Computational Economics **20:** 191–210, 2002.

Exchange-Rates Forecasting: A Hybrid Algorithm Based on Genetically Optimized Adaptive Neural Networks

ANDREAS S. ANDREOU [1], EFSTRATIOS F. GEORGOPOULOS [2] and SPIRIDON D. LIKOTHANASSIS [2,3]

[1] *University of Cyprus, Department of Computer Science, 75 Kallipoleos Str., P.O. Box 20537, CY1678, Nicosia, Cyprus; E-mail: aandreou@ucy.ac.cy;* [2] *University of Patras, Department of Computer Engineering & Informatics, Patras 26500, Greece; E-mail: likothan@cti.gr;*
[3] *Computer Technology Institute, 3 Kolokotroni Str., 2622 1, Patras, Greece*

(Accepted 10 May 2001)

Abstract. The use of neural networks trained by a new hybrid algorithm is employed on forecasting the Greek Foreign Exchange-Rate Market. Four major currencies, namely the U.S. Dollar (USD), the Deutsche Mark (DEM), the French Franc (FF) and the British Pound (GBP), versus the Greek Drachma, were used as experimental data. The proposed algorithm combines genetic algorithms and a training method based on the localized Extended Kalman Filter (EKF), in order to evolve the structure and train Multi-Layered Perceptron (MLP) neural networks. The goal of this effort is to predict, as accurately as possible, exchange-rates future behavior. Simulation results show that the method gives highly successful results, while the diversification of the structure between the four currencies has no effect on the performance.

Key words: exchange-rates, neural networks, genetic algorithms, filtering, forecasting

1. Introduction

The areas of Economics and Finance with special reference to predictability issues related to stock and foreign exchange markets seem to attract increasing interest during the last few years. The foreign-exchange market, in particular, offers ample room for macro-policy considerations, since the exchange-rate is often selected as a major policy instrument for attaining important targets, like price stability or balance-of-payment equilibrium. But even in cases where the exchange-rate is allowed to fluctuate in the international markets within margins that differ considerably between currencies, the fact rernains that central banks retain the power to interfere with the market prices, manipulating the exchange-rate of their respective currencies whenever the authorities consider it necessary. Such manipulations can take place either directly, as in the cases in which the exchange rate has been designed to attain a specific target value in year-end terms, or indirectly via interest

rate policy. Whatever the economic impact of such strategies may be, the fact remains that such methods of interference tend to increase the noise level which characterizes the behavior of the time series under study, an issue which introduces a considerable degree of difficulty when it comes to forecasting. Thus, one has to turn to models that either filter away the noise, or 'machines' that can learn how the system behaves, even in the presence of noise.

The relevant literature on currency forecasting issues includes a wide range of methods. In Marsh and Power (1996) the ability of 22 forecasters to predict movements in three major currencies against the U.S. Dollar is reported, including the random walk estimator. Apart from traditional statistical methods, the emerging technique of artificial neural networks seems to gain ground during the last five years. Kuan and Liu (1995) investigate the forecasting ability of feedforward and recurrent neural networks on empirical foreign data. Verkooijen and Daniels (1994) analyze long-run nonlinear correlation between economic fundamentals and the exchange rate of U.S. Dollar versus the Deutsche Mark, based on predictions made using a neural network architecture with projection pursuit regression. In Verkooijen (1996), a comparison between OLS estimation and neural network estimation favored the use of the latter in currency forecasting. Refenes and Zaidi (1995) present a hybrid system based on neural networks for managing trading strategies in the foreign-exchange markets and report results for the USD/DEM that are superior to those of classical techniques like moving average and mean value.

Based upon the successful use of neural networks reported in the literature so far, the authors turned to this heuristic methodology as a mean to investigate the forecasting performance of the Greek foreign exchange-rate market. An earlier work (Andreou et al., 1997) analyzed the behavior of the exchange rate of the same four currencies used in this paper. The primary goal was to reveal whether the time series involved exhibit chaotic behavior or not, a prerequisite for applying the well-known Farmer's algorithm for prediction and the secondary to test the level of predictability using the MLP artificial neural networks trained by Error Back-Propagation. The neural networks' results of this first approach were not satisfactory. In a subsequent paper (Adamopoulos et al., 1997), taking in mind the results of Andreou et al. (1997), two different algorithms for RBF neural networks training, a Kalman filtering based training algorithm and a modified genetic algorithm, were developed and used. The main goal of this work was to compare the performance of these two algorithms for the task of exchange rate forecasting. The results obtained here were much better than the previous ones, but still not as good as the authors would like to be. Recently, in Andreou et al. (1998), an MLP trained by two different methods, the momentum Error Back-Propagation and a specially designed genetic algorithm was developed. The goal of this effort was to check the predictability of these algorithms using two differently preprocessed currency data sets: (i) the fixed rates re-scaled in $[-1, 1]$ and (ii) the first differences of the natural logarithms of the fixed rates, re-scaled in $[-1, 1]$. Both algorithms exhibited very

satisfactory predicting performance for the case of the fixed rate data, while, for the case of the natural logarithms the results obtained were rather poor.

The current paper presents a novel hybrid algorithm, which employs a combination of Genetic Algorithms (GA's) and a technique based on a localized Extended Kalman Filter (EKF) for training MLP neural networks. This modified genetic algorithm is trying, evolving a population of MLP neural networks, to find a near optimum network architecture (number of inputs, number of hidden units) that solves a specific problem, while, at the same time the Kalman training algorithm is used in the networks' learning phase. The novelty of this effort lies on, apart from the combination of evolution programs with the localized Kalman training algorithm (Shah et al., 1992), the capability of the proposed method to search, not only for the optimal number of hidden units, but also, for the number of inputs needed for the problem at hand. As far as we know, this is the first attempt in the relevant literature to evolve both the numbers of hidden and input neurons of a network. Another important aspect of this work is the use of three different fitness functions for the evaluation of the neural networks evolved by the genetic algorithm. This task was performed aiming at investigating how the use of some specific fitness function affects the algorithm's performance. The final goal of this work is to predict as accurately as possible the future behavior of the currencies mentioned earlier on a daily predicting horizon, examining also the course of forecasting under a multistep horizon. A similar, but simpler, method was used successfully in Adamopoulos et al. (1998), for system structure identification, using single layer neural networks.

