Jenkins [1] a set of algebraic models, referred to as Auto-Regressive Integrated Moving Average (ARIMA) models, where it builds an accurate prediction for a given time series. The ARIMA model is based on two main models:

1. Auto-Regressive (AR), which is defined by

$$\widetilde{Z}_t = \phi_1 \widetilde{Z}_{t-1} + \phi_2 \widetilde{Z}_{t-2} + \cdots + \phi_p \widetilde{Z}_{t-p} + r_t, \tag{10}$$

   where $\widetilde{Z}_k = Z_k - \mu$, being μ defined as the mean of the time series. Terms ϕ_i (i = 1, 2,…, p) denotes the auto-regressive coefficients.

2. Moving Average (MA), which is defined by

$$Z_t = \mu + r_t - \theta_1 r_{t-1} - \theta_2 r_{t-2} - \cdots - \theta_q r_{t-q}. \tag{11}$$

   Assuming that $\widetilde{Z}_k = Z_k - \mu$, it has

$$\widetilde{Z} = (1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q) r_t = \Theta(B) a_t \tag{12}$$

   where $\Theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q$ represent the moving average operator.

The union of both AR and MA models build a model known as Auto-regressive Moving Average (ARMA) of order (p,q), which was proposed in the attempt to build the most parsimonious model as possible, given that, with the inclusion of auto-regressive and moving average terms in a unique model can be seen as a possible solution to obtain a small number of model parameters. In this way, the ARMA model is defined by

$$\widetilde{Z}_t = \phi_1 \widetilde{Z}_{t-1} + \cdots + \phi_p \widetilde{Z}_{t-p} + r_t - \theta_1 r_{t-1} - \theta_q r_{t-q}. \tag{13}$$

The ARIMA model basically consists of the application of data to the high-pass filter, which is sensible only to high frequencies of the function, which is applied to the ARMA model. Such a filter is represented by letter “I” (integrated) in the ARIMA notation and this is the main difference among the separated data by a constant distance d. This procedure, known as data differences, is performed to remove the data trends, building the time series as a stationary process, that is, an ARIMA(p,d,q) model is an algebraic study that show as a time series variable (X_t) is related with its past values (X_{t-1},X_{t-2},…,X_{t-p}) and the past noisy term values (r_{t-1}, r_{t-2},…, r_{t-p}), differentiated d times [15].

In this way, Box & Jenkins [1] proposed a procedure able to find an adequate prediction model to solve the time series prediction problem, as be seen in Figure 1.

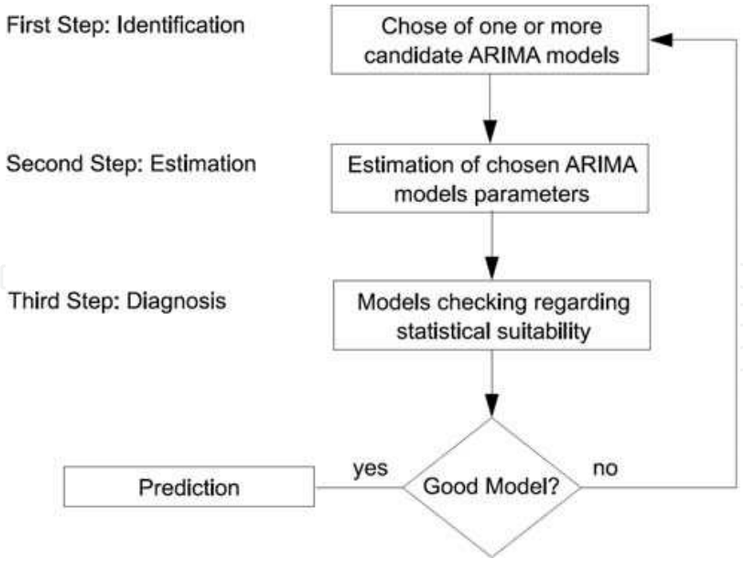


Fig. 1. Box & Jenkins method procedure.

According to Figure 1, it verifies that the first step (identification) uses two graphical devices to measure the correlation among the observations of the data set of the time series. Such devices are the Auto-Correlation Function (ACF) [1] and the Partial Auto-Correlation Function (PACF) [1]. In the second step (estimation), the model coefficients are estimated, and finally, in the third step (diagnosis), Box & Jenkins [1] proposed some checking procedures to determine the statistical suitability of the chosen model in previous steps. Then, the model that fails in these procedures will be rejected.

2.4 Nonlinear statistical models

As mentioned in the previous section, the ARIMA model [1] is one of the most common choices for the time series prediction. However, since the ARIMA models are linear and most real world applications involve nonlinear problems, this can introduce a limitation in the accuracy of the predictions generated, that is, this model assumes that the time series are stationary or generated by a linear process. However, it is not correct to generalize the linearity assumption of the time series due to the nonlinear nature of a given real-world phenomena, where nonlinear structures are found in historical time series data.
n this way, a time series can be modeled by

$$X_t = h(X_{t-1}, \ldots, X_{t-p}, r_{t-1}, \ldots, r_{t-q}) + r_t, \tag{14}$$

where r_t represents a random component, or noisy term. Terms p and q represent integer indexes that define the time windows of past terms of the time series and noise, respectively. The term $h(\cdot)$ denotes a nonlinear transference function, which build the mapping among the past and future values of the time series.
Therefore, to overcome the linear statistical models limitations, several nonlinear statistical models have been proposed in the literature, such as the bilinear models [2], the exponential auto-regressive models [4], the threshold autoregressive models [3], and the general state dependent models [5], amongst other.

The general state dependent models (GSD) [5] of (p, q) order is defined as an expansion and a local linearization of Equation 14 in Taylor series around a fixed time point, which can be defined by

$$X_t + \sum_{i=1}^{p} \phi_i(y_{t-1}) X_{t-1} = \mu(y_{t-1}) + r_t + \sum_{j=1}^{q} \theta_i(y_{t-1}) r_{t-1}, \tag{15}$$

where y_t = (r_{t-q+1},…, r_t, X_{t-q+1},.., X_t)′ is defined as a state vector, and the symbol "′"denotes a transposition operator.

A special class of such models, known as bilinear models [2], may be seen as a natural nonlinear extension of the ARMA model, making $\mu(x)$ and $\phi_i(x)$ constants and $\theta_j(y_{t-1}) = \theta_j + \sum_{v=1}^{Q} c_{jv} X_{t-v}$ ($j = 1, 2, \ldots$ and c_{jv} the coefficients to be adjusted). The general form of bilinear models of (p, q, P, Q) order is defined by

$$X_t + \sum_{i=1}^{p} \phi_i X_{t-1} = \mu + r_t + \sum_{j=1}^{q} \theta_j r_{t-j} + \sum_{u=1}^{P} \sum_{v=1}^{Q} c_{uv} X_{t-v} r_{t-u}. \tag{16}$$

According to Ferreira [15], it is verified that Equation 16 is linear in terms X_t and r_t and nonlinear regarding the cross term of Z and r. Thus, a bilinear model of first order can be built from the Equation 16 is given by

$$X_t = \alpha X_{t-1} + \beta r_t + \gamma X_{t-1} r_{t-1}, \tag{17}$$

where α, β and γ are the constant parameters to be determined.

Another particular class of such models, known as Exponential Auto-Regressive (EAR) models [4], of p order, is given by using a constant term $\mu(x)$, $\theta_j(x) = 0(\forall x)$ and $\phi_i(y_{t-1}) = \phi_i + \pi_i exp(-\gamma X_{t-1}^2)$ in Equation 15, and is formally defined by

$$X_t + \sum_{i=1}^{p} \{\phi_i + \pi_i exp(-\gamma X_{t-1}^2)\} X_{t-1} = \mu + r_t, \tag{18}$$

where γ denotes the time series scale factor.

Another class of nonlinear models which are used in time series predictions are the Threshold Auto-Regressive (TAR) models [3], where its parameters depend only on past values of its own process, and represent a finite set of possible AR models that a particular process could obey at any time point [15]. However, if the switch on of such models is determined by the data values location regarding thresholds, in this way the TAR model is known as Self-Excited Threshold Auto-regressive (SETAR) model [45].

A first-order TAR model is defined by

$$X_t = \begin{cases} \alpha_1 X_{t-1} + r_t, \text{ se } X_{t-1} < d \\ \alpha_2 X_{t-1} + r_t, \text{ se } X_{t-1} \geq d \end{cases}, \tag{19}$$

where the constant d is defined as the threshold.

The SETAR model can be defined $\mu(x) = \phi_0(j)$, $\theta_j(x) = 0(\forall x)$ and $\phi_i(y_{t-1}) = \acute{A}_i(j)$ if $X_{t-d} \in R_{(j)}$ (i = 1, 2,…, p; j = 1, 2,…, l), being d a positive integer and $R_{(j)}$ is a subset of real numbers (the threshold). Thus, Equation 15 can be rewritten in these terms, defining a SETAR model by

$$X_t + \phi_0^{(j)} + \sum_{i=1}^{p} \phi_i^{(j)} X_{t-i} = r_t^{(j)}, \text{ se } X_{t-d} \in R^{(j)}, \; j = 1, 2, \ldots, l, \tag{20}$$

where such equation represents a SETAR model of kind (l, p,…, p). Term $r_t^{(j)}$ denotes a white noise, being $r_t^{(j)}$ independent of $r_t^{(j')}$, with $j \neq j'$.

There are several other nonlinear models for time series prediction in literature, such as auto-regressive smooth models [46], auto-regressive models with time-dependent coefficients [46], auto-regressive conditional heteroscedastic models (ARCH) [47], amongst other. However, even with a wide number of nonlinear models proposed in the literature, De Gooijer and Kumar [46] do not find clear evidences, in terms of prediction performance, of nonlinear models when compared with the linear models. Clements et al. [45, 48] also argues that the prediction performance of such nonlinear models is more inferior than expected, and this problem still remains open.

According to Ferreira [15], a general accepted concept is that the environment of a given temporal phenomenon is nonlinear, and the fact that the nonlinear models do not achieve the expected results is due to inability of such models to describe the time series phenomenon more accurately than simple linear approximations. In particular, it is verified that the models applied in real world stock market and finance are highly nonlinear [48]. However, the problem of financial time series prediction is still considered a very di±cult problem due to several complex characteristics that often are present in these time series (irregularities, volatility, trend and noise).

Due to the complexity of the structures of relationships among time series data, there are several limitations of the nonlinear models when applied in real situations. One of these limitations is a high mathematical complexity, a factor that limits the nonlinear models to a performance similar to linear models, as well as the need, in most cases, of a problem specialist to validate the predictions generated by the model [6]. These factors suggest that new approaches must be developed in order to improve the prediction performance. Consequently, it is not surprising the great interest on the development of nonlinear models for time series prediction using new approaches and paradigms applied to the problem previously exposed.

2.5 Neural network models

The Artificial Neural Networks (ANN) are models that simulate biological neural systems behavior, particularly the human brain. The ANNs represent a parallel and distributed system composed of simple processing units, such as neurons, which calculate non linear mathematical functions.

The neurons are contained in a spatial arrangement generally composed of one or more layers interconnected by a wide number of connections. Generally, in most models, such connections are associated with weights, which are responsible for the storage of knowledge represented in the model, used as weights for the signals to be processed by neurons in the network.

Each ANN unit is conditioned to receive a signal, weighted by their respective input unit processing connections (ANN weights), which is processed by a mathematical function,

known as activation function or transfer function, and producing a new output signal which is propagated over the network.
In this way, making an analogy to the human brain, an ANN has the ability to learn through examples, as well as perform interpolations and extrapolations of the learned information. In the ANN learning process the main task is to determine the intensity of connections among neurons, which are adjusted and adapted by learning algorithms, which aims to make a fine tuned adjustment of connection weights, in order to better generalize the information contained in the pattern examples.
Therefore, a wide number of ANNs have been proposed in literature, which is worth mentioning,

- MultiLayers Perceptron (MLP) Neural Networks [49];
- Recursive Networks [49];
- Kohonen Networks [50, 51];
- Hopfield Networks [52];

Among the several kinds of ANNs, the MLPs are undoubtedly the most popular due to convenience, flexibility and efficiency, and can be applied to a wide range of problems [9, 49, 53].

2.6 MultiLayer perceptron neural networks

The MLP neural networks are typically composed of several neuron layers. The first layer is known as the input layer, where information is passed to the network. The last layer is called the output layer, where the model responses of a given information is then produced. Among input and output layers, there are one or more layers, which are referred to as intermediate layers.
Each layer is interconnected with the adjacent layer. If each neuron of a layer is connected to all neurons of the next layer, then it haas a fully connected MLP network (illustrated in Figure 2).

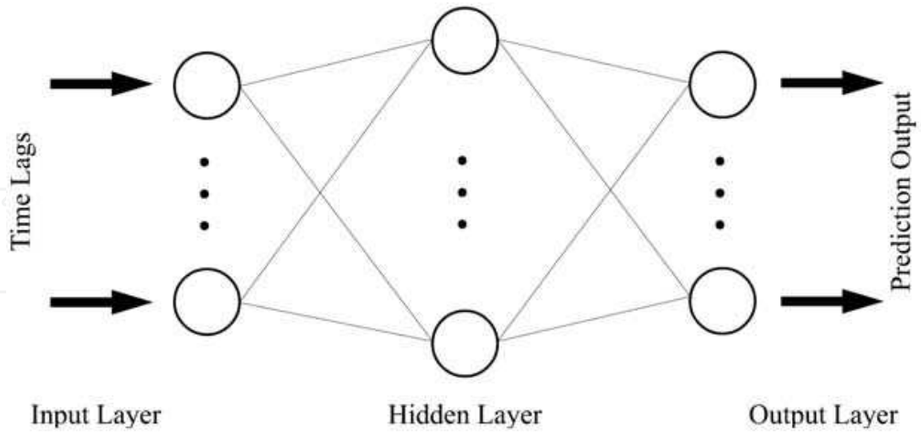


Fig. 2. Fully connected three layer MLP neural network.

