Tools for Non-linear Time Series Forecasting in Economics

– An Empirical Comparison of Regime Switching Vector Autoregressive Models and Recurrent Neural Networks –

Jane M. Binner

Aston University

Thomas Elger

Lund University

Birger Nilsson

Lund University

Jonathan A. Tepper

The Nottingham Trent University

Abstract

The purpose of this study is to contrast the forecasting performance of two non-linear models, a regime-switching vector autoregressive model (RS-VAR) and a recurrent neural network (RNN), to that of a linear benchmark VAR model. Our specific forecasting experiment is UK inflation and we utilize monthly data from 1969-2003. The RS-VAR and the RNN perform approximately on par over both monthly and annual forecast horizons. Both non-linear models perform significantly better than the VAR model.

Keywords: Inflation forecasting, regime-switching vector autoregressive model, recurrent neural network.

JEL: C22, C45, C53.

1. Introduction

Non-linear models for economics and time series modeling have gained in popularity over recent years. The main reason for this is the failure of linear models to capture non-linear dynamic relationships embedded in real-world data. Econometric developments in combination with increases in computing power have further spurred the use of non-linear models. The purpose of this study is to contrast the forecasting performance of two non-linear models, a regime-switching (RS) vector autoregressive model (VAR) and a recurrent neural network (RNN), to that of a linear benchmark VAR model. These models belong to different classes of non-linear models that are both econometrically challenging and therefore rarely compared.

Our specific forecasting experiment is UK inflation over 1969-2003. For this purpose, we obtain monthly observations of the retail price index, M0 and industrial production. The first part of the data set is used for estimation (training). The last five years of the data is used for out-of-sample forecasting and evaluation of the different models. There are three main motives for choosing to study UK inflation using this set of variables. Firstly, the amount of available data must be considered large. Our full sample contains over 400 observations. Macroeconomists are often fortunate if quarterly data for, say, 20-30 years is available. Secondly, visual inspection of the data indicates that the use of a linear model may be inappropriate. Over the full sample, we can identify long periods with high inflation and long periods with low inflation. The latter is particularly evident for the nineties, or, more specifically, the 'post-ERM' period. Finally, as evident in numerous papers and central bank reports, many people are interested in inflation forecasts. For them, this forecasting experiment may be interesting in its own right.

In the regime switching framework, the underlying idea is to allow for endogenous switches between different data generating processes at different points in time, where the switches are governed by an underlying discrete state Markov process. Markov-switching regression models were introduced in economics by Goldfeld and Quandt (1973), but time-series applications were not considered until the seminal papers by Hamilton (1988, 1989). Regime-switching models are inherently non-linear because the mean, as well as higher moments, are non-linear functions of the current state of the Markov process. The different states of the underlying Markov process have in economic applications, for example, represented periods of low and high exchange rate volatility (Klaassen, 2002), different level and volatility of real interest rates (Garcia and Perron, 1996), booms and slumps (Hamilton, 1989), and low and high volatility and correlation in stock markets (Ang and Bekaert, 2002). Particular applications to forecasting in the RS-VAR framework include Krolzig (2004), who models output and employment growth and Blix (1999), who studies inflation in a trivariate RS-VAR system.

Artificial neural networks (ANNs) (Rumelhart *et al*., 1986; Cheng and Titterington, 1994; Haykin, 1999), on the other hand, consist of simple interacting processing units that are arranged in arbitrary layers with variable patterns of interconnectedness. Knowledge is represented as connection strengths (or weights) between connected units. Each processing unit spreads its activation to connected units after combining and processing its input using some linear or non-linear activation function. Learning occurs when general recursive rules are applied to adapt these weights in order to produce desired output responses.

ANNs are becoming increasingly popular in economics and are used in a large variety of modeling and forecasting problems. The main reason for this increased popularity is that these models have been shown to be able to approximate any non-linear function arbitrarily close (Cybenko, 1989; Hornik, 1991). Hence when applied to a time series which is characterized by truly non-linear dynamic relationships, the ANN will provide a global approximation to this unknown non-linear relationship. Previous studies unveil the applicability of ANNs to modeling and forecasting in economics. Applications include returns forecasting (Gençay, 1996; Haefke and Helmenstein, 1996a, 1996b; Gençay and Stengos, 1998), option pricing (Qi and Maddala, 1995), exchange rates (Kuan and Liu, 1995; Tenti, 1996; Franses and van Griensven, 1998; Franses and van Homelen, 1998; Gençay, 1999; Giles *et al*., 2001), interest rates (Swanson and White, 1995) and inflation (Moshiri *et al*., 1999; Stock and Watson, 1999; Binner, Gazely and Chen, 2002).[1] Due to the inherent ability of ANNs with recurrent connections (i.e. RNNs) to implicitly learn the non-linear temporal dynamics of sequential time series data without recourse to additional temporally-dependent external memory mechanisms, this work uses and evaluates RNN models.

The paper is organized as follows. Section two introduces the VAR model, the RS-VAR model and the RNN. Section three contains a brief description of the data together with a discussion of estimation (training) results. Section four presents the inflation forecast evaluation results. Section five concludes the paper.

2. Non-linear Models

This section introduces the VAR model, the RS-VAR model and the RNN. The *raison d'être* for considering the latter models is an *a priori* belief that linear models will fair

badly in our forecasting experiment. It is therefore logical to include a linear model as a benchmark model in our analysis. A natural choice is the VAR model, which is commonly used in macroeconomic forecasting experiments.

In the VAR model, the dynamics of $\mathbf{x}_t$, a *k*-dimensional vector of dependent variables at time *t*, is governed by the following *p*:th order autoregressive process:[2]

$$\mathbf{x}_t = \boldsymbol{\alpha}_0 + \mathbf{A}_1\mathbf{x}_{t-1} + ... + \mathbf{A}_p\mathbf{x}_{t-p} + \boldsymbol{\varepsilon}_t, \tag{1}$$

where $\boldsymbol{\alpha}_0$ is a vector of intercepts, $\mathbf{A}_\ell$, $\ell = 1,..., p$ are $k \times k$ coefficient matrices and $\boldsymbol{\varepsilon}_t$ is a white-noise distributed disturbance vector. Estimates of the coefficient matrices are obtained using ordinary least squares. The model is said to be stable if its reverse characteristic polynomial has no roots in and on the complex unit circle.

For the VAR model, a conditional *t*+1 forecast of **x** is obtained from:

$$E_t(\mathbf{x}_{t+1}) = \boldsymbol{\alpha}_0 + \mathbf{A}_1\mathbf{x}_t + ... + \mathbf{A}_p\mathbf{x}_{t-p+1}, \tag{2}$$

for each *t* using the information set $\mathbf{X}_t = \{\mathbf{x}_t, \mathbf{x}_{t-1}, ...\}$. The conditional *t*+1 forecast in (2) is a special case of the conditional dynamic $t + \tau$ forecast where *forecasted* values of **x** should be used for certain lags (depending on the forecast horizon and the number of included lags).

2.1 The Regime Switching VAR model

We follow the common practice in the economic regime switching literature and assume that a discrete time-homogenous *s*-state first order Markov process governs the endoge-

nous switches between the regimes. This assumption implies that the probability of a switch between different regimes is described by a constant transition matrix, that there is a different VAR model in each regime and that only the most recent state of the Markov-process influences the transition probabilities. We also assume that $\mathbf{x}_t$ can be modeled as a discrete mixture of *k*-variate Gaussian distributions. This assumption implies that $\mathbf{x}_t$ is Normal distributed conditional on the prevailing regime and the information set $\mathbf{X}_{t-1}$. Finally, we restrict the variance-covariance matrix in each regime to be constant.