The data used for the training and the evaluation of the networks consist of daily exchange-rate observations of four major currencies, namely the U.S. Dollar (USD), the Deutsche Mark (DEM), the French Franc (FF) and the British Pound (GBP), versus the Greek Drachma. The rates are those determined during the daily 'fixing' sessions in the Bank of Greece. These data cover an 11-year period, from the 1st of January 1985 to the 31st of December 1996, consisting of 2660 daily observations. This amount of data is relatively small compared to the time series used in Natural Sciences, but large enough compared to other studies in Economics and Finance, in most of which the data length merely reach 2000 observations. The algorithm has been tested and evaluated using the raw data, that is, the actual fixing rates of each currency rescaled on $[-1, 1]$.

The rest of the paper is organized as follows: Section 2 describes the hybrid algorithm, while the numerical experiments are presented in section 3. Finally, section 4 discusses the conclusions derived.

2. The Hybrid Algorithm

2.1. THE LOCALIZED EXTENDED KALMAN FILTER

A category of MLP neural networks training algorithms are those using advanced filtering techniques. One of the most effective among the different methods belonging to this category, is the localized approach of the Extended Kalman Filter (Shah et al., 1992), briefly presented in this section.

Let us consider an MLP network characterized by a vector $\boldsymbol{w}$ representing the free parameters of the network (i.e. the weights and the thresholds of the nodes). The average cost function that should be minimized during the training phase, is defined in terms of N input-output patterns as follows:

$$E_{av}(\boldsymbol{w}) = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j \in C} \left[d_j(n) - y_j(n) \right]^2 , \tag{1}$$

where $d_j(n)$ is the desired response (actual sample value) and $y_j(n)$ the response of output neuron j when the input pattern n is presented, while the set C includes all the output neurons of the network. The cost function $E_{av}(\boldsymbol{w})$ depends on the vector $\boldsymbol{w}$ due to the fact that $y_j(n)$ itself depends on $\boldsymbol{w}$.

Concentrating on an arbitrary neuron i, which might be located anywhere in the hidden or output layers of the network, its behavior during the training phase may be viewed as a non-linear dynamic system, which in the context of Kalman filter theory may be described by the following state-measurement equations (Haykin, 1994; Shah et al., 1992):

$$\boldsymbol{w}_i(n+1) = \boldsymbol{w}_i(n) \tag{2}$$

$$d_i(n) = y_i(n) + e_i(n) \tag{3}$$

$$y_i(n) = \varphi(\boldsymbol{x}_i{}^T(n), \boldsymbol{w}_i(n)) , \tag{4}$$

where iteration n corresponds to the presentation of the n-th input pattern, $\boldsymbol{x}_i(n)$ and $\boldsymbol{y}_i(n)$ are the input and output vectors of neuron i respectively and $\boldsymbol{e}_i(n)$ is the error measurement at the output of neuron i, the instantaneous estimate of which is given by:

$$e_i(n) = -\frac{\partial E(n)}{\partial y_i(n)} , \tag{5}$$

where

$$E(n) = \frac{1}{2} \sum_{j \in C} \left[d_j(n) - y_j(n) \right]^2 \tag{6}$$

The differentiation in Equation (5) corresponds to the back-propagation of the global error to the output of neuron i.

The activation function $\varphi(\bullet)$ in (4) is responsible for the non-linearity in the neuron. The vector $\boldsymbol{w}_i$ of the optimum model for neuron i is to be 'estimated' through training with examples. The activation function is assumed to be differentiable. Accordingly, we can use Taylor series to expand Equation (3) about the current estimate of the weight vector and thereby linearize the equation as follows (Haykin, 1994):

$$\varphi\left(x_i^T(n)w_i(n)\right) \cong q_i^T(n)w_i(n) + \left[\varphi\left(x_i^T(n)\hat{w}_i(n)\right) - q_i^T(n)\hat{w}_i(n)\right] , \tag{7}$$

where

$$q_i(n) = \left[\frac{\partial\varphi\left(x_i^T(n)w_i(n)\right)}{\partial w_i(n)}\right]_{w_i(n)=\hat{w}_i(n)} = \left[1 - \hat{y}_i^2(n)\right]\frac{x_i(n)}{2} \tag{8}$$

$\hat{y}_i(n)$ is the output of neuron i that results from the use of the weight estimate. In Equation (8) we have assumed the use of the hyperbolic tangent function; other sigmoid functions, like the logistic, may be used as well. The first term of the right hand side of Equation (7) is the desired linear term, while the remaining term represents a modeling error. By substituting Equation (7) and (4) in (3) and by ignoring the modeling error we obtain:

$$d_i(n) = q_i^T(n)w_i(n) + e_i(n) , \tag{9}$$

where $e_i(n)$ and $q_i(n)$ are defined in Equations (5) and (8) respectively. Equations (2) and (9) describe the linearized behavior of neuron i.

Given the pair of Equations (2) and (9), we can make use of the standard Recursive Least Squares (RLS) algorithm equations (Haykin, 1994), which is a special case of the Kalman filter, to make an estimate of the weight vector $w_i(n)$ of neuron i. The resulting solution is defined by the following system of recursive equations (Haykin, 1994), that describe the Multiple Extended Kalman Algorithm (MEKA) (Shah et al., 1992):

$$\boldsymbol{r}_i(n) = \lambda^{-1}\boldsymbol{P}_i(n-1)\boldsymbol{q}_i(n) \tag{10}$$

$$\boldsymbol{k}_i(n) = \boldsymbol{r}_i(n)\left[1 + \boldsymbol{r}_i^T(n)\boldsymbol{q}_i(n)\right]^{-1} \tag{11}$$

$$\boldsymbol{w}_i(n+1) = \boldsymbol{w}_i(n) + e_i(n)\boldsymbol{k}_i(n) \tag{12}$$

$$\boldsymbol{P}_i(n+1) = \lambda^{-1}\boldsymbol{P}_i(n) - \boldsymbol{k}_i(n)\boldsymbol{r}_i^T(n) \tag{13}$$

where, $n = 1, \ldots, N$ is the iteration number and N is the total number of examples. The vector $\boldsymbol{q}_i(n)$ represents the linearized neuron activation function given in

Equation (8), $\boldsymbol{P}_i(n)$ is the current estimate of the inverse of the covariance matrix of $\boldsymbol{q}_i(n)$ and $\boldsymbol{k}_i(n)$ is the Kalman gain. The parameter λ is a forgetting factor which takes values in the range (0, 1], and $e_i(n)$ is the localized measure of the global error. Equation (13) is called the Riccatti difference equation. Each neuron in the network perceives its own effective input $\boldsymbol{q}_i(n)$ hence it has to maintain its own copy of $\boldsymbol{P}_i(n)$, even in the case in which it may share some of its inputs with other neurons in the network.