An MLP is able to mapp past observations (network input) in their future values (network output). However, before having the capability to perform a given task, it is necessary that the network passes through a training or learning process. The MLP is typically trained by a supervised process and an external supervisor presents the input patterns and adjusts the weights of the network, according to the ANN success degree. For each pair of input-output,

the network will be adjusted to make the mapping between input patterns and desired output (output pattern).
The ANN training is usually a very complex process, and according to a given problem, it requires a large number of input patterns. Each pattern of these vectors is presented to a neuron of the network input layer. In the time series prediction problem, the number of processing units in the ANN input layer is determined by the number of time lags of a given time series.
The set of patterns (or historical data) are usually divided into three sets according to Prechelt [54]: training set (50% of the points), validation set (25% of the points) and test set (25% of the points). Thus, the ANN training uses these three sets. Initially, the set of training examples are presented to the network, then the information passed between the input, hidden and output layers, the response is calculated, then a training algorithm is perfomed to minimize the global error achieved by the network, calculating new values for the weights of the network, taking the difference between the desired output and the output obtained by the network, as in Sum of Squared Errors (SSE), given by

$$\frac{1}{2}\sum_{n=1}^{N}(target_i - output_i)^2, \tag{21}$$

where $target_i$ is the real value of the i-th pattern, $output_i$ is the response obtained by the network to the i-th pattern, and the factor $\frac{1}{2}$ is just a term for the simplification derived from the expressions in Equation 21, often calculated in training algorithms such as BackPropagation in [49].

2.7 Genetic algorithms

Evolutionary Algorithms (EAs) are a powerful class of stochastic optimization algorithms and have been widely used to solve complex problems which cannot be solved analytically. The most popular EA is the Genetic Algorithm (GA) [55, 56]. The GAs were developed by Holland [55] motivated by Charles Darwin's Evolution Theory [57], where its main goal was to find ways in which the mechanisms of natural adaptation might be integrated into computer systems. The GAs are used successfully in several kind of real-world applications due to their high search power in state spaces, being widely applied to optimization and learning machine problems. The GAs work with a set of attempt solutions (initial states) for the problem. This set, referred to as population, is evolved towards a sub-optimal or optimal solution to a given problem by performing a search in the multiple trajectory simultaneously.
Standard Genetic Algorithm. In this section, a brief description of Standard Genetic Algorithm (SGA) procedure is presented, which is illustrated in Figure 3. More details will be supplied as follows. For further details see [56, 58-60].
According to Figure 3, the SGA procedure starts with the creation of an individuals' population, or more specifically, the solutions set. Then, each individual is evaluated by a fitness function (or cost function), which is a heuristic function that guides the search for an optimal solution in state space. After evaluating the SGA population, it is necessary to use some procedures to select the individual parent pairs, which will be used to perform the genetic operators (crossover and mutation). There are some procedures to perform this selection, and is worth mentioning the rank-based selection, elitist strategies, steady-state

```
1  begin SGA
2      τ = 0;                  // τ: iteration number
3      initialize p;           // p: population
4      evaluate f(p);          // f(·): fitness or cost function
5      while not termination condition do
6          τ = τ+1;
7          select individuals parents pairs from p;
8          perform crossover operator with the selected individuals parents pairs;
9          generate a new population (np);
10         perform the mutation operator with the np;
11         evaluate f(np);
12         p = np;
13     end
14 end
```

Fig. 3. Standard genetic algorithm procedure.

election and tournament selection [16], amongst others. The next step is responsible for performing the crossover genetic operator. Usually, the crossover operator mixes the parent genes for exchanging genetic information from both, obtaining its individual offspring. There are some procedures to perform the crossover operator such as one-point, two-point or multi-point crossover, arithmetic crossover, heuristic crossover [16], amongst others. After crossover operator, all offspring individuals will be the new population, which contains relevant characteristics of all individual parent pairs obtained in the selection process. The next step is to mutate the new population. The mutation operator is responsible for the individual genes aleatory modification, allowing the population diversification and enabling SGA to escape from the local minima (or maxima) of the surface of the cost function (fitness). After that, the new mutated population is evaluated. This procedure is repeated until a stop condition has been reached.

Modified Genetic Algorithm. The Modified Genetic Algorithm (MGA) used here is based on the work of Leung et al. [16]. The MGA is a second version of the Standard Genetic Algorithm (SGA) [56, 58, 59] that was modified to improve search convergence. The SGA was first studied, and, then, was modified to accelerate its convergence through the use of modified crossover and mutation operators (described later). The algorithm is described in Figure 4.

According to Figure 4, the MGA procedure consists of selecting a parent pair of chromosomes and then performing crossover and mutation operators (generating the offspring chromosomes – the new population) until the termination condition is reached; then the best individual in the population is selected as a solution to the problem.

The crossover operator is used for exchanging information from two parents (vectors $\underline{p}_1$ and $\underline{p}_2$) obtained in the selection process by a roulette wheel approach [16]. The recombination process to generate the offsprings (vectors $\underline{C}_1$,$\underline{C}_2$,$\underline{C}_3$ and $\underline{C}_4$) is done by four crossover operators, which are defined by the following equations [16]:

$$\underline{C}_1 = \frac{\underline{p}_1 + \underline{p}_2}{2}, \tag{22}$$

$$\underline{C}_2 = \underline{p}_{max}(1-w) + max(\underline{p}_1, \underline{p}_2)w, \tag{23}$$

$$\underline{C}_3 = \underline{p}_{min}(1-w) + min(\underline{p}_1, \underline{p}_2)w, \tag{24}$$

$$\underline{C}_4 = \frac{(\underline{p}_{max} + \underline{p}_{min})(1-w) + (\underline{p}_1 + \underline{p}_2)w}{2}, \tag{25}$$

```
begin MGA
    τ = 0;                    // τ: actual iteration
    initialize population;
    evaluate f(population);   // f(·): fitness function
    while not termination condition do
        τ = τ+1;
        select a parent pair of chromosomes (p_1 and p_2) from population;
        begin crossover operator
            generate the offsprings C_1, C_2, C_3 and C_4 by Equations (22)-(25);
            the offspring with the best fitness function is denoted C^best;
        end
        begin mutation operator with C^best
            generate the mutated offsprings M_1, M_2 and M_3 by Equation (26);
        end
        // generate a new population
        insert C^best in the population;
        if random number < p_mut then
            the one among M_1, M_2 and M_3 with largest fitness value replaces the individual of the population with the
            smallest fitness value;
        else
            if f(M_1) > smallest fitness value in the population then
                M_1 replaces the individual of the population with the smallest fitness value;
            end
            if f(M_2) > smallest fitness value in the population then
                M_2 replaces the individual of the population with the smallest fitness value;
            end
            if f(M_3) > smallest fitness value in the population then
                M_3 replaces the individual of the population with the smallest fitness value;
            end
        end
        evaluate f(population);
    end
end
```

Fig. 4. The modified genetic algorithm procedure.

where $w \in [0, 1]$ denotes the crossover weight (the closer w is to 1, the greater is the direct contribution from parents), $max(\underline{p}_1, \underline{p}_2)$ and $min(\underline{p}_1, \underline{p}_2)$ denotes the vector whose elements are the maximum and the minimum, respectively, between the gene values of $\underline{p}_1$ and $\underline{p}_2$. The terms $\underline{p_{max}}$ and $\underline{p_{min}}$ denote a vector with the maximum and minimum possible gene values, respectively. After offspring generation by crossover operators, the offspring with the best evaluation (greater fitness value) will be chosen as the offspring generated by the crossover process and will be denoted by $\underline{C}^{best}$.

After the crossover operator, $\underline{C}^{best}$ is selected to have a mutation process, where three new mutated offsprings are generated and defined by the following equation [16]:

$$\underline{M}_j = C_i^{best} + \gamma_i \Delta M_i, \; j = 1, 2, 3 \text{ and } i = 1, 2, \ldots, NG, \tag{26}$$

where γ_i can only take the values 0 or 1, ΔM_i are randomly generated numbers such that $p_{min} \leq C_i^{best} + \Delta M_i \leq p_{max}$ and NG denotes the number of genes in the chromosome.

The first mutated offspring (M_1) is obtained according to (26) using only one term γ_i set to 1 (i is randomly selected within the range [1,NG]) and the remaining terms γ_i are set to 0. The second mutated offspring (M_2) is obtained according to (26) using some γ_i, randomly chosen, set to 1 and the remaining terms γ_i are set to 0. The third mutated offspring (M_3) is obtained according to (26) using all γ_i set to 1.

It is worth mentioning that the GA is not directly used for modeling and predicting time series, but it is applied to support other methods and techniques in the search for the optimal or sub-optimal parameters of the predictive model [15].

2.8 Intelligent hybrid models

Humans can be considered a good example of machines that have hybrid information. Their attitudes and actions are governed by a combination of genetic information and information

acquired through learning. In genetic information, known as genotype, the information that come with the individual in the form of genetic coding, are the features inherited from your parents. The phenotype is the combination of features given by genotype combined with the environmental influences. The information in our genes ensures the success of our survival, which has been proven and tested over millions of years of evolution. Human learning consists of a variety of complex processes that use information acquired from environmental interactions. It is the combination of these different types of methods of processing information that enables humans to succeed in their survival in dynamic environments that change all the time.

This kind of hybrid information processing has been replicated on adaptive machines generation, where in their main unit processing there are intelligent computing systems and some mechanisms inspired by nature. It is possible to find some examples: neural networks [49, 61], genetic algorithms [56, 58, 59], fuzzy systems [62], artificial immune systems [63], expert systems [64] and induction rules [65]. The IA techniques have produced encouraging results in some particular tasks, but some complex problems, such as time series prediction, can not be successfully solved by a single intelligent technique. Each of these techniques have strengths and weaknesses, which make them suitable for some and not other problems. These limitations have been the main motivation for the study of Hybrid Intelligent Systems (HIS) where two or more AI techniques are combined in order to overcome the particular limitations of an individual technique. Hybrid Intelligent systems are also important when considering a wide range of real world applications. Many areas have many complex components of different problems, each of them may require a different type of processing. Moreover, the HIS being can be combined with different techniques, including conventional computing systems. The reasons for the HIS built are numerous, but can be summarized in three [66]:

1. Intensification Techniques: the integration of at least two different techniques, with the purpose of offsetting the weakness of a technique with the strength of the other;
2. Multiplicity of Applications in Tasks: A HIS is built, with the purpose of a single technique not being applied to many sub-problems that some application might have;
3. Implementation of Multiple Feature: the HIS build exhibits the capacity for multiple processing information within an architecture. Functionally, these systems emulate or mimic different processing techniques.

There are many possible combinations of the various techniques of artificial intelligence for hybrid intelligent systems built, however the discussion outlined here will be limited to a combination of techniques such as artificial neural networks and genetic algorithms.

TAEF Model. The Time-delay Added Evolutionary Forecasting (TAEF) method [15] tries to reconstruct the phase space of a given time series by carrying out a search for the minimum dimensionality necessary to reproduce the generator phenomenon of the time series. The TAEF method is an intelligent hybrid system based on Artificial Neural Networks (ANNs) architectures trained and adjusted by a Modified Genetic Algorithm (MGA) which not only searches for the ANN parameters but also for the adequate embedded dimension represented in the time lags.

The scheme describing the TAEF algorithm is based on the iterative definition of the four main elements: (i) the underlying information necessary to predict the series (the minimum number of lags), (ii) the structure of the model capable of representing such underlying information for the purpose of prediction (the number of units in the ANN structure), (iii)

the appropriate algorithm for training the model, and (iv) a behavior test to adjust time phase distortions that appear in some time series.
Following this principle, the important parameters defined by the algorithm are:

1. The number of time lags to represent the series;
2. The number of units in the ANN hidden layer;
3. The training algorithm for the ANN.

The TAEF method starts with the user defining a minimum initial fitness value (*MinFit*) which should be reached by at least one individual of the population in a given MGA round. The fitness function is defined as

$$\text{Fitness Function} = \frac{1}{1 + \text{MSE}} \tag{27}$$

where MSE is the Mean Squared Error of the ANN and will be formally defined in Section 5.
In each MGA round, a population of M individuals are generated, each of them being represented by a chromosome (in Ferreira's works [15] $M = 10$ was used). Each individual is in fact a three-layer ANN where the first layer is defined by the number of time lags, the second layer is composed of a number of hidden processing units (sigmoidal units) and the third layer is composed by one processing unit (prediction horizon of one step ahead).
The stopping criteria for each one of the individuals are the number of epochs (NEpochs), the increase in the validation error (*Gl*) and the decrease in the training error (*Pt*).
The best repetition (the smallest validation error) is chosen to represent the best individual. Following this procedure, the MGA evolves towards an optimal or close to optimal fitness solution (which may not be the best solution yet), according to the stopping criteria: number of generations created (NGen) and fitness evolution of the best individual (BestFit).
After this point, when the MGA reaches a solution, the algorithm checks if the fitness of the best individual paired or overcame the initial value specified for the variable MinFit (minimum fitness requirement). If this is not the case, the value of MaxLags (maximum number of lags) is increased by one and the MGA procedure is repeated to search for a better solution.
However, if the fitness reached was satisfactory, then the algorithm checks the number of lags chosen for the best individual, places this value as MaxLags, sets MinFit with the fitness value reached by this individual, and repeats the whole MGA procedure. In this case, the fitness achieved by the best individual was better than the fitness previously set and, therefore, the model can possibly generate a solution of higher accuracy with the lags of the best individual (and with the MinFit reached by the best individual as the new target). If, however, the new value of MinFit is, again, not reached in the next round, MaxLags gets the same value defined for it, just before the round that found the best individual, increased by one (the maximum number of lags is increased by one). The state space for the lag search is then increased by one to allow a wider search for the definition of the lag set. This procedure goes on until the stop condition is reached. After that, the TAEF method chooses the best model found among all the candidates.
After the best model is chosen, when the training process is finished, a statistical test (*t*-test) is employed to check if the network representation has reached an "in-phase" matching (without a one step shift – the shape of the time series and the shape of the generated prediction has a time matching) or "out-of-phase" matching (with a one step shift { the shape of the time series and the shape of the generated prediction do not have a time

matching). If this test accepts the "in-phase" matching hypothesis, the elected model is ready for practical use. Otherwise, the method carries out a new procedure to adjust the relative phase between the prediction and the actual time series. The validation patterns are presented to the ANN and the output of these patterns are re-arranged to create new inputs that are both presented to the same ANN and set as the output (prediction) target.
It is worth mentioning that the variable *cont* just represents the current iteration of the TAEF method. The maximum of ten iterations of the TAEF method (given by expression *not cont* > 10), was chosen empirically according to previous experiments in order to generate an optimal prediction model.

