Stochastic nonlinear time series forecasting using time-delay reservoir computers: performance and universality

Lyudmila Grigoryeva[1], Julie Henriques[2], Laurent Larger[3], and Juan-Pablo Ortega[4]

Abstract

Reservoir computing is a recently introduced machine learning paradigm that has already shown excellent performances in the processing of empirical data. We study a particular kind of reservoir computers called time-delay reservoirs that are constructed out of the sampling of the solution of a time-delay differential equation and show their good performance in the forecasting of the conditional covariances associated to multivariate discrete-time nonlinear stochastic processes of VEC-GARCH type as well as in the prediction of factual daily market realized volatilities computed with intraday quotes, using as training input daily log-return series of moderate size. We tackle some problems associated to the lack of task-universality for individually operating reservoirs and propose a solution based on the use of parallel arrays of time-delay reservoirs.

Key Words: Reservoir computing, echo state networks, neural computing, time-delay reservoir, time series forecasting, universality, VEC-GARCH model, volatility forecasting, realized volatility, parallel reservoir computing.

1 Introduction

Reservoir computing (RC), also referred to in the literature as liquid state machines, echo state networks, or nonlinear transient computing, is a recently introduced machine learning paradigm [Jaeg 01, Jaeg 04, Maas 02, Maas 11, Croo 07, Vers 07, Luko 09] that has already shown excellent performances in the processing of empirical data. Examples of tasks that have already been explored are spoken digit recognition [Jaeg 07, Appe 11, Larg 12, Paqu 12, Brun 13], the NARMA model identification task [Atiy 00, Roda 11], and continuation of chaotic time series [Jaeg 04].

RC is a promising and fundamentally new approach to neural computing that can be seen as a modification of the traditional artificial recurrent neural networks (RNN) in which the architecture and the neuron weights of the network are created using different static or dynamic procedures and remain inaccessible during the training stage; this out of reach RNN is called the reservoir. The output signal is obtained out of the RNN via a linear readout layer that is trained using the teaching signal via a ridge regression.

[1]Laboratoire de Mathématiques de Besançon, Université de Franche-Comté, UFR des Sciences et Techniques. 16, route de Gray. F-25030 Besançon cedex. France. Lyudmyla.Grygoryeva@univ-fcomte.fr

[2]Cegos Deployment. 11 rue Denis Papin. F-25000 Besançon. jhenriques@deployment.org

[3]UMR CNRS FEMTO-ST 6174/Optics Department, Université de Franche-Comté, UFR des Sciences et Techniques. 16, route de Gray. F-25030 Besançon cedex. France. Laurent.Larger@univ-fcomte.fr

[4]Corresponding author. Centre National de la Recherche Scientifique, Laboratoire de Mathématiques de Besançon, Université de Franche-Comté, UFR des Sciences et Techniques. 16, route de Gray. F-25030 Besançon cedex. France. Juan-Pablo.Ortega@univ-fcomte.fr

In this paper we concentrate on a specific class of reservoirs whose architecture is dynamically created via the sampling of the solutions of a forced time-delay differential equation of the form

$$\dot{x}(t) = F(x(t), x(t-\tau), I(t)), \tag{1.1}$$

where $\tau > 0$ is the delay, and the external forcing $I(t)$ is used to insert the input signal that needs to be processed into the reservoir. We will refer to reservoirs constructed this way as time-delay reservoirs (TDRs). As we will see in detail in the next section, this procedure yields reservoir topologies that generalize the so called Simple Cycle Reservoirs (SCR) [Roda 11, Guti 12] that have already shown good performances in a variety of tasks. Moreover, as equations of the type (1.1) appear profusely in the modeling of concrete phenomena, this leads to the possibility of building physical realizations of the reservoir [Appe 11, Larg 12, Paqu 12, Dupo 12, Damb 12, Brun 13] which confers interesting possibilities to this approach. A particularly impressive example related to this feature are the results presented in [Brun 13] where an optoelectronic implementation of a TDR is capable of achieving the lowest documented error in the speech recognition task as well as the highest classification speed in an experiment design in which the setup carries out in parallel digit and speaker recognition.

Our work contains three main contributions:

- We show the pertinence of using TDRs in the forecasting of multivariate discrete time nonlinear stochastic processes, which is a non-deterministic extension of the frameworks where this technique has already proved to exhibit good performance. We carry out this assessment by exploring the applicability of this approach in volatility forecasting both with synthetic data generated using VEC-GARCH type models, as well as with actual market daily realized volatility matrices computed using intraday market price quotes. RC has already been used in the forecasting of factual time series from various origins (see for example [Ilie 07, Wyff 08, Wyff 10] and references therein); the underlying dynamics of the series chosen in most of these works exhibits significant seasonal and trend components that help in the forecasting. In contrast with these existing works, we will be dealing in our work with nonlinearly generated multidimensional white noise that lacks those helping features.

- We evidence the lack of task-universality for an individually operating TDR; more specifically, a reservoir that performs well in the forecasting of a stochastic dynamical model for a given horizon does not reproduce the same performance when the parameter values are modified and, to a lesser extent, when the prediction horizon changes. This observation has two serious negative consequences:

  - Adapting the reservoir to different tasks, even when they are closely related (like the forecasting of the same dynamical phenomenon at different horizons), requires the execution of expensive cross-validation based optimization routines.
  - It prevents the use of TDRs in a multi-tasking regime, which is one of the major potential advantages of reservoir computing [Maas 11].

- We propose the use of parallel arrays of reservoir computers in order to limit the computational effort necessary to adapt the device to the task at hand, to improve the performance for small training sample sizes, and to achieve robustness with respect to modifications in the task, hence making it more universal and appropriate for parallel reading. This approach is inspired by the theoretical results in [Boyd 85, Maas 00, Maas 11] that use basis filters to secure the defining features of a functioning RC, namely, the fading memory, separation, and approximation properties. The use of several reservoirs, each running at different regimes, functionally enriches the output of the global device. A parallel approach similar to ours with only two reservoirs has been explored in [Orti 12] where it is shown that the use of parallel nodes with different parameters is capable of providing memory capacity increases.

The paper is structured as follows: Section 2 explains in detail the construction of TDR computers and how they are adapted for forecasting tasks. Section 3 presents the volatility forecasting task that we are interested in, introduces the parametric family that we will be using as data generating process, and reviews the standard forecasting approach in this context that we use as a performance benchmark; additionally we introduce the parallel reservoir architecture that helps us solve challenges having to do with universality issues and small training sizes. The empirical results are presented in detail in Section 4. Section 5 concludes the paper.

Acknowledgments: We thank Yanne Chembo, Stéphane Chrétien, Maxime Jacquot, Christine de Mol, and Luis Pesquera for various insights about this project that have significantly improved it. Realized volatility matrices computed using intraday data were generously made available to us by Luc Bauwens. We acknowledge partial financial support of the Région de Franche-Comté (Convention 2013C-5493), the European project PHOCUS (FP7 Grant No. 240763), the Labex ACTION program (Contract No. ANR-11-LABX-01-01), and Deployment S.L. LG acknowledges financial support from the Faculty for the Future Program of the Schlumberger Foundation.

2 Construction of Time-Delay Reservoir (TDR) computers

A reservoir computer is formed by three modules that are depicted in Figure 1: an input layer (module A in the figure) that feeds the signal to be treated, the reservoir (module B) that processes it, and a readout layer (module C) that uses the reservoir output to produce the information needed. The input and readout layers are adapted to the specific features of the task under consideration; for example the construction of the input layer needs to take into consideration the dimensionality of the signal that needs to be processed; different ways to feed the signal can have an impact on the training speed of the system. Regarding the readout layer, we restrict to linear readings that are calibrated by minimizing the mean square error committed when comparing the output with a teaching signal, via a ridge regression. In this work we consider exclusively an offline (batch) type of training.

Construction of the time-delay reservoir (module B in Figure 1). We start by describing the construction of the dynamical time-delay reservoir out of a solution $x(t)$ of the differential equation (1.1). The state of the reservoir is labeled in time steps that are multiples of the delay τ and is characterized by the values of N ***virtual neurons*** obtained out of the regular sampling of $x(t)$ during a given time-delay interval. We will denote by $x_i(k)$ the ***value*** of the ith neuron of the reservoir at time $k\tau$ defined by

$$x_i(k) := x(k\tau - (N-i)\theta), \quad i \in \{1, \ldots, N\}, \tag{2.1}$$

where $\tau := \theta N$ and θ is referred to as the ***separation between neurons***. We will also say sometimes that $x_i(k)$ is the ***ith neuron value of the kth layer of the reservoir***.

Even though the justification for the TDR topology depicted in Figure 1 intuitively follows from the form of the differential equation (1.1), it can be also explained through its Euler time discretization as in [Appe 11, Guti 12]. More explicitly, consider the following special case of the time-delay differential equation (1.1):

$$\dot{x}(t) = -x(t) + f(x(t-\tau), I(t)). \tag{2.2}$$

Given an arbitrary natural number N, the Euler discretization of (2.2) with integration step $\theta := \tau/N$ is

$$\frac{x(t) - x(t-\theta)}{\theta} = -x(t) + f(x(t-\tau), I(t)). \tag{2.3}$$

We now construct a reservoir by using the same assignment of neuron values as in (2.1), that is, we define

$$x_i(k) := x(k\tau - (N-i)\theta) \quad \text{and} \quad I_i(k) := I(k\tau - (N-i)\theta).$$

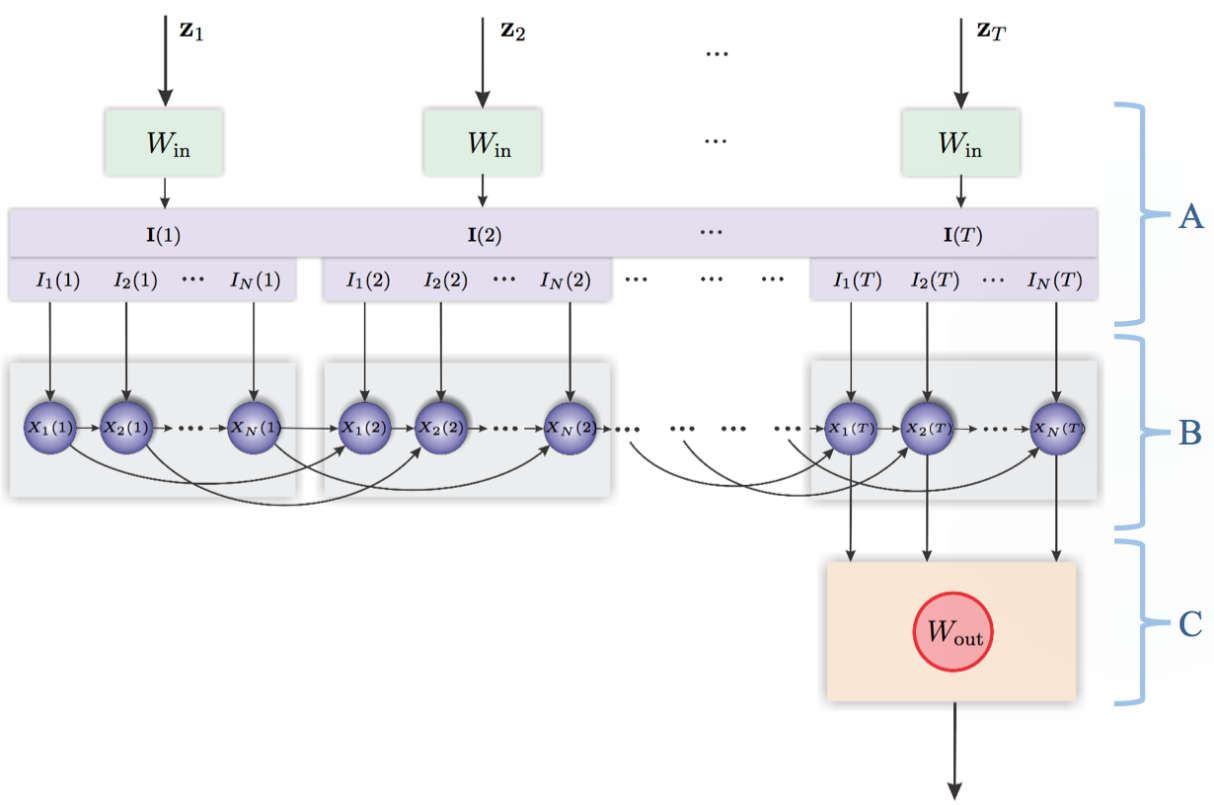

Figure 1: Neural diagram representing the architecture of the reservoir and the three modules of the reservoir computer: A is the input layer, B is the time-delay reservoir, and C is the readout layer.