2.2. THE EVOLUTION PROCESS

The proposed modified GA maintains a population of individuals (ANN) for each generation having random structure (random number of inputs and hidden units). The MEKA algorithm is employed for the training of each network for one epoch. Performance is measured with the fitness function, which is a function of the Mean Relative Error (MRE) and the size (number of nodes) of the network. Then a new population is created, by selecting the more fit individuals according to their fitness (select step). Some members of the population undergo transformations by means of genetic operators to form the new individuals. We use a mutation operator that changes, randomly, the structure of the network in order to preserve diversity. Also, there is a crossover operator, which creates new individuals by combining parts from two individuals. After some number of iterations the program converges at a near-optimum solution. The steps of the algorithm are briefly described in the following:

Step 1: Initialization

An initial population of individuals is created. Each individual, a neural network, is generated randomly (random number of inputs and hidden neurons). The population size is an important factor that influences the convergence of the algorithm. Generally, a large population size is preferable. In the experiments conducted we had to compromise with the computer limitations, so a population of one hundred individuals was developed in all currency cases. The connection weights of the individuals were initialized to random values within a certain interval.

Step 2: Selection

The selection is an essential operation in genetic algorithms; it constructs a new population with respect to the probability distribution based on fitness values of the individuals in the previous generation. In our experiments a variation of the classic Roulette Wheel Selection Operator (Michalewicz, 1996) was used. This variation saves the best (thus far) individual in a place outside the population and when the selection operation is applied we make sure that at least one copy of this individual passes to the next generation.

We have investigated the use of three different fitness functions. The first one takes into consideration only the performance of the network according to the Mean Relative Error (MRE). This function has the following form:

$$\text{Fitness} = 1/(1 + \text{MRE}) \tag{14}$$

The second one attempts to optimize the structure of the individuals by inserting the size of the individual directly into the fitness function. Hence, this function is a combination of the MRE and the size of the individual:

$$\text{Fitness} = 1/(1 + \text{MRE} + \text{size\_par*SIZE}) , \tag{15}$$

where SIZE is the total number of neurons in a network and size_par a parameter that controls the importance of SIZE in the evaluation of the fitness function. The objective is the size_par to take values that will lead to individuals with simple architectures, which will lead in its turn to reducing complexity and computational time, maintaining though good forecasting ability. The third function used is a variation of (15):

$$\text{Fitness} = 1/(1 + \text{MRE} + \text{size\_par*MRE*SIZE}) . \tag{16}$$

MRE is included in the third term of the denominator of Equation (16) to handle the strong dependency of the function on the size of the network (Equation (15)) as MRE gets smaller through generations.

Step 3: Crossover

The crossover or recombination operator was applied to the new population as follows: two parents were selected and after recombining parts of them one or two offspring were generated. The offspring replace their parents in the new population.

Analytically:

Let us assume that we have the two parents: I_1H_1O and I_2H_2O where I, H and O are the numbers of input, hidden and output nodes, respectively. Next we generate the random numbers:

i_1 = a uniform random number in $[0, I_1]$,
i_2 = a uniform random number in $[0, I_2]$,
h_1 = a uniform random number in $[0, H_1]$,
h_2 = a uniform random number in $[0, H_2]$.

Then we create a child with $(i_1 + i_2)$ input notes, $(h_1 + h_2)$ hidden nodes and O output nodes. If $(i_1 + i_2) = 0$ then we set the number of input nodes to 1; if $(h_1 + h_2) = 0$ we set the number of hidden nodes to 1. The weights of the child are initialized randomly in the same interval that was used in the initialization phase. The second child is created in the same manner.

Step 4: Mutation

The mutation process has been proved to be a very significant operator that greatly influences the convergence of the algorithm. The mutation operator used works as follows: it selects randomly a neural network (individual) from the population and changes its number of inputs and/or its number of hidden neurons by adding or deleting a random number (selected uniformly from a given interval) of inputs and/or hidden neurons.

3. Numerical Experiments

3.1. NEURAL NETWORKS RESULTS

All simulations conducted in this section were based on the following constant values for the variables involved:

The population size was set equal to 100, while the number of generations equal to 50. The weight values were initialized in the range $[-1.0, 1.0]$. The probabilities of applying the genetic operators of crossover, mutation of the number of inputs and mutation of the number of hidden neurons, were kept at the value of 0.1. Finally, the Kalman training phase was left to run for a single epoch.

The selection of the training and testing sets is very important. Some periods in a time series may lead to a very poor forecasting performance. This is due to some abrupt changes that may occur in the series, which provide for misleading information during the learning stage. We have used randomly selected periods for training our networks and we indeed observed significant variations in the ability of the networks to embody all the available information. Based on these findings we chose to exclude a large number of samples in the beginning of each currency series, using the last 1300 daily values for forecasting future behavior. This length was not chosen arbitrarily. We have investigated various different time periods regarding the start date of the training set. Performance was improving as the start date moved further towards more recent observations in time space. This finding was more or less expected. The fluctuations of the series during the first 4–5 years are much more pronounced, as a result of significant events in the Greek economy that led, in some cases. to drachma devaluation. Table I presents the prediction sensitivity on the data length variability by considering three training sets cases: The first set starts from the very first sample and numbers 1900 samples in total. The second excludes the first 200 points and includes the 1700 samples immediately after. The third training set excludes the first 400 samples and includes the 1500 samples immediately after. For each of the three training sets a testing set containing the last 100 samples was used, trying to forecast the behavior of our currency series during the most recent available time period. The results of Table I involve only the most accurate forecasting experiments that used the fitness function described in Equation (14) to keep the size of the table in reasonable bounds. The remaining two fitness functions (Equations (15) and (16)) resulted the

Table I. Forecasting results using the hybrid algorithm on the USD/GRD, GBP/GRD, DEM/GRD and FF/GRD series, optimized with (1/(1+MRE)) as fitness function, using different data sets on a daily predicting horizon.