3. Mathematical morphology

The Mathematical Morphology (MM) is based on two basic operations, the sum and substraction of Minkowski [67], which are respectively given by [68]

$$X \oplus B = \bigcup_{b \in B} X_b; \tag{28}$$

$$X \ominus B = \bigcap_{b \in B^r} X_b, \tag{29}$$

where $X_b = \{x + b : x \in X\}$ represents the input signal and $B_r = \{-b : b \in B\}$ is the reflected structuring element B.
All of MM transformations are based on combinations of four basic operations, which are defined by [68]

$$\text{Dilation: } \delta_B(X) = X \oplus B; \tag{30}$$

$$\text{Erosion: } \epsilon_B(X) = X \ominus B; \tag{31}$$

$$\text{Anti-Dilation: } \delta_B^a(X) = (X \oplus B^{rc})^{rc}; \tag{32}$$

$$\text{Anti-Erosion: } \epsilon_B^a(X) = (X \ominus B^{rc})^{rc}, \tag{33}$$

where $B_{rc} = \{-b : b \notin B\}$ represents the reflected complement of structuring element B.
According to Sousa [68], an operator of kind Ψ: $P(E) \rightarrow P(E)$, where $P(E)$ represents all subsets of $E = \mathbb{R}^N$, may be a translation invariant (Equation 34), increasing (Equation 35), decreasing (Equation 36) or window (Equation 37).

$$\Psi(X_{\underline{h}}) = (\Psi(X))_{\underline{h}}, \tag{34}$$

where $X_{\underline{h}} = \{x + h : x \in X\}$ represents the translation of $X \in P(E)$ by vector $h \in E$.

$$X_{\underline{h}} = \{x + \underline{h} : x \in X\} \tag{35}$$

$$X \supset Y \Rightarrow \Psi(X) \supset \Psi(Y), \forall X, Y \in P(E); \tag{36}$$

$$\forall x \in E\ ,\ x \in \Psi(X) \Leftrightarrow x \in \Psi(X \cap L_x), \tag{37}$$

where L_x is the translation of $L \in E$ finite.

3.1 Morphological-Rank-Linear (MRL) filter preliminaries

Definition 1 – Rank Function: the r-th rank function of the vector t = (t_1, t_2,…, t_n) $\in \mathbb{R}^n$ is the r-th element of the vector t sorted in decreasing order ($t_{(1)} \geq t_{(2)} \geq \ldots \geq t_{(n)}$). It is denoted by [31]

$$\mathcal{R}_r(\underline{t}) = t_{(r)}, \quad r = 1, 2, \ldots, n. \tag{38}$$

For example, given the vector t = (3, 0, 5, 7, 2, 1, 3), its 4-th rank function is $\mathcal{R}_4(\underline{t}) = 3$.

Definition 2 – Unit Sample Function: the unit sample function is given by [31]

$$q(v) = \begin{cases} 1, \text{ if } v = 0, \\ 0, \text{ otherwise.} \end{cases} \tag{39}$$

where $v \in \mathbb{R}$.

Applying the unit sample function to a vector $\underline{v}$ = (v_1, v_2,…, v_n) $\in \mathbb{R}^n$, yields a vector unit sample function ($Q(\underline{v})$), given by [31]

$$Q(\underline{v}) = [q(v_1), q(v_2), \ldots, q(v_n)]. \tag{40}$$

Definition 3 – Rank Indicator Vector: the r-th rank indicator vector c of t is given by [31]

$$\underline{c}(\underline{t}, r) = \frac{Q\left((z \cdot \underline{1}) - \underline{t}\right)}{Q\left((z \cdot \underline{1}) - \underline{t}\right) \cdot \underline{1}^T}, \tag{41}$$

where $z = \mathcal{R}_r(\underline{t})$, $\underline{1} = (1, 1, \ldots, 1)$ "$\cdot$" represents scalar product and the symbol T denotes transposition.

For example, given the vector $\underline{t}$ = (3, 0, 5, 7, 2, 1, 3), its 4-th rank indicator function is $\underline{c}(\underline{t}, 4) = \frac{1}{2}(1, 0, 0, 0, 0, 0, 1)$.

Definition 4 – Smoothed Rank Function: the smoothed r-th rank function is given by [31]

$$\mathcal{R}_{r,\sigma}(\underline{t}) = \underline{c}_\sigma(\underline{t}, r) \cdot \underline{t}^T, \tag{42}$$

with

$$\underline{c}_\sigma(\underline{t}, r) = \frac{Q_\sigma\left((z \cdot \underline{1}) - \underline{t}\right)}{Q_\sigma\left((z \cdot \underline{1}) - \underline{t}\right) \cdot \underline{1}^T}, \tag{43}$$

where c_σ is an approximation for the rank function $\underline{c}$ and $Q_\sigma(\underline{v}) = [q_\sigma(v_1), q_\sigma(v_2), \ldots, q_\sigma(v_n)]$ is a smoothed impulse function (where $q_\sigma(v)$ is like $sech^2(v/\sigma)$) (where $sech$ is the hyperbolic secant), $\sigma \geq 0$ is a scale parameter and "$\cdot$" represents the scalar product.

Thus, $\underline{c}_\sigma$ is an approximation for the rank indicator vector $\underline{v}$. Using ideas from the fuzzy set theory, $\underline{c}_\sigma$ can also be interpreted as a membership function vector [31]. For example, if the vector $\underline{t}$ = (3, 0, 5, 7, 2, 1, 3), $q_\sigma(v) = sech^2(\frac{v}{\sigma})$ and $\sigma = 0.5$ then its smoothed 4-th rank indicator function is

$$\underline{c}_\sigma(\underline{t},4) = \tfrac{1}{2}(0.9646, 0, 0.0013, 0, 0.0682, 0.0013, 0.9646),$$

where $\underline{c}(\underline{t},4) = \tfrac{1}{2}(1,0,0,0,0,0,1)$.

3.2 MRL filter definition

The MRL filter [31] is a linear combination between a Morphological-Rank (MR) filter [29, 30] and a linear Finite Impulse Response (FIR) filter [31].

Definition 5 – MRL Filter [31]: Let $\underline{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ represent the input signal inside an n-point moving window and let y be the output from the filter. Then, the MRL filter is defined as the shift-invariant system whose local signal transformation rule $\underline{x} \rightarrow y$ is given by [31]

$$y = \lambda\alpha + (1-\lambda)\beta, \tag{44}$$

with

$$\alpha = \mathcal{R}_r(\underline{x} + \underline{a}) = \mathcal{R}_r(x_1 + a_1, x_2 + a_2, \ldots, x_n + a_n), \tag{45}$$

and

$$\beta = \underline{x} \cdot \underline{b}' = x_1 b_1 + x_2 b_2 + \ldots + x_n b_n, \tag{46}$$

where $\lambda \in \mathbb{R}$, $\underline{a}$ and $\underline{b} \in \mathbb{R}^n$. Terms $\underline{a} = (a_1, a_2, \ldots, a_n)$ and $\underline{b} = (b_1, b_2, \ldots, b_n)$ represent the coefficients of the MR filter and the coefficients of the linear FIR filter, respectively. Term $\underline{a}$ is usually referred to "structuring element" because for $r = 1$ or $r = n$ the rank filter becomes the morphological dilation and erosion by a structuring function equal to $\pm\underline{a}$ within its support [31]. The structure of the MRL filter is illustrated in Figure 5.

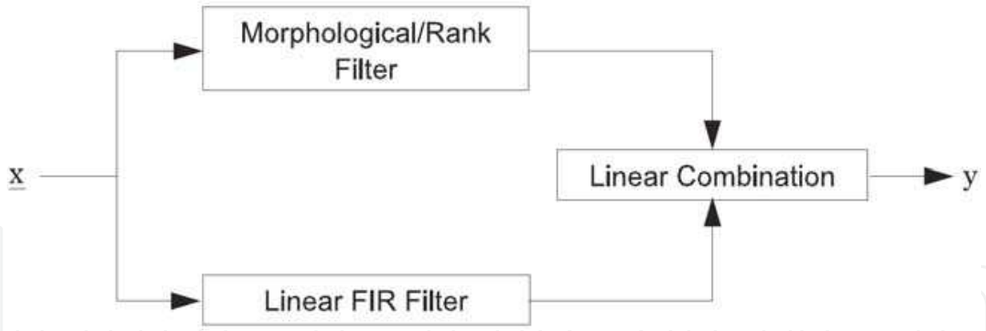


Fig. 5. Structure of the MRL filter.

3.3 MRL filter training algorithm

Pessoa and Maragos [31] presented an adaptive design of MRL filters based on the LMS algorithm [29, 30], the "rank indicator vector"" [31] and "smoothed impulses" [31] for overcoming the problem of nondifferentiability of rank operations.

Pessoa and Maragos [31] have shown that the main goal of the MRL filter is to specify a set of parameters ($\underline{a}$, $\underline{b}$, r, λ) according to some design requirements. However, instead of using the integer rank parameter r directly in the MRL filter definition equations (44-46), they argued that it is possible to work with a real variable ρ implicitly defined through the following rescaling [31]

$$r = round\left(n - \frac{n-1}{exp(-\rho)}\right), \tag{47}$$

where $\rho \in \mathbb{R}$, n is the dimension of the input signal vector $\underline{x}$ inside the moving window and round(·) denotes the usual symmetrical rounding operation. In this way, the weight vector to be used in the filter design task is defined by [31]

$$\underline{w} \equiv (\underline{a}, \underline{b}, \rho, \lambda). \tag{48}$$

The framework of the MRL filter adaptive design is viewed as a learning process where the filter parameters are iteratively adjusted. The usual approach to adaptively adjust the vector $\underline{w}$, and therefore design the filter, is to define a cost function $J(\underline{w})$, estimate its gradient $\nabla J(\underline{w})$, and update the vector $\underline{w}$ by the iterative formula

$$\underline{w} \equiv (\underline{a}, \underline{b}, \rho, \lambda). \tag{49}$$

where $\mu_0 > 0$ (usually called step size) and $i \in \{1, 2, \ldots\}$. The term μ_0 is responsible for regulating the tradeoff between stability and speed of convergence of the iterative procedure. The iteration of Equation 49 starts with an initial guess $\underline{w}(0)$ and stops when some desired condition is reached. This approach is known as the method of gradient steepest descent [31].

The cost function J must reflect the solution quality achieved by the parameters configuration of the system. A cost function J, for example, can be any error function, such as

$$J[\underline{w}(i)] = \frac{1}{M} \sum_{k=i-M+1}^{i} e^2(k), \tag{50}$$

where $M \in \{1, 2,\ldots\}$ is a memory parameter and $e(k)$ is the instantaneous error, given by

$$e(k) = d(k) - y(k), \tag{51}$$

where $d(k)$ and $y(k)$ are the desired output signal and the actual filter output for the training sample k, respectively. The memory parameter M controls the smoothness of the updating process. If we are processing noiseless signals, $M = 1$ is recommended [31]. However, when we use $M > 1$, the updating process tends to reduce the noise influence of noisy signals during the training [31].

Hence, the resulting adaptation algorithm is given by [31]

$$\underline{w}(i+1) = \underline{w}(i) + \frac{\mu}{M} \sum_{k=i-M+1}^{i} e^2(k) \frac{\partial y(k)}{\partial \underline{w}}, \tag{52}$$

where $\mu = 2\mu_0$ and $i \in \{1, 2,\ldots\}$. From Equations (44), (45), (46) and (48), term $\frac{\partial y(k)}{\partial \underline{w}}$ [31] may be calculated as

$$\frac{\partial y}{\partial \underline{w}} = \left(\frac{\partial y}{\partial \underline{a}}, \frac{\partial y}{\partial \underline{b}}, \frac{\partial y}{\partial \rho}, \frac{\partial y}{\partial \lambda}\right) \tag{53}$$

with

$$\frac{\partial y}{\partial \underline{a}} = \lambda \frac{\partial \alpha}{\partial \underline{a}}, \tag{54}$$

$$\frac{\partial y}{\partial \underline{b}} = (1-\lambda)\underline{x}, \tag{55}$$

$$\frac{\partial y}{\partial \rho} = \lambda \frac{\partial \alpha}{\partial \rho}, \tag{56}$$

$$\frac{\partial y}{\partial \lambda} = (\alpha - \beta), \tag{57}$$

where

$$\frac{\partial \alpha}{\partial \underline{a}} = \underline{c} = \frac{Q\left((\alpha \cdot \underline{1}) - \underline{x} - \underline{a}\right)}{Q\left((\alpha \cdot \underline{1}) - \underline{x} - \underline{a}\right) \cdot \underline{1}'}, \tag{58}$$

$$\frac{\partial \alpha}{\partial \rho} = 1 - \frac{1}{n} Q\left((\alpha \cdot \underline{1}) - \underline{x} - \underline{a}\right) \cdot \underline{1}', \tag{59}$$

where n is the dimension of $\underline{x}$ and $\alpha = \mathcal{R}_r(\underline{x} + \underline{a})$.
It is important to mention that the unit sample function Q is frequently replaced by smoothed impulses Q_σ, in which case an appropriate smoothing parameter σ should be selected (which will affect only the gradient estimation step in the design procedure [31]).