The equation (2.3) amounts to saying that the neuron values $x_i(k)$ satisfy the following recursive relation:

$$x_i(k) := e^{-\xi} x_{i-1}(k) + (1-e^{-\xi}) f(x_i(k-1), I_i(k)), \quad \text{with} \quad x_0(k) := x_N(k-1), \quad \text{and} \quad \xi := \log(1+\theta), \tag{2.4}$$

that shows how, as depicted in Figure 1, the ith neuron value of the kth layer of the reservoir is a convex linear combination between the previous neuron value in the same layer and a nonlinear function of the same neuron value in the previous layer and the input.

The recursion (2.4) shows how the separation between neurons influences the memory of the reservoir regarding previous neuron values and the input signal; indeed, when the distance θ is small, the neuron value $x_i(k)$ is mainly determined by the previous neuron value $x_{i-1}(k)$, while large distances between neurons give predominance to the previous layer and foster the input gain.

We emphasize that a major feature of the TDR approach to information processing is the possibility of constructing physical realizations instead of simulating them using a computer. Actual experimental systems with proven RC capabilities have been built in [Appe 11] using electronic oscillators or in [Larg 12, Paqu 12, Dupo 12, Damb 12, Brun 13] with optoelectronic devices. The availability of physical implementations makes the TDR approach very promising since its use is an efficient way to overcome the speed and scaling problems associated with computer generated reservoirs [Luko 09].

Construction of the readout layer (module C in Figure 1). Suppose that the training is carried out with the help of T^* input layers $\mathbf{I} := \{\mathbf{I}(1), \ldots, \mathbf{I}(T^*)\}$, prepared to be used in a TDR with N neurons, that is, for each input layer $\mathbf{I}(k) := (I_1(k), \ldots, I_N(k))$, $k \in \{1, \ldots, T^*\}$, there is a corresponding

teaching signal $\mathbf{y}(k) \in \mathbb{R}^n$. As we already said, the construction of the input signal layers $\mathbf{I}$ is adapted to each specific task and will be described in detail below for the stochastic time series forecasting task. Typically, the number of neurons N is taken much higher than the dimension n of the teaching signal.

Since we restrict ourselves to linear readouts, for each input signal $\mathbf{I}(k)$ and reservoir layer $\mathbf{x}(k) := (x_1(k), \ldots, x_N(k))$, a n-dimensional output $\mathbf{y}(k)^* \in \mathbb{R}^n$ is obtained by using an ***output matrix*** $W_{\text{out}} \in \text{Mat}(N, n)$ and by setting $\mathbf{y}(k)^* := \mathbf{x}(k)^T \cdot W_{\text{out}}$. The training consists of finding the output matrix W_{out} that minimizes the distance between this output and the teaching signal, measured via the L^2 norm. This amounts to solving the following regularized optimization problem

$$\begin{aligned} W_{\text{out}} &:= \underset{W \in \text{Mat}(N,n)}{\arg\min} \left(\sum_{k=1}^{T^*} \|\mathbf{y}(k)^* - \mathbf{y}(k)\|^2 + \lambda \|W\|^2_{\text{Frob}} \right) \\ &= \underset{W \in \text{Mat}(N,n)}{\arg\min} \left(\sum_{k=1}^{T^*} \|\mathbf{x}(k)^T \cdot W - \mathbf{y}(k)\|^2 + \lambda \|W\|^2_{\text{Frob}} \right), \end{aligned} \tag{2.5}$$

whose solution is given by

$$W_{\text{out}} = (XX^T + \lambda \mathbb{I}_N)^{-1} X\mathbf{y}, \tag{2.6}$$

where $X \in \text{Mat}(N, T^*)$ is the reservoir output given by $X_{i,j} := x_i(j)$ and $\mathbf{y} \in \text{Mat}(T^*, n)$ is the teaching matrix containing the vectors $\mathbf{y}(k)$, $k \in \{1, \ldots, T^*\}$, organized by rows. The symbol $\|\cdot\|_{\text{Frob}}$ denotes the Frobenius norm $\|W\|^2_{\text{Frob}} := \text{trace}\left(W^T W\right)$ [Meye 00] and the ridge (or Tikhonov) penalty term $\lambda \|W\|^2_{\text{Frob}}$ [Tikh 43] is introduced in order to regularize the regression problem by limiting the norm of the solution hence helping to avoid overfitting. This regularization requires introducing a new parameter λ that controls the intensity of the penalty and that needs to be optimized at training via cross-validation.

The linear RC training philosophy that we adopt in this work has a profound positive impact in its practical implementation when compared to more traditional machine learning approaches like recursive neural networks because it allows us to replace convoluted and sometimes ill-defined optimization algorithms by simple regressions.

Construction of the input layer (module A in Figure 1) and the teaching signal for forecasting tasks. Given a multivariate time series sample $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_{T_{\text{train}}}\}$, $\mathbf{z}_i \in \mathbb{R}^n$, the goal of the forecasting task consists in training the RC so that it is capable of predicting the time series values h time steps in advance in an out-of-sample configuration. The value h will be referred to as the ***forecasting horizon***.

The first choice that has to be made in the construction of the input layer is the number N_I of time series values that are assigned to each reservoir layer; this choice may have an impact on the training speed of the system. We pick N_I so that it divides the number of neurons N and we note $K_I := N/N_I$.

Before the sample is inserted in the reservoir, it is preprocessed using a random input mask $W_{\text{in}} \in \text{Mat}(K_I, n)$ that produces a new signal $\mathbf{u} = \{\mathbf{u}_1, \ldots, \mathbf{u}_{T_{\text{train}}}\}$, defined by $\mathbf{u}_i := W_{\text{in}}\mathbf{z}_i \in \mathbb{R}^{K_I}$. This operation has two main roles:

- It takes care of the temporal and dimensional multiplexing of the signal by distributing the information contained in the same time series value to among K_I neurons.
- If the average of the mask values is zero then the preprocessed signal $\mathbf{u}$ has zero mean even if the original signal $\mathbf{z}$ does not have this property; this is particularly convenient in order to eliminate the need for an intercept in the ridge regression (2.5).

The choice of the input mask W_{in} may have a major impact in the performance of the TDR computer and the parameters that rule the working regime of the reservoir need, in general, to be optimized with respect

to the particular W_{in} selected. After the preprocessing, the input signal layers $\mathbf{I} = \{\mathbf{I}(1), \ldots, \mathbf{I}(T^*)\}$ are constructed via the assignment

$$\mathbf{I}(k) := \left(\mathbf{u}^T_{N_I(k-1)+1}, \mathbf{u}^T_{N_I(k-1)+2}, \ldots, \mathbf{u}_{N_I(k-1)+N_I-1}, \mathbf{u}^T_{kN_I}\right)^T \in \mathbb{R}^N,$$

with $k \in \{1, \ldots, T^*\}$. In order to simplify the presentation, we will assume without loss of generality that the entire sample $\mathbf{z}$ is used for training and hence $N_I T^* = T_{\text{train}}$.

If the goal of the forecasting task is, for instance, guessing the value of the time series value h time steps in advance it is natural to use as teaching signal

$$\mathbf{y}(k) := \mathbf{z}_{kN_I+h} \in \mathbb{R}^n, \qquad k \in \{1, \ldots, T\}. \tag{2.7}$$

Another example is provided in Section 3.2 where we carry out a volatility forecasting exercise whose teaching signal is specified in (3.5).

Once the system has been trained and the output matrix $W_{\text{out}}(h)$ adapted to the forecasting horizon h has been computed using (2.6), actual forecasting can be carried out of a separate testing data set $\widetilde{\mathbf{z}} := \{\widetilde{\mathbf{z}}_1, \ldots, \widetilde{\mathbf{z}}_{T_{\text{test}}}\}$ by generating a new reservoir output $\widetilde{X} \in \text{Mat}(N, T_{\text{test}}/N_I)$ out of it and by setting $\widetilde{\mathbf{y}} := \widetilde{X}^T W_{\text{out}}(h)$; the rows of $\widetilde{\mathbf{y}}$ contain the h-steps ahead forecasts for the elements of the form $\{\widetilde{\mathbf{z}}_{N_I}, \widetilde{\mathbf{z}}_{2N_I}, \ldots, \widetilde{\mathbf{z}}_{T_{\text{test}}}\}$ in $\widetilde{\mathbf{z}}$ (we assume that T_{test} is a multiple of the number N_I of time series values used for each input layer of the reservoir).

3 Stochastic nonlinear time series forecasting using time-delay reservoirs

In this section we explore the performance of RC in the forecasting of multidimensional nonlinear stochastic time series. Stochastic time series forecasting is a well established topic in statistical modeling with major applications in a large spectrum of applied research fields: sociology, epidemiology, ecology, econometrics, medicine, finance, engineering, transportation, meteorology, or material sciences to name a few. Most of the research activity in this field takes place in the parametric statistics setup. Using this approach, various families of models developed over the years and adapted to different kinds of temporal phenomena are fitted (estimated) to the available historical data and are subsequently used for forecasting via the minimization of different objective functions (usually the mean square error). Many textbooks explain this methodology, usually referred to as the Box-Jenkins approach [Box 76], at various levels and with different applications in mind; see [Broc 06, Broc 02, Lutk 05, Tsay 10]. We emphasize that this strategy is intrinsically empirical in the sense that the observed data are used for the selection and estimation of the models that are used later on for forecasting. Its implementation consists in the following steps:

- Model selection: this stage requires the choice of a specific parametric family of models for the problem at hand, hundreds of which have been developed over the years to account for many phenomena arising in different contexts. In practice, there are not many theoretical results that can be used for the selection of a particular family and this choice usually involves much trial and error. Additionally, once a model has been picked, one needs to determine its order; there exist asymptotic estimators for that purpose based on information theoretical arguments (AIC, BIC), forecasting efficiency (FPE, first prediction error), or others (Hannan-Quinn and Schwarz criteria). See [Lutk 05] for a careful presentation of all these criteria.

- Model estimation: it provides the values of the model parameters that fit best the available historical data. It is usually carried out via maximum likelihood estimation (MLE) which involves sometimes solving complicated, non-convex, high dimensional constrained optimization problems.

- Diagnostic checking: it consists of testing the statistical properties of the estimation residuals in order to asses the quality of the model choice. It usually involves using various tests of randomness adapted to the specific problems at hand.

- Forecasting: carried out via the minimization of an objective function. When this function is the mean square error, the solution of the optimization problem is provided by the conditional expectation of the time series value at the desired forecasting horizon with respect to the information set generated by the known history of the process. There are formulas available for most parametric families that allow the construction of this conditional expectation. A more complicated task is the assessment of the quality of the forecast since the error associated with it involves aggregating the errors committed not only at the forecasting stage but also at the model selection and estimation steps. For linear models there are available results based on asymptotic theory that provide information in this context [Lutk 05, Grig 13] while in more general contexts bootstrapping provides good answers (see [Pasc 01, Ruiz 02, Pasc 04, Pasc 05, Pasc 06, Rodr 09] and references therein).