Currency	Data Set Used (Total 2000 samples)	Network[a]	CC	NRMSE	MRE
USD	Training: First 1900 samples Testing: Last 100 samples	13-26-1	0.621	0.729	0.362
	Training: 1700 samples (first 200 skipped) Testing: Last 100 samples		0.734	0.678	0.287
	Training: 1500 samples (first 400 skipped) Testing: Last 100 samples		0.866	0.589	0.178
GBP	Training: First 1900 samples Testing: Last 100 samples	20-40-1	0.598	0.781	0.388
	Training: 1700 samples (first 200 skipped) Testing: Last 100 samples		0.754	0.623	0.256
	Training: 1500 samples (first 400 skipped) Testing: Last 100 samples		0.897	0.553	0.159
DEM	Training: First 1900 samples Testing: Last 100 samples	4-28-1	0.699	0.687	0.331
	Training: 1700 samples (first 200 skipped) Testing: Last 100 samples		0.810	0.609	0.253
	Training: 1500 samples (first 400 skipped) Testing: Last 100 samples		0.906	0.445	0.125
FF	Training: First 1900 samples Testing: Last 100 samples	8-46-1	0.679	0.693	0.344
	Training: 1700 samples (first 200 skipped) Testing: Last 100 samples		0.807	0.611	0.254
	Training: 1500 samples (first 400 skipped) Testing: Last 100 samples		0.913	0.332	0.096

[a] *m-d-n* stands for *m* inputs, *d* hidden neurons and *n* outputs.

same forecasting accuracies and thus were omitted. A close look at Table I reveals the improvement of forecasting accuracies as we move from the first to the third training set used. Thus, after continuous experimentation we decided to consider only the last 5–6 years, which combine the typical movements of a currency in the Greek exchange-rate market with the more recent economic policy followed in Greece.

The results reported hereafter regard the architectures that performed best with a 1200-sample length for the training data set (approximately 5 years, in-sample data) and a 100-sample (out-of-sample data) length for the testing set, starting immediately after the last sample value used in the training set. Objective evaluation is achieved through particular error measures reported below, on this testing set of data, which was not previously presented to the network during any of the intermediate training stages.

Evaluation of forecasting is performed through three well-known error measures, the Normalized Root Mean Squared Error (NRMSE), the Correlation Coefficient (CC) and the Mean Relative Error (MRE).

The NRMSE is calculated using the root-mean-squared-error (rmse) given by:

$$\sigma_{\Delta}(T) = \left\langle \left[x_{\text{pred}}(t, T) - x_{\text{act}}(t, T) \right]^2 \right\rangle^{\frac{1}{2}} , \tag{17}$$

where T is the number of samples being predicted. The rmse is normalized by the rmse deviation of the data $\sigma \left\langle (x - \langle x \rangle)^2 \right\rangle^{\frac{1}{2}}$ producing the normalized error NRMSE $= \sigma_{\Delta}(T)/\sigma_X$. If NRMSE = 0 then predictions are perfect; NRMSE = 1 indicates that prediction is no better than taking x_{pred} equal to the x-mean.

Prediction was also tested using the correlation coefficient (CC) between the actual and predicted series. The CC measures the ability of the predicted samples to follow the upward or downward jumps of the original series. A CC value near 1 in absolute terms is interpreted as a perfect follow up of the original series by the forecasted one. A negative CC sign indicates that the forecasting series follows the same ups or downs of the original with a negative mirroring, that is, with a 180° rotation about the time-axis. When the original series moves up the forecasting moves down at the same time-period and vice versa.

The MRE is given by the formulae:

$$\text{MRE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{o_i - d_i}{d_i} \right| , \tag{18}$$

where o_i is the output of the network, d_i is the actual value when pattern i is presented and n is the total number of patterns. MRE shows the percentage of accuracy of predictions expressing it in a stricter way, since it focuses on the sample being predicted. Thus, we are able to estimate prediction error as a fraction of the actual value.

Table II summarizes the best results obtained for each currency, under the use of the three different fitness functions mentioned earlier. One can observe that, in general, the performance of the various network architectures shows a remarkable level of accuracy in all currency and function cases. Correlation coefficient ranges from 92% (worst case) to 99% (best case), evaluated on the testing period. The NRMSE figure is also very low, reaching in some cases at 0.07, while the MRE remains below the value of 0.03. These results suggest that the proposed method produces forecasts that successfully follow the original series both in values and fluctuations. The predictive ability is also by far superior compared to the simple mean (NRMSE). The accuracy of predictions is easily observed through Figures 1 to 4, where the best network architecture of each currency is graphed, including both the training (in-sample) and the testing (out-of-sample) phase (training lies on the left hand-side of the dashed line, while testing on the right). Figures 2, 3 and 4

Table II. Forecasting results using the hybrid algorithm on the USD/GRD, GBP/GRD, DEM/GRD and FF/GRD series, for different fitness functions on a daily predicting horizon. The data set used includes the last 1300 samples.

Currency	Fitness Function	Size_par	Network[a]	CC	NRMSE	MRE
USD	1/(1 + MRE)	–	13-26-1	0.964	0.289	0.018
		–	23-26-1	0.940	0.370	0.018
	1/(1 + MRE + size_par*SIZE)	0.060	1-1-1	0.975	0.219	0.028
		0.030	1-1-1	0.975	0.228	0.029
	1/(1 + MRE + size_par*MRE*SIZE)	0.100	1-3-1	0.974	0.223	0.028
		0.400	1-1-1	0.974	0.239	0.031
GBP	1/(1 + MRE)	–	20-40-1	0.943	0.340	0.014
		–	21-41-1	0.925	0.423	0.017
	1/(1 + MRE + size_par*SIZE)	0.030	1-1-1	0.961	0.288	0.014
		0.010	1-4-1	0.965	0.268	0.012
	1/(1 + MRE + size_par*MRE*SIZE)	0.250	1-3-1	0.966	0.258	0.012
		0.400	1-2-1	0.961	0.292	0.013
DEM	1/(1 + MRE)	–	4-28-1	0.996	0.083	0.001
		–	5-21-1	0.996	0.083	0.001
	1/(1 + MRE + size_par*SIZE)	0.030	1-1-1	0.996	0.168	0.003
		0.010	1-1-1	0.995	0.525	0.013
	1/(1 + MRE + size_par*MRE*SIZE)	0.100	1-11-1	0.996	0.069	0.001
		0.010	1-22-1	0.995	0.110	0.002
FF	1/(1 + MRE)	–	2-31-1	0.985	0.183	0.007
		–	8-46-1	0.996	0.080	0.007
	1/(1 + MRE + size_par*SIZE)	0.030	1-1-1	0.986	0.178	0.007
		0.001	1-4-1	0.987	0.180	0.007
	1/(1 + MRE + size_par*MRE*SIZE)	0.200	1-6-1	0.987	0.157	0.006
		0.200	1-5-1	0.987	0.162	0.006