4. The proposed morphological-rank-linear time-lag added forecasting (MRLTAEF) model

The approach model in this work, referred to as Morphological-Rank-Linear Time-lag Added Evolutionary Forecasting (MRLTAEF) model, uses an evolutionary search mechanism in order to train and adjust the Morphological-Rank-Liner (MRL) filter applied to financial time series prediction. It is based on the definition of the four main elements necessary for building an accurate forecasting system [15]:

- The underlying information necessary to predict the time series;
- The structure of the model capable of representing such underlying information for the purpose of prediction;
- The appropriate algorithm for training the model
- The behavior statistical test to adjust time phase distortions

It is important to consider the minimum possible number of time lags in the representation of the series because the model must to be as parsimonious as possible, avoiding the overfitting problem and decreasing the computacional cost.
Based on that definition, the proposed method consists of a hybrid intelligent morphological-rank-linear model composed of a MRL filter [31] with a MGA [16], which searches for:

1. The minimum number of time lags to represent the series: initially, a maximum number of time lags ($MaxLags$) is pre-defined and then the MGA will search for the number of time lags in the range [1,$MaxLags$] for each individual of the population;
2. The initial (sub-optimal) parameters of the MRL filter (mixing parameter (λ), rank (r), linear Finite Impulse Response (FIR) filter ($\underline{b}$) and the Morphological-Rank (MR) filter ($\underline{a}$) coefficients.

Then, each element of the MGA population is trained via LMS algorithm [31] to further improve the parameters supplied by the MGA, that is, the LMS is used, for each individual candidate, to perform a local search around the initial parameters supplied by MGA. The main idea used here is to conjugate a local search method (LMS) to a global search method (MGA). While the MGA makes it possible to test of varied solutions in different areas of the solution space, the LMS acts on the initial solution to produce a fine-tuned forecasting model. The proposed method is described in Figure 6.

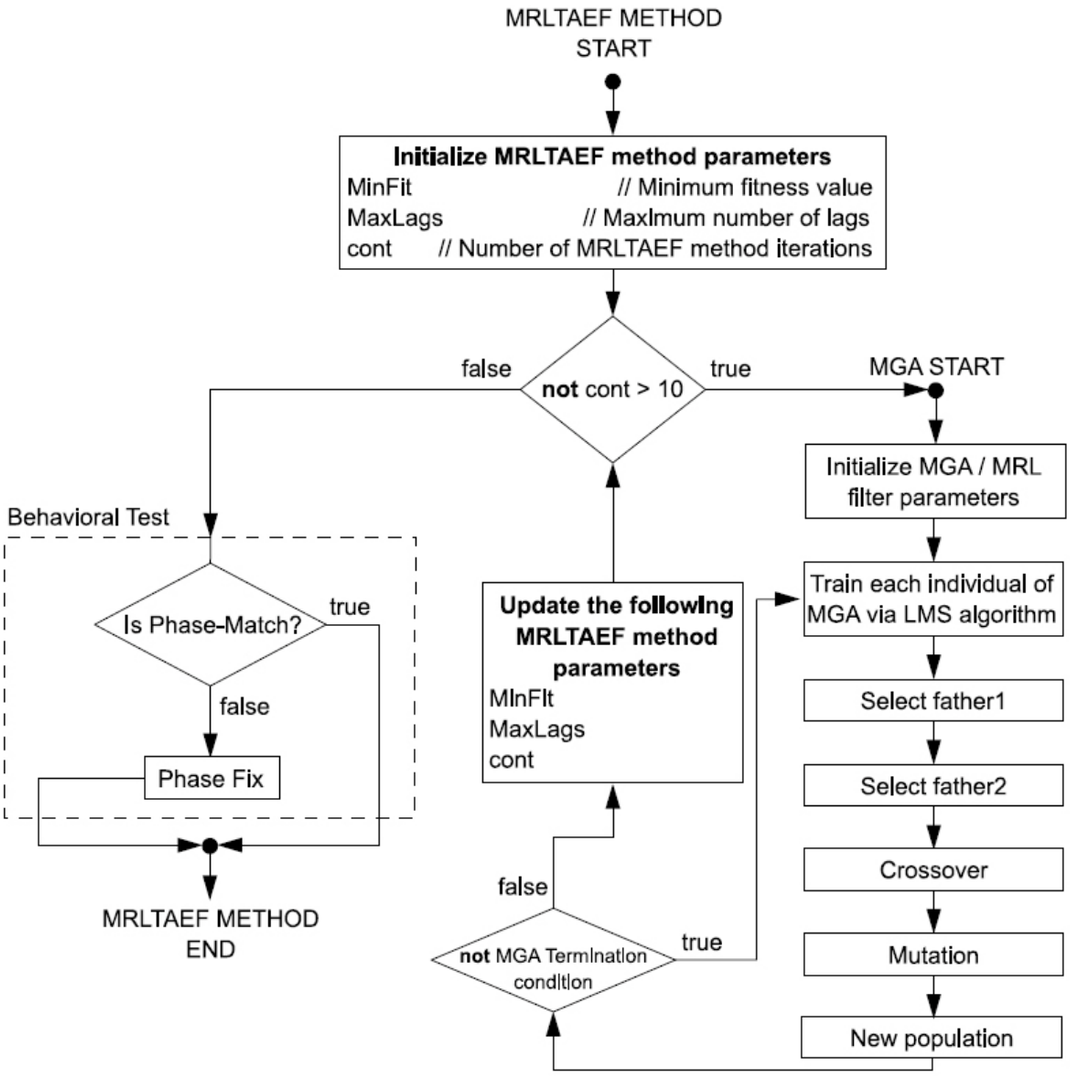


Fig. 6. The proposed method.

Such a process is able to seek the most compact MRL filter, reducing computational cost and probability of model overfitting. Each MGA individual represents a MRL filter, where its input is defined by the number of time lags and its output represents the prediction horizon of one step ahead.

Most works found in the literature have the fitness function (or objective function) based on just one performance measure, like Mean Square Error (MSE). However, Clements et al. [69], since 1993 has shown that the MSE measure has some limitations of availability and comparing the prediction model performance. Information about the prediction, as the absolute percentage error, the accuracy in the future direction prediction and the relative gain regarding naive prediction models (like random walk models and mean prediction) are not described using MSE measure.

In order to provide a more robust forecasting model, a multi-objective evaluation function is defined, which is a combination of five well-known performance measures: Prediction Of Change In Direction (POCID), Mean Square Error (MSE), Mean Absolute Percentage Error (MAPE), Normalized Mean Square Error (NMSE) or U of Theil Statistic (THEIL) and Average Relative Variance (ARV), where all these measures will be formally defined in Section 5. The multi-objective evaluation function used here is given by

$$\text{Fitness Function} = \frac{\text{POCID}}{1 + \text{MSE} + \text{MAPE} + \text{THEIL} + \text{ARV}}. \tag{60}$$

Whereas there are linear and nonlinear metrics in the such evaluation function and each one of these metrics can contribute to different forms for the evolution process, the Equation 60 was built from empirical form to have all information necessary to describe as well as allow the time series generator phenomenon.

After MRL filter adjusting and training, the proposed method uses the phase fix procedure presented by Ferreira [15], where a two step procedure is introduced to adjust time phase distortions observed ("out-of-phase" matching) in financial time series. Ferreira [15] has shown that the representations of some time series (natural phenomena) were developed by the model with a very close approximation between the actual and the predicted time series (referred to as "in-phase" matching), whereas the predictions of other time series (mostly financial time series) were always presented with a one step delay regarding the original data (referred to as "out-of-phase" matching).

The proposed method uses the statistical test (t-test) to check if the MRL filter model representation has reached an in-phase or out-of-phase matching (in the same way of TAEF method [15]). This is conducted by comparing the outputs of the prediction model with the actual series, making use only of the validation data set. This comparison is a simple hypothesis test, where the null hypothesis is that the prediction corresponds to in-phase matching and the alternative hypothesis is that the prediction does not correspond to in-phase matching (or corresponds to out-of-phase matching).

If this test accepts the in-phase matching hypothesis, the elected model is ready for practical use. Otherwise, the proposed method performs a new procedure to adjust the relative phase between the prediction and the actual time series. The phase fix procedure has two steps (described in Figure 7): (i) the validation patterns are presented to the MRL filter and the output of these patterns are re-arranged to create new inputs patterns (reconstructed patterns), and (ii) these reconstructed patterns are represented to the same MRL filter and the output set as the prediction target. This procedure of phase adjustment considers that

the MRL filter is not a random walk model, it just shows a behavior characteristic of a random walk model: the $t + 1$ prediction is taken as the t value (Random Walk Dilemma).
If the MRL filter was like a random walk model, the phase adjust procedure would not work. Such phase fix was originally proposed by Ferreira [15], where he observed the fact that when Artificial Neural Network (ANN – Multilayer Perceptron like) is correctly adjusted (TAEF method), the one step shift distortion in the prediction can be softened.

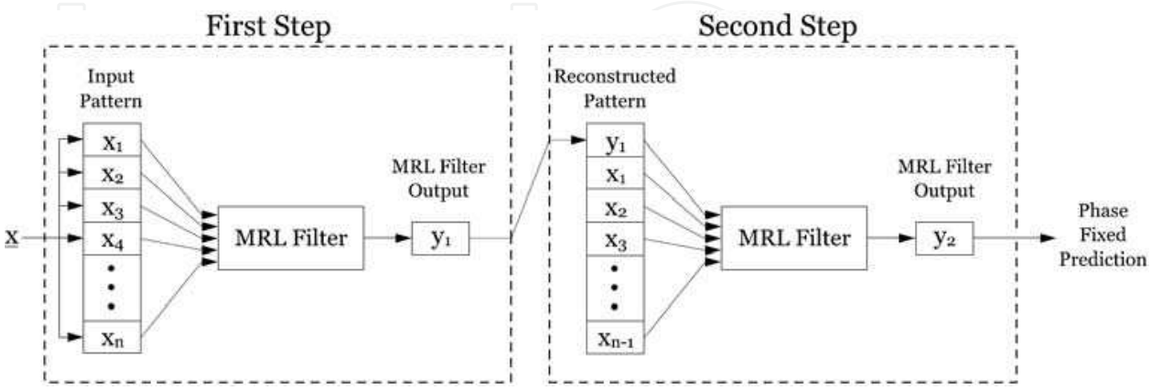


Fig. 7. Phase fix procedure.

The termination conditions for the MGA are:

1. Minimum value of fitness function: $fitness \geq 40$, where this value mean the accuracy to predict direction around 80% ($POCID \gtrsim 80\%$) and the sum of the other errors around one ($MSE + MAPE + THEIL + ARV \cong 1$);
2. The increase in the validation error or generalization loss (Gl) [54]: $Gl > 5\%$;
3. The decrease in the training error process training (Pt) [54]: $Pt \leq 10^{-6}$.

Each individual of the MGA population is a MRL filter represented by the data structure with the following components (MRL filter parameters):

- $\underline{a}$: MR filter coefficients;
- $\underline{b}$: linear FIR filter coefficients;
- ρ: variable used to determine the rank r;
- λ: mixing parameter;
- NLags: a vector, where each position has a real-valued codification, which is used to determine if a specific time lag will be used ($\text{NLags}_i > 0$) or not ($\text{NLags}_i \leq 0$).

5. Performance metrics

Many performance evaluation criteria are found in literature. However, most of the existing literature on time series prediction frequently employ only one performance criterion for prediction evaluation. The most widely used performance criterion is the Mean Squared Error (MSE), given by

$$\text{MSE} = \frac{1}{N}\sum_{j=1}^{N}(\text{target}_j - \text{output}_j)^2, \tag{61}$$

where N is the number of patterns, target_j is the desired output for pattern j and output_j is the predicted value for pattern j.

The MSE measure may be used to drive the prediction model in the training process, but it cannot be considered alone as a conclusive measure for comparison of different prediction models [69]. For this reason, other performance criteria should be considered for allowing a more robust performance evaluation.

A measure that presents accurately identifying model deviations is the Mean Absolute Percentage Error (MAPE), given by

$$\mathrm{MAPE} = \frac{1}{N}\sum_{j=1}^{N}\left|\frac{\mathrm{target}_j - \mathrm{output}_j}{x_j}\right|, \tag{62}$$

where x_j is the time series value at point j.

The random walk dilemma can be used as a naive predictor ($X_{t+1} = X_t$), commonly applied to financial time series prediction. Thus, a way to evaluate the model regarding a random walk model is using the Normalized Mean Squared Error (NMSE) or U of Theil Statistic (THEIL) [70], which associates the model performance with a random walk model, and given by

$$\mathrm{THEIL} = \frac{\sum_{j}^{N}(\mathrm{target}_j - \mathrm{output}_j)^2}{\sum_{j}^{N}(\mathrm{target}_j - \mathrm{target}_{j-1})^2}, \tag{63}$$

where, if the THEIL is equal to 1, the predictor has the same performance than a random model. If the THEIL is greater than 1, then the predictor has a performance worse than a random walk model, and if the THEIL is less than 1, the predictor is better than a random walk model. In the perfect model, the THEIL tend to zero.