Taken together, if the prevailing regime at time *t* is $j \in \{1, ..., s\}$, then the VAR process is:

$$\mathbf{x}_t = \boldsymbol{\alpha}_0^{(j)} + \mathbf{A}_1^{(j)}\mathbf{x}_{t-1} + ... + \mathbf{A}_{p(j)}^{(j)}\mathbf{x}_{t-p(j)} + \boldsymbol{\varepsilon}_t^{(j)}, \tag{3}$$

where $p(j)$ is the order of the VAR model, i.e. the number of included lags in regime *j*. Further, the matrix of transition probabilities, $\mathbf{P} = \{p_{ij}\}$, where $i, j = 1, \ldots, s$ is determined by the equations:

$$p_{ij} = \Pr[S_t = j \mid S_{t-1} = i], \tag{4}$$

where the state variable S_t denotes the regime prevailing at time *t*.

This Markov switching model can be estimated via maximum likelihood as described in Hamilton (1994). A well-known problem of regime switching models is that the likelihood surface is multimodal. For this reason, a strategy of how to be able to re-

port the global maximum should be advised. Our solution to this problem is to use simulated annealing to maximize the likelihood function. Simulated annealing is a derivative-free stochastic search algorithm that, in contrast to conventional gradient based algorithms, is able to escape from local maxima and hence is very well suited for estimation of regime switching models.[3]

As the regimes are unobservable, S_t can be regarded as missing data. However, the probability that a given observation belongs to a particular regime can be computed. From this information, we can construct an optimal forecast probability that the next observation belongs to a particular regime.

Let $\Pr_t(S_t = j)$ denote the updated (filtered) probability that $S_t = j$, given the information set $\mathbf{X}_t$. The forecast probabilities for time t+1, given the information available at time t, are denoted $\Pr_t(S_{t+1} = j)$. Following Hamilton (1994), let $\boldsymbol{\xi}_{t|t}$ denote the vector of updated probabilities and $\boldsymbol{\xi}_{t+1|t}$ the vector of forecast probabilities. The time t likelihood function value is obtained as a by-product from the following iterations to calculate the optimal forecast probabilities:

$$\text{(5a)} \qquad \boldsymbol{\xi}_{t|t} = \frac{\boldsymbol{\xi}_{t|t-1} \circ \boldsymbol{\varphi}_t}{\mathbf{1}'\left(\boldsymbol{\xi}_{t|t-1} \circ \boldsymbol{\varphi}_t\right)},$$

$$\text{(5b)} \qquad \boldsymbol{\xi}_{t+1|t} = \mathbf{P}'\boldsymbol{\xi}_{t|t},$$

where $\boldsymbol{\varphi}_t$ is the vector of conditional densities, $\mathbf{1}$ is a unit column vector and $\circ$ denotes (vector) element-by-element multiplication. This filter is proposed in Hamilton (1989) and can be thought of as a non-linear Kalman filter, where the probabilities are the state

variables. The filter outputs an optimal inference about the predicted and updated probabilities through one set of prediction equations and another set of updating equations. The time *t* likelihood function is given by the denominator in (5a). In words, the likelihood function is the weighted sum of the conditional densities, with weights given by the forecast probabilities.

2.1.1 Forecasting Method

For the RS-VAR model, the conditional $t+1$ forecast of $\mathbf{x}$ is the weighted average of the *s* different forecasted within regime means, with weights given by the probability of each regime to prevail in the next period, i.e.:

$$(6) \qquad E_t(\mathbf{x}_{t+1}) = \sum_{j=1}^{s}\sum_{i=1}^{s} \Pr_t(S_t = j) p_{ij}\left(\boldsymbol{\alpha}_0^{(j)} + \sum_{\ell=1}^{p(j)} \mathbf{A}_\ell^{(j)} \mathbf{x}_{t-\ell+1}\right),$$

for each *t* using the information set $\mathbf{X}_t = \{\mathbf{x}_t, \mathbf{x}_{t-1}, ...\}$. In an *s*-regime model there are s^τ possible outcomes for each dependent variable τ periods ahead and s^τ associated probabilities. Hence, the $t+\tau$ forecast of $\mathbf{x}$ is calculated as the probability weighted average by traversing the non-recombining tree generated by the underlying Markov-chain along all possible paths. Note that the probability trees should be multiplied by the *s* updated probabilities according to (6). This reflects the uncertainty of the prevailing regime today, or, econometrically, the fact that the Markov chain is unobserved even *ex post*. [4]

2.2 Recurrent Neural Networks

Recurrent neural networks based on gradient descent learning[5] are typically adaptations of the traditional feed-forward multi-layered perceptron (FF-MLP) trained with the back-

propagation algorithm[6] (Rumelhart *et al*., 1986). This particular class of RNN extends the FF-MLP architecture to include recurrent connections that allow network activations to feedback (as internal input at subsequent time steps) to units within the same or preceding layer(s). Units that receive such feedback values are referred to as *context* or *state* units.

This internal memory enables the RNN to construct internal representations of temporal order and dependencies since the temporal structure embedded within the data will be encoded within the spatial structure of the RNN. This is due to the sequence of weights to each unit connected to the input layer being convolved with different sequences of input examples (Haykin, 1999). Assuming non-linear activation functions are used, the universal function approximation properties of FF-MLPs naturally extends to RNNs.

The pattern of recurrent connectivity strongly influences the computational power of RNNs where certain classes of RNNs have been shown to theoretically simulate Turing machines (Siegelmann and Sontag, 1991). More specifically, it has been shown that RNNs employing recurrent connections from the output layer back to the input layer (see Jordan, 1986) are analogous to infinite impulse response filters (IIRs) (Khan and Unal, 1995). Likewise, those RNNs that contain recurrent connections from the hidden layer back to the input layer, such as Elman's Simple Recurrent Networks (SRNs) (Elman, 1990) are similar in computational power to Hidden Markov Models (HMMs) (Lee, 1989). More generally, RNNs are considered dynamical systems that can represent at least auto-regressive with moving average (ARMA) estimators (Connor and Martin, 1994).

RNNs can be expressed generally as (modified from Chappelier *et al*., 2000):

(7a) $\mathbf{r}_t = f(\mathbf{r}_{t-1}, \mathbf{x}_t, t, \Theta_t^f)$,

(7b) $\mathbf{y}_t = g(\mathbf{r}_t, t, \Theta_t^g)$,

where **r** represents the context vector (recurrent variables); **y** refers to the activation vector on the output layer (regressands); **x** is the input vector (input variables or regressors); *t* refers to the current time step; Θ_t^f refers to the set of weights connecting the input layer to the hidden layer at time *t*; and Θ_t^g refers to the set of weights connecting the hidden layer to the output layer at time *t*; finally, functions *f* and *g* represent the activation vectors from the hidden and output layers respectively.

For the purposes of this paper, we use an RNN architecture that combines the relative strengths of the Jordan network (Jordan, 1986) with Elman's SRN (Elman, 1990) to form a hybrid RNN that feeds both the hidden unit activations and output unit activations back to the context units of the input layer. This is shown in Figure 1. To reduce complexity and the number of parameters, our model does not utilize a recurrent input layer or self-loops.

[FIGURE 1 ABOUT HERE]

We also use a common extension of the backpropagation algorithm for training RNNs called *backpropagation-through-time* (BPTT) (Rumelhart *et al.*, 1986; Werbos, 1990; Williams and Peng, 1990), which has proven more powerful than standard backpropagation for training RNNs such as Jordan and Elman networks. BPTT works under

the assumption that for every recurrent MLP network, a feedforward MLP with identical behavior can be obtained by 'unfolding the network over time', i.e. a recurrent network state at time *t* can be viewed as though it was obtained from the *t:*th layer output of a corresponding FF-MLP network with *T* layers (Rumelhart *et al.*, 1986), where *T* is the length of the sequence. When at the end of a sequence or *h* patterns have been propagated through the network, the external error is calculated and the errors and local gradients are backpropagated through the network from *n* until the initial time step. During this back-error propagation phase, an additional error term for recurrent connections can be introduced to emphasize unit-error contributions from subsequent time steps. Each individual weight change is then calculated as in standard backpropagation but applied and summed across all time steps resulting in one large weight change calculation for each weight. The weights are then adjusted as normal. In the case of *epoch-wise* BPTT (Williams and Peng, 1990) the context units are reset to some initial value at the beginning of each sequence. Due to these reset operations, we implement a discrete dynamical RNN as opposed to a continually running one.