[a] *m-d-n* stands for *m* inputs, *d* hidden neurons and *n* outputs.

are hard to distinguish between the actual (solid) and the forecasted (dashed) series. This is not due to the scaling used for the graphs, it results from the fact that actual and predicted values are almost identical. The algorithm forecasted future values with negligible deviation from the original ones, especially in the case of FF and DEM. Performance is slightly inferior in the case of USD and GBP, where a very small diversification from the original series is observed during the training phase for a period of 700 days for the former and 200 for the latter. These periods, as one can easily observe, are characterized by high-frequency fluctuations. Hence, we can also remark that the proposed algorithm presents a slightly weak point regarding the learning of 'wild' fluctuations. This, however, can not be regarded as a serious drawback, since its performance on these periods is still highly successful. CC was calculated only for the periods of fluctuations and resulted the values of

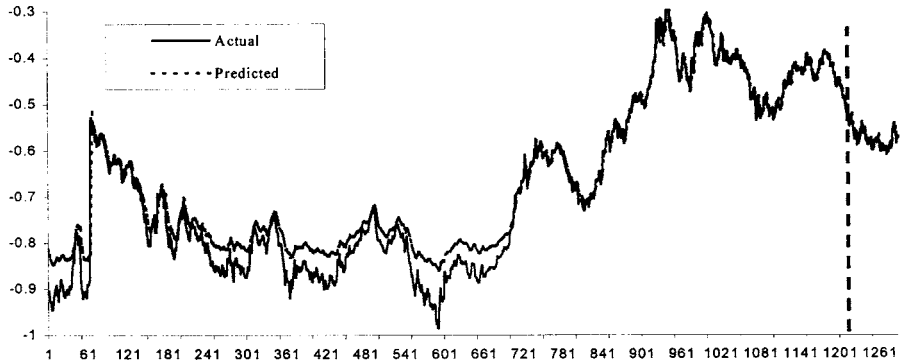

Figure 1. Actual versus predicted series of the USD/GRD exchange rates, using a neural network with 1 input and 1 hidden node, optimized with (1/(1+MRE+size_par*SIZE)) as fitness function and size_par equal to 0.060. The training phase (in-sample) covers the left side of the dashed line, while the evaluation phase (out-of-sample) the right side.

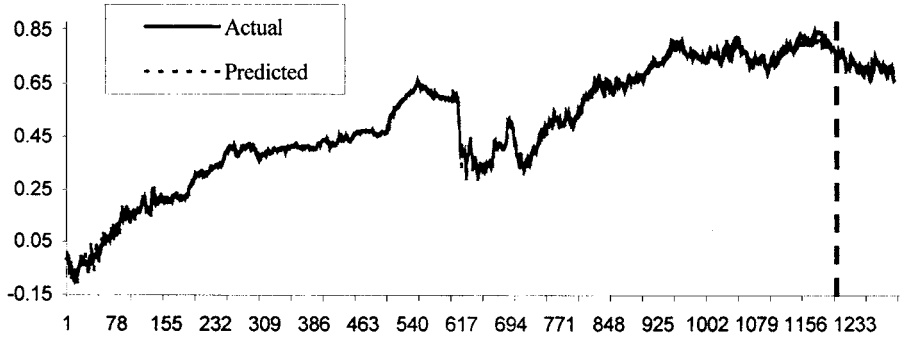

Figure 2. Actual versus predicted series of the GBP/GRD exchange rates, using a neural network with 1 input and 3 hidden nodes, optimized with (1/(1+MRE+size_par*SIZE)) as fitness function and size_par equal to 0.250. The training phase (in-sample) covers the left side of the dashed line, while the evaluation phase (out-of-sample) the right side.

0.96 for USD and 0.98 for GBP. Thus, we can conclude that the networks ability of learning in highly fluctuated periods is satisfactory.

USD and GBP forecasts, depicted in Figures 1 and 2 respectively, were obtained using a neural network with only one input node. One might consider this as a simple transfer function from today's rate to tomorrow's rate. Nevertheless, there is more behind the learning process than just a simple mapping of one input to one output. The nonlinear, recursive Equations (10) to (13) describing the MEKA are responsible for determining the internal node weights in the neural network. Therefore, prediction is indeed simple on one hand because it is based solely on one past sample, but it is internally complex at another, as weight estimates are produced using the advanced Kalman filtering technique. It should be noted, however, that the GA reached to this single input network topology only for the USD and GBP series signifying that the necessary knowledge for reaching to accurate

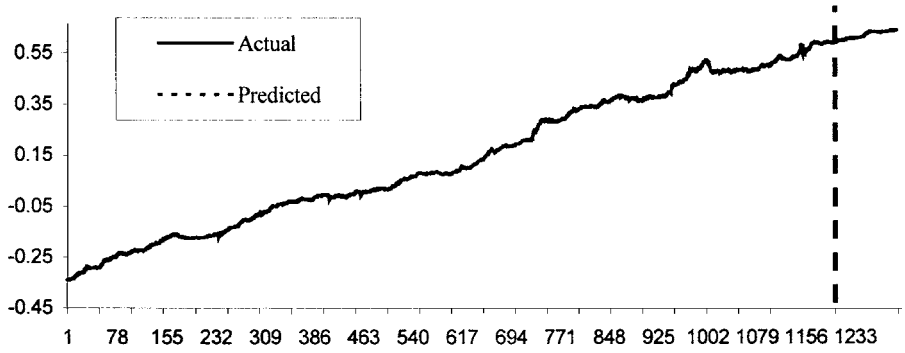

Figure 3. Actual versus predicted series of the DEM/GRD exchange rates, using a neural network with 5 input and 21 hidden nodes, optimized with (1/(1+MRE)) as fitness function. The training phase (in-sample) covers the left side of the dashed line, while the evaluation phase (out-of-sample) the right side.