Another interesting measure maps the accuracy in the future direction prediction of the time series or, more specifically, the ability of the method to predict if the future series value (prediction target) will increase or decrease with respect to the previous value. This metric is known as the Prediction Of Change In Direction (POCID) [15], and is given by

$$\mathrm{POCID} = \frac{100}{N}\sum_{j=1}^{N} D_j, \tag{64}$$

where

$$D_j = \begin{cases} 1, \text{ if } (\mathrm{target}_j - \mathrm{target}_{j-1})(\mathrm{output}_j - \mathrm{output}_{j-1}) > 0 \\ 0, \text{ otherwise} \end{cases} \tag{65}$$

The last measure used associates the model performance with the mean of the time series. The measure is the Average Relative Variance (ARV), and given by

$$\mathrm{ARV} = \frac{\sum_{j=1}^{N}(\mathrm{target}_j - \mathrm{output}_j)^2}{\sum_{j=1}^{N}(\mathrm{output}_j - \overline{\mathrm{target}})^2}, \tag{66}$$

where, $\overline{\text{target}}$ is the mean of the time series. If the ARV is equal to 1, the predictor has the same performance of the time series average prediction. If the ARV is greater than 1, then the predictor has a performance worse than the time series average prediction, and if the ARV is less than 1, the predictor is better than the time series average prediction. In the ideal model, ARV tend to zero.

6. Simulations and experimental results

A set of six real world financial time series (Dow Jones Industrial Average (DJIA) Index, National Association of Securities Dealers Automated Quotation (NASDAQ) Index, Standard & Poor 500 Stock (S&P500) Index and Petrobras Stock Prices, General Motors Corporation Stock Prices and Google Inc Stock Prices) were used as a test bed for evaluation of the proposed method. All time series investigated were normalized to lie within the range $[0, 1]$ and divided into three sets according to Prechelt [54]: training set (50% of the points), validation set (25% of the points) and test set (25% of the points).

For all the experiments, the following initialization system parameters were used: $cont$ = 1, $MinFit$ = 40 and $MaxLags$ = 4. The MGA parameters used in the proposed MRLTAEF method are a maximum number of MGA generations, corresponding to 10^4, crossover weight w = 0.9 (used in the crossover operator), mutation probability equals to 0.1. The MR filter coefficients and the linear FIR filter coefficients ($\underline{a}$ and $\underline{b}$, respectively) were normalized in the range [–0.5, 0.5]. The MRL filter parameters λ and ρ were in the range [0, 1] and [–$MaxLags$,$MaxLags$], respectively.

Next, the simulation results involving the proposed model will be presented. In order to establish a performance study, results previously published in the literature with the TAEF Method [15] were examined in the same context and under the same experimental conditions. For each time series, ten experiments were done, where the experiment with the best validation fitness function is chosen to represent the prediction model.

In order to establish a performance study, results previously published in the literature with the TAEF Method [15] on the same series and under the same conditions are employed for comparison of results. In addition, experiments with MultiLayer Perceptron (MLP) networks and Morphological-Rank-Linear (MRL) filters were used for comparison with the MRLTAEF method. The Levenberg-Marquardt Algorithm [71] and the LMS algorithm [31] were employed for training the MLP network and the MRL filter, respectively. In all of the experiments, ten random initializations for each architecture were carried out, where the experiment with the best validation fitness function is chosen to represent the prediction model. The statistical behavioral test, for phase fix procedure, was also applied to all the MLP, MRL and TAEF models in order to guarantee a fair comparison among the models.

It is worth mentioning that the results with ARIMA models were not presented in our comparative analysis since Ferreira [15] has shown that MLP networks obtained results better than ARIMA models, for all financial time series used in this work. Therefore, only MLP networks were used in our comparative analysis.

Furthermore, in order to analyze time lag relations in the studied time series, the graphical methodology proposed by [42, 72], referred to as lagplot [72] or phase portrait [42], was employed. This consists of dispersion graph constructions relating the different time lags of the time series (X_t vs X_{t-1}, X_t vs X_{t-2}, X_t vs X_{t-3}, …), and allow observations of possible relative strong relationships between any pair of time lags (when a structured appearance is

shown in the graph). Although such technique is very limited since it depends on human interpretation of the graphs. However, its simplicity is a strong argument for its utilization [15].

6.1 Dow Jones Industrial Average (DJIA) index series

The Dow Jones Industrial Average (DJIA) Index series corresponds to daily records from January 1st 1998 to August 26th 2003, constituting a database of 1,420 points. Figure 8 shows the DJIA Index lagplot.

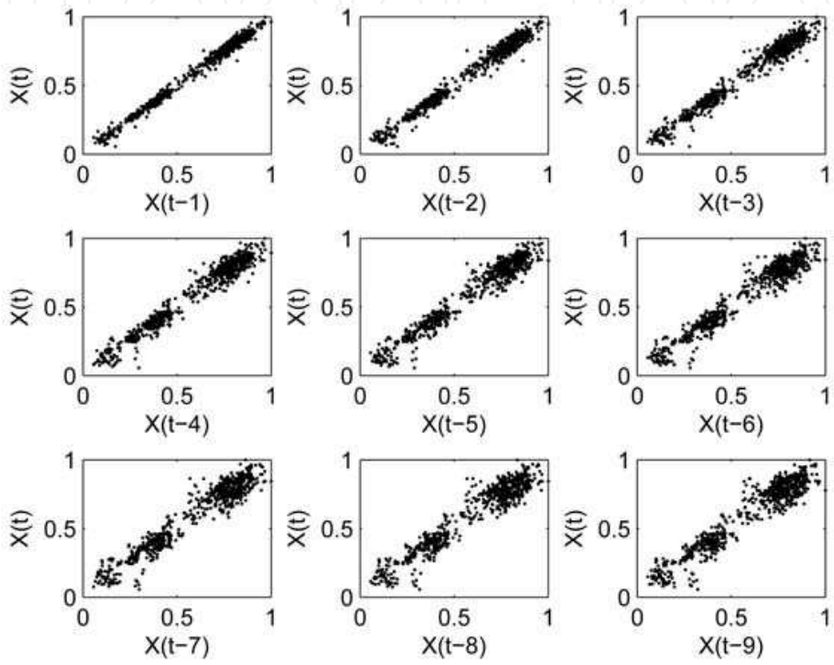


Fig. 8. DJIA Index series lagplot.

According to Figure 8, it is seen that for all the time lags of DJIA Index series there is a clear linear relationship among the lags. However, with the increase in the time lag degree, the appearance of the structure towards the graph center indicates a nonlinear relationship among the lags.

For the DJIA Index series prediction (with one step ahead of prediction horizon), the proposed method automatically chose the lag 2 as the relevant time lag (n = 1), defined the parameters ρ = 1.6374 and λ = 0.0038, and classified the best model as the "out-of-phase" model. Table 1 shows the results (with respect to the test set) for all the performance measures for the MLP, MRL, TAEF and MRLTAEF models.

	MLP		MRL		TAEF		MRLTAEF	
	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase
MSE	0.0827	0.0830	8.2763e-4	8.2148e-4	8.4183e-4	2.6841e-5	8.4886e-4	4.4636e-6
MAPE	9.3700	9.3788	9.6601	9.6578	10.1529	0.1993	9.6068	0.1833
NMSE	0.9878	0.9885	0.9889	0.9916	1.0006	0.0318	1.0005	0.0053
ARV	3.3877e-2	3.4204e-2	3.4227e-2	3.3981e-2	0.0346	0.0007	3.4685e-2	1.8272e-4
POCID	46.74	46.59	46.71	46.82	47.57	97.14	46.32	99.43
Fitness	4.0734	4.0567	3.9977	4.0071	3.9027	78.8584	3.9784	83.6398

Table 1. Results for the DJIA Index series.

Figure 9 shows the actual DJIA Index values (solid line) and the predicted values generated by the MRLTAEF out-of-phase model (dashed line) for the last 100 points of the test set.

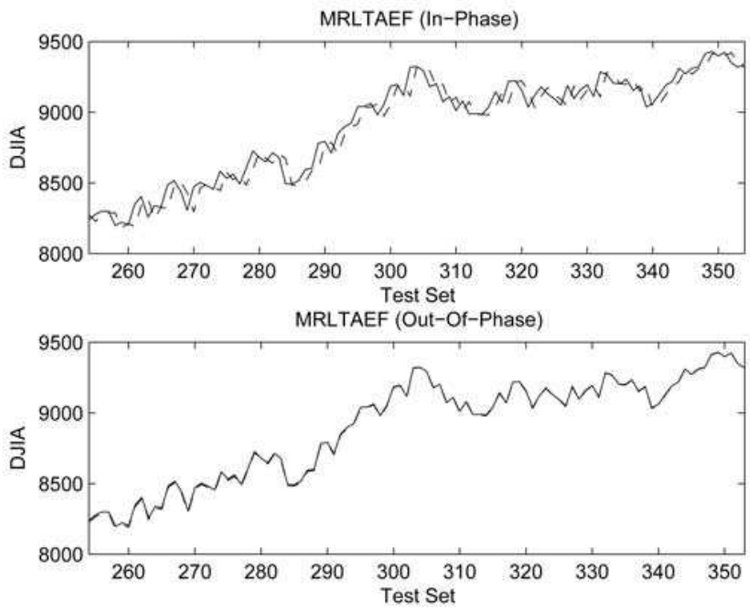


Fig. 9. Prediction results for the DJIA Index series (test set): actual values (solid line) and predicted values (dashed line).

Another relevant aspect to notice is that the MRLTAEF model chose the parameter λ = 0.0038, which indicates that it used 99.62% of the linear component of the MRL filter and 0.38% of the nonlinear component of the MRL filter, supporting the assumption (through lagplot analysis) that the DJIA Index series has a strong linear component mixed with a nonlinear component.

6.2 National association of securities dealers automated quotation (NASDAQ) index series

The National Association of Securities Dealers Automated Quotation (NASDAQ) Index series corresponds to daily observations from February 2nd 1971 to June 18th 2004, constituting a database of 8428 points. Figure 10 shows the NASDAQ Index lagplot.

According to Figure 10, it is seen that the time lags of NASDAQ Index series present a clear linear relationship among them, which, in theory, can contribute to a better forecasting result.

For the NASDAQ Index series prediction (with one step ahead of prediction horizon), the proposed method automatically chose the lag 2 as the relevant time lag (n = 1), defined the parameters ρ = 1.5581 and λ = 0.0005, and classified the model as "out-of-phase" matching. Table 2 shows the results (of the test set) for all performance measures for MLP, MRL, TAEF and MRLTAEF models.

Figure 11 shows the actual NASDAQ Index values (solid line) and the predicted values generated by the MRLTAEF out-of-phase model (dashed line) for the last 100 points of the test set.

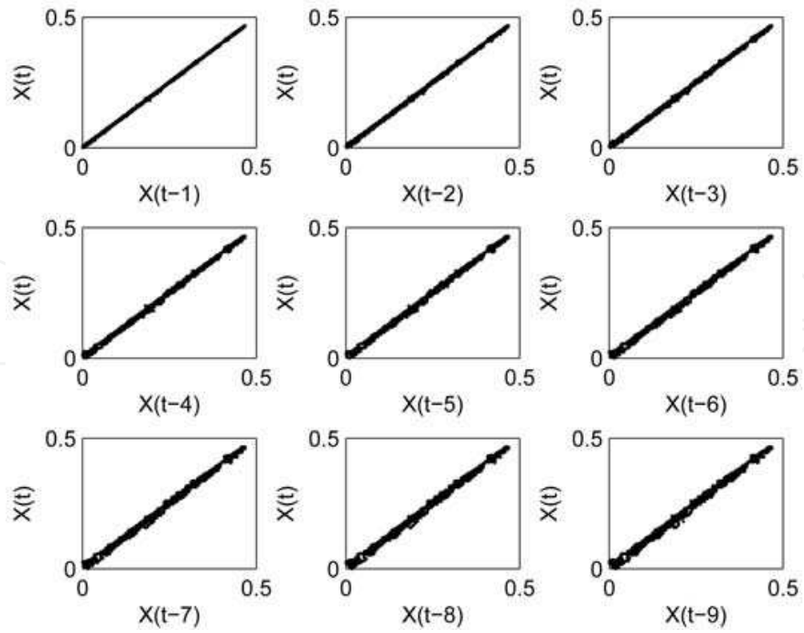


Fig. 10. NASDAQ Index series lagplot.

	MLP		MRL		TAEF		MRLTAEF	
	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase
MSE	0.0022	0.0023	2.0730e-5	2.0728e-5	2.1449e-5	3.2374e-6	1.8751e-5	1.3003e-10
MAPE	2.6988e-3	2.7001e-3	0.4100	0.4097	0.2012	0.0774	0.3979	1.3350e-3
NMSE	1.1728	1.1759	1.1057	1.1043	1.1441	0.1726	1.0002	6.9324e-6
ARV	3.4977e-3	3.5011e-3	3.3038e-3	3.2912e-3	0.0034	5.1500e-4	2.9884e-3	2.0713e-8
POCID	53.06	53.94	52.89	53.21	52.70	89.63	52.70	99.95
Fitness	24.2123	24.6932	20.9962	21.1376	22.4377	78.8584	21.9482	99.8160

Table 2. Results for the NASDAQ Index series.

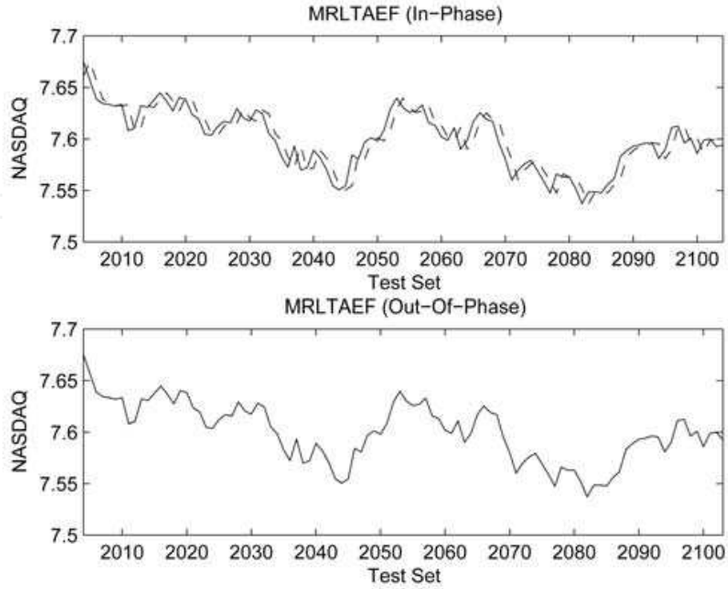


Fig. 11. Prediction results for the NASDAQ Index series (test set): actual values (solid line) and predicted values (dashed line).