The equation for the hybrid Jordan-Elman RNN used in this work can be expressed generally as (modified from equations (7a) and (7b) above):

(8a) $\mathbf{r}_t = f(\mathbf{c}_t, \mathbf{x}_t, \Theta_t^f)$,

(8b) $\mathbf{y}_t = g(\mathbf{r}_t, \Theta_t^g)$,

where $\mathbf{r}_t$ now represents the hidden unit activation vector at time *t*; $\mathbf{c}_t$ refers to the concatenation of the previous hidden state vector, $\mathbf{r}_{t-1}$, and the previous external output re-

sponse vector, $\mathbf{y}_{t-1}$; $\mathbf{y}_t$ refers to the external output activation vector at time *t*; as illustrated in Figure 1, $\mathbf{x}_t$ refers to the external vector of input variables at time *t*; Θ_t^f refers to the set of weights connecting the input layer to the hidden layer; and Θ_t^g refers to the set of weights connecting the hidden layer to the output layer; finally, functions, as before, *f* and *g* represent the activation vectors from the hidden and output layers, respectively.

2.2.1 Forecasting Method

When applying the selected RNN method to our forecasting task we follow the common practice of: i) normalizing the input variables to accelerate the learning process by avoiding time consuming trajectories across the error surface caused by groups of successive observations sharing the same sign and thus direction; ii) generating the training (or estimation) data from some in-sample using a sliding window of a pre-determined time-lag; iii) using the volume of training data to set an upper-limit on the number of allowable free parameters (weights) and thus to constrain network size in a way that will reduce over-fitting and subsequently increase generalization ability; iv) implementing an appropriate training regime with a learning rate schedule that encourages fast, stable and convergent learning; v) defining a suitable stopping criteria; and vi) using the sliding window technique to generate forecasts of the required forecast horizon.

The fundamental purpose of the training phase is for the RNN to learn to forecast (on its output vector **y**) each value of the dependent variables in **x** at time *t*+1 given a sequence of input vectors $\mathbf{X}_t = \{\mathbf{x}_t, \mathbf{x}_{t-1}, ...\}$.

We use the hyperbolic tangent function for calculating the activation level for all processing units (hidden and output units) of the network. The hyperbolic tangent function is simply the logistic function rescaled and biased with user defined constants. For

training the RNNs, we employ a '*search and converge*' learning rate regime as defined by Darken and Moody (1991):

$$(9) \qquad \eta = \frac{\eta_0}{\frac{n}{N/2} + \frac{c_1}{\max(1, (c_1 - \frac{\max(0, c_1(n - c_2 N))}{(1 - c_2)N}))}},$$

where η is the learning rate, η_0 is the initial learning rate which we set to 0.01, N is the total number of training epochs, n refers to the current training epoch, c_1 and c_2 are user defined constants. This provides an effective time varying learning rate constant that guarantees convergence of stochastic approximation algorithms and has proven effective for temporal domains (for example, see Lawrence *et al.*, 2000). The momentum term constant is fixed at 0.9 due to the low learning rates. Note that since our model is a discrete-time RNN, the context units ($\mathbf{c}_t$) must be reset to some initial value after each $\mathbf{x}_{t+1}$ forecast during training (estimation) and testing (forecast evaluation).

For generating dynamic forecasts of each dependent variable τ periods ahead, we maintain two input streams: stream A containing all real observations found in the forecast evaluation (test) set and stream B consisting of the first $t - \tau$ real observations found in the training set, where t initially refers to the first observation to be forecast in the evaluation (test) sample. Stream B will grow dynamically over time as it accommodates the RNNs t+1 forecasts. We first initialize the RNNs context units and then using the standard sliding window technique we pass the first τ observations in stream B through the network, generating dynamic observations at each step and appending them to stream B. Clearly, the first $t + \tau$ dynamic forecast cannot be generated until the last real observation in stream B is in the first position of the sliding window (i.e. becomes $\mathbf{x}_t$). From

thereon both streams are synchronously indexed. Dynamic $t+\tau$ forecasts are generated when the window of input observations becomes $\hat{\mathbf{X}}_t = \{\mathbf{x}_t, \mathbf{x}_{t+1} = \mathbf{y}_t, ..., \mathbf{x}_{t+\tau-1} = \mathbf{y}_{t+\tau-2}\}$, where $\mathbf{x}_t$ is always a real observation extracted from stream A. Note that each element of $\hat{\mathbf{X}}_t$ is presented sequentially to the RNN. We continue processing in this way until all observations in stream A have been represented in $\mathbf{x}_t$.

3. Data and Estimation Results

We obtain a monthly data set that covers June 1969 to October 2003 from EcoWin, yielding a total of 413 observations. The three time-series we obtain are i) the Retail Price Index (*P*); ii) M0 (*M*); and iii) Industrial Production (*Y*).[7] Each of these series is logarithmically transformed and differenced, so that $\mathbf{x}_t' = (dP_t, dM_t, dY_t)$. For the RNN, all observations for each of the dependent input variables are independently normalized such that each variable in **x** has a zero mean and a unit standard deviation. The data is shown in Figure 2.

[FIGURE 2 ABOUT HERE}

The data up to October 1998 is used for estimation (training). The last five years of the data (November 1998 – October 2003; 60 observations) is used for forecast evaluation.

To limit the scope of our study somewhat and to facilitate comparison of forecasts from the different models, we use a fixed lag-length of 12 for our two vector autoregressive models and for the RNNs, a temporal window containing 12 observations such that

each observation within the window is presented sequentially (rather than simultaneously) to the RNNs input layer.

Before turning to a formal comparison of the forecasting performance, we briefly present the estimation results for the different models. The estimated dynamic structure of the price equation in the VAR(12) model is visualized in Figure 3. The parameters are depicted jointly with their two standard error bands.

[FIGURE 3 ABOUT HERE]

As can be seen, the estimated coefficient for the constant term is insignificant. The first lag of inflation is (relative to remaining coefficients) large and highly significant. We also find various higher lags of inflation as well as changes in M0 and industrial production to be significant. Finally, it can be verified that the estimated model satisfies the theoretical stability restrictions.

3.1 RS-VAR

In the RS-VAR framework, we find that allowing for a different intercept in each regime, while leaving the dynamic structure unchanged across regimes, provides the best models for inflation forecasting purposes. When allowing for a different dynamic structure in different regimes, the forecasting performance deteriorates significantly, which may be interpreted as a classical case of in-sample over-fitting. This result is in accordance with Krolzig (2004), who argues that forecast errors of economic time series from linear models are mainly due to shifts in the level or drift. Hence, we only report estimation (and

forecasting) results for a two-regime RS-VAR(12), with a different intercept and the same dynamic structure in each regime.

[FIGURE 4 ABOUT HERE]

The estimated dynamic structure of the price equation in the RS-VAR model can be found in Figure 4. The point estimate of the intercept in the second regime is obviously much higher than in the first regime, although both are statistically insignificant. Turning to the lag structure, we find that the first, sixth and twelfth lag of inflation is significant as well as the fourth, fifth and ninth lag of changes in M0 and the fifth lag of industrial production. This pattern is very similar to the one found for the linear VAR model (Figure 3).