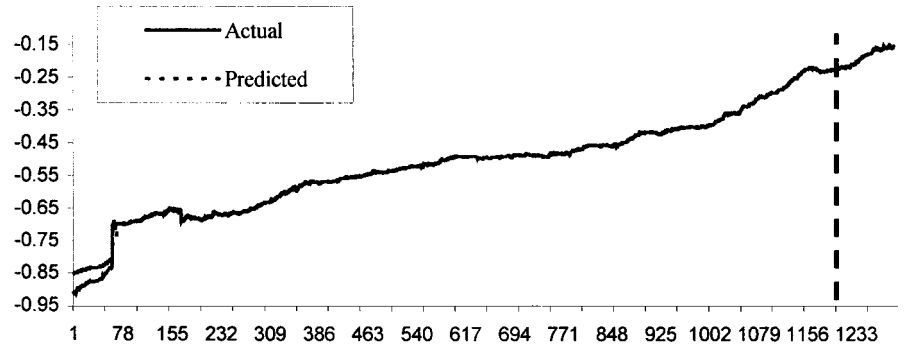

Figure 4. Actual versus predicted series of the FF/GRD exchange rates, using a neural network with 8 input and 46 hidden nodes, optimized with (1/(1+MRE)) as fitness function. The training phase (in-sample) covers the left side of the dashed line, while the evaluation phase (out-of-sample) the right side.

predictions for these series is embedded in only one past rate. This is perfectly logical considering the work of Andreou et al. (2000), which reports a high level of noise present in these two currency series. This noise mixture disturbs the quality of knowledge in the sample data, hence using only one past sample less amount of noise creeps into the learning process and results higher prediction accuracies.

Based on the error figures and the graphs we can easily conclude that DEM and FF series are described by a structure that allows for a slightly better forecasting ability compared to the one of USD and GBP. This finding is consistent with the work of Karytinos et al. (2000) and Andreou et al. (2000) reporting evidences of a deterministic, low-dimensional, noisy chaotic system for DEM/GRD and FF/GRD, and a random explanation for the USD/GRD and GBP/GRD series.

It is also clear that Equation (15) is the leader, among the different fitness functions used by the GA to optimize the network architectures, regarding the convergence to simpler topologies with high predictive performance.

3.2. COMPARISON TO OTHER MODELS

The nature of the learning techniques for training neural networks, as well as the course of the input/output mapping achieved at the end of the training phase are often characterized as a black box procedure. Thus, evaluation of results obtained is best presented in comparison to other well-known models. In this subsection we present the forecasting results of five model estimators for each currency by means of stationary AutoRegressive (AR), Moving Average (MA) and AutoRegressive Moving Average (ARMA) processes, and two naive predictors based on the mean of past values (NPMV) and the value of the previous sample (NPLV).

Our goal in this evaluation is not by all means to present an analytical time-series framework for developing such stationary models. Instead, we will focus on selecting the appropriate order of the model, correctly fitting the model on the data series investigated and evaluating the results in contrast with the ones obtained with the proposed GA method.

The process $\{X_t, t = 0, \pm 1, \pm 2, \ldots\}$ is said to be an ARMA(p, q) process if $\{X_t\}$ is stationary and if for every t,

$$X_t - \varphi_1 X_{t-1} - \ldots - \varphi_p X_{t-p} = Z_t + \theta_1 Z_{t-1} + \ldots + \theta_q Z_{t-q} \tag{19}$$

where $\{Z_t\} \sim \text{WN}(0, \sigma^2)$. Equation (19) can be written in the more compact form

$$\varphi(B) X_t = \theta(B) Z_t, \quad t = 0, \pm 1, \pm 2, \ldots \tag{20}$$

where φ and θ are the pth and qth degree polynomials

$$\varphi(z) = 1 - \varphi_1 z - \varphi_2 z^2 - \ldots - \varphi_p z^p \tag{21}$$

and

$$\theta(z) = 1 + \theta_1 z + \theta_2 z^2 + \ldots + \theta_q z^q \tag{22}$$

and B is the backward shift operator defined by

$$B^j X_t = X_{t-j}, \quad t = 0, \pm 1, \pm 2, \ldots \tag{23}$$

The moving average process of order q (or MA(q)) is obtained with $\varphi(z) \equiv 1$, while the autoregressive process of order p (or AR(q)) with $\theta(z) \equiv 1$.

Our series are clearly non-stationary data and had to be transformed before attempting to fit a stationary model. The transformation carried out includes the following steps (Brockwell and Davis, 1996):

Step 1: Box-Cox Transformation

This transformation is suitable when the variability of the data increases or decreases with the level. Using the conversion

$$f(y) = \left\{ \begin{array}{l} \dfrac{y^{\lambda} - 1}{y}, \lambda \neq 0 \\ \log(y), \lambda = 0 \end{array} \right\} \tag{24}$$

and by suitable choice of λ the variability can be made nearly constant. We have used $\lambda = 1.5$ to convert our series.

Step 2: Seasonal Component and Trend Elimination

All possible seasonal components were eliminated using the technique of differencing. The idea is to consider the differences between pairs of observations with appropriate time-separations (lags). The transformed series are generated by taking

$$Y_i = X_t - X_{t-T} \tag{25}$$

where T is the period of the seasonal component. Differencing at lag 1 can eliminate a linear trend.

We performed differencing with the lag being equal to 250 for the USD and GBP, 200 for the DEM and 120 for the FF series, after visual inspection of each series. Linear trend was also removed from all series.

Step 3: Mean Subtraction

The models intended to be used, e.g. ARMA, belong to the class of stationary zero mean processes. In order to fit such models to our data we have subtracted the sample rnean of the transformed data from each sample observation.

Selection of the initial order of our models was based on the sample Autocorrelation and Partial Autocorrelation Functions. The Akaike's Information Criterion Corrected (AICC) statistic was then used for deciding between rival candidates for 'best' model to represent a particular series. The AICC is a bias-corrected modification of the Akaike's IC statistic, which introduces a penalty for increasing the number of modal parameters (Brockwell and Davis, 1996). The order of the ARMA(p, q) model is selected so as to minimize the AICC statistic. Once such a model is formed, estimation of non-zero coefficients is performed on the basis of maximizing the Gaussian likelihood.

Finally, our model family is completed with two naive estimators. The first one predicts a future value by taking the mean of the past n values (NPMV(n)), while the second one by just taking the previous value as the new, forecasted one.