It is worth mention that, as the MRLTAEF model chose λ= 0.0005, it used 99.95% of the linear component of the MRL filter and 0.05% of the nonlinear component of the MRL filter. This result can indicate that there is a nonlinear relationship among the time lags, a fact which could not be detected by the lagplot analysis.

6.3 Standard & Poor 500 (S&P500) index series

The Standard & Poor 500 (S&P500) Index is a pondered index of market values of the most negotiated stocks in the New York Stock Exchange (NYSE), American Stock Exchange (AMEX) and Nasdaq National Market System. The S&P500 series used corresponds to the monthly records from January 1970 to August 2003, constituting a database of 369 points. Figure 12 shows the S&P500 Index lagplot.

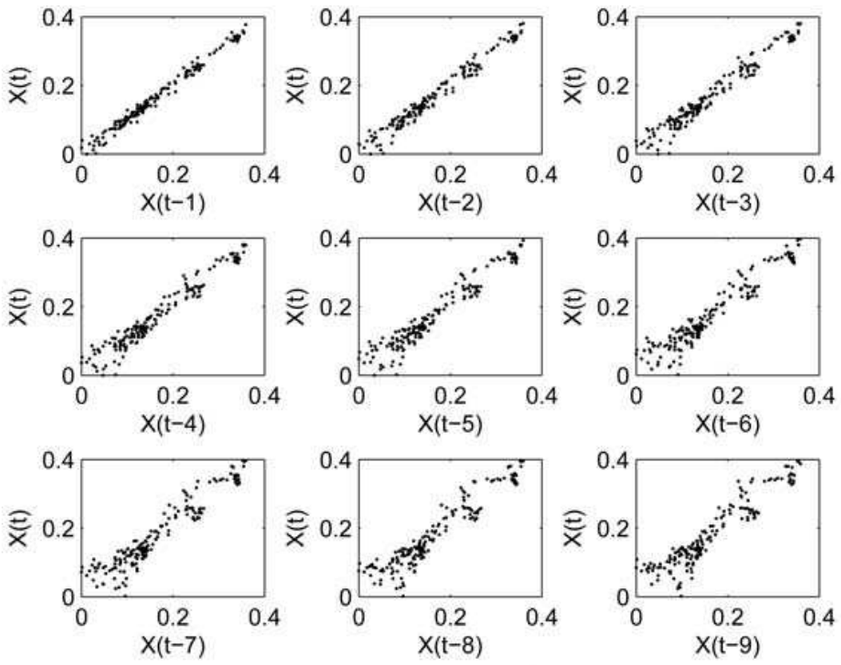


Fig. 12. S&P500 Index series lagplot.

According to Figure 12, it is also seen that for all the time lags of S&P500 Index series there is a clear linear relationship among the lags. However, with the increase in the time lag degree, the appearance of the structure towards the upper corner on the right hand side of the graph indicates a nonlinear relationship among the lags.

For the S&P500 Index series prediction (with one step ahead of prediction horizon), the proposed method automatically chose the lags 2, 3 and 10 as the relevant time lags (n = 3), defined the parameters ρ = 1.2508 and λ = 0.0091, and classified the best model as "out-of-phase" matching. Table 3 shows the results (for the test set) for all the performance measures for the MLP, MRL, TAEF and MRLTAEF models.

Figure 13 shows the actual S&P500 Index values (solid line) and the predicted values generated by the MRLTAEF out-of-phase model (dashed line) for the 90 points of the test set.

	MLP		MRL		TAEF		MRLTAEF	
	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase
MSE	0.0095	0.0096	9.9894e-5	1.0982e-4	7.4290e-4	8.0263e-4	1.1032e-4	1.3488e-6
MAPE	1.0100	1.0103	0.9400	1.0214	1.0431	1.0228	1.0222	0.0935
NMSE	0.9166	0.9179	0.9590	1.0397	7.2412	7.0883	1.0522	0.0120
ARV	7.2728e-3	7.2875e-3	7.7320e-3	8.4926e-3	0.0100	0.0012	8.6381e-3	1.0690e-4
POCID	51.11	50.98	67.77	52.18	50.54	100.00	50.56	98.86
Fitness	17.3644	17.3101	23.3140	16.9983	5.4373	10.9732	16.3988	89.4168

Table 3. Results for the S&P500 Index series.

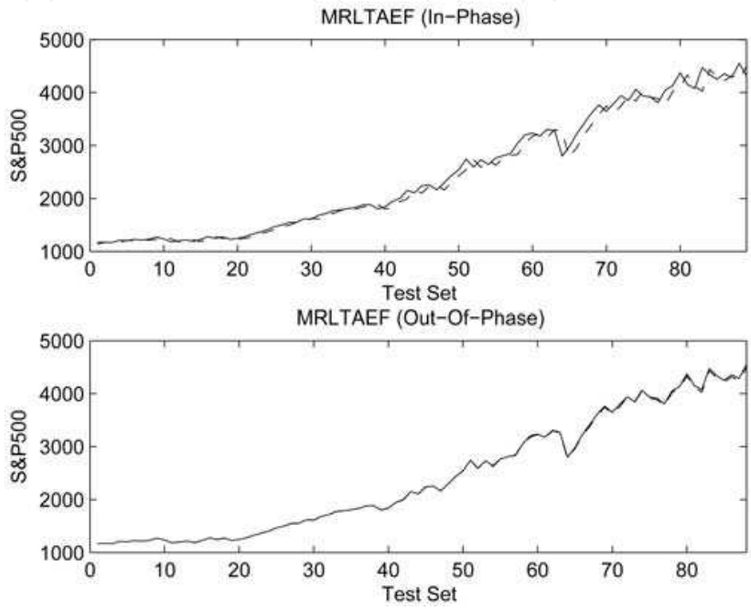


Fig. 13. Prediction results for the S&P500 Index series (test set): actual values (solid line) and predicted values (dashed line).

The proposed MRLTAEF chose λ = 0.0091, implying that it used 99.01% of the linear component of the MRL filter and 0.91% of the nonlinear component of the MRL filter, confirming the assumption (through lagplot analysis) that the S&P500 Index series has a strong linear component mixed with a nonlinear component.

6.4 Petrobras stock prices series

The Petrobras Stock Prices series corresponds to the daily records of Brazilian Petroleum Company from January 1st 1995 to July 3rd 2003, constituting a database of 2,060 points. Figure 14 shows the Petrobras Stock Prices lagplot.

According to Figure 14, it is seen that for all the time lags of the Petrobras Stock Prices series there is a clear linear relationship among the lags. However, with the increase in the time lag degree, the appearance of the structure towards the graph center indicates a nonlinear relationship among the lags.

For the Petrobras Stock Prices series prediction (with one step ahead of prediction horizon), the proposed method chose the lag 3 as the relevant time lag (n = 1), defined the parameters ρ = 1.9010 and λ = 0.0070, and classified the best model as “out-of-phase" matching. Table 4 shows the results (for the test set) of all the performance measures for the MLP, MRL, TAEF and MRLTAEF models.

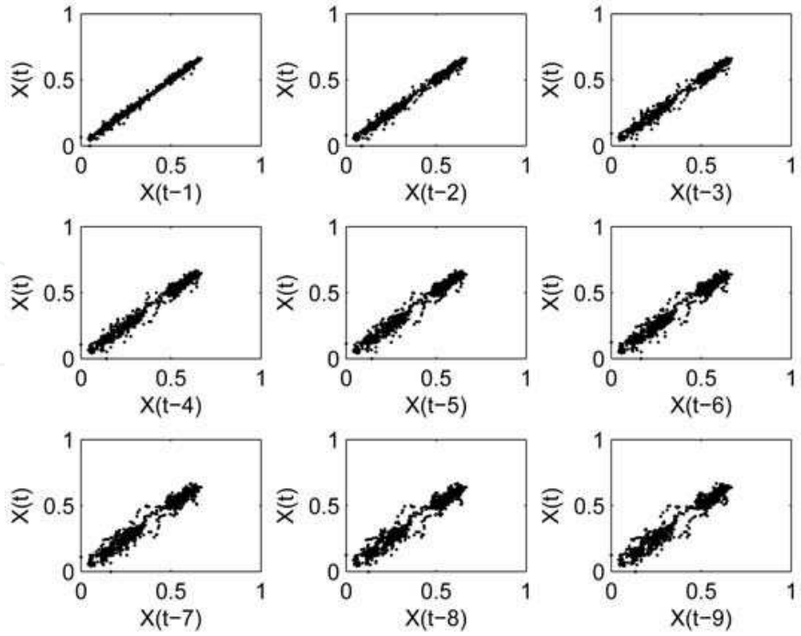


Fig. 14. Petrobras Stock Prices series lagplot.

	MLP		MRL		TAEF		MRLTAEF	
	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase
MSE	0.0095	0.0095	8.8100e-5	8.8101e-5	7.5951e-5	1.9049e-5	6.6044e-5	2.6435e-6
MAPE	0.5100	0.5101	0.8002	0.8000	0.5480	0.2850	0.6609	0.1722
NMSE	1.5124	1.5130	1.3992	1.3998	1.2077	0.3023	1.0521	0.0419
ARV	0.0890	0.0895	0.0826	0.0829	0.0050	0.0019	0.0618	2.4770e-3
POCID	51.63	51.64	52.02	52.08	52.79	97.68	51.53	99.03
Fitness	16.5433	16.5401	15.8496	15.8645	19.1214	61.4641	18.5702	81.4003

Table 4. Results for the Petrobras Stock Prices series.

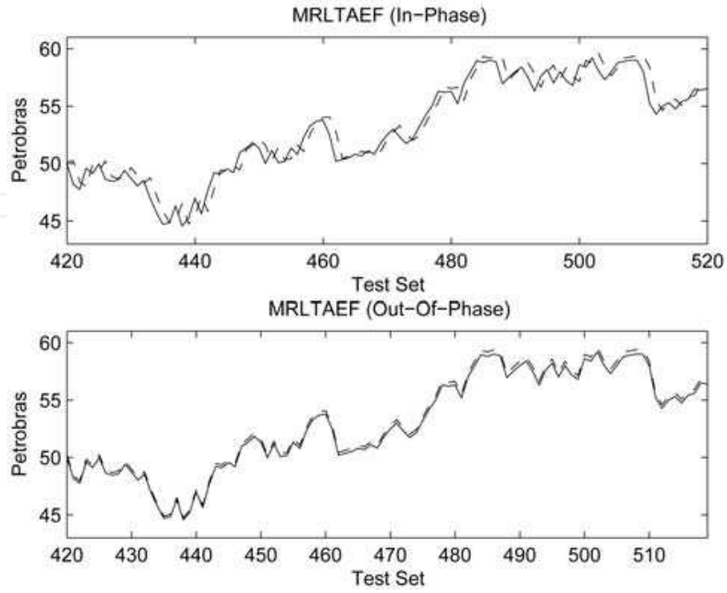


Fig. 15. Prediction results for the Petrobras Stock Prices series (test set): actual values (solid line) and predicted values (dashed line).

Figure 15 shows the actual Petrobras Stock Prices (solid line) and the predicted values generated by the MRLTAEF model out-of-phase (dashed line) for the 100 points of the test set.

For this series the proposed MRLTAEF chose λ = 0.0070, which means that it used 99.30% of the linear component of the MRL filter and 0.7% of the nonlinear component of the MRL filter, confirming the assumption (through lagplot analysis) that the Petrobras Stock Prices series has a strong linear component mixed with a nonlinear component.

6.5 General motors corporation stock prices series

The General Motors Corporation Stock Prices series corresponds to the daily records of General Motors Corporation from June 23th 2000 to June 22th 2007, constituting a database of 1,758 points. Figure 16 shows the General Motors Corporation Stock Prices lagplot.

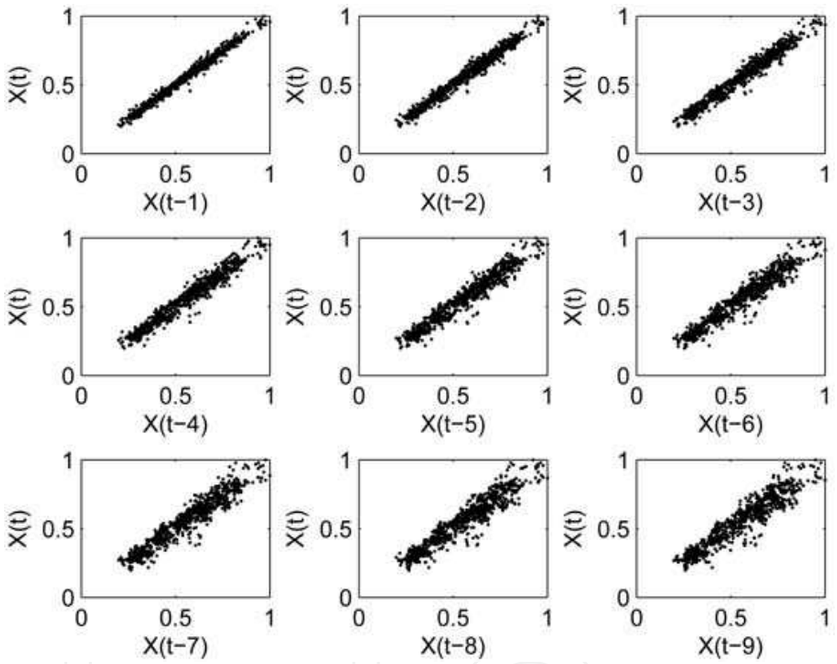


Fig. 16. General Motors Corporation Stock Prices series lagplot.