[TABLE 1 ABOUT HERE]

The estimated probabilities for remaining in each regime are presented in Table 1. The expected duration for each regime is determined by $(1-p_{jj})^{-1}$ where p_{jj} is the probability of remaining in regime *j* once there. For the preferred two-regime model, both regimes are relatively persistent, with expected durations of about 47 and 23 months, for regime one and regime two, respectively.

[FIGURE 5 ABOUT HERE]

Time-series plots of the updated probabilities can be found in Figure 5. The probabilities attached to regime one and two vary (in principle) between zero and one. As is evident from Table 1, volatility is higher in the second regime for both inflation, money growth and growth in industrial production. This is consistent with the time-series plots of the updated probabilities, in which it can be seen that the second regime is dominating up until the beginning of the eighties, while the first regime has dominated since then. In other words, the second regime seems to capture the period of high inflation and high inflation volatility during the first, say, 12 years of the sample, while the first regime captures the period of lower inflation and lower inflation volatility during the next 20 years. However, one must remember that we simultaneously model inflation, money growth and growth in industrial production, which means that the regimes must be interpreted as different regimes of the economy, rather than different regimes of inflation (only).

3.2 RNN

All RNN models contain three input units (to present $\mathbf{x}_t$ to the processing units) and three output units (to forecast $\mathbf{x}_{t+1}$). The number of hidden units is empirically established.

Using the sliding window technique to generate training sequences from the in-sample observations we obtain a training set consisting of the following total number of patterns:

(10) $$z = \frac{q-m}{m},$$

where z refers to the total number of input pairs (a pair refers to an input vector $\mathbf{x}$ and an output vector encoding the desired response, i.e. values of $\mathbf{x}$ at the next time period), q is the total number of observations in the time series and m is the window size. The resulting number of training sequences (collections of training pairs) is simply:

$$v = \frac{z}{m}. \tag{11}$$

For our purposes, q=352 and m=12, thus z=4080 and v=340.

The number of patterns generated (z) form a basis for our upper limit constraint on the number of allowable parameters (weights) and thus on the number of hidden units allowed. The architecture is therefore fixed in accordance with the data and with the complexity of the underlying problem. We expect over-fitting to occur as the number of free parameters approaches z, i.e. 4,080 (e.g. 60 hidden units or above).

For all training (or estimation) experiments weight initializations were in the range of [-0.1, 0.1]. Our stopping condition is a function of the root mean squared error (*RMSE*) and number of training cycles performed. A training cycle is where the network has processed all training patterns and sequences once (a single pass of the whole data through the network) and is referred to as an epoch. An RNN is stopped training when either the number of epochs reaches 2,000 or the rate of change of the *RMSE* is sufficiently small, whichever comes first. During each epoch, successive sequences are randomly selected to encourage a wider search of the weight space. The standard summed squared error (quadratic cost) function was used for all error calculations.

After performing a number of training experiments with various hidden unit configurations (below 60) we found that an RNN configuration with 50 hidden units, RNN(50), and 3,003 free parameters (26% fewer than available training cases) provides

an optimum fit for the in-sample data with respect to subsequent dP_{t+1} performance on the evaluation forecast data (generalization). After 2,000 epochs, the RNN(50) obtained an *RMSE* of 5.0093 for the whole in-sample data set and an *RMSE* of 0.2826 for the *dP* observations.

We also assessed an RNN with more free parameters than there were training cases to confirm our intuitions that classic over-fitting would naturally occur. After 2,000 epochs, an RNN with 100 hidden units, RNN(100), and 11,003 free parameters (170% more than available training cases) obtained a tighter fit of the whole in-sample data set with an *RMSE* of 4.2246. As expected, it obtained a tighter fit (approximately 26% closer) for the in-sample *dP* observations with an *RMSE* of 0.2102 at the expense of yielding poorer generalization performance for dP_{t+1} forecasts.

4. Forecast Evaluation

Turning to a formal comparison of the forecasts, we use three common forecast evaluation criteria; mean errors (*ME*), root mean squared errors (*RMSE*), and mean absolute errors (*MAE*). *ME* can give an indication as to whether the forecast is biased. *RMSE* is the most frequently used measure, while *MAE* is known to be less sensitive to outliers. Let:

$$ME = \frac{1}{K} \sum_{t=1998:10+\tau}^{2003:10} \left(E_{t-\tau}[dP_t] - dP_t \right), \tag{12a}$$

$$MAE = \frac{1}{K} \sum_{t=1998:10+\tau}^{2003:10} \left| E_{t-\tau}[dP_t] - dP_t \right|, \tag{12b}$$

$$RMSE = \left[\frac{1}{K} \sum_{t=1998:10+\tau}^{2003:10} \left(E_{t-\tau}[dP_t] - dP_t \right)^2 \right]^{\frac{1}{2}}, \tag{12c}$$

where K is the total number of out-of-sample forecasts and τ is the forecast horizon. In our specific study K=60 or 49, depending on whether τ equals one or twelve. We complement the raw measures in (12b-c) by presenting ratios of the *MAE* and *RMSE* for each model to that of the best performing model.

4.1 One-month ahead forecasts

We present the t+1 forecasts as a 2×2-panel in Figure 6. Forecasts (black line) are compared to actual inflation (gray line) in each panel.

[FIGURE 6 ABOUT HERE]

We immediately note that the VAR-model performs reasonably well in comparison with the non-linear models. It tends, however to overshoot large increases in monthly inflation on occasions. The forecasts from the RS-VAR model are quite similar to those obtained from the VAR-model. We note, however, that there are fewer tendencies to overshoot large changes in inflation compared to the standard VAR. Although the pattern of the forecasts obtained from the RNN(50) and RNN(100) models differs from the forecasts obtained by the two VAR-models, these models appear to give good forecasts of inflation as well. An interesting observation is that all models poorly capture the fall in the price level in January and July up to 2001.

[TABLE 2 ABOUT HERE]

As is evident from the calculated *ME* in Table 2, all models but RNN(50) are slightly biased upwards. For the two VAR-models and the RNN(100) model, the bias is approximately 0.1 percent, but substantially lower for the RNN(50) model. Turning next to *RMSE* and *MAE*, both criteria rank the different models under evaluation similarly. The RNN(50) model and the RS-VAR model are the best models. They perform approximately as well, with the RS-VAR model yielding slightly lower values for both *MAE* as well as *RMSE*. The RNN(100) model is the worst performing model, yielding higher values of both *MAE* and *RMSE* than are obtained from the linear VAR model. In fact, looking at ratios of the forecast-evaluation criterion, it performs up to 33% worse than the best performing RS-VAR model.

4.2 One-year ahead forecasts

We present the t+12 forecasts as a 2×2-panel in Figure 7. As before, forecasts (black line) are compared to actual inflation (gray line) in each panel.

[FIGURE 7 ABOUT HERE]

A noticeable difference between the two VAR models and the two RNNs is that forecasts from the VAR models appear to be biased upwards. This tendency seems to be more pronounced compared with the one-month ahead forecasts.

[TABLE 3 ABOUT HERE]

The fact that the VAR-models overshoot actual inflation for the one-year ahead forecasts is confirmed by the *ME* criterion presented in Table 3. The RNN(50) undershoots actual inflation. The RNN(100) still overshoots inflation, although slightly less than was the case on a one-month ahead basis. The RNN(100) model is now, judging from *MAE* and *RMSE* the best forecaster. The second best is the RS-VAR model followed by the RNN(50) model. The VAR model performs significantly worse than the other models.

Finally, we observed an interesting behavior when comparing RNN(50) and RNN(100) dP_{t+12} dynamic forecasts. Contrary to our intuitions, we found that the RNN(100) yielded superior performance with an *RMSE* of 0.3018, 9% better than that generated by the RNN(50) (i.e. an *RMSE* of 0.3332). This is a good example of the un-predictable nature of RNNs and further research is required to assess which factors within its complex internal dynamics is causing such perturbations in expected behavior. We are particularly interested in the approach taken by Lawrence *et al.* (2000) and Giles *et al.* (2001), for extracting discrete finite state automata from RNNs to help understand its be-havioral patterns and intend to pursue this further.

5. Conclusions

In this study, we have compared the forecasting performance of two non-linear models, a RS-VAR model and a RNN model, with that of a benchmark linear VAR model. Our specific forecast experiment is UK inflation. We find that the RNN model and RS-VAR model outperform the VAR model for both monthly and annual forecast horizons. The RS-VAR and the RNN perform approximately on par over both forecast horizons. For the RS-VAR model, we find that imposing the restriction that only the intercept is allowed to

vary across regimes provides the best forecasts. For the RNN-model, the forecasting performance depends on the number of hidden units and thus free parameters included.

References

Ang, A., Bekaert, G., 2002. International Asset Allocation with Regime Shifts. Review of Financial Studies 15, 1137--1187.

Binner, J.M., Gazely, A.M., Chen, S.H., 2002. Financial innovation and Divisia monetary indices in Taiwan: a neural network approach. European Journal of Finance 8, 238--247.

Blix, M., 1999. Forecasting Swedish Inflation with a Markov Switching VAR. Sveriges Riksbank Working Paper, No 76.

Chappelier, J-C., Gori, M., Grumbach, A., 2000. Time in Connectionist Models, in: Sun, R., Giles, L. (Eds.), Sequence Learning: Paradigms, Algorithms, and Applications. Springer Verlag, pp. xxx--yyy.

Cheng, B., Titterington, D.M., 1994. Neural networks: a review from a statistical perspective. Statistical Science 9, 2--54.

Connor, J., Martin, R., Atlas, L., 1994. Recurrent Neural Networks and Robust Time Series Prediction. IEEE Transactions on Neural Networks 5, 240--254.

Corona, A., Marchesi, M., Martini, C., Ridella, S., 1987. Minimizing Multimodal functions of Continuous Variables with the 'Simulated Annealing' Algorithm. ACM Transactions of Mathematical Software 13, 262--280.

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems 2, 303--314.

Darken, C., Moody, J.E., 1991. Note on learning rate schedules for stochastic optimization, in: Lippmann, R.P., Moody, J.E., Touretzky, D.S., (Eds.). Advances in Neural Information Processing Systems 3, 832--838.

Elman, J., 1990. Finding Structure in Time. Cognitive Science 14, 179--211.

Enders, W., 1995. Applied Econometric Time Series. John Wiley and Sons.

Fishman, G., 1996. Monte Carlo Concepts, Algorithms and Applications. Springer Verlag.

Franses, P.H., van Griensven, K., 1998. Forecasting exchange rates using neural networks for technical trading rules. Studies in Nonlinear Dynamics and Econometrics 2, 109--116.

Franses, P.H., van Homelen, D., 1998. On forecasting exchange rates using neural networks. Applied Financial Economics 8, 589--596.

Garcia, R., Perron, P., 1996. An Analysis of the Real Interest Rate under Regime Shifts. Review of Economics and Statistics 78, 111--125.

Gençay, R., 1996. Non-linear prediction for security returns with moving average rules. Journal of Forecasting 15, 165--174.

Gençay, R., 1999. Linear, non-linear and essential foreign exchange rate predictions with simple technical trading rules. Journal of International Economics 47, 91--107.

Gençay, R., Stengos, T., 1998. Moving average rules, volume and the predictability of security returns with feedforward networks. Journal of Forecasting 17, 401--414.

Giles, C.L., Lawrence, S., Tsoi, A.C., 2001. Noisy time series prediction using a recurrent neural network and grammatical inference. Machine Learning 44, 161--183.

Goffe, W., Ferrier, G., Rogers, J., 1994. Global Optimization of Statistical Functions with Simulated Annealing. Journal of Econometrics 60, 65--99.

Goldfeld, S., Quandt, R., 1973. A Markov Model for Switching Regressions. Journal of Econometrics 1, 3--16.

Haefke, C., Helmenstein, C., 1996a. Forecasting Austrian IPOs: an application of linear and neural network error-correction models. Journal of Forecasting 15, 237--251.

Haefke, C., Helmenstein, C., 1996b. Neural networks in the capital markets: an application to index forecasting. Computational Economics 9, 37--50.

Hamilton, J., 1988. Rational-Expectations Econometric Analysis of Changes in Regime: An Investigation of the Term Structure of Interest Rates. Journal of Economic Dynamics and Control 12, 385--423.

Hamilton, J., 1989. A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle. Econometrica 57, 357--384.

Hamilton, J., 1994. Time Series Analysis. Princeton University Press.

Haykin, S., 1999. Neural Networks: A Comprehensive Foundation, 2nd edition. Prentice Hall, Upper Saddle River, NJ.

Hornik, K., 1991. Approximation Capabilities of Multilayer Feedforward Networks. Neural Networks 4, 251--257.

Jordan, M., 1986. Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. Proceedings of the 8th Annual Conference of the Cognitive Science Society, 531--545.

Khan, E., Unal, F., 1995. Recurrent fuzzy logic using neural networks, in: Furuhashi, T. (Ed.), Advances in fuzzy logic, neural networks, and genetic algorithms, Lecture Notes in AI. Springer Verlag, pp. xxx--yyy.

Klaassen, F., 2002. Improving GARCH volatility forecasts with regime-switching GARCH. Empirical Economics 27, 363--394.

Krolzig, H-M., 2004. Predicting Markov-Switching Vector Autoregressive Processes. Working Paper, Nuffield College, Oxford.

Kovalerchuk, B., Vityaev, E., 2000. Data Mining in Finance: Advances in Relational and Hybrid Methods. Kluwer.

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

Lawrence, S., Giles, C.L., Fong, S., 2000. Natural language grammatical inference with recurrent neural networks. IEEE Transactions on Knowledge and Data Engineering 12, 126--140.

Lee, K., 1989. Hidden Markov Models: Past, Present and Future. Proceedings of Eurospeech 89, 148--155.

Lütkepohl, H., 1991. Introduction to Multiple Time Series Analysis. Springer Verlag.

Moshiri, S., Cameron, N., Scuse, D., 1999. Static, dynamic and hybrid ANN models in forecasting inflation. Computational Economics 14, 219--235.

Qi, M., Maddala, G.S., 1995. Option pricing using ANN: the case of S&P 500 index call options. Neural Networks in Financial Engineering: Proceedings of the 3rd International Conference on Neural Networks in the Capital Markets, 78--91.

Pearlmutter, B.A., 1995. Gradient calculations for dynamic recurrent neural networks: a survey. IEEE Transactions on Neural Networks 6, xxx--yyy.

Robert, P.C., Casella, G., 1999. Monte Carlo Statistical Methods. Springer-Verlag.

Rumelhart, D., Hinton, G., Williams, R., 1986. Learning Internal Representations by Error Propagation, in: Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Vol. 1. Princeton University Press, pp. 318--362.

Shadbolt, J., Taylor, J.G., 2002. Neural Networks and the Financial Markets: Predicting, Combining and Portfolio Optimisation (Perspectives in Neural Computing). Springer Verlag UK.

Siegelmann, H.T., Sontag, E.D., 1991. Turing computability with neural nets. Applied Mathematics Letters 4, 77--80.

Stock, J.H., Watson, M.W., 1999. A comparison of linear and non-linear univariate models for forecasting macroeconomic time series, in: Engle, R., White, R. (Eds.), Cointegration, causality and forecasting: A festschrift in honor of Clive W.J. Granger. Oxford: Oxford University Press, pp. 1--44.

Swanson, N.R., White, H., 1995. A model selection approach to assessing the information in the term structure using linear models and artificial neural networks. Journal of Business & Economic Statistics 13, 265--275.

Tenti, P., 1996. Forecasting foreign exchange rates using recurrent neural networks. Applied Artificial Intelligence 10, 567--581.

Werbos, P.J., 1990. Backpropagation through time: What it does and how to do it, in: Proceedings of IEEE Special Issue on Neural Networks, Vol. 78, pp. 1550--1560.

Williams, R.J., Peng, J., 1990. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. Neural Computation 2, 490--501.

Figure 1: The Jordan-Elman hybrid network architecture.

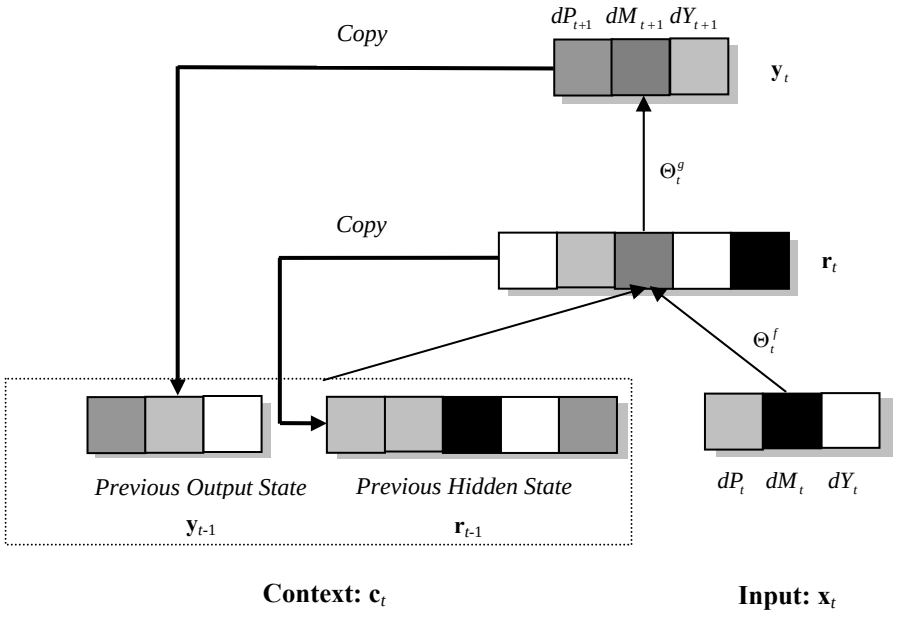

Figure 2: The log-differenced time series.

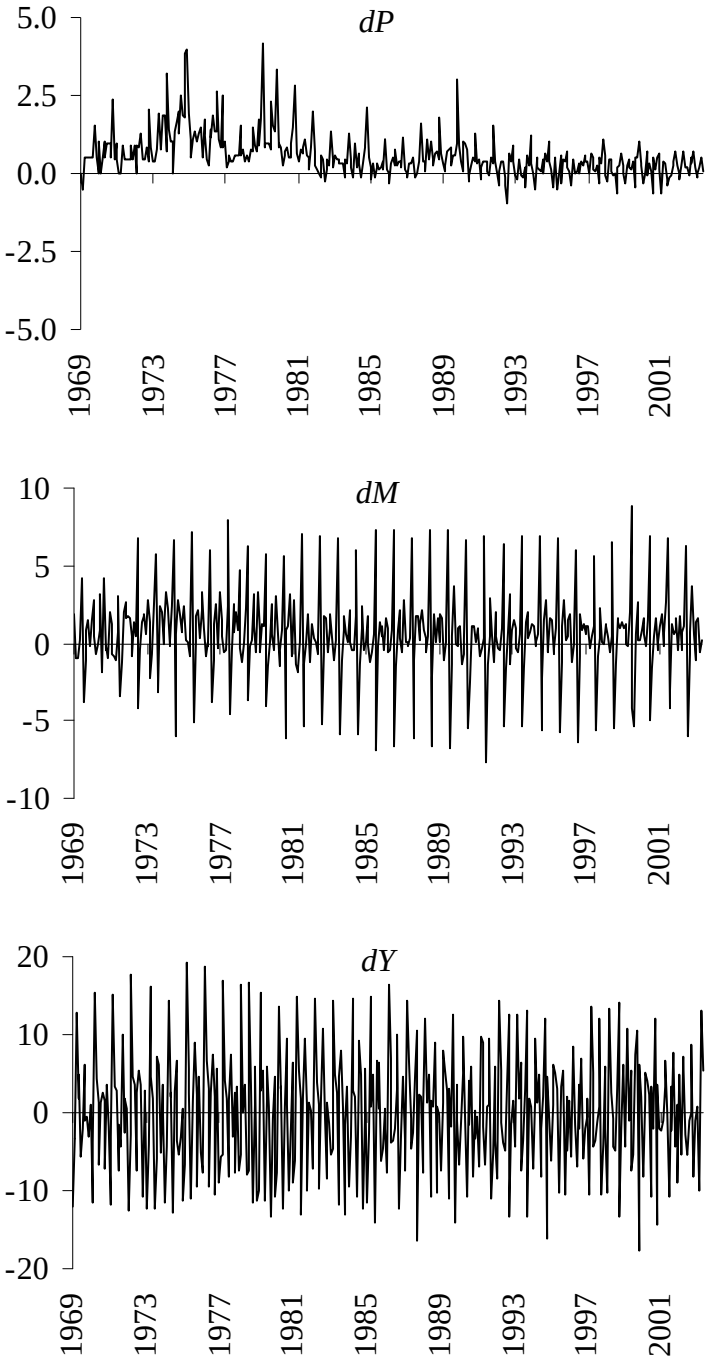

Figure 3: Estimated coefficients for price equation in VAR-model with two standard error bands.

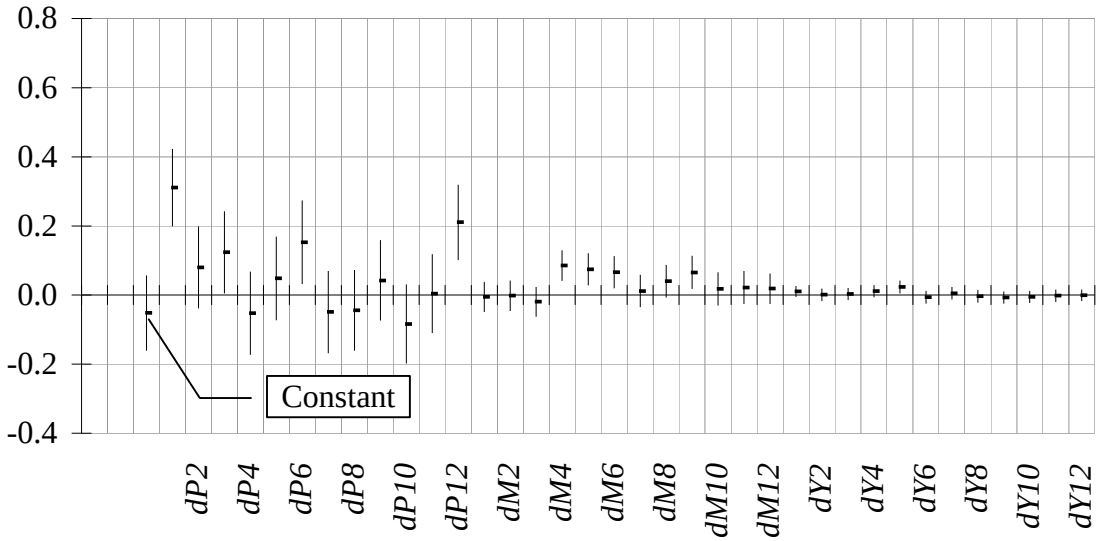

Figure 4: Estimated coefficients for price equation in RS-VAR-model with two standard error bands.

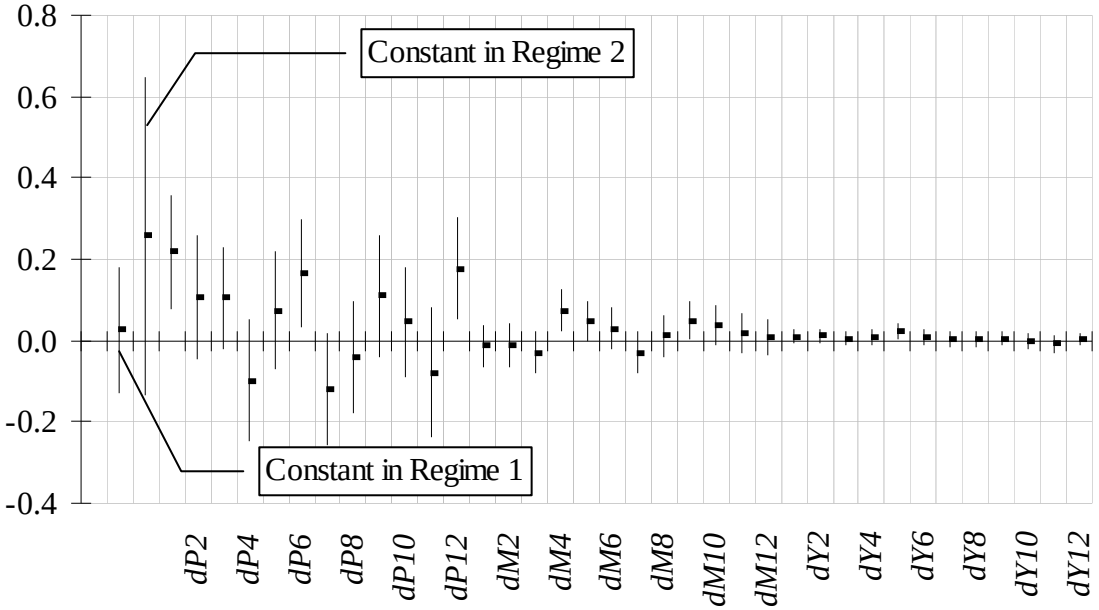

Figure 5: Filtered probabilities of regimes in RS-VAR-model.

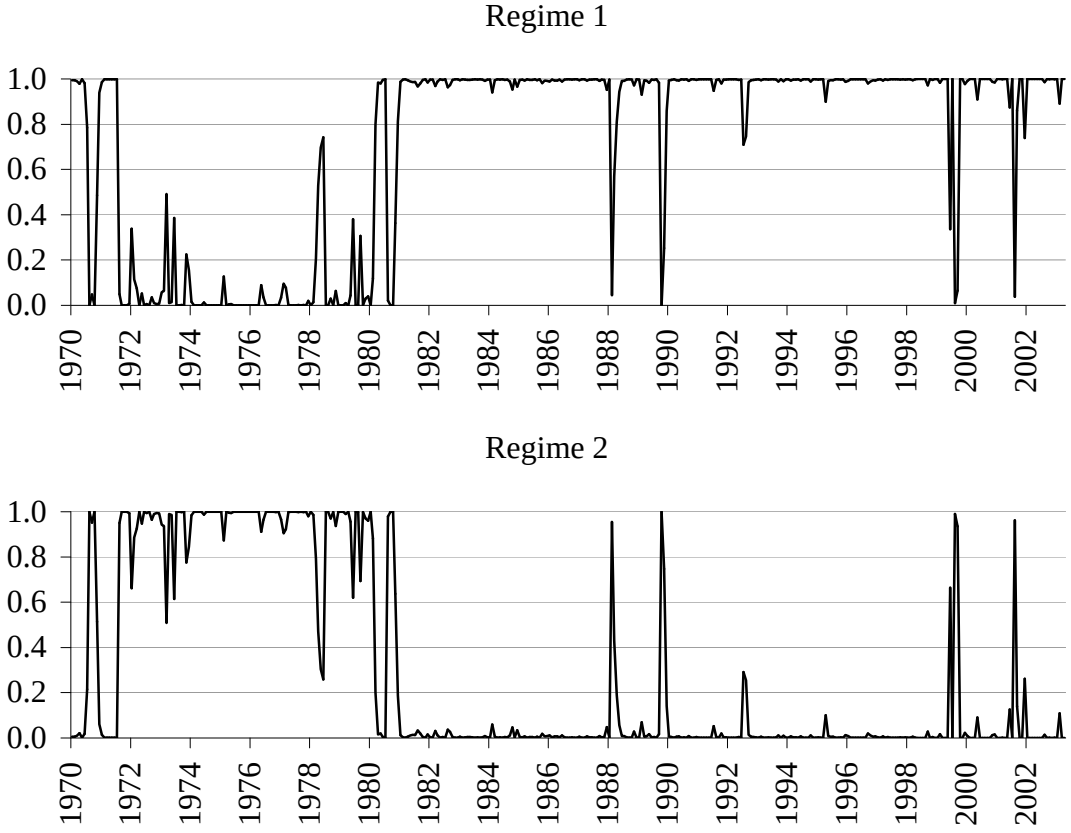

Figure 6: Forecasted (*t*+1) and actual values of inflation.

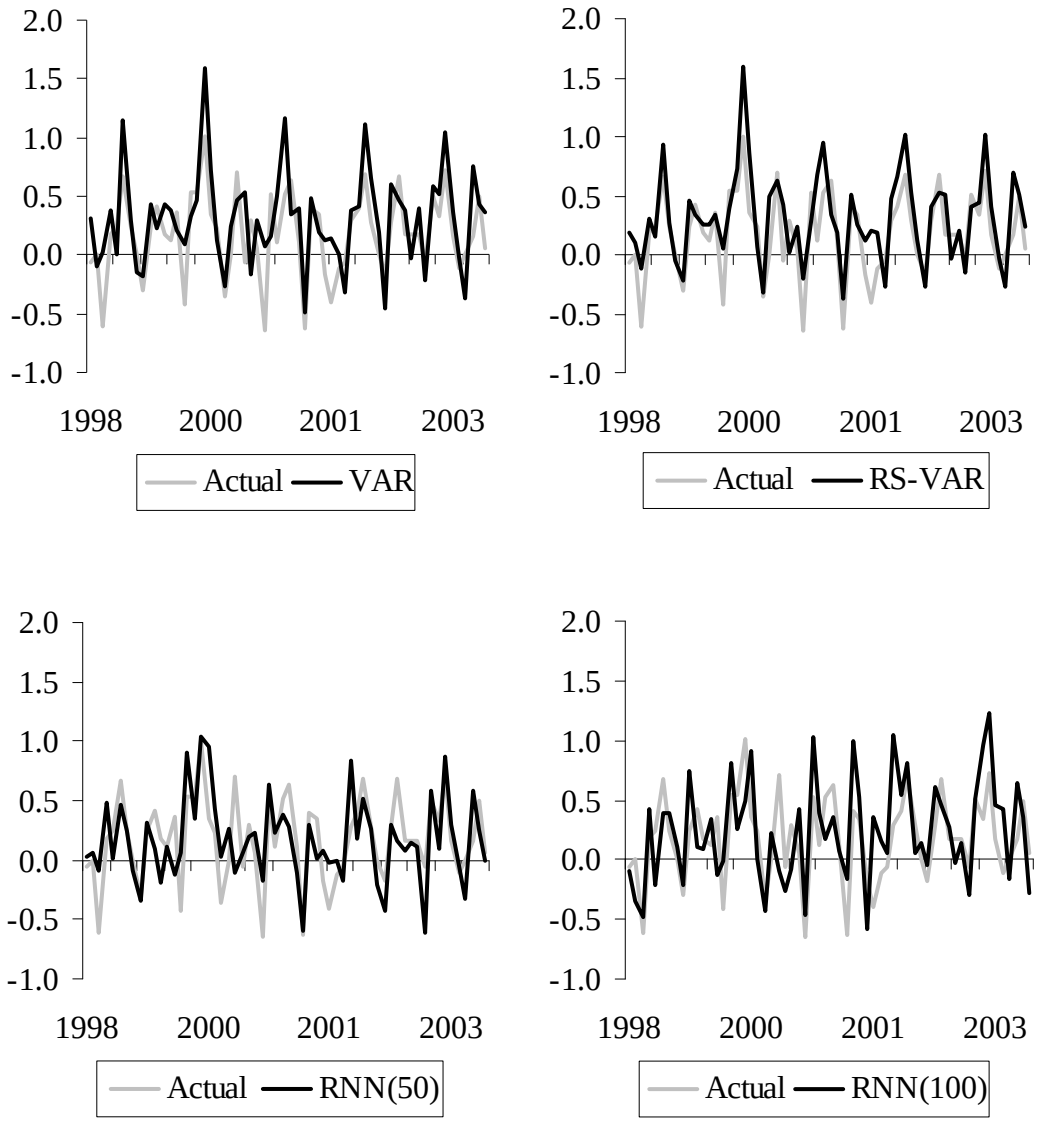

Figure 7: Forecasted (*t*+12) and actual values of inflation.

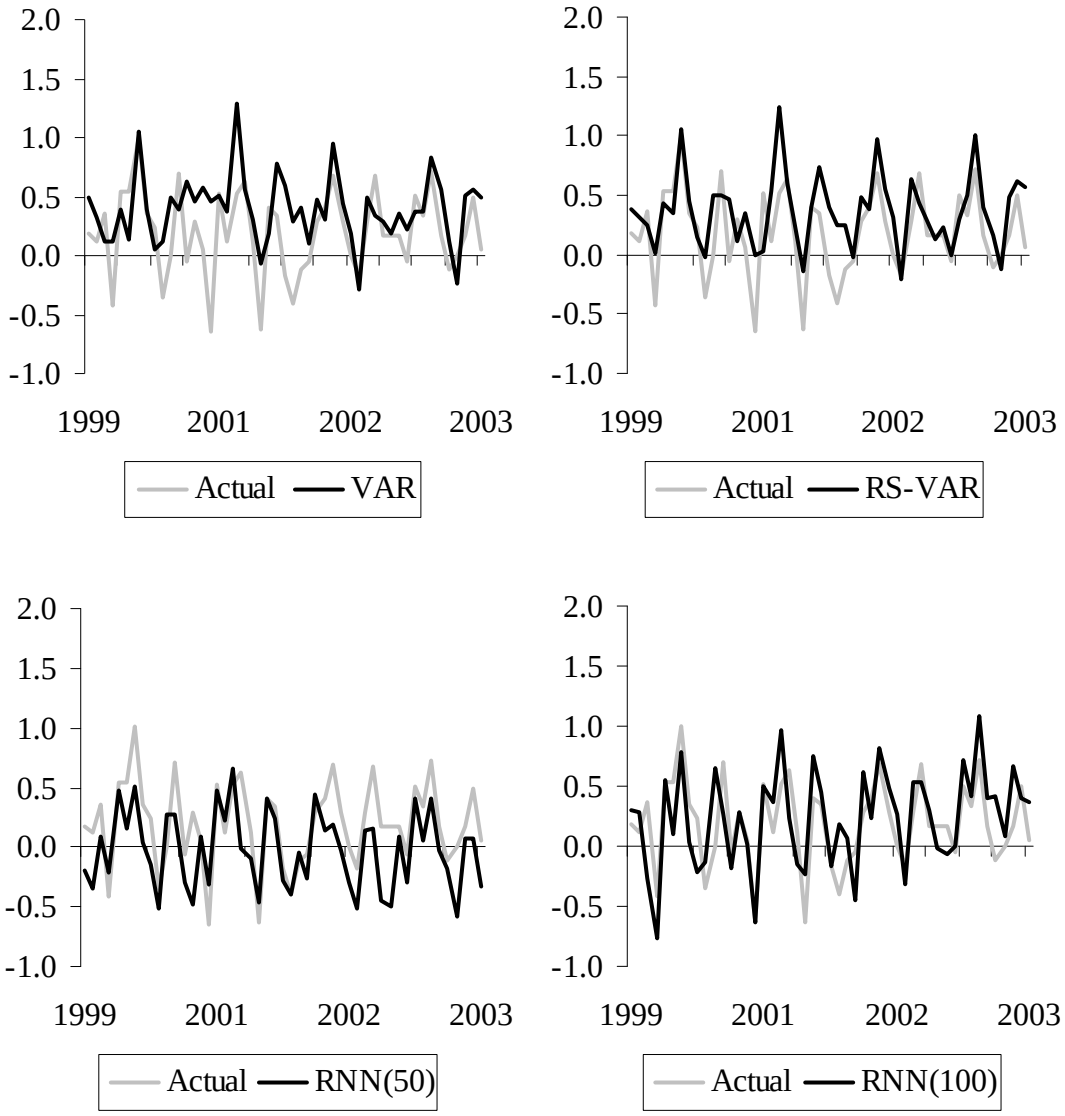

Table 1: Estimates of transition probabilities and volatilities.

	p_{jj}	σ_{jdP}	σ_{jdM}	σ_{jdY}
Regime 1	0.9787	0.2295	0.6614	2.2302
	(0.0130)	(0.0138)	(0.0457)	(0.1551)
Regime 2	0.9567	0.6680	1.3194	3.7160
	(0.0296)	(0.0443)	(0.1114)	(0.3669)

Note: Standard errors in parentheses.

Table 2: Evaluation criteria for *t*+1 forecasts.

	VAR	RS-VAR	RNN(50)	RNN (100)
ME	0.1285	0.1375	-0.0097	0.0616
MAE	0.2712	0.2245	0.2267	0.2987
RMSE	0.3186	0.2752	0.2889	0.3533
MAE-ratio	120.80%	100.00%	100.98%	133.03%
RMSE-ratio	115.77%	100.00%	104.97%	128.39%

Table 3: Evaluation criteria for *t*+12 forecasts.

	VAR	RS-VAR	RNN(50)	RNN (100)
ME	0.1963	0.1739	-0.2160	0.0428
MAE	0.2984	0.2579	0.2771	0.2498
RMSE	0.3761	0.3177	0.3332	0.3018
MAE-ratio	119.44%	103.25%	110.91%	100.00%
RMSE-ratio	124.63%	105.28%	110.41%	100.00%

[1] For a detailed account of vector-based machine learning models applied to financial prediction, see Shadbolt and Taylor (2002) and for hybrid models, see Kovalerchuck and Vityaev (2000).

[2] For thorough textbook treatments of VAR-models, see for example Enders (1995), Hamilton (1994) and Lütkepohl (1991).

[3] Relevant references to simulated annealing include Corona *et al*. (1987), Goffe *et al*. (1994), Fishman (1996, pp. 384-406) and Robert and Casella (1999, pp. 194-202). The program used in this paper is a C++ implementation of the algorithm in Goffe *et al*. (1994).

[4] From a computational perspective, there are therefore $s \cdot 2^{\tau}$ different probabilities τ periods ahead, each of which should be multiplied by the corresponding τ periods ahead forecast of $\mathbf{x}$.

[5] A detailed treatment of gradient descent learning for RNNs is provided in Pearlmutter (1995) and for brevity is not repeated here.

[6] From hereon simply referred to as FF-MLP.

[7] Our sample is restricted backward in time by the availability of data on M0 in EcoWin.