The problems associated with parametric families, namely the intrinsic difficulties that go with model selection and estimation have motivated the use of alternative approaches, in particular neural networks based techniques. Artificial neural networks (ANNs) are intrinsically nonparametric and they hence provide an attractive solution in situations where there is a high danger of model misspecification, that is, they are capable of learning from examples and capturing subtle functional relationships among the data even when they are unknown or hard to describe. As a consequence, one finds ANNs in the forecasting of specific time series like airborne pollen, commodity prices, environmental temperature, helicopter component loads, international airline passenger traffic, macroeconomic indices, ozone level, personnel inventory, rainfall, river flow, student grade point averages, tool life, total industrial production, trajectories and transportation, and wind pressure profile, to name a few. The paper [Zhan 98] provides a good description of the state of the art and hundreds of references regarding the application of standard neural methods to forecasting.

Despite the plethora of studies having to do with the use ANNs for forecasting, there is no universal criterion that describes the situations in which ANNs outperform the parametric approach; a work in this direction is [Alme 91] which claims that for series with short memory, neural networks outperform the Box-Jenkins methodology. A comprehensive review that evaluates the forecasting performance of ANNs in 48 studies is [Adya 98]. Most of these works evoke the design problem as the major difficulty in the practical implementation of ANN based forecasting methods [Kaas 96, Balk 00, Cron 05] and even though many of these references present recipes for the conception of reasonable network architectures, it is clear that these rules are the outcome of trial and error analysis.

The objective of this section is proving that TDRs are capable of forecasting performances comparable to those attained using the Box-Jenkins approach when the nonlinear VEC-GARCH volatility models proposed by Bollerslev et al [Boll 88] are taken as data generating process (DGP).

The VEC-GARCH family is the straightforward multivariate extension of the one-dimensional generalized autoregressive conditionally heteroscedastic (GARCH) models [Engl 82, Boll 86]. These models are widely used tools to forecast volatility due to their ability to capture most of the stylized empirical facts that can be observed in financial time series: leptokurticity, volatility clustering, and asymmetric response to volatility shocks. The main difficulty associated to the practical implementation of these models is their lack of parsimony that, even in low dimensions, makes them extremely complicated to calibrate. Indeed, in n dimensions the VEC(1,1) family requires $n(n+1)(n(n+1)+1)/2$ parameters and hence it is rare to find these models at work beyond two or three dimensions and even then, additional limitations are imposed on the model to make it artificially parsimonious. See [Alta 03] for an illustration of the estimation of a three dimensional DVEC-GARCH model (VEC-GARCH model with diagonal parameter matrices [Boll 88]) using constrained non-linear programming; in [Chre 13], Bregman divergences based optimization methods are used to estimate VEC(1,1) models of dimensions up to

eight with considerable computational effort. Another reason that makes these models good candidates for testing the performance of TDRs, is the availability of explicit expressions for the optimal volatility forecasts whose associated errors can be subsequently used as a benchmark to asses the performance of the RC system.

As we will see in the following sections, the main mathematical difficulties that one encounters when forecasting in the VEC-GARCH context with standard methods, namely, the need for the solution of high dimensional nonconvex optimization problems, is replaced in the RC approach by topology selection questions. In that sense, time-delay reservoirs simplify the problem since they reduce it to the task of tuning the parameters of the nonlinear delay kernel. This intrinsic parameter set is parsimonious and, in some cases, the performance of the RC system is reasonably robust with respect to its values; moreover, we will show that it is possible to obtain additional parameter independence by using parallel pools of TDRs. We emphasize that once the TDR has been designed, its training does not in principle need the solution of sophisticated optimization problems and hence a good RC performance in the VEC-GARCH forecasting task is a nontrivial achievement that improves the state of the art technology.

3.1 The VEC-GARCH family and volatility forecasting

Consider the n-dimensional conditionally heteroscedastic discrete-time process $\{\mathbf{z}_t\}$ determined by the relation

$$\mathbf{z}_t = H_t^{1/2}\boldsymbol{\epsilon_t} \qquad \text{with} \qquad \{\boldsymbol{\epsilon_t}\} \sim \text{IIDN}(\mathbf{0}, \boldsymbol{I}_n).$$

The symbol $\text{IIDN}(\mathbf{0}, \boldsymbol{I}_n)$ refers to the set of n-dimensional independent and identically distributed normal variables with mean $\mathbf{0}$ and covariance matrix $\boldsymbol{I}_n$. Additionally, in this expression $\{H_t\}$ is assumed to be a predictable positive semidefinite matrix process, that is, for each $t \in \mathbb{N}$, the matrix random variable H_t is $\mathcal{F}_{t-1}$-measurable, with $\mathcal{F}_{t-1} := \sigma(\mathbf{z}_0, \ldots, \mathbf{z}_{t-1})$ the information set (sigma algebra) generated by $\{\mathbf{z}_0, \ldots, \mathbf{z}_{t-1}\}$. In these conditions it is easy to show that the conditional mean $E[\mathbf{z}_t \mid \mathcal{F}_{t-1}] = \mathbf{0}$ and that the conditional covariance matrix process of $\{\mathbf{z}_t\}$ coincides with $\{H_t\}$.

Different specifications for the time evolution of the conditional covariance matrix $\{H_t\}$ determine different vector conditional heteroscedastic models. Here we focus on the **VEC-GARCH model**, introduced in [Boll 88] as the direct generalization of the univariate GARCH model [Boll 86] in the sense that every conditional variance and covariance is a function of all lagged conditional variances and covariances as well as all squares and cross-products of the lagged time series values. More specifically, the VEC(q,p) model is determined by

$$\boldsymbol{h}_t = \mathbf{c} + \sum_{i=1}^{q} A_i \boldsymbol{\eta}_{t-i} + \sum_{i=1}^{p} B_i \boldsymbol{h}_{t-i},$$

where $\boldsymbol{h}_t := \text{vech}(H_t)$, $\boldsymbol{\eta}_t := \text{vech}(\mathbf{z}_t\mathbf{z}_t^T)$, $\mathbf{c}$ is a N-dimensional vector, with $N := n(n+1)/2$ and $A_i, B_i \in \mathbb{M}_N$. The linear operator $\text{vech} : \mathbb{S}_n \to \mathbb{R}^N$ stacks the lower triangular part of a symmetric matrix including its main diagonal into a vector [Lutk 05]. We consider the case $p = q = 1$, that is:

$$\left\{ \begin{array}{rcl} \mathbf{z}_t & = & H_t^{1/2}\boldsymbol{\epsilon_t} \qquad \text{with} \qquad \{\boldsymbol{\epsilon_t}\} \sim \text{IIDN}(\mathbf{0}, \boldsymbol{I}_n), \\ \boldsymbol{h}_t & = & \mathbf{c} + A\boldsymbol{\eta}_{t-1} + B\boldsymbol{h}_{t-1}. \end{array} \right. \tag{3.1}$$

As we already mentioned, in this case the model needs $N(2N+1) = \frac{1}{2}n(n+1)(n^2+n+1)$ parameters for a complete specification.

A major complication in the estimation of these models via maximum likelihood is that the general prescription for the VEC-GARCH model spelled out in (3.1) does not guarantee that it has stationary solutions or that the resulting conditional covariance matrices $\{H_t\}_{t\in\mathbb{N}}$ are necessarily positive semidefinite. These structural requirements call for constraints that should be imposed on the parameter

matrices $\mathbf{c}, A$, and B. Unlike the situation encountered in the one-dimensional case, only sufficient conditions for positivity and stationarity are available in the literature for the multidimensional setup. For example, [Gour 97] shows that (3.1) admits a unique second order stationary solution if all the eigenvalues of $A+B$ lie strictly inside the unit circle. Regarding positive semidefiniteness of $\{H_t\}_{t\in\mathbb{N}}$, it can be shown that (see [Chre 13, Proposition 3.1]) this is ensured by the positive semidefiniteness of $\mathrm{math}(\mathbf{c}), \Sigma(A)$, and $\Sigma(B)$, where $\mathrm{math}:\mathbb{R}^N\to\mathbb{S}_n$ is the inverse of the vech operator and Σ is the linear operator introduced in [Chre 13, Definition 2.2].

Volatility forecasting. The main use of the VEC(1,1) model (3.1) is the description and the forecasting of the volatility of financial time series. In that context, the process $\{\mathbf{z}_t\}$ describes the dynamical evolution of market log-returns and $\{H_t\}$ are the associated conditional covariance matrices. The log-returns $\{\mathbf{z}_t\}$ are observable but the covariance matrices $\{H_t\}$ are not; nevertheless, the estimation and forecasting of these random variables is of paramount importance in portfolio and risk management, as well as in scenarios generation. The volatility prediction task at time T with a horizon of h time steps consists of providing the optimal forecast $\widehat{H_{T+h}}$ of the conditional covariance matrix H_{T+h} based on the information set $\mathcal{F}_T$ generated by the log-returns up to time T, that is, $\mathcal{F}_T := \sigma(\mathbf{z}_0,\dots,\mathbf{z}_T)$. This estimate is produced by minimizing the mean square forecasting error (MSFE) defined as

$$\mathrm{MSFE}(h) := E\left[\left(\widehat{\mathbf{h}_{T+h}}-\mathbf{h}_{T+h}\right)\left(\widehat{\mathbf{h}_{T+h}}-\mathbf{h}_{T+h}\right)^T\right],$$

where $\mathbf{h}_{T+h} := \mathrm{vech}(H_{T+h})$ is the vectorized conditional covariance matrix at time $T+h$, and $\widehat{\mathbf{h}_{T+h}} := \mathrm{vech}(\widehat{H_{T+h}})$. The $\mathrm{MSFE}(h)$ is a positive semidefinite matrix of dimension $N := \frac{1}{2}n(n+1)$ hence, given two different forecasts of $\mathbf{h}_{T+h}$ with associated errors $\mathrm{MSFE}(h)$ and $\widetilde{\mathrm{MSFE}}(h)$, respectively, we will say that $\mathrm{MSFE}(h) \le \widetilde{\mathrm{MSFE}}(h)$ when the difference $\widetilde{\mathrm{MSFE}}(h) - \mathrm{MSFE}(h)$ is a positive semidefinite matrix. A standard result [Hami 94, Chapter 4] shows that the optimal forecast of a random variable with respect to the MSFE and a given information set is given by its conditional expectation with respect to that information set. In particular, the optimal forecast $\widehat{\mathbf{h}_{T+h}}$ for the covariance matrix $\mathbf{h}_{T+h}$ knowing the log-returns $\{\mathbf{z}_0,\dots,\mathbf{z}_T\}$ is given by:

$$\widehat{\mathbf{h}_{T+h}} := \underset{\widetilde{\mathbf{h}_{T+h|\mathcal{F}_T}}}{\arg\min}\, E\left[\left(\mathbf{h}_{T+h}-\widetilde{\mathbf{h}_{T+h|\mathcal{F}_T}}\right)\left(\mathbf{h}_{T+h}-\widetilde{\mathbf{h}_{T+h|\mathcal{F}_T}}\right)^T\right] = E\left[\mathbf{h}_{T+h}\mid\mathcal{F}_T\right]. \tag{3.2}$$

An important advantage of the VEC-GARCH family is the possibility of explicitly computing the conditional expectation in (3.2) that yields the optimal forecast $\widehat{\mathbf{h}_{T+h}}$ via the following recursion on the horizon h:

$$\begin{aligned}
\widehat{\mathbf{h}_{T+1}} &= \mathbf{h}_{T+1} = \mathbf{c} + A\boldsymbol{\eta}_T + B\boldsymbol{h}_T,\\
\widehat{\mathbf{h}_{T+2}} &= \mathbf{c} + (A+B)\boldsymbol{h}_{T+1},\\
\widehat{\mathbf{h}_{T+3}} &= \mathbf{c} + (A+B)\widehat{\mathbf{h}_{T+2}},\\
&\vdots\\
\widehat{\mathbf{h}_{T+i}} &= \mathbf{c} + (A+B)\widehat{\mathbf{h}_{T+i-1}},\\
&\vdots\\
\widehat{\mathbf{h}_{T+h}} &= \mathbf{c} + (A+B)\widehat{\mathbf{h}_{T+h-1}}.
\end{aligned} \tag{3.3}$$

We emphasize that this recursion shows that the functional dependence of the forecast $\widehat{\mathbf{h}_{T+h}}$ on the elements $\{\mathbf{z}_0,\dots,\mathbf{z}_T\}$ that generate the information set $\mathcal{F}_T$ is highly nonlinear. Since it is the series

$\{\mathbf{z}_0, \ldots, \mathbf{z}_T\}$ that are used as an input in the TDR for the forecasting task, the performance of the RC scheme, if positive, cannot be exclusively associated to the linear readout layer that is estimated via a ridge regression and requires the contribution of the nonlinear reservoir module.

3.2 TDR based volatility forecasting

The objective of this section is assessing the performance and challenges of TDRs in the forecasting of volatilities synthetically generated by VEC-GARCH models of the type introduced in the previous subsection. We work in dimension $n = 3$ which requires the use of 78 different parameters in the construction of the model (3.1). The forecasting task will be carried out with a TDR of the type presented in Section 2 using the same nonlinear function used in [Appe 11] for the design of an experimental electronics based reservoir computer, namely,

$$f(x(t-\tau), u(t), \eta, \gamma, p) = \frac{\eta\,(x(t-\tau) + \gamma u(t))}{1 + (x(t-\tau) + \gamma u(t))^p}, \tag{3.4}$$

where γ, η, and p are real value parameters.

We work with reservoirs made out of N neurons with each layer constructed using a single time series value ($N_I = 1$, $T^* = T_{\text{train}}$), that is, if $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_{T_{\text{train}}}\}$, $\mathbf{z}_i \in \mathbb{R}^n$, is the training sample generated with a given three dimensional VEC-GARCH model, then the i-th input signal layer $\mathbf{I}(i) \in \mathbb{R}^N$ is obtained via the assignment $\mathbf{I}(i) := W_{\text{in}}\mathbf{z}_i$ with $W_{\text{in}} \in \text{Mat}(N, n)$ a random input mask whose entries are uniformly distributed in the centered interval $[-a, a]$, $a \in \mathbb{R}$.

We emphasize that the VEC-GARCH process (3.1) is a white noise, that is the series $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_{T_{\text{train}}}\}$ exhibits no serial autocorrelation. This makes pointless the forecasting of those time series values in the sense described in Section 2. In exchange, as we already mentioned, the main use of VEC-GARCH type models is the forecasting of conditional covariances and hence, in order to construct a readout layer adapted to this task, it suffices to use in (2.7) as teaching signal the h-shifted versions of the conditional covariance matrices $\{\mathbf{h}_{1+h}, \ldots, \mathbf{h}_{T_{\text{train}}}\}$ generated by the model together with the time series values $\{\mathbf{z}_1, \ldots, \mathbf{z}_{T_{\text{train}}-h}\}$, that is,

$$\mathbf{y}(k) := \mathbf{h}_{k+h} \in \mathbb{R}^n, \qquad k \in \{1, \ldots, T_{\text{train}} - h\}. \tag{3.5}$$

The regression (2.6) using this teaching signal and the reservoir output, yields an output matrix W_{out} that is subsequently used for out-of-sample forecasting.

Reservoir design and parameter optimization. Given a reservoir architecture it is in general challenging to find a universal set of optimal parameters (θ, γ, η) that offers top performance for a large array of tasks presented to it. A particularly detailed description of this phenomenon can be found in [Damb 12] where a detailed quantitative analysis is carried out in which the dependence of the total memory capacity of a RC system is studied as a function of its design parameters. We have realized in our work that our particular task is not an exception and that tuning the reservoir parameters to adapt them to it may have a major impact on the final performance of the method. In the particular case of VEC-GARCH volatility forecasting, the empirical results in Section 4 show that this statement holds true when forecasting is carried out for different processes, that is, different sets of parameters $\mathbf{c}$, A, and B, and to a lesser extent when the forecasting horizon changes. This observation has two important implications from the practical implementation point of view:

- **Numerical cost:** The parameter optimization is carried out via a computationally expensive cross validation procedure.
- **Parallel reading (multi-tasking):** one of the potential advantages of RC is the ability to carry out tasks in parallel (see for example the introduction in [Maas 11]); the strategy behind this

observation consists of processing a single signal with the reservoir and consequently using its output for different tasks via adapted readout matrices estimated with the pertinent teaching signal. In other words, there is a single signal processing step but various tasks are simultaneously accomplished with the output by adapting only the linear regression step to each of them. This is obviously only possible if the reservoir is functioning in a regime with good performance for all the tasks. In the particular case of forecasting problems, parallel reading can be useful at the time of simultaneously predicting at various horizons out of a single input signal; however, we emphasize this is only feasible whenever there is a set of reservoir parameters for which the forecasting performance is acceptable for all the horizons of interest.

3.3 Parallel reservoir computing and universality

In the following paragraphs we propose a solution to the problems described above that consists of presenting the input signal to a parallel array of reservoirs, each of them running with randomly chosen parameter values; the union of the outputs of these reservoirs is then used to construct a single readout layer via a ridge regression. Figure 2 provides an illustration of the parallel reservoir computing strategy.

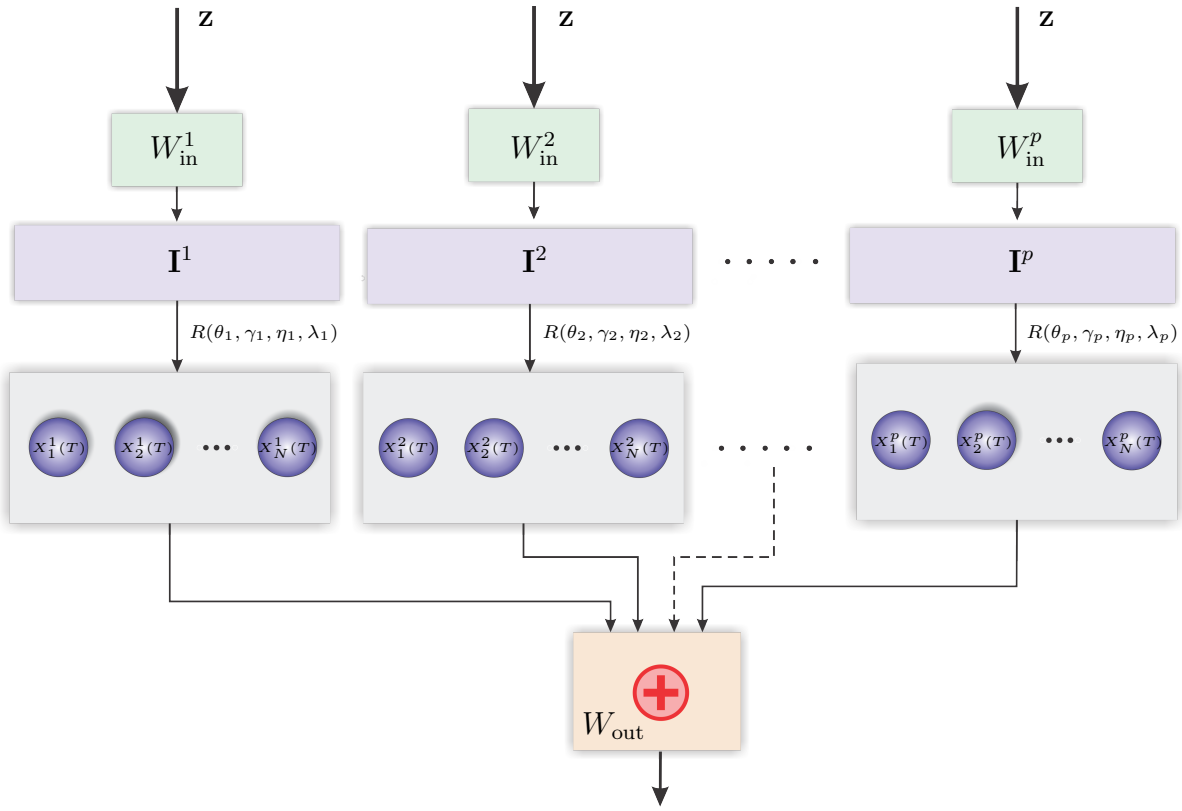

Figure 2: Diagram representing the architecture of a parallel reservoir computer.

The rationale behind this suggestion is the set of mathematical arguments that are used to prove that reservoir computers can be constructed so that they satisfy three key information processing features usually associated to this computational paradigm, namely the separation, approximation, and fading memory properties (see for example [Maas 02, Maas 11] and also [Boyd 85]). Indeed, in those results it is always required that the reservoir is capable of generating an output that is sufficiently rich from

the functional point of view so that standard approximation results like the Stone-Weierstrass theorem can be applied. A more explicit statement is provided in [Boyd 85] where it is shown how to carry out approximations using Volterra series generated by ensembles of delay operators. A parallel approach similar to ours with only two reservoirs has been explored in [Orti 12] where it is shown that the use of parallel nodes with different parameters is capable of providing memory capacity increases.

As we empirically show in the next section, this approach presents several advantages when compared to the use of a single optimized reservoir with the same total number of neurons, namely, limited computational effort for parameter optimization, better performance for smaller training sample sizes, and improved universality with respect to changes in the forecasting horizon and in the model specification.

4 Empirical results

In this section we conduct two empirical experiments that show the good performance of TDRs in the volatility forecasting task and substantiate the claims in the previous paragraphs as to the different implementation strategies. The tests will be conducted using both synthetic time series as well as real market data:

- **Performance assessment with synthetic data:** we consider a generic VEC(1,1) model as in (3.1) with parameters $\mathbf{c}$, A, and B that satisfy the constraints spelled out in Section 3.1 and that ensure that the model is stationary and produces positive semidefinite conditional covariance matrices. This model will be used as a data generating process for both the training and testing datasets when forecasting using the TDRs. In order to test the volatility forecasting capabilities of the different TDR implementation strategies, we generate two pairs of VEC-GARCH time series with their corresponding conditional covariance matrices, each of length 100 000 time steps. These pairs are subsequently used for the training and for the testing, respectively, noticing that we use the first 1 000 points in both as a washout period; the first pair will be used for parameter optimization in the different configurations (see below) and the second one for performance assessment.
- **Performance assessment with market data. Realized volatility forecasting:** the goal of this exercise is testing the ability of the TDR in the forecasting of market realized daily covariance matrices out of the observed corresponding market log-returns. The realized covariances are computed using intraday price quotes. The main challenge of this task when compared to the previous one are the limitations in sample size since we only have 2 500 available time steps for training and testing.

The different forecasting performances will be measured using the scalar standardized mean square forecasting error (sMSFE) defined as follows:

$$\text{sMSFE}(h) := \frac{E\left[\left(\mathbf{h}_{T+h} - \widehat{\mathbf{h}_{T+h}}\right)^T \left(\mathbf{h}_{T+h} - \widehat{\mathbf{h}_{T+h}}\right)\right]}{\text{var}\left(\mathbf{h}_{T+h}\right)},$$

where $\mathbf{h}_{T+h}$ is the vectorized covariance matrix that we are interested in and $\widehat{\mathbf{h}_{T+h}}$ the corresponding forecast h time steps into the future. We will consider forecasting horizons of up to twenty time steps.

4.1 Reservoir configurations

The different devices whose performances will be compared are constructed according to the following architectures:

(i) TDR with 400 neurons and grid optimized parameters: a TDR is built according to the prescription in (2.1) using the nonlinear kernel in (3.4). A preliminary study showed that the forecasting performance of this device is much dependent on the parameter values $\theta := \tau/N$, γ, and η of the kernel (3.4). We hence carry out a grid search for the optimal parameters (θ, γ, η) by using the first pair of series to conduct training/testing routines for each of the parameter combinations in the grid that yields, for each forecasting horizon h, the triple (θ, γ, η) that minimizes the corresponding sMSFE(h). The optimization grid is constructed according to the following prescription: $\theta, \gamma \in [0.1, 2]$, $\eta \in [0.1, 1.5]$ with a search step of 0.1.

(ii) TDR with 400 neurons and random optimized parameters: In this case we randomly simulate one hundred different sets of parameters and keep for each forecasting horizon the one that produces the smallest error in the testing phase. The parameters are randomly drawn from the uniform distribution on the intervals $[0.01, 2]$ for θ and γ, and $[0.01, 1.5]$ for η. For each of these parameter sets we simulate a new random input mask W_{in}.

(iii) Parallel array of 40 TDRs of 10 neurons each with random optimized parameters: we randomly generate one hundred parallel TDRs containing 40 reservoirs each with 10 neurons. The parameters of each TDR are generated using the same prescription as in the previous point and the different random input masks W_{in} are used for each draw.

(iv) Parallel array of 80 TDRs of 5 neurons each with random optimized parameters: the procedure is identical to the one in the previous point but this time with 80 reservoirs with 5 neurons each.

The choice of the total number of neurons for the individually operating TDR (configurations **(i)** and **(ii)**) has been made after several trials that show 400 nodes suffice to appropriately handle the complexity of the forecasting task at hand without incurring in overfitting. The size of this neural array also allows the construction of the arrays in points **(iii)** and **(iv)** with a sizeable number of parallel units, each of them with a reasonable number of neurons.

Concerning the parameter p in the nonlinear kernel (3.4), we explore the performances obtained with the values $p = 1$ and $p = 2$. The value $p = 1$ presents a potential difficulty due to the fact that there are non-generic combinations of reservoir inputs and outputs such that the denominator $1 + (x(t-\tau) + \gamma u(t))^p$ in (3.4) may become close to zero and cause a divergence. We avoid this problem in practice by using a random input mask $W_{\text{in}} \in \text{Mat}(N, n)$ that takes values in the interval $[-a, a]$, with a sufficiently small; the use of this device amounts to a global normalization of the input/output series that makes the term $x(t-\tau) + \gamma u(t)$ small with respect to 1 and hence makes the cancelling of $1 + x(t-\tau) + \gamma u(t)$ very unlikely. In what follows we will consider random input masks W_{in} with values uniformly distributed in the interval $[-0.1, 0.1]$ for the kernel with $p = 1$; in the case $p = 2$ this interval will be replaced by $[-1, 1]$.

4.2 Volatility forecasting with synthetic data

Using the generic VEC(1,1) model described above as data generating process for both time series and covariance matrices, we construct the different optimized reservoir configurations that we just listed. Table 4.1 reports the sets of optimal parameters obtained for each horizon and for both the grid and the random optimized individually operating TDR; additionally, the two last columns report the set labels of the arrays of parallel TDRs that optimize the forecasting performance among 100 randomly generated. In the case of 40 (respectively 80) parallel TDRs with 10 neurons (respectively 5) each, 2 (respectively 4) different pools are needed to optimize the performance for all the horizons considered.

We emphasize that the optimal parameters reported in Table 4.1 are related to a specific choice of random input mask W_{in}. Unreported simulations show that the same forecasting performances can be

attained with different random masks but this requires slight modifications of the reservoir parameters in order to achieve optimality. On other words, the optimal reservoir parameters depend on the random mask chosen.

The sMSFE committed when forecasting using these different devices is compared with the one associated to the optimal forecast that we described in (3.3). Unlike the situation for linear models, there are no explicit formulas to compute the corresponding error in the VEC-GARCH context and we hence estimate it via an expensive Monte Carlo simulation using two million different predictions. We emphasize that the comparison of this error with the one committed via the different RC based procedures presents an unjustified advantage in favor of the theoretical forecast because this one makes use of a total knowledge about the model and its parameters which is unknown to the RC strategy; in a more realistic situation the model needs to be selected and estimated which adds error to the forecast.

	Individually operating TDR of 400 neurons								**Parallel TDRs**	
	grid optimized				**random optimized**				**40 TDRs**	**80 TDRs**
Horizon	η	γ	θ	#	η	γ	θ	#	#	#
1	1.20	2.00	1.30	1	0.87	1.90	1.39	1	1	1
2	1.20	2.00	1.30	1	1.23	1.23	1.72	2	1	2
3	1.20	2.00	1.30	1	1.23	1.23	1.72	2	1	2
4	1.10	0.80	1.30	2	1.23	1.23	1.72	2	1	3
5	1.10	0.80	1.30	2	1.23	1.23	1.72	2	1	2
6	1.10	0.60	1.30	3	1.23	1.23	1.72	2	1	2
7	1.20	1.90	1.30	4	1.23	1.23	1.72	2	1	3
8	1.20	1.90	1.30	4	1.23	1.23	1.72	2	1	2
9	1.20	1.90	1.30	4	1.23	1.23	1.72	2	1	3
10	1.20	1.90	1.30	4	1.23	1.23	1.72	2	2	4
11	1.20	1.90	1.30	4	1.23	1.23	1.72	2	2	4
12	1.20	1.90	1.30	4	1.23	1.23	1.72	2	2	4
13	1.20	1.90	1.30	4	1.23	1.23	1.72	2	2	4
14	1.20	1.90	1.30	4	1.23	1.23	1.72	2	2	3
15	1.20	1.40	1.30	5	1.23	1.23	1.72	2	2	3
16	1.20	1.50	1.30	6	1.23	1.23	1.72	2	2	3
17	1.20	1.50	1.30	6	1.23	1.23	1.72	2	2	3
18	1.20	1.50	1.30	6	1.23	1.23	1.72	2	2	3
19	1.20	1.30	1.30	7	1.36	1.48	1.41	3	2	3
20	1.20	1.30	1.30	7	1.36	1.48	1.41	3	2	3

Table 4.1: The two sets of columns on the left of the table report the TDR grid and random optimized parameters corresponding to the different forecasting horizons for a single reservoir of 400 neurons. The grid (respectively random) optimized reservoir requires 7 (respectively 3) different parameter sets to achieve top performance for all the horizons considered; the set labels appear in the columns with the symbol #. The two last columns on the right of the table report the set labels of the arrays of parallel TDRs that optimize the forecasting performance among 100 randomly generated. In the case of 40 (respectively 80) parallel TDRs with 10 neurons (respectively 5) each, 2 (respectively 4) different pools are needed to optimize the performance for all the horizons considered. All the TDRs considered have been generated using the nonlinear kernel (3.4) with $p = 1$.

The three main conclusions resulting from our empirical comparison between the different RC based methods and the standard time series approach are:

- **TDRs are capable of competitively accomplishing forecasting tasks for stochastic**

nonlinear processes without the need to solve the sophisticated model selection and estimation problems associated to this type of parametric models (see the introduction to this section). As we show in Figure 3, the difference between the theoretical and the RC based errors is of modest size for all the approaches and it is justified by the fact that the theoretical error does not take into consideration the model selection and estimation errors to which the RC is exposed. The good forecasting performance is visible not only from the point of view of the sMSFE (Figure 3) but, as we illustrate in the fancharts in Figure 4, also in the error distributions.

- **The good performance in the forecasting task is mostly due to the presence of the reservoir** and cannot be explained by the regression step in the construction of the readout layer. Indeed, it can be checked that if the TDR module is suppressed in the forecasting scheme and we just carry out the regression step, a sMSFE is committed of around 10.86, independently of the horizon considered. We emphasize at this point that this observation holds mainly due to the nonlinear character of the VEC-GARCH data generating process that we use; indeed, simulations not reported in which we carried out a standard forecasting task using linear multidimensional VARMA models [Lutk 05], showed that the performance contribution of the reservoir in this setup is very limited, most of the forecasting effort being accomplished by the linear step in the information treatment.

- **The parameters of the TDR kernel need to be optimized in order to achieve adequate predictions.** As we will show later on when we discuss universality questions, the TDR based forecasting performance deteriorates rapidly when one separates from the optimal regime. This optimization can be carried out either via a systematic grid search or by randomly simulating parameter sets. As Figure 3 shows, the second option produces better results when we consider not a single reservoir but certain parallel arrays of them; in this case, a pool of 40 parallel reservoirs with 10 neurons each produces particularly good results. The computational cost of the random optimization is much lower than the grid search which, together with other considerations that we will spell out later on, points to the advantages of using arrays of parallel reservoirs instead of single ones.

Advantages of parallel reservoir computing. We have conducted a series of numerical experiences that show how the parallel reservoir architecture helps in minimizing some of the difficulties in the practical implementation of RC and lead us to believe that this is an interesting novel concept that deserves further investigation in the context of other machine learning tasks. The main merits of the parallel RC approach are:

- **Possibility of random parameter optimization and limitation of the computational effort.** An optimal parallel reservoir chosen out of a randomly generated family of parallel pools of moderate size exhibits performances similar to those of a grid optimized single reservoir with an equivalent number of neurons and systematically outperforms optimal single reservoirs picked out of a random family with the same number of draws (see Figure 3). Grid optimization is numerically very costly so the possibility of choosing a TDR configuration out of a randomly generated family of limited size is very valuable.

- **Better performance for smaller training sample sizes.** As it is shown in Figure 5, when the training sample size shrinks, the forecasting error remains more stable for a parallel reservoir array configuration than for a single optimized reservoir with the same number of neurons. This makes the parallel reservoir array particularly convenient in the presence of limited available historical data, as it will be the case in the empirical study with factual data carried out in the next section. This increase in performance cannot be attributed to the presence of overfitting

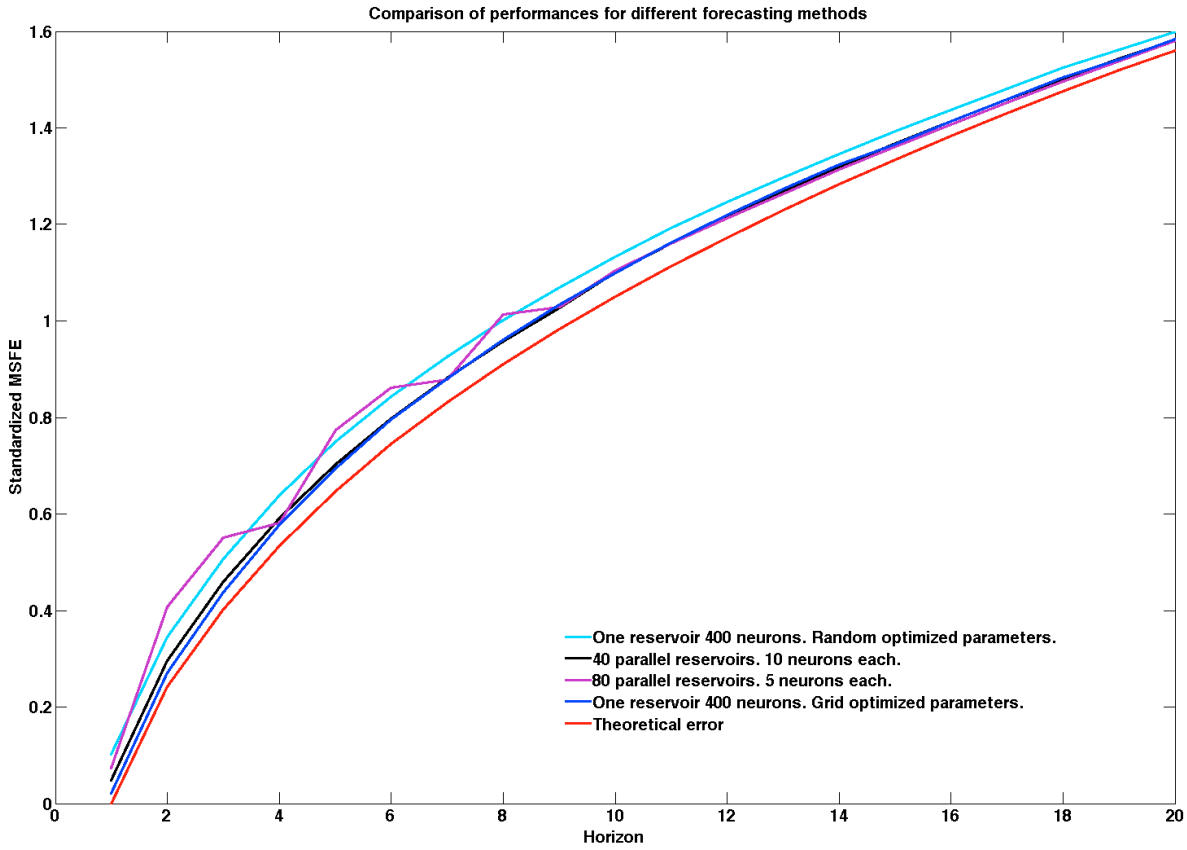

Figure 3: Forecasting performance in the three dimensional VEC-GARCH volatility forecasting task achieved with the different RC based methods in comparison with the theoretical error as a function of the forecasting horizon. All the TDRs considered have been generated using the nonlinear kernel (3.4) with $p = 1$. The training sample size is 100 000.

in the other competing configurations since the total number of neurons and the corresponding number of covariates in the ridge regression is identical for all of them. Consequently, the gain is certainly related to the richer functional output of this architecture that, as it has already been documented in the literature [Orti 12], yields improved memory capacities that increase the forecasting performance.

- **Improved universality with respect to changes in the forecasting horizon and in the model specification.** The optimal parameters for the prediction task are not the same neither for different forecasting horizons nor for different data generating processes. This variability is reduced by the use of parallel arrays of RCs.

  Regarding the dependence on the forecasting horizon, we already saw in Table 4.1 how a single grid optimized TDR with 400 neurons requires 7 different parameter sets to optimize the performance for the 20 horizons considered, while for the parallel array of 40 TDRs with 10 neurons each, only 2 suffice. Nevertheless, the sensitivity of the performance with respect to the set of parameters chosen for each forecasting horizon is limited for both configurations. In order to illustrate this claim, in Figure 6, we have taken only one set of parameters for all the forecasting horizons both

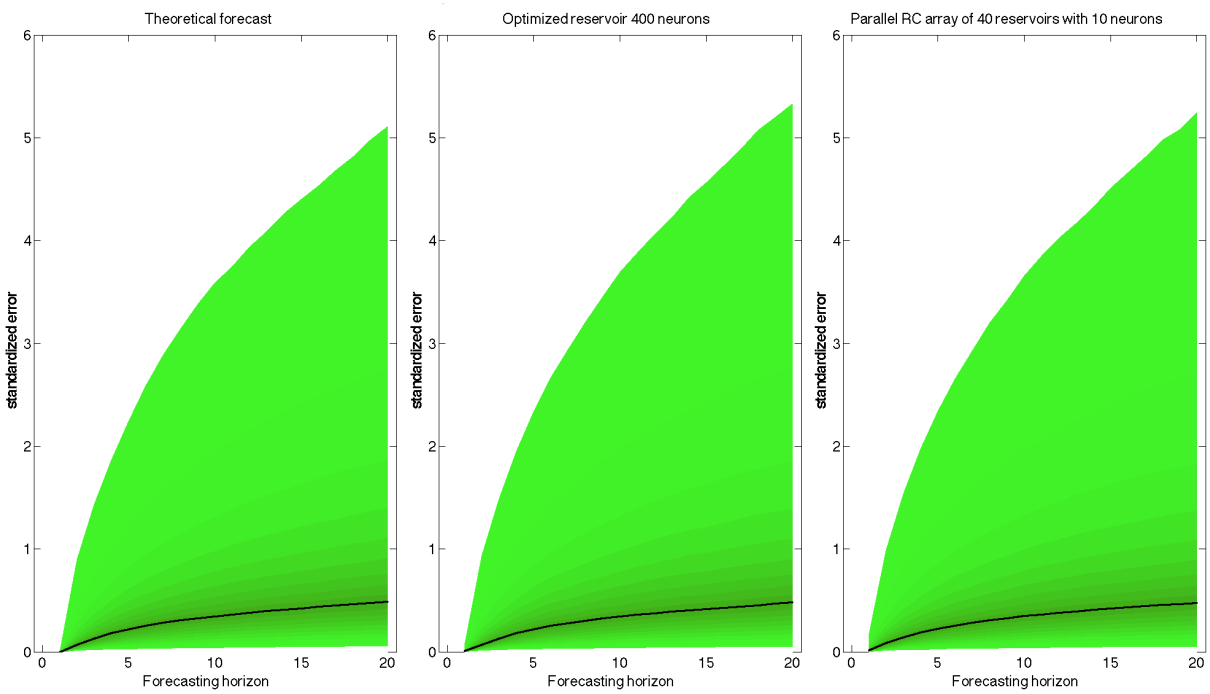

Figure 4: Fancharts representing the standardized error distributions committed in the volatility forecasting task for different forecasting horizons. The chart in the left corresponds to the error committed when using the optimal forecast spelled out in (3.3), the one in the middle was obtained using a grid optimized TDR with four hundred neurons, and the one in the right using the best of one hundred randomly generated parallel pools of forty reservoirs with ten neurons each. The black curves represent the standardized mean square forecasting errors committed at each forecasting horizon. All the TDRs considered have been generated using the nonlinear kernel (3.4) with $p = 1$. The training sample size is 100 000.

for the single and the parallel RC setup and we have tested the performance obtained. For the two devices we took the parameter configurations that appear more frequently in Tables 4.1 and, as it can be seen, the forecasting efficiency remains satisfactory for all horizons and for both approaches.

The situation is radically different when we modify the data generating process even if we keep using the same parameter family. Indeed, in Figure 7 we have considered volatility forecasting for three new VEC(1,1) models obtained by modifying the parameters $\mathbf{c}$, A, and B in (3.1). However, this time we use reservoirs, single or in parallel, tuned with the parameters that optimized the forecasting performance for the VEC-GARCH model that we considered up until now. As Figure 7 shows, the performance efficiency considerably deteriorates even though for the three models considered the array of parallel reservoirs presents a more stable behavior. In this particular exercise we have used TDRs generated using the nonlinear kernel (3.4) with $p = 2$; additional analogous simulations whose results we do not report here showed that the choice $p = 1$ produced an even more acute decline in the forecasting performance in the presence of misspecified data generating processes.

We emphasize that this observation proves in particular the claim about the need for parameter adjustments in the RC with respect to the task that needs to be accomplished and that we have put forward all along the paper. Using the terminology of time series modeling, Figure 7 also shows that parallel reservoir computing improves the performance of this computational paradigm when working in the presence of model misspecification.

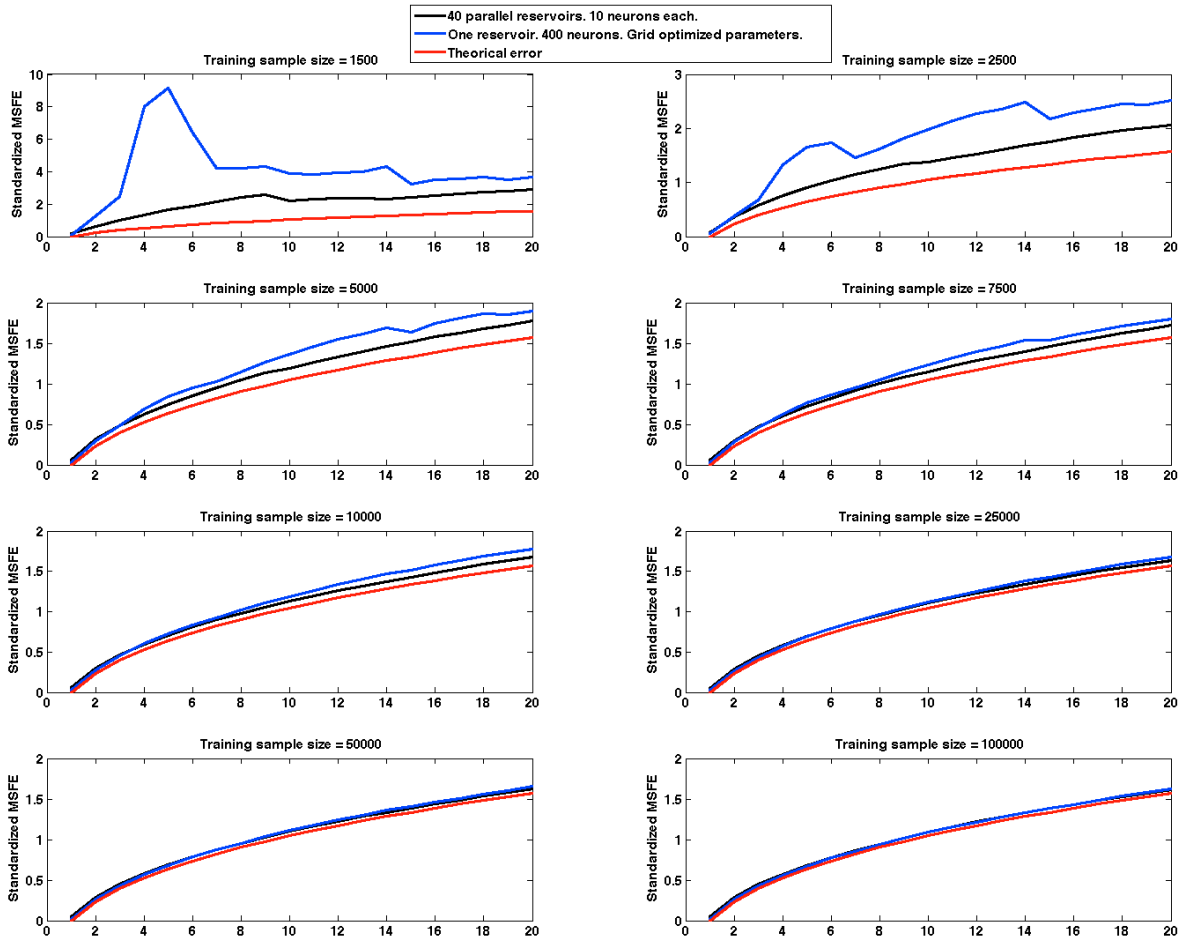

Figure 5: Comparison of the sMSFE committed for different training sample sizes by a single grid optimized TDR with 400 neurons and by a parallel array of 40 reservoirs with 10 neurons each. All the TDRs considered have been generated using the nonlinear kernel (3.4) with $p = 1$.

4.3 Market realized volatility forecasting

As we already pointed out, the VEC-GARCH processes that we worked with in the previous section are used in the context of financial econometrics to simultaneously model the time evolution of financial log-returns $\{\mathbf{z}_t\}$ and their conditional covariance matrices $\{H_t\}$. The log-returns are computed using quoted prices $\{\mathbf{S}_t\}$ via $\mathbf{z}_t := \log(\mathbf{S}_t/\mathbf{S}_{t-1})$ and they are hence observable; this is not the case for the conditional covariance matrices that are just inferred out of the model. This observation poses a problem when working in a purely non-parametric forecasting context like RC since the lack of an underlying model imposes the choice of an estimation method or a proxy for $\{H_t\}$. A natural candidate is the empirical covariance matrix using the log-returns observed during a certain time window; in this section we will work with daily quotes and the conditional covariance matrix H_t for each day t will be computed using intraday log-returns sampled at six minutes frequency. More specifically, if $\{\mathbf{z}_t^1, \ldots, \mathbf{z}_t^{d_t}\}$ are the

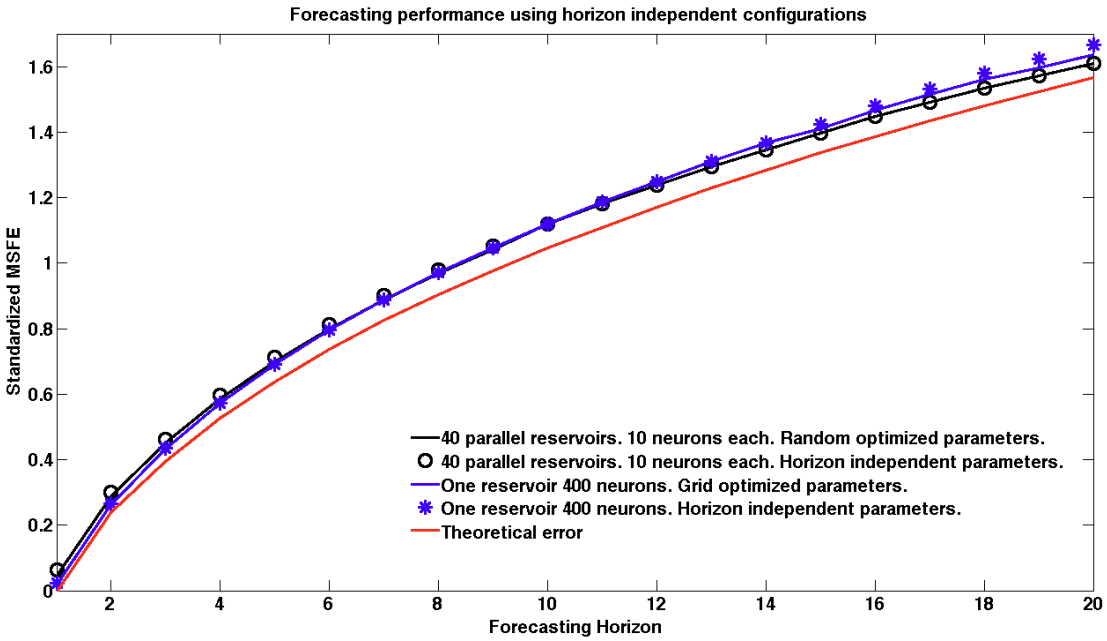

Figure 6: Comparison of the forecasting performances obtained by using horizon adapted parameter configurations and constant parameters. The constant parameters chosen are those that appear more frequently in Table 4.1. All the TDRs considered have been generated using the nonlinear kernel (3.4) with $p = 1$. The training sample size is 100 000.

intraday log-returns registered during the day t, then

$$\mathbf{z}_t = \sum_{i=1}^{d_t} \mathbf{z}_t^i \quad \text{and} \quad H_t := \frac{1}{d_t} \sum_{i=1}^{d_t} \left(\mathbf{z}_t^i - \overline{\mathbf{z}_t}\right) \left(\mathbf{z}_t^i - \overline{\mathbf{z}_t}\right)^T, \quad \text{with} \quad \overline{\mathbf{z}_t} := \frac{1}{d_t} \mathbf{z}_t.$$

The matrices H_t computed according to this prescription will be referred to as realized covariance matrices.

The dataset. We consider daily log-returns $\{\mathbf{z}_t\}$ for the NYSE quoted assets with Yahoo tickers AAPL, ABT, AXP, BA, BAC, BMY, BP, C, and CAT between January 6th, 1999 and December 31st, 2008. This yields 2 483 daily observations for which we construct realized covariance matrices $\{H_t\}$ using six minutes sampled intraday data. The resulting dataset $\{\mathbf{z}_t, H_t\}$ is then split into two parts, the first one comprising the first 2 000 points (from January 6th, 1999 to January 23rd, 2007) is used for training/estimation purposes and the remaining 483 points are used for an out-of-sample test. We emphasize that the testing period includes the high-volatility financial events of Fall 2008.

Even though we have nine different assets available, we work in a three dimensional context so that we stay in the same conditions as in Section 4.2. More specifically, we consider the 84 different possible three dimensional combinations of the nine assets and we evaluate for each of them the realized volatility forecasting abilities of the different RC configurations and compare them with a standard VEC-GARCH parametric approach.

The competing forecasting methods. We carry out realized volatility forecasting using two sets of approaches:

- **RC based forecasting:** we construct the three RC architectures spelled out in points **(ii)**, **(iii)**, and **(iv)** of Section 4.1. For all of them we carry out a random parameter optimization

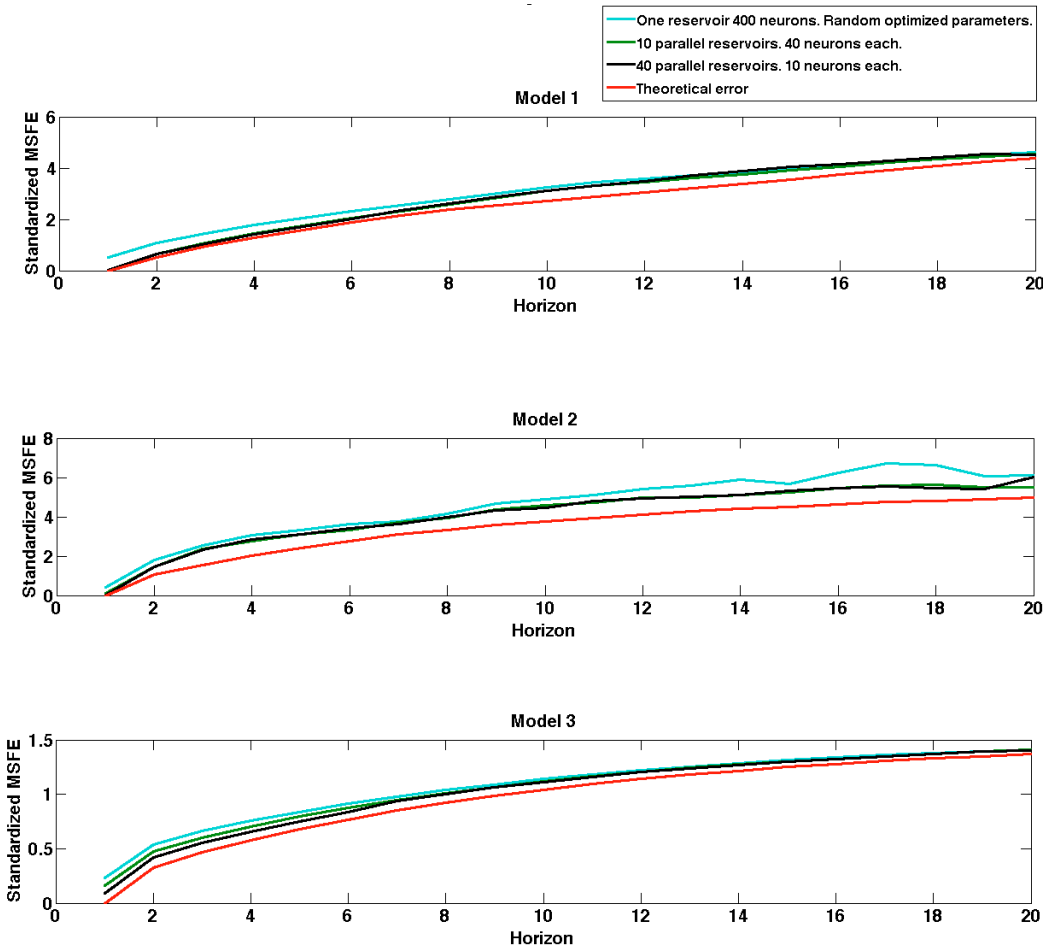

Figure 7: Forecasting performance under model misspecification. All the TDRs considered have been generated using the nonlinear kernel (3.4) with $p = 2$. Unreported analogous simulations showed that the choice $p = 1$ produced an even more acute decline in the forecasting performance in the presence of misspecified data generating processes. The training sample size is 100 000.

based on the in-sample forecasting performance within the 2 000 days long training period. This optimization is carried out for each prediction horizon and for each of the different 84 three dimensional combinations of assets. For each different RC architecture and for each forecasting horizon we select the reservoir parameters that yield the smallest training error among the 84 combinations of assets and we use that same reservoir for the testing part of the exercise for all the asset combinations. The sMSFE for each horizon averaged across the different 84 asset combinations is reported in Figure 8 only for the single operating reservoir with 400 neurons and for the 40 parallel reservoirs with 10 neurons each since other configurations offer much worse performances.

- **VEC-GARCH based forecasting:** for each three dimensional asset combination we estimate via the constrained maximum likelihood estimation algorithm in [Chre 13] a VEC(1,1) model as in (3.1) with the log-returns data included in the 2 000 days long training period. These models are used to forecast conditional covariance matrices according to (3.3) and we then compare this

forecasted values in the testing period with the corresponding realized volatility matrices. In this part of the exercise we are assuming in the spirit of [Ande 03, Laur 09, Laur 11] that realized covariances are a proxy for the conditional covariances; there are other modeling approaches that consider these two quantities as separate variables (see for instance [Shep 10, Nour 11]). The sMSFE for each horizon averaged across the different 84 asset combinations is reported in Figure 8.

The results reported in Figure 8 show the good performance and the improved universality properties of parallel RC in this forecasting task. More specifically, a parallel array of 10 reservoirs with 40 neurons each yields systematically better results than the VEC-GARCH model and the individually operating reservoir. Regarding the universality properties, we emphasize that for each forecasting horizon, it is the same array of parallel reservoirs that carries out the prediction for all 84 three dimensional combinations of assets while the VEC-GARCH models are adapted to each of them via MLE.

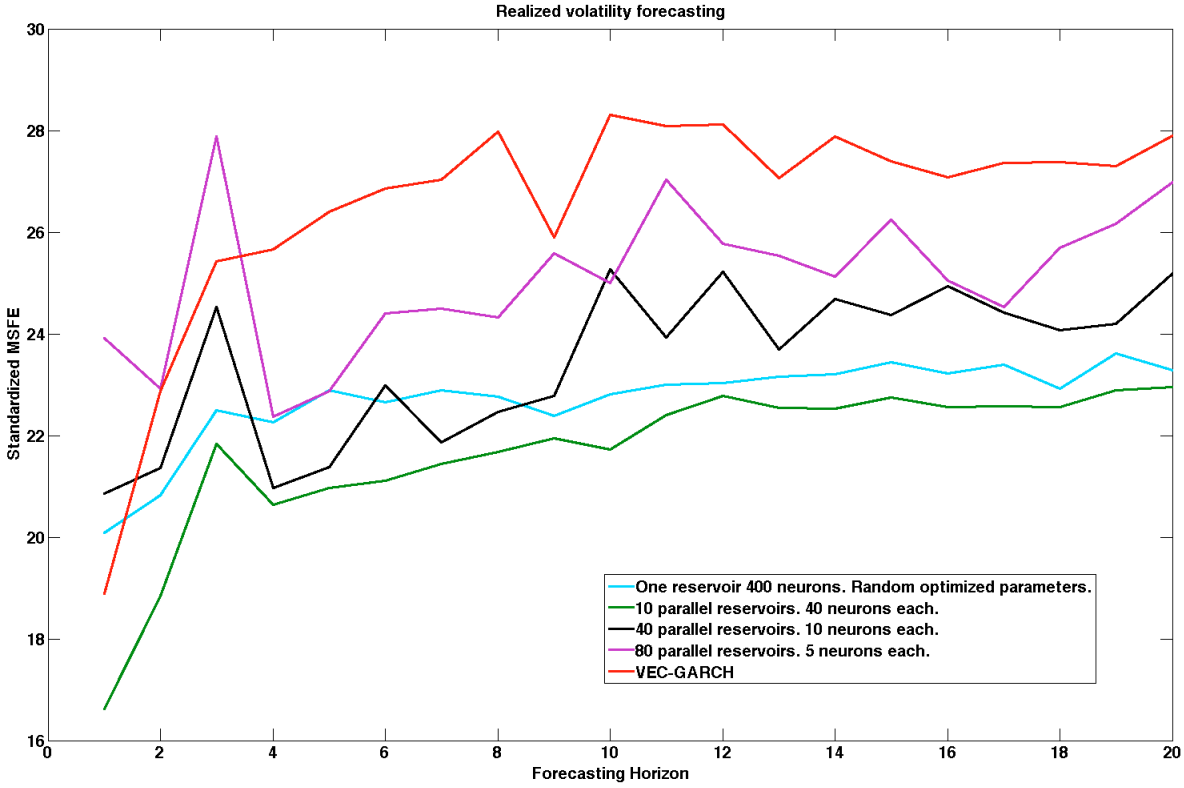

Figure 8: Average realized volatility forecasting performance using RC and VEC(1,1) models estimated via maximum likelihood (MLE). The sMSFE reported is an average over the sMSFEs associated to the realized volatility forecasting task for the 84 different three dimensional combinations of the nine assets considered. For each asset combination, forecasting with the VEC(1,1) approach is carried out with a different model estimated via MLE using the historical evolution of that particular combination. In the case of RC, for a given forecasting horizon, a single parameter set is used for all the combinations, chosen by minimizing the training error. All the TDRs considered have been generated using the nonlinear kernel (3.4) with $p = 2$.

5 Conclusions

We have shown that newly introduced parallel configurations of time-delay reservoirs (TDRs) are capable of good performances in the forecasting of the conditional covariances associated to multivariate discrete-time nonlinear stochastic processes of VEC-GARCH type as well as in the prediction of factual daily market realized volatilities computed with intraday quotes using as training input daily log-return quotes.

Our study has shown that this type of reservoir computers does not exhibit task universality in this context in the sense that reservoir parameter configurations that yield good results for a given model or forecasting horizon do not for others (see Figures 7 and 6, and Table 4.1). This observation has as negative consequences that TDRs need to be adapted for each task via a computational expensive cross validation procedure; additionally multi-tasking through parallel reading [Maas 11] is not available since it requires acceptable reservoir performance for each of the tasks that are being simultaneously processed via parallel read out.

We propose as a solution to these difficulties the use of *parallel arrays of TDRs*, each running at a different regime. This configuration reduces the computational effort necessary to adapt to the task at hand by using random optimization. It presents a better performance for small training sample sizes and is less sensitive to modifications in the task hence making it more universal and appropriate for multi-tasking.

References

[Adya 98] M. Adya and F. Collopy. "How effective are neural networks at forecasting and prediction? A review and evaluation". *Journal of Forecasting*, Vol. 17, pp. 481–495, 1998.

[Alme 91] C. de Almeida and P. A. Fishwick. "Time series forecasting using neural networks vs. Box-Jenkins methodology". *Simulation*, Vol. 57, No. 5, pp. 303–310, Nov. 1991.

[Alta 03] A. Altay-Salih, M. c. Pinar, and S. Leyffer. "Constrained nonlinear programming for volatility estimation with GARCH models". *SIAM Rev.*, Vol. 45, No. 3, pp. 485—-503 (electronic), 2003.

[Ande 03] T. G. Andersen, T. Bollerslev, F. X. Diebold, and P. Labys. "Modeling and Forecasting Realized Volatility". *Econometrica*, Vol. 71, No. 2, pp. 579–625, March 2003.

[Appe 11] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer. "Information processing using a single dynamical node as complex system.". *Nature communications*, Vol. 2, p. 468, Jan. 2011.

[Atiy 00] A. F. Atiya and A. G. Parlos. "New results on recurrent network training: unifying the algorithms and accelerating convergence.". *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, Vol. 11, No. 3, pp. 697–709, Jan. 2000.

[Balk 00] S. D. Balkin and J. K. Ord. "Automatic neural network modeling for univariate time series". *International Journal of Forecasting*, Vol. 16, No. 4, pp. 509–515, Oct. 2000.

[Boll 86] T. Bollerslev. "Generalized autoregressive conditional heteroskedasticity". *Journal of Econometrics*, Vol. 31, No. 3, pp. 307–327, 1986.

[Boll 88] T. Bollerslev, R. F. Engle, and J. M. Wooldridge. "A capital asset pricing model with time varying covariances". *Journal of Political Economy*, Vol. 96, pp. 116–131, 1988.

[Box 76] G. E. P. Box and G. M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

[Boyd 85] S. Boyd and L. Chua. "Fading memory and the problem of approximating nonlinear operators with Volterra series". *IEEE Transactions on Circuits and Systems*, Vol. 32, No. 11, pp. 1150–1161, Nov. 1985.

[Broc 02] P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting*. Springer, 2002.

[Broc 06] P. J. Brockwell and R. A. Davis. *Time Series: Theory and Methods*. Springer-Verlag, 2006.

[Brun 13] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer. "Parallel photonic information processing at gigabyte per second data rates using transient states". *Nature Communications*, Vol. 4, No. 1364, 2013.

[Chre 13] S. Chrétien and J.-P. Ortega. "Multivariate GARCH estimation via a Bregman-proximal trust-region method". *Computational Statistics and Data Analysis*, Vol. To appear, 2013.

[Cron 05] S. F. Crone. "Stepwise Selection of Artificial Neural Network Models for Time Series Prediction". *Journal of Intelligent Systems*, Vol. 15, 2005.

[Croo 07] N. Crook. "Nonlinear transient computation". *Neurocomputing*, Vol. 70, pp. 1167–1176, 2007.

[Damb 12] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar. "Information processing capacity of dynamical systems". *Scientific reports*, Vol. 2, No. 514, 2012.

[Dupo 12] F. Duport, B. Schneider, A. Smerieri, M. Haelterman, and S. Massar. "All-optical reservoir computing". *Optics Express*, Vol. 20, No. 20, pp. 22783–95, 2012.

[Engl 82] R. F. Engle. "Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation". *Econometrica*, Vol. 50, No. 4, pp. 987–1007, 1982.

[Gour 97] C. Gouriéroux. *ARCH models and financial applications. Springer Series in Statistics*, Springer-Verlag, New York, 1997.

[Grig 13] L. Grigoryeva and J.-P. Ortega. "Hybrid forecasting with estimated temporally aggregated linear processes". *http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2148895*, 2013.

[Guti 12] J. M. Gutiérrez, D. San-Martín, S. Ortín, and L. Pesquera. "Simple reservoirs with chain topology based on a single time-delay nonlinear node". In: *20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pp. 13–18, 2012.

[Hami 94] J. D. Hamilton. *Time series analysis*. Princeton University Press, Princeton, NJ, 1994.

[Ilie 07] I. Ilies, H. Jaeger, O. Kosuchinas, M. Rincon, V. Sakenas, and N. Vaskevicius. "Stepping forward through echoes of the past: forecasting with Echo State Networks". Tech. Rep., 2007.

[Jaeg 01] H. Jaeger. "The 'echo state' approach to analysing and training recurrent neural networks". Tech. Rep., German National Research Center for Information Technology, 2001.

[Jaeg 04] H. Jaeger and H. Haas. "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication". *Science*, Vol. 304, No. 5667, pp. 78–80, 2004.

[Jaeg 07] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert. "Optimization and applications of echo state networks with leaky-integrator neurons". *Neural networks*, Vol. 20, No. 3, pp. 335–352, 2007.

[Kaas 96] I. Kaastra and M. Boyd. "Designing a neural network for forecasting financial and economic time series". *Neurocomputing*, Vol. 10, pp. 215–236, 1996.

[Larg 12] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutierrez, L. Pesquera, C. R. Mirasso, and I. Fischer. "Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing". *Optics Express*, Vol. 20, No. 3, p. 3241, Jan. 2012.

[Laur 09] S. Laurent and J. Rombouts. "On loss functions and ranking forecasting performances of multivariate volatility models". *Sciences-New York*, 2009.

[Laur 11] S. Laurent, J. V. K. Rombouts, and F. Violante. "On the forecasting accuracy of multivariate GARCH models". *Journal of Applied Econometrics*, pp. n/a–n/a, Apr. 2011.

[Luko 09] M. Lukoševičius and H. Jaeger. "Reservoir computing approaches to recurrent neural network training". *Computer Science Review*, Vol. 3, No. 3, pp. 127–149, 2009.

[Lutk 05] H. Lütkepohl. *New Introduction to Multiple Time Series Analysis*. Springer-Verlag, Berlin, 2005.

[Maas 00] W. Maass and E. D. Sontag. "Neural Systems as Nonlinear Filters". *Neural Computation*, Vol. 12, No. 8, pp. 1743–1772, Aug. 2000.

[Maas 02] W. Maass, T. Natschläger, and H. Markram. "Real-time computing without stable states: a new framework for neural computation based on perturbations". *Neural Computation*, Vol. 14, pp. 2531–2560, 2002.

[Maas 11] W. Maass. "Liquid state machines: motivation, theory, and applications". In: S. S. Barry Cooper and A. Sorbi, Eds., *Computability In Context: Computation and Logic in the Real World*, Chap. 8, pp. 275–296, 2011.

[Meye 00] C. Meyer. *Matrix Analysis and Applied Linear Algebra Book and Solutions Manual*. Society for Industrial and Applied Mathematics, 2000.

[Nour 11] D. Noureldin, N. Shephard, and K. K. Sheppard. "Multivariate high-frequency-based volatility (HEAVY) models". *Journal of Applied Econometrics*, pp. n/a–n/a, Aug. 2011.

[Orti 12] S. Ortin, L. Pesquera, and J. M. Gutiérrez. "Memory and nonlinear mapping in reservoir computing with two uncoupled nonlinear delay nodes". In: *Proceedings of the European Conference on Complex Systems*, pp. 895–899, 2012.

[Paqu 12] Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar. "Optoelectronic reservoir computing". *Scientific reports*, Vol. 2, p. 287, Jan. 2012.

[Pasc 01] L. Pascual, J. Romo, and E. Ruiz. "Effects of parameter estimation on prediction densities: a bootstrap approach". *International Journal of Forecasting*, Vol. 17, No. 1, pp. 83–103, Jan. 2001.

[Pasc 04] L. Pascual, J. Romo, and E. Ruiz. "Bootstrap predictive inference for ARIMA processes". *Journal of Time Series Analysis*, Vol. 25, No. 4, pp. 449–465, July 2004.

[Pasc 05] L. Pascual, J. Romo, and E. Ruiz. "Bootstrap prediction intervals for power-transformed time series". *International Journal of Forecasting*, Vol. 21, No. 2, pp. 219–235, Apr. 2005.

[Pasc 06] L. Pascual, J. Romo, and E. Ruiz. "Bootstrap prediction for returns and volatilities in GARCH models". *Computational Statistics & Data Analysis*, Vol. 50, No. 9, pp. 2293–2312, May 2006.

[Roda 11] A. Rodan and P. Tino. "Minimum complexity echo state network.". *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, Vol. 22, No. 1, pp. 131–44, Jan. 2011.

[Rodr 09] A. Rodriguez and E. Ruiz. "Bootstrap prediction intervals in state-space models". *Journal of Time Series Analysis*, Vol. 30, No. 2, pp. 167–178, March 2009.

[Ruiz 02] E. Ruiz and L. Pascual. "Bootstrapping Financial Time Series". *Journal of Economic Surveys*, Vol. 16, No. 3, pp. 271–300, July 2002.

[Shep 10] N. Shephard and K. Sheppard. "Realising the future: forecasting with high-frequency-based volatility (HEAVY) models". *Journal of Applied Econometrics*, Vol. 25, No. 2, pp. 197–231, March 2010.

[Tikh 43] A. N. Tikhonov. "On the stability of inverse problems". *Dokl. Akad. Nauk SSSR*, Vol. 39, No. 5, pp. 195–198, 1943.

[Tsay 10] R. S. Tsay. *Analysis of Financial Time Series*. Wiley, 2010.

[Vers 07] D. Verstraeten, B. Schrauwen, M. DHaene, and D. Stroobandt. "An experimental unification of reservoir computing methods". *Neural Networks*, Vol. 20, pp. 391–403, 2007.

[Wyff 08] F. Wyffels, B. Schrauwen, and D. Stroobandt. "Using reservoir computing in a decomposition approach for time series prediction". 2008.

[Wyff 10] F. Wyffels and B. Schrauwen. "A comparative study of Reservoir Computing strategies for monthly time series prediction". *Neurocomputing*, Vol. 73, No. 10, pp. 1958–1964, 2010.

[Zhan 98] G. Zhang, B. Eddy Patuwo, and M. Y. Hu. "Forecasting with artificial neural networks: the state of the art". *International Journal of Forecasting*, Vol. 14, No. 1, pp. 35–62, March 1998.