According to Figure 16, it is virified that for all the time lags of the General Motors Corporation Stock Prices series there is a clear linear relationship among the lags. However, with the increase in the time lag degree, the appearance of the structure towards the upper corner on the right hand side of the graph indicates a nonlinear relationship among the lags.

For the General Motors Corporation Stock Prices series prediction (with one step ahead of prediction horizon), the proposed method chose the lags 2, 4, 5 and 8 as the relevant time lags (n = 4), defined the parameters ρ = 0.0617 and λ = 0.0011, and classified the best model as "out-of-phase" matching. Table 5 shows the results (for the test set) of all the performance measures for the MLP, MRL, TAEF and MRLTAEF models.

Figure 17 shows the actual General Motors Corporation Stock Prices (solid line) and the predicted values generated by the MRLTAEF model out-of-phase (dashed line) for the 100 points of the test set.

	MLP		MRL		TAEF		MRLTAEF	
	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase
MSE	1.8224e-4	1.7033e-4	1.6854e-4	1.6936e-4	1.6987e-4	4.0497e-6	1.6832e-4	2.2979e-6
MAPE	0.1628	0.1448	0.1357	0.1345	0.1511	3.8906e-2	0.1358	8.0446e-3
NMSE	1.0993	1.0273	1.0174	1.0223	1.0250	0.0243	1.0162	0.0137
ARV	2.6998e-2	2.5233e-2	2.4999e-2	2.5121e-2	2.5164e-2	6.0196e-4	2.4968e-2	3.4201e-4
POCID	52.63	52.40	52.50	52.48	53.54	94.26	52.51	99.54
Fitness	22.9897	23.8452	24.1017	24.0503	23.3205	88.6058	24.1188	97.3887

Table 5. Results for the General Motors Corporation Stock Prices series.

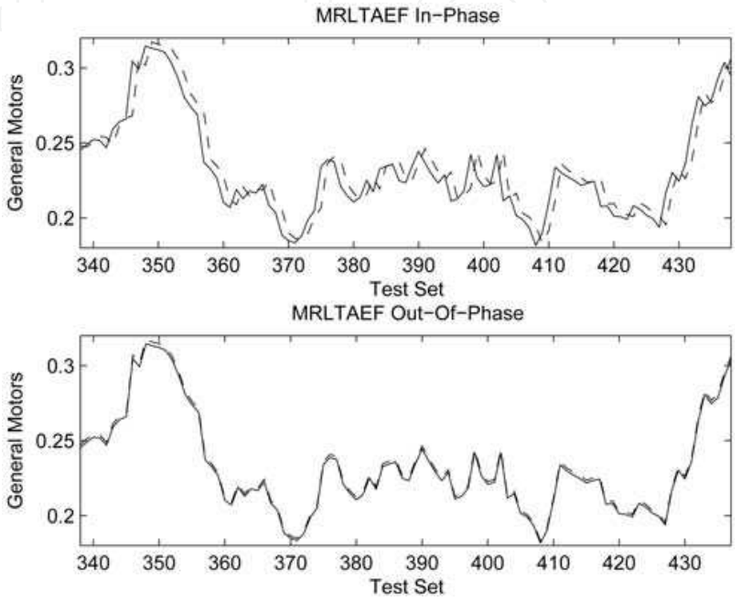


Fig. 17. Prediction results for the General Motors Corporation Stock Prices series (test set): actual values (solid line) and predicted values (dashed line).

For this series the proposed MRLTAEF chose λ = 0.0011, which means that it used 99.89% of the linear component of the MRL filter and 0.11% of the nonlinear component of the MRL filter, confirming the assumption (through lagplot analysis) that the General Motors Corporation Stock Prices series has a strong linear component mixed with a nonlinear component.

6.6 Google Inc Stock Prices series

The Google Inc Stock Prices series corresponds to the daily records of Google Inc from August 19th 2004 to June 21th 2007, constituting a database of 715 points. Figure 18 shows the Google Inc Stock Prices lagplot.

According to Figure 14, it is seen that for all the time lags of the Google Inc Stock Prices series there is a clear linear relationship among the lags. However, with the increase in the time lag degree, the appearance of the structure towards the graph center indicates a nonlinear relationship among the lags.

For the Google Inc Stock Prices series prediction (with one step ahead of prediction horizon), the proposed method chose the lags 2, 3 and 10 as the relevant time lags (n = 3), defined the parameters ρ = –1.5108 and λ = 0.0192, and classified the best model as "out-of-phase" matching. Table 6 shows the results (for the test set) of all the performance measures for the MLP, MRL, TAEF and MRLTAEF models.

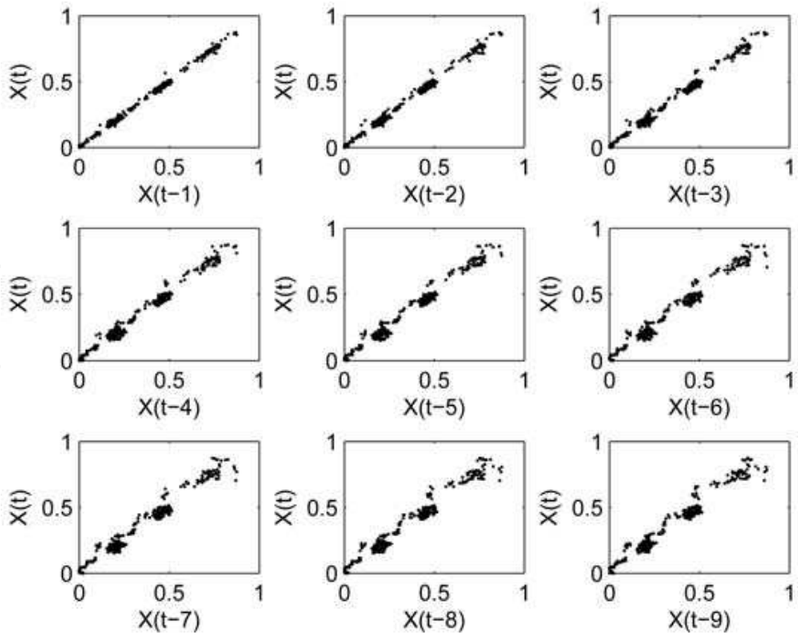


Fig. 18. Google Inc Stock Prices series lagplot.

	MLP		MRL		TAEF		MRLTAEF	
	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase	In-Phase	Out-Of-Phase
MSE	2.8899e-4	3.1912e-4	2.9851e-4	3.1680e-4	3.1500e-4	4.5046e-5	2.8399e-4	5.4079e-6
MAPE	1.4049e-2	1.4549e-2	1.3232e-2	1.4211e-2	1.4257e-2	6.5056e-3	1.3272e-2	2.5518e-3
NMSE	1.0249	1.1271	1.0638	1.1345	1.1117	0.1607	1.0143	0.0195
ARV	0.1210	0.1336	0.1250	0.1259	0.1319	1.9405e-2	0.1189	2.3296e-3
POCID	40.90	43.18	43.75	43.26	42.04	94.85	43.18	99.42
Fitness	18.9330	18.9754	19.8653	19.0159	18.6168	79.9305	20.2266	97.0531

Table 6. Results for the Google Inc Stock Prices series.

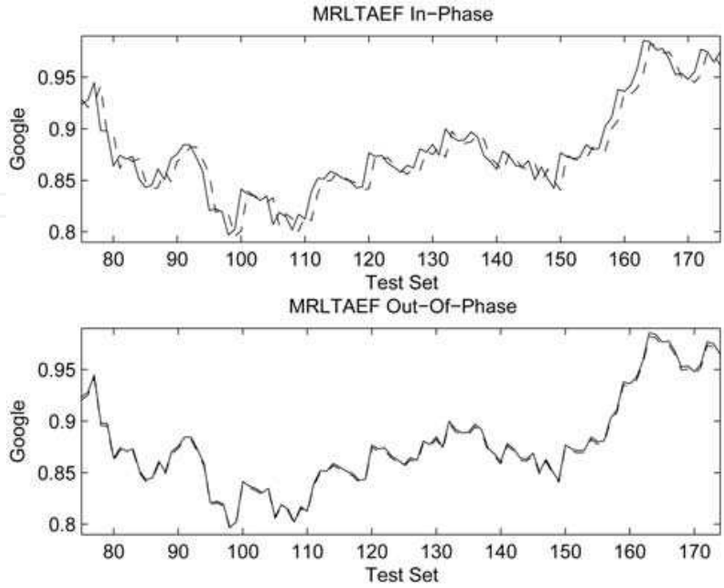


Fig. 19. Prediction results for the Google Inc Stock Prices series (test set): actual values (solid line) and predicted values (dashed line).

Figure 19 shows the actual Google Inc Stock Prices (solid line) and the predicted values generated by the MRLTAEF model out-of-phase (dashed line) for the 100 points of the test set.

For this series the proposed MRLTAEF chose $\lambda = 0.0192$, which means that it used 98.08% of the linear component of the MRL filter and 1.92% of the nonlinear component of the MRL filter, confirming the assumption (through lagplot analysis) that the Google Inc Stock Prices series has a strong linear component mixed with a nonlinear component.

In general, all generated prediction models using the phase fix procedure to adjust time phase distortions shown forecasting performance much better than the MLP model and MRL model, and slightly better than the TAEF model. The proposed method was able to adjust the time phase distortions of all analyzed time series (the prediction generated by the out-of-phase matching hypothesis is not delayed with respect to the original data), while the MLP model and MRL model were not able to adjust the time phase. This corroborates with the assumption made by Ferreira [15], where he discusses that the success of the phase fix procedure is strongly dependent on an accurate adjustment of the prediction model parameters and on the model itself used for prediction.

7. Conclusions

This work presented a new approach, referred to as Morpological-Rank-Linear Time-lag Added Forecasting (MRLTAEF) model, to overcome the RW dilemma for financial time series forecasting, which performs an evolutionary search for the minimum dimension to determining the characteristic phase space that generates the financial time series phenomenon. It is inspired on Takens Theorem and consists of an intelligent hybrid model composed of a Morpological-Rank-Linear (MRL) filter combined with a Modified Genetic Algorithm (MGA), which searches for the minimum number of time lags for a correct time series representation and estimates the initial (sub-optimal) parameters of the MRL filter (mixing parameter (λ), rank (r), linear Finite Impulse Response (FIR) filter ($\underline{b}$) and the Morphological-Rank (MR) filter ($\underline{a}$) coefficients). Each individual of the MGA population is trained by the averaged Least Mean Squares (LMS) algorithm to further improve the MRL filter parameters supplied by the MGA. After adjusting the model, it performs a behavioral statistical test and a phase fix procedure to adjust time phase distortions observed in financial time series.

Five different metrics were used to measure the performance of the proposed MRLTAEF method for financial time series forecasting. A fitness function was designed with these five well-known statistic error measures in order to improve the description of the time series phenomenon as much as possible. The five different evaluation measures used to compose this fitness function can have different contributions to the final prediction, where a more sophisticated analysis must be done to determine the optimal combination of such metrics.

An experimental validation of the method was carried out on four real world financial time series, showing the robustness of the MRLTAEF method through a comparison, according to five performance measures, of previous results found in the literature (MLP, MRL and TAEF models). This experimental investigation indicates a better, more consistent global performance of the proposed MRLTAEF method.

In general, all generated predictive models with the MRLTAEF method using the phase fix procedure (to adjust time phase distortions) showed forecasting performance much better

than the MLP model and MRL model, and slightly better than the TAEF model. The MRLTAEF method was able to adjust the time phase distortions of all analyzed time series, while the MLP model and MRL model were not able to adjust the time phase. This fact shows that the success of the phase fix procedure is strongly dependent on the accurate adjustment of parameters of the predictive model and on the model itself used for forecasting. It was also observed that the MRLTAEF model reached a much better performance when compared with a random walk like model, overcoming the random walk dilemma for the analyzed financial times series.

The models generated by the MRLTAEF method are not random walk models. This affirmation is shown with the phase fix procedure. If the MRL filter models were random walk models, the phase fix procedure would generate the same result of the original prediction, since in the random walk model the t+1 value is always the t value.

It is worth mentioning that the first time lag is never selected to predict any time series used in this work. However, a random walk structure is necessary for the phase fix procedure to work, since the key of this procedure is the two step prediction (described by the phase fix procedure) in order to adjust the one step time phase.

Also, one of the main advantages of the MRLTAEF model (apart from its predictive performance when compared to all analyzed models) is that not only they have linear and nonlinear components, but they are quite attractive due to their simpler computational complexity when compared to other approaches such as [33, 34], other MLP-GA models [15] and other statistical models [2-5].

Furthermore, another assumption made by Ferreira [15] was confirmed through the analyzes of the MRL filter mixing parameter (λ). It was argued that through lagplot analysis it is possible to notice in financial time series indicative structures of some nonlinear relationship among the time lags even though they are super-imposed by a dominant linear component. In all the experiments, the MRLTAEF model set a strong linear component mixed with a weak nonlinear component (it uses ~99% of the linear component of MRL filter and ~1% of the nonlinear component of the MRL filter). Since the MRLTAEF method defines a MRL filter like model, which has the ability to select the percentage of use of the linear and nonlinear components, it is believed that it improves the prediction performance through a balanced estimation of the linear and nolinear relationships.