The results of all models used are presented in Table III. The best performed NN for each currency is also provided for comparison purposes. The main conclusion

drawn from comparing these results to those obtained by the use of the hybrid algorithm on neural networks is that the latter technique is obviously superior. The Correlation Coefficient is much lower for the stationary models, while in most cases it reveals a reverse follow-up of the original series (negative sign). When it comes to the naive estimators, CC is high enough, lower though compared to the one given by neural networks. NRMSE has almost the same behavior as CC, with the naive estimators giving the lowest error values, still higher than the ones of the NNs. Finally, MRE stays at low levels, due to the small deviations between actual and predicted samples, also higher though than those obtained by the NNs.

These findings suggest that the stationary models used failed in successfully predicting the data, in consistency with numerous reports in literature that describe this inability as the outcome of high frequency noise and/or chaotic motion in such systems. Naive predictors on the other hand proved higher predictive ability due to their structure, which was developed to consider 5 past sample values the most. This leads to an artificially nice fitting as a result of the small changes observed in currency prices as time increases.

3.3. MULTISTEP PREDICTION

This subsection is devoted to investigating the predictive ability of the proposed hybrid algorithm on a multistep forecasting horizon. All the experiments conducted so far proved a very successful predictive performance over a one-step ahead daily currency value. One might argue, though, that if the time interval under which predictions are attempted was increased, accuracy would decrease. This is a somewhat expected result due to the fact that the non-linear structure of currency markets (Andreou et al., 1997; Andreou et al., 1999; Peters, 1991) causes an almost exponential divergence between the actual values and the ones predicted by the model used.

Our case study was based on predicting with three different time lags ahead: A two-days lag, a five-days lag (corresponding to one calendar week) and a twenty-days lag (corresponding to one calendar month). Training of the networks developed was carried out using consecutive sample values and the sample predicted is chosen to be k values ahead from the last observation participated in the input vector, with k corresponding to the time-lags mentioned. The optimization performed by the GA was based on the fitness functions given by Equations (15) and (16) that yielded simpler network architectures in the previous subsection. The choice of the parameter size_par, was a very difficult task and was made empirically after numerous experiments, aiming primarily at achieving the highest forecasting performance rather than the simpler network architecture.

The best results of the multistep predicting horizon experiments are summarized in Table IV. The decrease in prediction accuracy as the time-lag increases is clearly discernible in both fitness functions used. CC and NRMSE proved a quite successful performance even in the one-month case, where only the USD series suffered a strong deviation between actual and predicted samples. MRE stayed

Table III. Forecasting evaluation of stationary models and naive estimators on the USD/GRD, GBP/GRD, DEM/GRD and FF/GRD series. The best NN performance is also reported.

Currency	Model	CC	NRMSE	MRE
USD/GRD	AR(6)	–0.840	1.680	0.205
	MA(6)	–0.840	1.675	0.205
	ARMA(6,6)	–0.841	1.672	0.204
	NPMV(2)	0.957	0.302	0.048
	NPMV(5)	0.902	0.473	0.074
	NPLV	0.967	0.256	0.039
	NN(1-1-1)	0.975	0.219	0.028
GBP/GRD	AR(11)	–0.803	5.708	0.093
	MA(11)	–0.804	5.753	0.094
	ARMA(11,11)	–0.795	5.597	0.091
	NPMV(2)	0.799	0.648	0.009
	NPMV(5)	0.708	0.798	0.011
	NPLV	0.829	0.600	0.008
	NN(1-3-1)	0.966	0.258	0.012
DEM/GRD	AR(6)	–0.838	6.317	0.050
	MA(26)	–0.861	6.108	0.048
	ARMA(6,26)	–0.859	7.531	0.059
	NPMV(2)	0.938	0.365	0.002
	NPMV(5)	0.907	0.491	0.003
	NPLV	0.943	0.338	0.002
	NN(5-21-1)	0.996	0.083	0.001
FF/GRD	AR(21)	0.604	1.664	0.012
	MA(4)	0.782	0.707	0.004
	ARMA(21,4)	0.610	1.614	0.011
	NPMV(2)	0.831	0.572	0.004
	NPMV(5)	0.746	0.693	0.004
	NPLV	0.847	0.550	0.003
	NN(8-46-1)	0.996	0.080	0.007

low in all cases as in the one-step experiments. Finally the use of the fitness function described in Equation (15) resulted, in general, simpler network architectures compared to the one given by Equation (16), yet maintaining good performance.

Table IV. Forecasting results using the hybrid algorithm on the USD/GRD, GBP/GRD, DEM/GRD and FF/GRD series, for different fitness functions and predicting horizons.

Currency	Fitness Function	Horizon	Network[a] (days)	CC	NRMSE	MIRE
USD	1/(1 + MRE + size_par*SIZE)	2	1-3-1	0.955	0.326	0.044
	size_par = 0.03	5	1-2-1	0.350	1.659	0.092
		20	1-1-1	0.385	1.296	0.074
	1/(1 + MIRE + size_par*MRE*SIZE)	2	1-1-1	0.955	0.336	0.045
	size_par = 0.1	5	1-1-1	0.855	0.519	0.064
		20	1-3-1	0.375	0.928	0.050
GBP	1/(1 + MRE + size_par*SIZE)	2	1-1-1	0.939	0.393	0.019
	size_par = 0.03	5	1-1-1	0.825	1.053	0.049
		20	1-1-1	0.839	0.840	0.035
	1/(1 + MRE + size_par *MRE*SIZE)	2	1-4-1	0.939	0.353	0.017
	size_par =0.1	5	1-3-1	0.875	0.514	0.025
		20	2-2-1	0.866	0.498	0.023
DEM	1/(1 + MRE + size_par *SIZE)	2	1-1-1	0.992	1.045	0.025
	size_par =0.03	5	1-2-1	0.969	0.776	0.015
		20	1-1-1	0.867	0.986	0.016
	1/(1 + MRE + size_par*MRE*SIZE)	2	1-7-1	0.991	0.127	0.002
	size_par = 0.1	5	1-1-1	0.968	0.324	0.006
		20	2-2-1	0.872	0.503	0.009
FF	1/(1 + MRE + size_par*SIZE)	2	1-1-1	0.970	0.433	0.019
	size_par = 0.03	5	1-1-1	0.917	1.006	0.046
		20	1-1-1	0.575	1.133	0.047
	1/(1 + MRE + size_par*MRE*SIZE)	2	1-3-1	0.969	0.276	0.010
	size_par=0.1	5	1-4-1	0.913	0.418	0.016
		20	1-4-1	0.654	0.767	0.030

[a] *m-d-n* stands for *m* inputs, *d* hidden neurons and *n* outputs.

4. Conclusions

This paper has presented a new hybrid algorithm based on a Genetic Algorithm for the evolution of the architecture of MLP neural networks and a localized version of the Extended Kalman Filter for the training. The application of this algorithm on the task of exchange rate forecasting of four major currencies, namely the U.S. Dollar (USD), the Deutsche Mark (DEM), the French Franc (FF) and the British Pound (GBP), versus the Greek Drachma, was very positive and encouraging. The data used involve raw prices, without elimination of trends present or noise imposed. It is noteworthy that in all the analyzed time series, a remarkable prediction success has been achieved in both, a one-step ahead and a multistep predicting horizon. The latter, being the most difficult and at the same time interesting problem, proved the strong influence of noise in a long-term forecasting. The fact remains, though, that the hybrid algorithm showed a stable performance, at least for one-week ahead predictions, with significant success.

The results obtained, are by far more successful when compared to any related works on the same data series (Andreou et al., 1997; Adamopoulos et al., 1997; Andreou et al., 1998) or other currencies in world literature. The networks produced by the hybrid genetic algorithm succeeded to track the time series fluctuations for all currencies tested, with very small to negligible accuracy deviation. FF and DEM series proved to have structural characteristics that led to better predictive ability than USD and GBP. Comparison to other stationary forecasting models, as well as naive predictors, clearly favored the proposed hybrid method.

A slightly weak point of the algorithm concerning its learning ability, proved to be a small sensitivity in periods characterized by highly frequent and abrupt fluctuations. This interesting characteristic was observed only during the learning phase, not the generalization phase (out-of-sample evaluation). During these periods, the ability of embodying the input information was slightly reduced compared to other smoother periods, in the order of 2–3%. A possible explanation could be the delay of the training algorithm to adjust the weights and intermediate errors in internal nodes before the next reversal of the series around its local mean is fed to the network.

Although the proposed hybrid algorithm exhibited great learning and generalization ability and forecasting results obtained were highly satisfactory, additional research may be conducted. Future steps may include the implementation of more refined genetic operators and a deeper investigation on the ways the values of the genetic and other heuristic parameters influence the performance of the algorithm and the generalization ability of the generated networks. Further research could also be concentrated on the appropriate choice of the parameter size_par in the fitness functions used by the GA to optimize the architecture of the networks. This could be handled by a meta-genetic algorithm, responsible only for the selection of this parameter. It would also be very interesting for one to see whether including the Correlation Coefficient in the denominator of a fitness function, in a similar manner as with MRE, could improve prediction. Finally, the application of the proposed hybrid algorithm on major wider traded exchange-rates, like U.S. Dollar versus Deutsche Mark, or U.S. Dollar versus Japanese Yen, would also be of great interest, since these currencies serve as the basis for worldwide transactions and small countries economies and are more free to fluctuate in the international markets than the Greek Drachma.

References

Adamopoulos, A., Andreou, A., Georgopoulos, E., Ioannou, N. and Likothanassis, S. (1997). Currency forecasting using recurrently RBF networks optimized by genetic algorithms. *Proceedings of the Fifth International Conference on Computational Finance 1997 (CF '97)*, London Business School, London.

Adamopoulos, A., Georgopoulos E., Manioudakis, G. and Likothanassis, S. (1998). An evolutionary method for system structure identification using neural networks. *Proceedings of the Neural Computation '98*, Vienna.

Andreou, A. S., Karytinos A. and Pavlides G. (2001). Non-linear time-series analysis of the Greek exchange-rate market. *International Journal of Bifurcation and Chaos*, **10** (7), 1729–1758.

Andreou, A., Georgopoulos, E., and Likothanassis, S. and Polidoropoulos, P. (1997). Is the Greek foreign exchange-rate market predictable? A comparative study using chaotic dynamics and neural networks. *Proceedings of the Fourth International Conference on Forecasting Financial Markets*, Banque Nationale de Paris and Imperial College, London.

Andreou, A., Georgopoulos, E., Zombanakis, G. and Likothanassis, S. (1998). Testing currency predictability using an evolutionary neural network model. *Proceedings of the Fifth International Conference on Forecasting Financial Markets*, Banque Nationale de Paris and Imperial College, London.

Brockwell, J.B. and Davis, A.R. (1996). *Time Series: Theory and Methods.* Springer-Verlag, New York.

Haykin, S. (1994). *Neural Networks – A Comprehensive Foundation*. McMillan College Publishing Company, New York.

Karytinos A., Andreou, A.S. and Pavlides G. (2000). 'Long-term dependence in exchange rates. *Journal of Discrete Dynamics in Nature and Society*, **4/1**, 1–20.

Kuan, C.M., and Liu T. (1995). Forecasting exchange rates using feedforward and recurrent neural networks, *Journal of Applied Econometrics*, **10**, 347–364.

Likothanassis, S.D., Georgopoulos, E. and Fotakis, D. (1997). Optimizing the structure of neural networks using evolution techniques. *5th Int. Conference on Applications of High Performance Computers in Engineering*, Spain, July 2–4.

Marsh, I.W. and Power, D.M. (1996). A note on the performance of foreign exchange forecasters in a portfolio framework, *Journal of Banking Finance*, **20**, 605–613.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag.

Peters, E. (1991), *Chaos and Order in the Capital Markets*, John Wiley, New York.

Refenes, A.P. and Zaidi, A. (1995). Managing exchange-rate predictions strategies with neural networks. In A.P. Refenes (ed.), *Neural Networks in the Capital Markets*, Wiley & Sons.

Shah, S., Palmieri, F. and Datum, M. (1992), Optimal filtering algorithms for fast learning in feed forward neural networks, *Neural Networks*, **5**, 779–787.

Verkooijen, W. and Daniels, H. (1994), Connectionist projection pursuit regression, *Computational Economics*, **7**, 155–161.

Verkooijen, W. (1996). A neural network approach to long-run exchange rate prediction, *Computational Economics*, **9**, 51–65.