Future works will consider the development of further studies in order to formalize properties of the proposed model using the phase fix procedure. Also, other financial time series with components such as trends, seasonalities, impulses, steps and other nonlinearities can be used for the efficiency confirmation of the proposed method, as well as, further studies, in terms of risk and financial return, can be developed in order to determine the additional economical benefits, for an investor, with the use of the proposed method.

8. Acknowledgements

The authors are thankful to Mr. Chè Donavon David Davis for English support.

9. References

[1] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice Hall, New Jersey, third edition, 1994.

[2] T. Subba Rao and M. M. Gabr. Introduction to Bispectral Analysis and Bilinear Time Series Models, volume 24 of Lecture Notes in Statistics. Springer, Berlin, 1984.
[3] T. Ozaki. Nonlinear Time Series Models and Dynamical Systems, volume 5 of Hand Book of Statistics. North- Holland, Amsterdam, 1985.
[4] M. B. Priestley. Non-Linear and Non-stationary Time Series Analysis. Academic Press, 1988.
[5] D. E. Rumelhart and J. L. McCleland. Parallel Distributed Processing, Explorations in the Microstructure of Cognition, volume 1 & 2. MIT Press, 1987.
[6] M. P. Clements, P. H. Franses, and N. R. Swanson. Forecasting economic and financial time-series with non-linear models. *International Journal of Forecasting*, 20(2):169-183, 2004.
[7] T. C. Myhre. Financial forecasting at martin marietta energy systems, inc. *The Journal of Business Forecasting Methods & Systems*, 11(1):28-30, April 1992.
[8] M. Crottel, B. Girard, Y. Girard, M. Mangeas, and C. Muller. Neural modeling for time series: a statistical stepwise method for weight elimination. *IEEE Transaction on Neural Networks*, 6(6):1355-1364, 1995.
[9] G. Zhang, B. E. Patuwo, and M. Y. Hu. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14:35-62, 1998.
[10] A. Khotanzad, H. Elragal, and T.-L. Lu. Combination of artificial neural-network forecasters for prediction of natural gas consumption. *Neural Networks, IEEE Transactions on*, 11(2):464-473, Mar 2000.
[11] Renate Sitte and Joaquin Sitte. Neural networks approach to the random walk dilemma of financial time series. *Applied Intelligence*, 16(3):163-171, May 2002.
[12] Marko Hocevar, Brane Širok, and Bogdan Blagojevic. Prediction of cavitation vortex dynamics in the draft tube of a francis turbine using radial basis neural networks. *Neural Computing & Applications*, 14(3):229-234, September 2005.
[13] Arie Preminger and Raphael Franck. Forecasting exchange rates: A robust regression approach. *International Journal of Forecasting*, 23(1):71-84, January-March 2007.
[14] D.M. Zhang, G.P.; Kline. Quarterly time-series forecasting with neural networks. *Neural Networks, IEEE Transactions on*, 18(6):1800-1814, Nov. 2007.
[15] Tiago A. E. Ferreira, Germano C. Vasconcelos, and Paulo J. L. Adeodato. A new intelligent system methodology for time series forecasting with artificial neural networks. In *Neural Processing Letters*, volume 28, pages 113-129, 2008.
[16] F. H. F. Leung, H. K. Lam, S. H. Ling, and P. K. S. Tam. Tuning of the structure and parameters of the neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks*, 14(1):79-88, January 2003.
[17] T. A. E. Ferreira, G. C. Vasconcelos, and P. J. L. Adeodato. A hybrid intelligence system approach for improving the prediction of real world time series. In *IEEE Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 736-743, Portland, Oregon, 2004. IEEE.
[18] Mariano Matilla-García and Carlos Argüello. A hybrid approach based on neural networks and genetic algorithms to the study of profitability in the spanish stock market. *Applied Economics Letters*, 12(5):303-308, April 2005.
[19] T. A. E. Ferreira, G. C. Vasconcelos, and P. J. L. Adeodato. A new evolutionary method for time series forescasting. In *ACM Proceedings of Genetic Evolutionary Computation Conference - GECCO 2005*, Washington D.C., USA, 2005. ACM.

[20] R. A. Araújo, Germano C. Vasconcelos, and T. A. E. Ferreira. Hybrid differential evolutionary system for financial time series forecasting. In *Proceedings of the IEEE Congress on Evolutionary Computation*, Singapore, 2007.

[21] P. Maragos. A representation theory for morphological image and signal processing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11:586-599, 1989.

[22] J. Serra. *Image Analysis And Mathematical Morphology*. Academic Press, London, 1982.

[23] E. J. Coyle and J. H. Lin. Stack filters and the mean absolute error criterion. *IEEE Trans. Acoust. Speech Signal Processing*, 36:1244-1254, 1988.

[24] S. S. Wilson. Morphological networks. Proceedings of The SPIE Visual Communication and Image Processing IV, 1199:483-493, 1989.

[25] R. P. Loce and E. R. Dougherty. Facilitation of optimal binary morphological filter design via structuring element libraries and design constraints. *Opt. Eng.*, 31:1008-1025, May 1992.

[26] P. Yang and P. Maragos. Character recognition using min-max classifiers designed using an LMS algorithm. *Visual Communications and Image Processing*, 92(1818):674-685, nov. 1992.

[27] J. L. Davidson and F. Hummer. Morphology neural networks: An introduction with applications. *Circuits, System and Signal Process*, 12(2):179-210, 1993.

[28] C. B. Herwing and R. J. Shalkoff. Morphological image processing using artificial neural networks. In C. T. Leondes, editor, *Control and Dynamic Systems, Vol. 67*, pages 319-379. Academic Press, 1994.

[29] P. Salembier. Structuring element adaptation for morphological filters. *Journal for Visual Communication and Image Representation*, 3(2):115-136, June 1992.

[30] P. Salembier. Adaptive rank order based filters. *Signal Process.*, 27(1):1-25, 1992.

[31] L. F. C. Pessoa and P. Maragos. MRL-filters: A general class of nonlinear systems and their optimal design for image processing. *IEEE Transactions on Image Processing*, 7:966-978, 1998.

[32] L. F. C. Pessoa and P. Maragos. Neural networks with hybrid morphological/rank/linear nodes: A unifying framework with applications to handwritten character recognition. *Pattern Recognition*, 33:945-960, 2000.

[33] R. A. Araújo, F. Madeiro, R. P. Sousa, L. F. C. Pessoa, and T. A. E. Ferreira. An evolutionary morphological approach for financial time series forecasting. In *Proceedings of the IEEE Congress on Evolutionary Computation*, Vancouver, Canada, 2006.

[34] R. A. Araújo, R. P. Sousa, and T. A. E. Ferreira. An intelligent hybrid approach for designing increasing translation invariant morphological operators for time series forecasting. In *ISNN (2)*, volume 4492 PART II of *Lecture Notes in Computer Science*, pages 602-611. Springer-Verlag, 2007.

[35] G. Matheron. *Random Sets and Integral Geometry*. Wiley, New York, 1975.

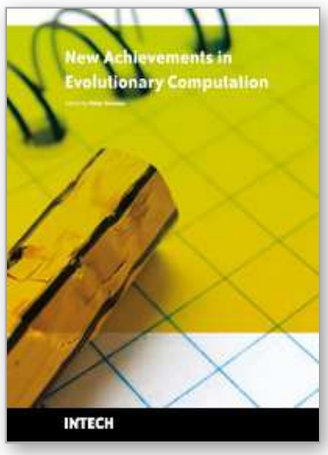
[36] G. J. F. Banon and J. Barrera. Minimal representation for translation invariant set mappings by mathematical morphology. *SIAM J. Appl. Math.*, 51(6):1782-1798, 1991.

[37] F. Takens. Detecting strange attractor in turbulence. In A. Dold and B. Eckmann, editors, *Dynamical Systems and Turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366-381, New York, 1980. Springer-Verlag.

[38] R. Savit and M. Green. Time series and dependent variables. *Physica D*, 50:95-116, 1991.

[39] H. Pi and C. Peterson. Finding the embedding dimension and variable dependences in time series. *Neural Computation*, 6:509-520, 1994.
[40] N. Tanaka, H. Okamoto, and M. Naito. Estimating the active dimension of the dynamics in a time series based on a information criterion. *Physica D*, 158:19-31, 2001.
[41] T. C. Mills. *The Econometric Modelling of Financial Time Series*. Cambridge University Press, Cambridge, 2003.
[42] Haolger Kantz and Thomas Schreiber. *Nonlinear Time Series analysis*. Cambridge University Press, New York, NY, USA, second edition, 2003.
[43] Pedro A. Morettin e Clélia M. Toloi. *Modelos para Previsão em Séries Temporias*, volume 1 e 2. IMPA – Instituto de Matemática Pura e Aplicada, São Paulo, 1981.
[44] Pedro A. Morettin e Clélia M. Toloi. *Séries Temporais*. Coleção Métodos Quantitativos. Atual Editora, São Paulo, segunda edition, 1987.
[45] Michalem P. Clements, Philip Hans Franses, Jeremy Smith, and Dick Van Dijk. On setar non-linearity and forecasting. *Journal of Forecasting*, 22(5):359-375, Aug 2003.
[46] J. G. De Gooijer and K. Kumar. Some recent developments in non-linear time series modelling, testing, and forecasting. *International Journal of Forecasting*, 8:135-156, 1992.
[47] R. F. Engle. Autoregressive conditional heteroskedasticity with estimates of the variance of uk onflation. *Econometrica*, 50:987-1008, 1982.
[48] M. P. Clements, P. H. Franses, and N. R. Swanson. Forecasting economic and finalcial time-seires with non-linear models. *International Journal of Forecasting*, 20:169-183, 2004.
[49] Simon Hakykin. *Redes Neurais - Princípos e Prática*. Bookman, Porto Alegre - Brasil, 2a edition, 2002.
[50] T. Kohonen. Self-organized formation of topologically conect feature maps. *Biological Cybernetics*, 43:5-69, 1982.
[51] Teuvo Kohonen. *Self-Organization and Associative Memory*. Springer-Verlog, Berlin, 1984.
[52] J. J. Hopfield. Neural networks and physical systems with emergent coletive computational abilities. In *Proceedings of the National Academy of the Sciences of tha U.S.A.*, volume 79, pages 2554-2558, 1982.
[53] Tom M. Mitchell. *Machine Learning*. WCB McGraw-Hill, Boston, 1997.
[54] Lutz Prechelt. Proben1: A set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, 1994.
[55] John H. Holland. *Adaptation in Natual and Artificial Systems*. University of Michigan Press, Michigan, 1975.
[56] David E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.
[57] Charles Darwin. *The origin of species*. United King, 1859.
[58] John H. Holland. Genetic algorithms. *Scientific Amerian*, pages 66-72, july 1992.
[59] M. Mitchell. *A Introduction to Genetic Algorithms*. MIT Press, Canbridge, 1999.
[60] Mitsuo Gen and Runwei Cheng. *Genetic Algorithms and Engineering Design*. John Wiley and sons, New York, 1997.
[61] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing*, volume 1: Foundations. The MIT Press, 1986.
[62] Q. Liang and J. M. Mendel. Interval type-2 fuzzy logic systems: Theory and design. *IEEE Transaction on Fuzzy Systems*, 8(5):535-550, October 2000.

[63] D. Dasgupta. *Artificial Immune System and the Applications*. Springr-Verlog, Berlin, 1999.
[64] Donald Arthur Waterman. *A guide to expert systems*. Addison-Wesley, 1986.
[65] Maria Carolina Monard and José Augusto Baranauskas. *Indução de Regras e Árvores de Decisão*, chapter 5 – Sistemas Inteligentes – Fundamentos e Aplicações, pages 115-139. Malone, São Paulo, 2003.
[66] Suran Goonatilake and Sukhdev Khebbal. *Intelligence Hybrid Systems*. John Wiley & Son, 1995.
[67] H. Minkowski. *Gesammelte Abhandlungen*. Teubner Verlag, Leipzig-Berlin, 1911.
[68] R. P. Sousa. Design of translation invariant operators via neural network training. PhD thesis, UFPB, Campina Grande, Brazil, 2000.
[69] M. P. Clements and D. F. Hendry. On the limitations of comparing mean square forecast errors. *Journal of Forecasting*, 12(8):617-637, Dec. 1993.
[70] T. H. Hann and E. Steurer. Much ado about nothing? exchange rate forecasting: Neural networks vs. linear models using monthly and weekly data. *Neurocomputing*, 10:323-339, 1996.
[71]. M. Hagan and M. Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989-993, November 1994.
[72] Donald B. Percival and Andrew T.Walden. Spectral Analysis for Physical Applications-Multitaper and Conventional Univariate Techniques. Cambridge University Press, New York, 1998.



New Achievements in Evolutionary Computation

Edited by Peter Korosec

ISBN 978-953-307-053-7
Hard cover, 318 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

Evolutionary computation has been widely used in computer science for decades. Even though it started as far back as the 1960s with simulated evolution, the subject is still evolving. During this time, new metaheuristic optimization approaches, like evolutionary algorithms, genetic algorithms, swarm intelligence, etc., were being developed and new fields of usage in artificial intelligence, machine learning, combinatorial and numerical optimization, etc., were being explored. However, even with so much work done, novel research into new techniques and new areas of usage is far from over. This book presents some new theoretical as well as practical aspects of evolutionary computation. This book will be of great value to undergraduates, graduate students, researchers in computer science, and anyone else with an interest in learning about the latest developments in evolutionary computation.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Ricardo de A. Araújo, Gláucio G. de M. Melo, Adriano L. I. de Oliveira and Sergio C. B. Soares (2010). Morphological-Rank-Linear Models for Financial Time Series Forecasting, New Achievements in Evolutionary Computation, Peter Korosec (Ed.), ISBN: 978-953-307-053-7, InTech, Available from: http://www.intechopen.com/books/new-achievements-in-evolutionary-computation/morphological-rank-linear-models-for-financial-time-series-forecasting

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen