Journal of Computational and Applied Mathematics 236 (2011) 411–421

Contents lists available at SciVerse ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Investigating asymptotic properties of vector nonlinear time series models

Ioana Banicescu [a,*], Ricolindo L. Cariño [b], Jane L. Harvill [d], John Patrick Lestrade [c]

[a] Department of Computer Science and Engineering, Mississippi State University, United States
[b] Center for Advanced Vehicular Systems, United States
[c] Department of Physics and Astronomy, Mississippi State University, United States
[d] Department of Statistical Science, Baylor University, United States

ARTICLE INFO

Keywords:
Vector FCAR models
Dynamic loop scheduling

ABSTRACT

Analyses and simulations of vector nonlinear time series typically run into weeks or even months because the methods used are computationally intensive. Statisticians have been known to base empirical results on a relatively small number of simulation replications, sacrificing precision and reliability of results in the interest of time and productivity. The simulations are amenable for parallelization. However, parallel computing technology has not yet been widely used in this specific research area. This paper proposes an approach to the parallelization of statistical simulation codes to address the challenge of long running times. Requiring minimal code revision, this approach takes advantage of recent advances in dynamic loop scheduling to achieve high performance on general-purpose clusters, even with the presence of unpredictable load imbalance factors. Preliminary results of applying this approach in the simulation of normal white noise and threshold autoregressive model obtains efficiencies in the range 95%–98% on 8–64 processors. Furthermore, previously unobserved properties of the statistical procedures for the models are uncovered by the simulation.

1. Introduction

Correct application of traditional statistical analysis methods rely heavily on the assumption that observations are independent. However, there are many situations when the independence assumption is violated—for example, time series data. Time series models describe the behavior of the present as a function of past values of the series and random noise. When multiple series are observed at each time point, the series is called a vector time series. Vector time series are prevalent in many areas and have been shown to out-perform their univariate counterparts in modeling co-dependent characteristics; see for example, [1,2] or [3]. Examples of vector time series are common in applications such as population dynamics, economics and finance, physics and astronomy, or meteorology.

Nonlinear time series models are quite effective in modeling complex dynamic systems. A particularly powerful model in vector nonlinear time series is the vector functional coefficient autoregressive (VFCAR) model. Because of its general nature, statistical inference using this model is nonparametric. On a single processor, numerical investigation of statistical properties of estimators or inferential procedures requires execution times of weeks, even months. Consequently statisticians have based empirical results on a relatively small number of replications and on less than optimal computing parameters, sacrificing precision, accuracy, and possibly compromising the reliability of results in the interest of time or "computational

* Corresponding author.
E-mail address: ioana@cse.msstate.edu (I. Banicescu).

0377-0427/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2011.07.018

expediency" [4]. Techniques which require thousands of replications for accuracy often have at most a few hundred. Fortunately the replications are amenable for computation in parallel. Thus parallel processing technology can be exploited to enable the extensive simulation of a variety of models within reasonable running time limits. This paper demonstrates an approach to the simulation of such models which takes advantage of recent advances in dynamic loop scheduling on clusters of workstations, in order to shorten the simulation time.

The remainder of this paper is organized as follows. Section 2 defines the vector functional coefficient autoregressive (VFCAR) models. In Section 2.1, the most common statistical method for estimating the parameters in VFCAR models are discussed, along with the computational issues related to that procedure. Section 2.2 presents a computational bootstrap hypothesis testing procedure for model selection, and a discussion of using the nonparametric VFCAR model for predicting M steps into the future. Section 3 provides an approach to facing the challenges in implementing the procedures in the previous sections numerous times, as would be done in a typical Monte Carlo study for any of the procedures in the earlier sections. This section also shows preliminary results which demonstrate the high parallel performance achieved by this approach. Section 4 discusses advances in understanding of the structure of the power function of the test that becomes possible by applying the computing methods of Section 3. Concluding remarks are given in Section 5.

2. The VFCAR model

In this section, we define the vector functional coefficient autoregressive (VFCAR) model. For specifics, the reader is referred to the series of papers in [5,6] or the paper in [7]. In what follows, the univariate time series observed at time t is denoted $Y_t,\ t = 1, 2, \ldots, T$. When k multiple processes are concurrently observed, individual processes are denoted $Y_{i,t}$, where $i = 1, \ldots, k$ indicates the process, and the set of observations on all k processes at time t, is denoted by the bold-face $\boldsymbol{Y}_t = (Y_{1,t}, \ldots, Y_{k,t})'$, where $'$ denotes transpose.

A nonlinear autoregressive model is one where the time series at $Y_{i,t}$ is written as some measurable, real-valued function η_i of past values of each component of the series plus some error term; that is,

$$Y_{i,t} = \eta_i(\boldsymbol{Y}_{t-1}, \ldots, \boldsymbol{Y}_{t-p}) + \varepsilon_{i,t}, \tag{1}$$

where, for each i, the series $\{\varepsilon_{i,t}\}_{t=1}^{T}$ is white noise, independent of $\{\boldsymbol{Y}_t\}_{t=1}^{T}$. We consider the case where all η_i in Eq. (1) are additive; that is,

$$\eta_i = \sum_{j=1}^{p} \boldsymbol{f}_i^{(j)}(\boldsymbol{Z}_t)\boldsymbol{Y}_{t-j}, \tag{2}$$

(j) is a superscript, and $\boldsymbol{Z}_t$ is an exogenous predictor, e.g., time, or lagged values of the series $\boldsymbol{Y}$. There is little or no information about the specific forms of the $\boldsymbol{f}_i^{(j)}$. When components of Y are correlated and the error terms have positive contemporaneous correlation, simultaneously nonparametrically estimating the $\boldsymbol{f}_i^{(j)}$ using local regression methods provides improved efficiency compared to fitting separate (unrelated) univariate models.

Combining Eqs. (1) and (2) yields the vector functional coefficient autoregressive model of order p, VFCAR(p), defined by

$$\boldsymbol{Y}_t = \boldsymbol{f}^{(0)}(\boldsymbol{Z}_t) + \sum_{j=1}^{p} \boldsymbol{f}^{(j)}(\boldsymbol{Z}_t)\boldsymbol{Y}_{t-j} + \boldsymbol{\varepsilon}_t, \quad t = p+1, \ldots, T, \tag{3}$$

where $\boldsymbol{Z}_t$ is the functional variable of dimension $m \geq 1$ and the $\boldsymbol{\varepsilon}_t$ are independent, identically distributed random variables with mean vector $\boldsymbol{0}$ and $k \times k$ covariance matrix $\boldsymbol{\Sigma}$, $\boldsymbol{\varepsilon}_t$ independent of $\boldsymbol{Y}_s$ and $\boldsymbol{Z}_s$ for all $s < t$. The $\boldsymbol{Z}_t$ are as defined for Eq. (2), and $\boldsymbol{f}^{(j)},\ j = 1, \ldots, p$ are $k \times k$ matrices with elements $[f_{il}^{(j)}]$ that are real-valued measurable functions that change as a function of $\boldsymbol{Z}_t$ and which have continuous second derivatives. If the variable $\boldsymbol{Z}_t$ is lagged values of $\boldsymbol{Y}_{t-d}$ with $d \leq p$, the inclusion of the intercept function $\boldsymbol{f}^{(0)}$ results in a non-identifiable model, giving unstable estimates of the functional coefficients. In such cases, the intercept should be omitted.

2.1. Estimation of VFCAR parameters

Since there is little or no information about the form of the coefficients $\boldsymbol{f}^{(j)}$ in (3), estimation of the $\boldsymbol{f}^{(j)},\ j = 1, \ldots, p$ at observations $\{\boldsymbol{Z}_t, \boldsymbol{Y}_t\}_{t=1}^{T}$ is accomplished using locally constant or local linear multivariate regression in a neighborhood of $\boldsymbol{Z}_t$ with a specified kernel and numerically determined bandwidth matrix. Section 2.1.1 outlines the method for bandwidth and autoregressive (AR) order (p) selection.

At time t, denote the fitted AR order by p^*, and the kp^*-vector of predictors by $\boldsymbol{X}_t$; that is, let $\boldsymbol{X}_t = [\boldsymbol{1}, \boldsymbol{Y}_{t-1}, \boldsymbol{Y}_{t-2}, \ldots, \boldsymbol{Y}_{t-p^*}]'$, where $\boldsymbol{Y}_{t-j} = [Y_{1,t-j}, Y_{2,t-j}, \ldots, Y_{k,t-j}]'$, for $j = 1, \ldots, p^*$, and let $\boldsymbol{f}(\boldsymbol{Z}_t)$ be defined as $\boldsymbol{f}(\boldsymbol{Z}_t) = [\boldsymbol{f}^{(0)}, \boldsymbol{f}^{(1)}(\boldsymbol{Z}_t), \ldots, \boldsymbol{f}^{(p^*)}(\boldsymbol{Z}_t)]'$. Then model (3) can be written as

$$\boldsymbol{Y}_t = \boldsymbol{f}(\boldsymbol{Z}_t)\boldsymbol{X}_t + \boldsymbol{\varepsilon}_t, \quad t = p^* + 1, \ldots, T.$$

For the sake of discussion, we temporarily restrict the dimension of $\boldsymbol{Z}_t$ to $m = 1$. Since all elements of $\boldsymbol{f}$ have continuous second-order derivatives, each $f_{il}^{(j)}(\cdot)$ may be approximated locally at z_0 by a linear function $f_{il}^{(j)}(z) = \alpha_{il}^{(j)} + \beta_{il}^{(j)}(z - z_0)$. Writing the coefficient matrices in the form $[\boldsymbol{\alpha} \mid \boldsymbol{\beta}]$, the local linear least squares kernel estimator of $\boldsymbol{f}(Z_t)$ is defined as $\hat{\boldsymbol{f}}(z_0) = \hat{\boldsymbol{\alpha}}$, where $[\hat{\boldsymbol{\alpha}} \mid \hat{\boldsymbol{\beta}}]$ is the solution to $[\boldsymbol{\alpha} \mid \boldsymbol{\beta}]$ that minimizes the sum of weighted squares

$$\sum_{t=p^*+1}^{T} \left[\boldsymbol{Y}_t - [\boldsymbol{\alpha} \mid \boldsymbol{\beta}] \begin{pmatrix} \boldsymbol{X}_t \\ \boldsymbol{U}_t \end{pmatrix}\right] \left[\boldsymbol{Y}_t - [\boldsymbol{\alpha} \mid \boldsymbol{\beta}] \begin{pmatrix} \boldsymbol{X}_t \\ \boldsymbol{U}_t \end{pmatrix}\right]' K_h(Z_t - z_0). \tag{4}$$

Here $\boldsymbol{U}_t$ is a partitioned matrix, with the first partition being $[\boldsymbol{X}_{p^*+1}, \ldots, \boldsymbol{X}_T]'$, and the second partition is the first multiplied by $(Z_t - z_0)$. The function K is a specified kernel, $h > 0$ is the bandwidth, and $K_h(u) = h^{-1}K(u/h)$. If the intent is to use the VFCAR model for model testing (see Section 2.2), a boundary kernel is recommended to avoid trimming the functional coefficient estimates. The solution to (4) is a straightforward least squares problem,

$$\begin{bmatrix} \hat{\boldsymbol{\alpha}} \\ \hat{\boldsymbol{\beta}} \end{bmatrix} = (\boldsymbol{U}'\boldsymbol{W}\boldsymbol{U})^{-1}\boldsymbol{U}'\boldsymbol{W}\boldsymbol{Y},$$

where $\boldsymbol{W} = \mathrm{diag}\{K_h(Z_{p^*+1} - z_0), \ldots, K_h(Z_n - z_0)\}$. If the dimension of Z is $m > 1$, the first kp^* rows of $\boldsymbol{U}_t$ are the product of $\boldsymbol{X}_t$ and $(Z_{1,t} - z_{1,0})$, the second kp^* rows of $\boldsymbol{U}_t$ are the product of $\boldsymbol{X}_t$ and $(Z_{2,t} - z_{2,0})$, *etc.* In this case, K is a specified m-variate kernel function, $\boldsymbol{H}^{1/2}$ is the bandwidth matrix, and $K_{\boldsymbol{H}}(\boldsymbol{u}) = |\boldsymbol{H}|^{-1/2}K(\boldsymbol{H}^{-1/2}\boldsymbol{u})$.

2.1.1. Bandwidth matrix

The accuracy and precision of the estimators in the previous section are highly affected by the choice of bandwidth $\boldsymbol{H}^{1/2}$. The optimal bandwidth, along with the optimal value for p, are selected numerically using a modified multi-fold cross-validation criterion, that numerically minimizes accumulated prediction error. This method has been successfully used for nonlinear time series modeling in the univariate framework [8,4,5] and in the multivariate framework [5,6].

Let r and Q be two positive integers such that $T > rQ$. To find an optimal value for the bandwidth, the first Q subseries of lengths $T - rq$ $(q = 1, \ldots, Q)$ are used to estimate the unknown coefficient functions. Using the estimated coefficients, one-step forecasting errors are computed based on the next section of length r of the time series. The value chosen h_{opt} for h is the value minimizing the average mean square prediction error (APE), defined by

$$\mathrm{APE}(h) = \sum_{q=1}^{Q} \mathrm{APE}_q(h), \tag{5}$$

where for $q = 1, \ldots, Q$, $\mathrm{APE}_q(h)$ is trace of the estimated error covariance matrix; that is, for $\boldsymbol{e} = \boldsymbol{Y}_q^* - \hat{\boldsymbol{f}}_q(\boldsymbol{Z}_t)\boldsymbol{X}_q^*$, $\mathrm{APE}_q(h) = \mathrm{trace}\left(\boldsymbol{e}\boldsymbol{e}'\right)$, where $\boldsymbol{Y}_q^*$ and $\boldsymbol{X}_q^*$ are the matrices containing the observations indexed from $T - qr + 1$ to $T - qr + r$ and the $\{\hat{\boldsymbol{f}}_q(\cdot)\}$ are computed from the sample $\{(\boldsymbol{Z}_t, \boldsymbol{Y}_t),\ 1 \le t \le T - qr\}$ with bandwidth $h = [T/(T - qr)]^{1/5}$.

Cai et al. [4] present a univariate version of the methods described above with $Q = 4$ and "$r = [0.1T]$ rather than $r = 1$ simply because of computational expediency". The two papers in the literature in [5,6] follow suit. Since the number of computations increases exponentially when the method moves from the univariate framework to the vector framework, it is here we encounter the first computational challenge in applying the methods.

2.1.2. Smoothing variable selection

Smoothing variable selection can also affect the model's performance. Obviously, if knowledge of the physical process is available, that knowledge should be a guiding force in selecting $\boldsymbol{Z}_t$. However, if no prior knowledge is available, the smoothing variable can be chosen by data-driven methods. Fan et al. [9] explore choosing a univariate smoothing variable as a linear function of given explanatory variables, i.e. a single index model, according to some optimal criterion. Harvill and Ray [6] choose the functional variable as the component of Y at lag d that minimizes the accumulated prediction error defined in (5).

2.2. Model testing procedure

Specific choices $\boldsymbol{g}$ for the elements of the matrices $\boldsymbol{f}$ in the VFCAR model can result in parametric time series models. This feature of the VFCAR model is particularly useful in validating specific choices of functional forms to model the observed data. Consider testing

$$H_0 : \boldsymbol{f}^{(l)}(\boldsymbol{Z}) = \boldsymbol{g}^{(l)}(\boldsymbol{Z}; \boldsymbol{\theta}) \quad \text{vs.} \quad H_1 : \boldsymbol{f}^{(l)}(\boldsymbol{Z}) \neq \boldsymbol{g}^{(l)}(\boldsymbol{Z}; \boldsymbol{\theta}), \tag{6}$$

where $\boldsymbol{g}^{(l)}(\cdot; \boldsymbol{\theta})$ is a given family of functions indexed by unknown parameter vector $\boldsymbol{\theta}$.

The power of the test is the probability that H_1 is correctly declared as truth. To find a mathematical expression for the power of the test, the probability distribution of the test statistic is modified to satisfy the conditions of H_1. In nonparametric settings, it is often impossible to develop an analytic expression for the distribution of the test statistic, and thus to compute the power of the test.

2.2.1. Test statistic

Let $\hat{\boldsymbol{\theta}}$ be an estimator of $\boldsymbol{\theta}$ in $\boldsymbol{g}^{(l)}(\boldsymbol{Z};\boldsymbol{\theta})$. Under the null hypothesis, the estimated error covariance matrix is

$$\mathrm{RSS}_0 = n^{-1}\left[\boldsymbol{Y} - \boldsymbol{g}(\boldsymbol{Z}_t;\hat{\boldsymbol{\theta}})\boldsymbol{X}\right]\left[\boldsymbol{Y} - \boldsymbol{g}(\boldsymbol{Z}_t;\hat{\boldsymbol{\theta}})\boldsymbol{X}\right]'. \tag{7}$$

Under the VFCAR model, the estimated error covariance matrix is

$$\mathrm{RSS}_1 = n^{-1}\left[\boldsymbol{Y} - \hat{\boldsymbol{f}}(\boldsymbol{Z}_t)\boldsymbol{X}\right]\left[\boldsymbol{Y} - \hat{\boldsymbol{f}}(\boldsymbol{Z}_t)\boldsymbol{X}\right]'. \tag{8}$$

Then the likelihood ratio test statistic is given by

$$\boldsymbol{\Lambda} = \left[\frac{\mathrm{trace}(\mathrm{RSS}_0)}{\mathrm{trace}(\mathrm{RSS}_1)}\right]^{1/2}, \quad \text{and} \quad \boldsymbol{L} = \frac{1-\boldsymbol{\Lambda}}{\boldsymbol{\Lambda}}. \tag{9}$$

The null hypothesis is rejected for large values of $\boldsymbol{L}$. If a boundary kernel is not used, the residual sums of squares should be computed using a trimmed version to avoid inflating the statistic due to edge effects.

2.2.2. Bootstrap procedure

The distribution of the test statistic (9) is difficult to derive for finite sample size. Even in the univariate case, the best asymptotic approximation for the null distribution does not necessarily give a good approximation for finite samples (see p. 412 of [10]). An alternative for finding p-values for testing (6) uses bootstrap resampling. Harvill and Ray [6] extend the methods of [4] to the vector case series, and present an investigation of the properties of the test. The bootstrap procedure is outlined below. For a more thorough discussion, the reader is referred to the originating papers cited previously.

1. Sample bootstrap residuals $\{\boldsymbol{\varepsilon}_t\}_{t=1}^{T}$ from the empirical distribution of the centered residuals $\{\hat{\boldsymbol{\varepsilon}}_t - \bar{\hat{\boldsymbol{\varepsilon}}}\}_{t=1}^{T}$, where $\bar{\hat{\boldsymbol{\varepsilon}}}$ is the k-vector of means of the component residuals; that is,

$$\bar{\hat{\boldsymbol{\varepsilon}}}_i = T^{-1}\sum_{t=1}^{T}\hat{\boldsymbol{\varepsilon}}_{t,i}, \quad i = 1,\ldots,k,$$

   where $\boldsymbol{\varepsilon}_t = \boldsymbol{Y}_t - \hat{\boldsymbol{f}}(\boldsymbol{Z}_t)\boldsymbol{X}_t,\ t = p^* + 1,\ldots,T$.
2. Compute the bootstrap sample

$$\boldsymbol{Y}_t^B = \boldsymbol{g}(\boldsymbol{Z}_t;\hat{\boldsymbol{\theta}})\boldsymbol{X}_t + \boldsymbol{\varepsilon}_t^B.$$

3. Replacing the $\boldsymbol{Y}$ in (7) and (8) with $\boldsymbol{Y}_t^B$, compute the bootstrap test statistics $\boldsymbol{L}^B$.
4. Reject the null hypothesis in (6) when $\boldsymbol{L}$ is greater than the upper α point of the conditional distribution of $\boldsymbol{L}^B$ given the original data.

The p-value of the test is the relative frequency of the event $\{\boldsymbol{L}^B \geq \boldsymbol{L}\}$ in the replications of the bootstrap sampling.

A careful reading of the literature which introduces bootstrap tests reveals that computational expediency allowed only limited numerical studies of the properties of the estimators. For example, in most cases, the number of replications was small (200 or 400). In all of the papers, the time series lengths T were limited in either the number of lengths considered, or size of the lengths. The number of nonlinear behaviors examined was few, with the exception of [6], who consider only moderate series lengths for vector nonlinear series ($T = 100$ and 250) and a few values for error cross-correlation. However, the numerical properties of statistical estimators and hypothesis tests are best examined for a variety of series lengths across multiple behaviors and parameter perturbations.

2.3. Forecasting

Forecasting with nonparametrically fitted models necessitates the use of computationally intensive techniques. Even though these models can outperform their simpler counterparts [11,5], the mathematical and computational complexity incurred from using these models for forecasting is the primary reason that currently only three papers exist in the statistics literature. For the model given by

$$Y_t = f_0(Z_t) + \sum_{j=1}^{p} f_j(Z_t)Y_{t-j} + \varepsilon_t, \quad t = p+1,\ldots,T, \tag{10}$$

the goal of prediction is to find an estimator of the conditional expectation

$$\mathrm{E}[Y_{T+M} \mid Y_T, \ldots, Y_{T-p}] = \sum_{j=1}^{p} f_j(Z_{T+M})\hat{Y}_{T+M-j}, \tag{11}$$

assuming $f_j(\cdot)$ is known and Z_t is exogenous. The three methods described below for finding an M-step predictive estimate hinge on different ways of dealing with this complication.

2.3.1. Naive plug-in predictor

The naive plug-in predictor ignores the fact that the expectation in (11) is not a linear function of Y_{t+M-j} for $M \geq 2$ and simply plugs $\hat{Y}_{t+M-j}$ into the forecast equation. The form of the functional coefficient is determined using only the within-sample series values. In other words, for $Z_t = Y_{t-d}$,

$$\hat{Y}_{T+M} = \sum_{j=1}^{p^*} \hat{f}_j(\hat{Y}_{T+M-d})\hat{Y}_{T+M-j}, \tag{12}$$

where $\hat{Y}_t = Y_t,\ t \leq T$ and $\hat{f}_j(\cdot)$ are the values $\hat{\alpha}_j$ minimizing (4).

2.3.2. Bootstrap predictor

Like the plug-in estimator, the bootstrap prediction method uses only within-sample values to compute the functional coefficients and evaluates these coefficients at the predicted values. However, the predicted values are obtained as $\hat{Y}_{T+M} = \sum_{j=1}^{p^*} \hat{f}_j(\hat{Y}_{T+M-d})\hat{Y}_{T+M-j} + \epsilon^b$, where ϵ^b is a bootstrapped value of the within-sample residuals from the fitted FCAR model. The bootstrapped forecast is obtained for $b = 1, \ldots, B$ and the average across all bootstrap predictions is used as the M-step ahead point forecast. The predictive density of Y_{t+M} can be obtained using the complete set of bootstrap predictions. Note that care must be taken when $\hat{Y}_{t+M-d}$ falls outside or near the boundary of the range of the original Y_{t-d}, as the estimated functional coefficients may be very unreliable in this case.

2.3.3. Multi-stage predictor

The multi-stage predictor is a modification of the naive predictor in which the functional coefficients are updated at each step to incorporate the information from Y_t encoded in the predicted response at time $T+j,\ j = 1, \ldots, M-1$. Specifically,

$$\hat{Y}_{T+M} = \sum_{j=1}^{p^*} \hat{f}_j^M(\hat{Y}_{T+M-d})\hat{Y}_{T+M-j}, \tag{13}$$

where $\hat{Y}_t = Y_t,\ t \leq T$ and $\hat{f}_j^M$ are the values $\hat{\alpha}_j$ minimizing

$$\sum_{t=s^*+1}^{T+M-1} \left\{ Y_t - \sum_{j=1}^{p} \Big[\alpha_j + \beta_j(Z_t - z) \Big] Y_{t-j} \right\} K_h(Z_t - z). \tag{14}$$

Numerical studies of these forecasting methods are presented in [5], having the same limitations as the hypothesis testing computational studies.

3. Simulation

Fig. 1 illustrates a high-level flowchart of a simulation procedure to investigate the statistical properties of the methods used in the VFCAR model. The procedure synthetically generates a large number of time series samples for a given combination of data model, error cross correlation and series length. Then, for each sample, the procedure estimates the model parameters and applies the hypothesis test.

Although the procedure above can be accomplished in the *R* system [12], the total computation time would be enormous when using a large number of replications, more bootstrap samples, and more parameter choices. One solution to reduce this computation time is by utilizing parallel processing technology. Thus, compromises in parameter selection for the sake of expediency become less important. Simple parallel implementation of work in the *R* language is developed in [13]. However, because of the nature of the FCAR computations, a Fortran implementation still offers significant improvement in the computing time.

An initial Fortran 90 implementation of the simulation running on a desktop computer with a 1 GHz Pentium 4 processor and 256 MB RAM, for 1000 replications of a single bivariate model using a sample size of 400 takes approximately nine (9) days. An investigation of various models using more error cross correlation values and larger sample sizes or number of replications will run into months on a single machine.

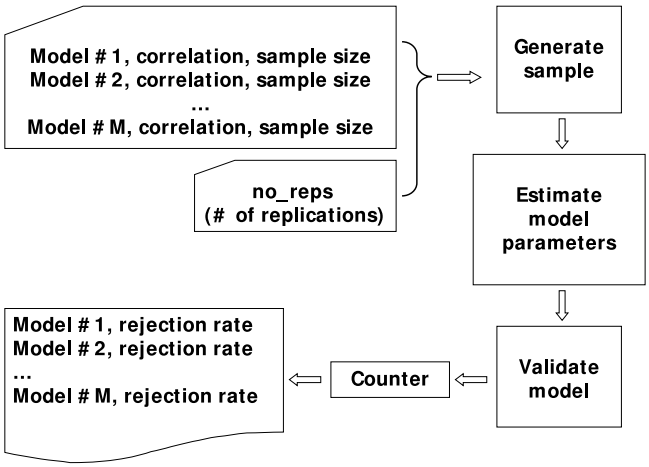


Fig. 1. Outline of VFCAR simulation procedure.

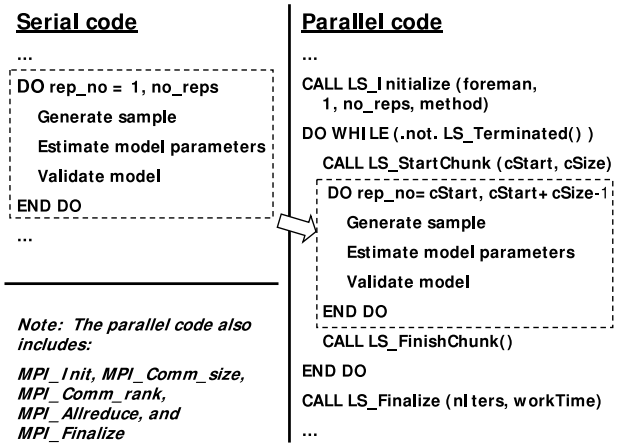


Fig. 2. Parallelization of replication loop, with the serial code largely intact.

3.1. Parallelization

The simulation procedure presents two opportunities for exploiting coarse grain parallelism. Up to `M` data models can be investigated concurrently. Also, for a given data model, the `no_reps` replications can be distributed among, say, P processors. Typically, `no_reps` $\gg P$, hence the focus of this paper on parallel execution of the replications. In general, equally distributing the replications among the processors leads to high performance if the replications have the same execution time and if the processors are homogeneous. However, this static assignment may result in load imbalance due to heterogeneity of processors, irregular iteration execution times, or unpredictable systemic effects like operating system interference leading to varying effective processor speeds. Load imbalance is typically indicated by highly uneven processor finishing times, and is a major cause of performance degradation in parallel applications. Dynamic loop scheduling is therefore necessary in order to balance processor loads and achieve improved performance.

The parallel simulation program was obtained after minor alterations to the initial Fortran 90 code. Essentially, three (3) lines of code are added before and after the original replication loop code, and the loop extents were changed, as illustrated by Fig. 2. The routines `LS_Initialize`, `LS_Finalize`, etc., are contained in a module specifically designed for quick integration of dynamic loop scheduling into sequential programs containing parallel loops. This module evolved from previously developed code for dynamic loop scheduling in scientific applications that utilize the Message Passing Interface (MPI) library [14]. Prior versions of the module were used to improve the performance of applications such as the profiling of automatic quadrature routines [15,16] and simulation of wave packets by the quantum trajectory method [17].

The routine `LS_Initialize` signals the start of dynamic scheduling of loop iterations: `foreman` specifies the rank of the process that will serve as the scheduler, `(1,no_reps)` is the loop extent, and `method` identifies the scheduling technique. The logical function `LS_Terminated` tests for loop termination. `LS_Finalize` synchronizes the processes, and returns `nIters` and `workTime`, the count and total execution time, respectively, of the replications computed by the

Table 1
Sample job times of performance tests.

Cluster	Test Id	Size	Reps	P	STAT	AF
Sun	TAR30	30	10 000	8	11:09	11:25
Solaris	TAR75	75	10 000	16	27:47	29:05
	WN150	150	10 000	32	31:15	23:28
Intel	WN500	500	10 000	64	5:56	4:47
Linux	TAR500	500	10 000	64	15:42	15:59

Notes: Size = Sample size; Reps = `no_reps`; P = Processors; STAT = job time, no loop scheduling; AF = job time with loop scheduling using AF.

calling process. These two quantities are useful in analyzing the performance of load balancing; ideally, the work times of the processors should be approximately equal.

The routine `LS_StartChunk` returns the start `cStart` and size `cSize` of a chunk of iterations to be executed by the calling processor. These quantities are determined according to the scheduling technique specified by `method`. After the chunk is finished, the routine `LS_FinishChunk` is invoked. Internally, the calling processor sends performance data pertaining to the recently executed chunk to the scheduler. The scheduler receives the performance data, and if there are remaining iterations, it computes and sends the next (`cStart`, `cSize`) to the processor; otherwise, the scheduler sends a termination message.

In addition to the changes in the replication loop, other modifications to enable parallelization pertain to the use of MPI. The usual calls to MPI_Init(), MPI_Comm_size() and MPI_Comm_rank() are added at the start of the program, and MPI_Finalize() added just before the program ends. Also, each processor computes only a fraction of the total number of replications, resulting in partial results being stored in each processor. The full set of results is obtained by performing the reduction operation MPI_Allreduce().

3.2. Parallel performance testing

Performance tests of the Fortran 90+MPI implementation of the VFCAR simulation procedure were conducted on the general-purpose Solaris and Linux clusters at the Mississippi State University Engineering Research Center. The Solaris cluster consists of Myrinet-interconnected Sun Microsystems SMPs. Each SMP has four (4) UltraSPARC 400 MHz processors, and the cluster has 16 nodes for a total of 64 processors. However, the cluster usage policy allows a job to utilize no more than 32 processors and no more than 48 h execution time. The Linux cluster, on the other hand, has a total of 1038 Pentium III (1.0 GHz and 1.266 GHz) processors, runs the Red Hat Linux operating system, and is interconnected via fast Ethernet switches with Gigabit Ethernet uplinks. The general submission queue allows jobs to request up to 64 processors for 48 h. The queuing system (PBS) attempts to assign homogeneous compute nodes to a job, but this is not guaranteed. Other jobs were also running on the clusters along with the experiments, thus network traffic volume may have varied during the experiments, with unpredictable effects on the performance tests.

Table 1 summarizes the job times of some of the tests. WNnnn and TARnnn denote a normal white noise and a threshold autoregressive model, respectively, with 'nnn' sample size. The last two columns give the job execution time (in hh: mm format) without loop scheduling (STAT) or with loop scheduling using the adaptive factoring technique (AF) [18–21]. The AF was chosen since it is the most sophisticated technique, as it addresses all sources of load imbalance. However, the AF has a higher overhead than other techniques, which may cause the AF to perform slightly less than other techniques for problems that exhibit no load imbalance.

The job times (STAT, AF) demonstrate the dramatic reduction in simulation time achieved by the simple code modification. For instance, running the original code for the TAR500 model on one of the processors of the Linux cluster would take approximately $64 * 15.7$ h, or 42 days.

Except for WN150 and WN500, the parallelization without loop scheduling (STAT) consumed slightly lesser time than with loop scheduling using the AF technique. This indicates that application-induced load imbalance was not a significant issue. However, the cases of the WN150 and WN500 provide clear evidence for the influence of system-induced load imbalance. Fig. 3 for WN150 illustrates this influence. For STAT, where each processor executed $(\texttt{no\_reps} + P - 1)/P$ replications (the gray bars), the plots of the processor work times (the gray triangles) show unusually longer times for processors 4, 12, 20 and 28. Most likely, these processors belonged to the same SMP node, and that this node had an extraneous process. With the AF technique, system-induced imbalance was also present, as evidenced by the obviously unequal number of replications executed by the processors. The almost flat trend line of the processor work times for the AF technique (dark squares) indicates that the load imbalance was successfully addressed.

The cluster resources allocated to the simulation were very efficiently utilized, as indicated in Table 2 which shows that up to 98% parallel efficiency is achievable. The lowest is 77% for WN150 with STAT, which may be attributed to severe system-induced load imbalance. This load imbalance is successfully addressed by AF obtaining 98% efficiency for the same problem. The speedup (Spd) and efficiency (Eff) metrics were computed as follows. Denote by t_r the time spent by processor r on the modified replication loop, and w_r the `workTime`. A t_r is the difference of the times taken just before `LS_Initialize` and right after `LS_Finalize`. A w_r is the cumulative time spent doing useful work. The parallel time t_{par} for the replication

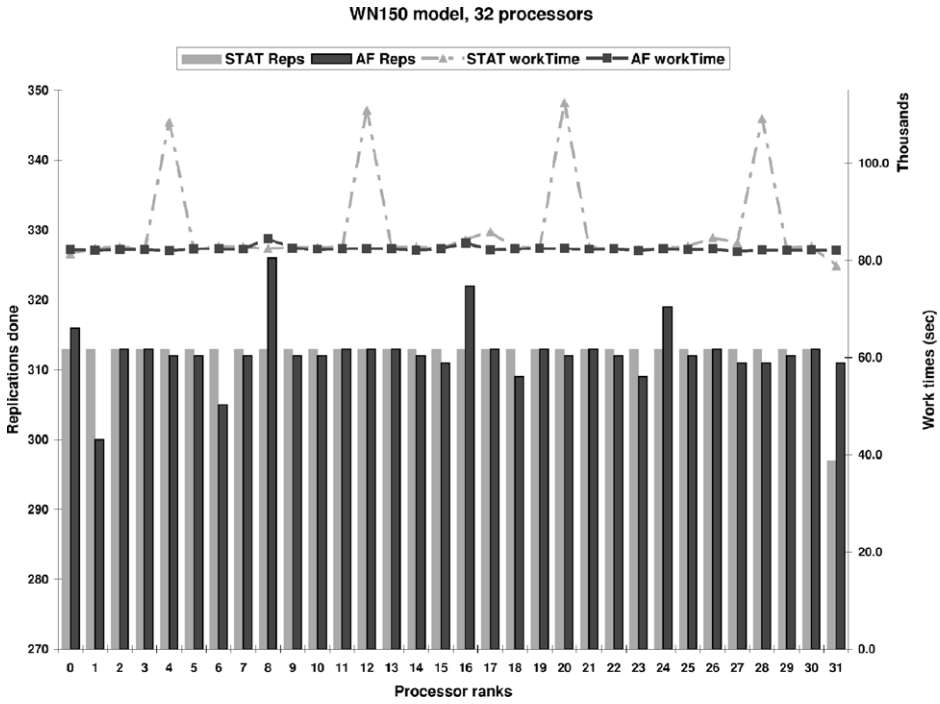


Fig. 3. Loop scheduling summary for WN150 on the Solaris cluster.

Table 2
Speedup and efficiency.

Test Id	P	STAT		AF	
		Spd	Eff	Spd	Eff
TAR30	8	7.9	0.98	7.6	0.95
TAR75	16	15.9	0.99	15.2	0.95
WN150	32	24.5	0.77	31.2	0.98
WN500	64	49.9	0.78	61.9	0.97
AR500	64	63.4	0.99	62.3	0.97

loop is $t_{\text{par}} = \max t_r$, and an estimate of the serial time t_{ser} is given by $t_{\text{ser}} = \sum_{\forall r} w_r$. This estimate for t_{ser} is accurate if the processors are homogeneous. From P, t_{par} and t_{ser}, Spd=$t_{\text{ser}}/t_{\text{par}}$ and Eff=$t_{\text{ser}}/(P \times t_{\text{par}})$.

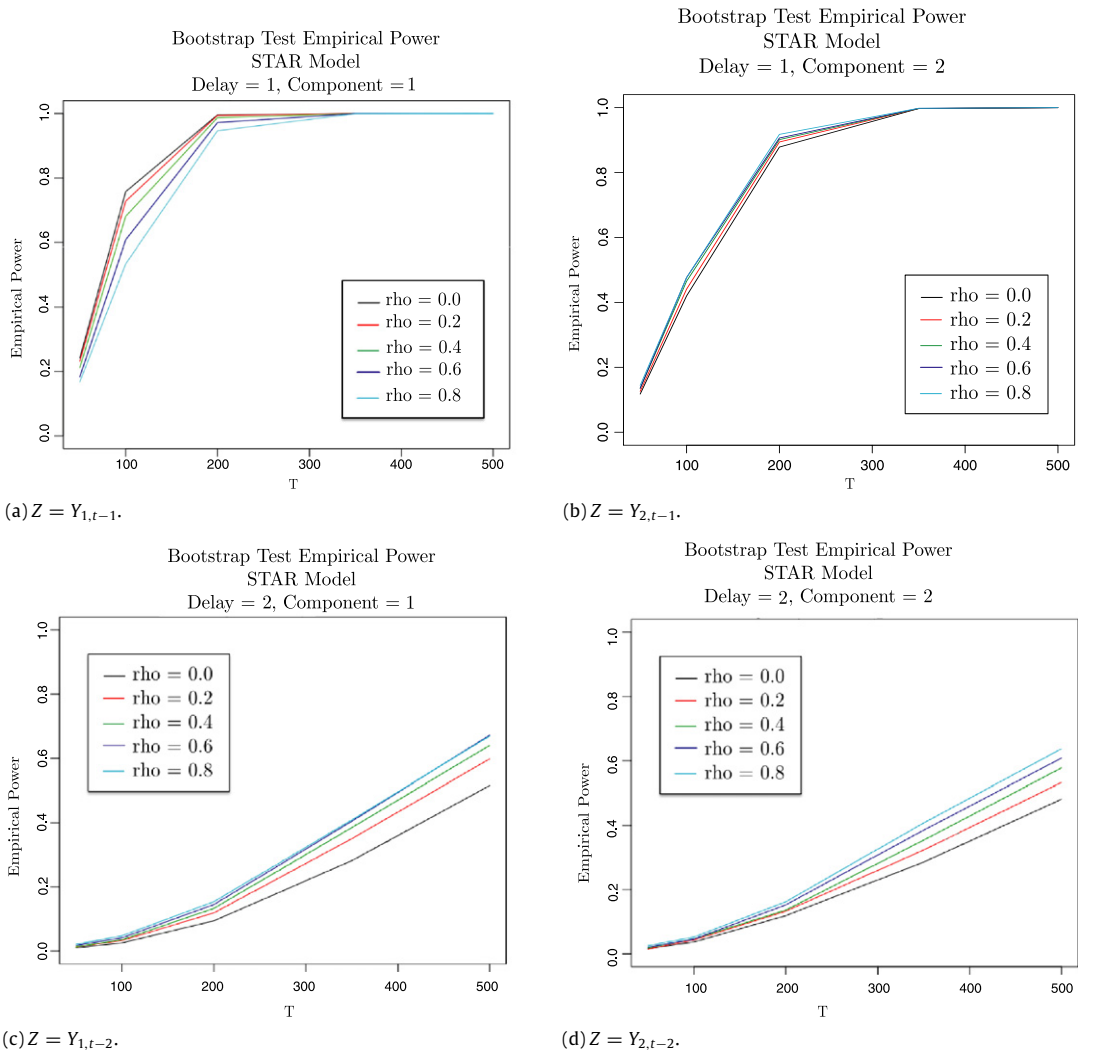
4. Advances in nonlinear time series

Obviously the simulations described in 3.2 could be completed on a single processor. On the other hand, if a numerical investigation using the same parameters was run for one model on a single processor, the time to complete the study would run into months, and may not be possible for the larger sample sizes. In the current statistics literature, numerical studies of the complex behaviors described in Sections 2.2 and 2.3 are completed using 500 replications for each of two moderate sample sizes ($T = 100$ and 250), three error cross-correlations ($\rho = 0.0$, 0.4, and 0.8), and 400 bootstrap replications. Small studies such as these, while providing important information, are far from complete, lack precision, and possibly lack accuracy. Consider the paper in [6]. The discussion of the numerical results indicates the behavior of the test to be quite complex, depending upon the model, the choice of smoothing variable, the length of the series (T), and the error cross-correlation (ρ).

Using the parallel code with dynamic scheduling techniques presented in Section 3, a more thorough study of the bootstrap test of Section 2.2 was conducted in a matter of days. The resulting study is not only more complete in its ability to describe the relationship between the power of the test for varying values of T and ρ, but the precision of the results is increased by at least a factor 20—a substantial result for numbers that are between 0 and 1. In the following paragraphs, we illustrate the improved results by considering two models also in [6].

Consider testing a null hypothesis that specifies the model is linear vs. an alternative of non-linearity; that is, the hypothesis in (6) are given by

$$H_0 : \boldsymbol{f}^{(l)}(\boldsymbol{Z}) = \boldsymbol{c}^{(l)} \quad \text{versus} \quad H_1 : \boldsymbol{f}^{(l)}(\boldsymbol{Z}) \neq \boldsymbol{c}^{(l)},$$



(a) $Z = Y_{1,t-1}$. (b) $Z = Y_{2,t-1}$.

(c) $Z = Y_{1,t-2}$. (d) $Z = Y_{2,t-2}$.

Fig. 4. Empirical power of bootstrap test for STAR model when choice of smoothing variable is correct (top right panel) and incorrect choice of smoothing variable (remaining three panels).

or the functional coefficients are *constants as a function of* $\boldsymbol{Z}$. Since the distribution of the test statistics when the null hypothesis is true is unknown, then the first numerical investigation of any hypothesis test begins by considering probabilities of a Type I error. The null hypothesis is true under two conditions.

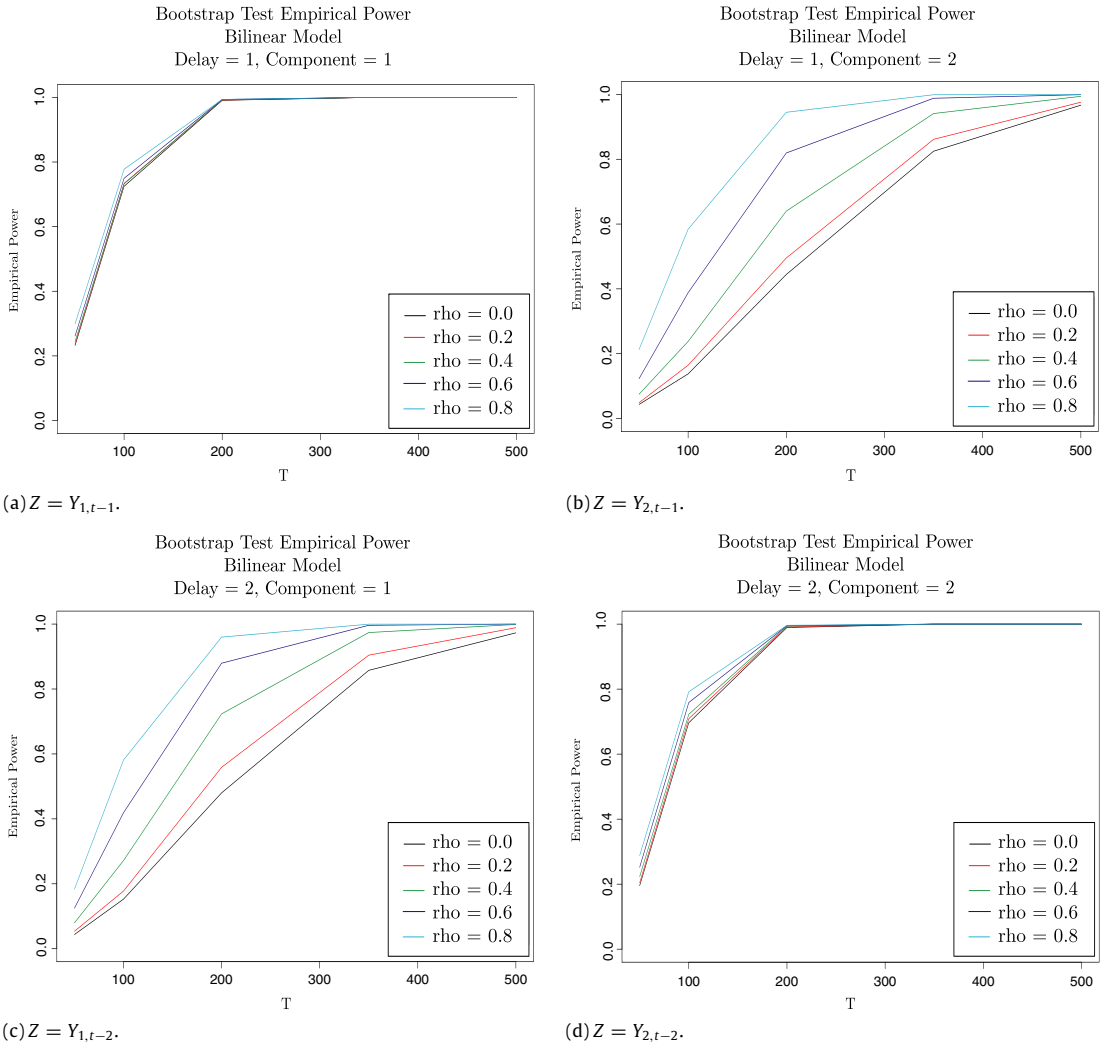
1. All $\boldsymbol{c}^{(l)}$ are zero, in which case the linear model reduces to white noise. An investigation of the size of the test reveals that the test is conservative, rejecting with a smaller probability than the specified, nominal size of $\alpha = 0.05$. Moreover, it is noted that the empirical size slowly increases as T increases, and as $\rho \to 1$. However, it is still less than nominal.
2. Some $\boldsymbol{c}^{(l)}$ are non-zero, resulting in a linear vector autoregressive model. In this case, the empirical size of the test is also less than nominal, but is larger than the size of the test when the model generating the data is white noise. In this case, the empirical size appears unaffected by sample size or error cross-correlation.

In both cases, the size does not appear to be a function of the choice of smoothing variable. While the empirical sizes of the tests run in this simulation were slightly smaller than Harvill and Ray [6], the general conclusions are the same.

The numerical investigation of the power of the test occurs when the null hypothesis is false (i.e., the model generating the data is not linear). One parametric nonlinear model that can easily be written in the form of an FCAR model is the smoothed threshold autoregressive model (STAR), first introduced in [22]. The bivariate STAR considered in our study is the same as the model considered in [6], which can be expressed in VFCAR form with $Z = Y_{1,t-1}$ and functional coefficients

$$f_{1,1}^{(1)}(Z) = 0.6 - \left(1 + e^{-5Z}\right)^{-1}, \qquad f_{2,1}^{(1)}(Z) = 0.3 - 0.3\left(1 + e^{-5Z}\right)^{-1},$$
$$f_{1,1}^{(2)}(Z) = 0.3 - 0.3\left(1 + e^{-5Z}\right)^{-1}, \qquad f_{2,1}^{(2)}(Z) = 0.6 - \left(1 + e^{-5Z}\right)^{-1}.$$

Fig. 4(a)–(d) are graphs depicting the behaviors of the empirical power of the test.



(a) $Z = Y_{1,t-1}$. (b) $Z = Y_{2,t-1}$.

(c) $Z = Y_{1,t-2}$. (d) $Z = Y_{2,t-2}$.

Fig. 5. Empirical power of the bootstrap test for a bilinear model when the choice of smoothing variable is correct (top right panel) and incorrect choice of smoothing variable (remaining three panels).

A number of properties of the empirical power of the test are immediately obvious from this more thorough study that are difficult to see in the abbreviated one. First, it is noted that for large ($T = 500$) and correct choice of the delay for the smoothing variable, the empirical power of the test is extremely close to 1. The value $T = 500$ is problematic in terms of memory and time on a reasonably power single processor desktop machine. Additionally, it is much easier to spot interesting behaviors not recognized in previous studies. Specifically, if the correct choice of smoothing variable in terms of time series component ($Y_{1,.}$) is chosen, then as the error cross-correlation approaches one, the empirical power of the test decreases for the smaller sample sizes. On the other hand, when the second component $Y_{2,.}$ was selected as the smoothing variable, the empirical power of the test increases as $\rho \to 1$ for the smaller sample sizes. The shape of the empirical power curves also suggests that the choice of delay for the smoothing variable has an influence on how quickly the power converges to one for all values of ρ, with correct choice of d resulting in a test that had power converging to 1 much faster than an incorrect choice of d. On the other hand, incorrectly choosing the component for the smoothing variable seemed to have little effect on the power of the test.

Another model considered in [6] is a bivariate bilinear model. For that particular model, it is possible choose any of $Y_{2,t-1}$, $Y_{2,t-1}$, $\varepsilon_{1,t-1}$ or $\varepsilon_{1,t-2}$ as smoothing variables. Moreover, since $Y_{1,t}$ is represented as a function of $Y_{t,t-2}$, and $Y_{2,t}$ as a function of $Y_{1,t-1}$, the iterative nature of the model makes pinpointing a *single* smoothing variable challenging. The plots in Fig. 5(a)–(d) are the empirical powers of the bootstrap test.

The empirical power of the test quickly goes to one with either $Y_{1,t-1}$ or $Y_{2,t-2}$ selected for the smoothing variable. In fact, the power still increases to one when the incorrect variable is selected, but more slowly, and more dependent upon the value of the cross correlation between the error terms, probably due to the iterative nature of the model. For the bilinear model, the empirical power of the bootstrap test appears to be greater than that of the STAR model.

5. Conclusion

This paper presents an approach to the parallelization of simulations to investigate statistical properties of methods using VFCAR models. The parallelization allows improved accuracy and precision of statistical results, and is based on novel techniques for improving performance of scientific computing in parallel and distributed environments. The approach is capable of successfully overcoming the challenge of long running times and requires very minimal revisions to an existing sequential code. The revisions generate parallel code which incorporates dynamic load balancing. The parallel code addresses factors that may give rise to load imbalance, such as those that may be inherent in the computations or induced by the computing environment. Tests on a homogeneous Sun Solaris cluster and a heterogeneous Linux cluster indicate that the parallel code achieves very high performance, even with unpredictable system-induced load imbalance, obtaining efficiencies in the range 95%–98% for the normal white noise and threshold autoregressive model, with sample sizes up to 500 and 10 000 replications, on up to 64 processors. This combination of sample size and replication count, to the best of our knowledge, has not been previously attempted when using simulation to investigate properties of statistical methods using VFCAR models. As illustrated in Section 4, this ability provides a more precise set. But more importantly, the ability to numerically investigate the size and power of the test provides a much clearer picture of the effect on the empirical size and powers of the test of the choice of model parameters when conducting the test – results that were previously unavailable. Future work will include experiments with new models and more error correlation values. Furthermore, this interdisciplinary research work finds application in a variety of other statistical disciplines. The authors plan to apply these state of-the-art techniques to an important problem in astrophysics, the classification of gamma-ray bursts.

Acknowledgments

This work was partially sponsored by NSF Award #0313274 at Mississippi State University. The project aims to develop new methods in statistics and high performance computing, and to apply these methods to analyze gamma-ray burst time profiles. Additionally, the work was also supported by NSF through the following grants: CAREER #9984465 and EPS #0132618.

References

[1] H. Lütkepohl, Introduction to Multiple Time Series, Springer, New York, 1993.
[2] J.L. Harvill, B.K. Ray, A note on tests for nonlinearity in vector time series, Biometrika 86 (3) (1999) 728–734.
[3] B. Rajagopalan, M.E. Mann, U. Lall, A multivariate frequency-domain approach to long-lead climatic forecasting, Weather Forecasting 13 (1) (1998) 58–74.
[4] Z. Cai, J. Fan, Q. Yao, Functional coefficient regression models for nonlinear time series, J. Amer. Statist. Assoc. 95 (451) (2000) 941–956.
[5] J.L. Harvill, B.K. Ray, A note on multi-step forecasting with functional coefficient autoregressive models, Int. J. Forecast. 4 (21) (2005) 717–727.
[6] J.L. Harvill, B.K. Ray, Functional coefficient autoregressive models for vector time series, Comput. Statist. Data Anal. 50 (12) (2005) 3547–3566.
[7] I. Banicescu, R. Cariño, J. Harvill, J. Lestrade, Simulation of vector nonlinear time series models on clusters, in: Proceedings of the IEEE International Parallel and Distributed Processing Symposium, 2005, pp. 4–8.
[8] R. Chen, L. Liu, Functional coefficient autoregressive models: estimation and tests of hypotheses, J. Time Ser. Anal. 22 (2) (2001) 151–173.
[9] J. Fan, Q. Yao, Z. Cai, Adaptive varying-coefficient linear models, J. R. Stat. Soc. Ser. B 65 (1) (2003) 57–80.
[10] J. Fan, Q. Yao, Nonlinear Time Series: Nonparametric and Parametric Methods, Springer, New York, 2003.
[11] R. Chen, T.S. Tsay, Functional coefficient autoregressive models, J. Amer. Statist. Assoc. 88 (421) (1993) 298–308.
[12] R Development Core Team. R: a language and environment for statistical computing, *R* Foundation for Statistical Computing, Vienna, Austria, 2004. 3-900051-07-0.
[13] L. Tierney, A.J. Rossini, N. Li, The snow package, 2004. http://cran.us.r-project.org/doc/packages/snow.pdf.
[14] R.L. Cariño, I. Banicescu, A load balancing tool for distributed parallel loops, Cluster Computing: The Journal of Networks, Software Tools and Applications 8 (4) (2005) 313–321.
[15] R.L. Cariño, I. Banicescu, Dynamic scheduling parallel loops with variable iterate execution times, in: Proceedings of the 16th IEEE International Parallel and Distributed Processing Symposium—3rd Workshop on Parallel and Distributed Scientific and Engineering Computing With Applications (IPDPS-PDSECA 2002) CDROM, IEEE Computer Society, 2002.
[16] R.L. Cariño, I. Banicescu, Load balancing parallel loops on message-passing systems, in: S.G. Akl, T. Gonzales (Eds.), Proceedings of the 14th IASTED International Conference on Parallel and Distributed Computing and Systems, PDCS 2004, ACTA Press, 2002, pp. 362–367.
[17] R.L. Cariño, I. Banicescu, R.K. Vadapalli, C.A. Weatherford, J. Zhu, Message-passing parallel adaptive quantum trajectory method, in: L.T. Yang, Y. Pan (Eds.), High performance Scientific and Engineering Computing: Hardware/Software Support, Kluwer Academic Publishers, 2004, pp. 127–139.
[18] I. Banicescu, Z. Liu, Adaptive factoring: a dynamic scheduling method tuned to the rate of weight changes, in: Proceedings of the High Performance Computing Symposium, HPC 2000, 2000, pp. 122–129.
[19] I. Banicescu, V. Velusamy, Load balancing highly irregular computations with the adaptive factoring, in: Proceedings of the 16th IEEE International Parallel and Distributed Processing Symposium—11th Heterogeneous Computing Workshop (IPDPS-HCW 2002) CDROM, IEEE Computer Society, 2002.
[20] I. Banicescu, V. Velusamy, J. Devaprasad, On the scalability of dynamic scheduling scientific applications with adaptive weighted factoring, Cluster Computing: The Journal of Networks, Software Tools and Applications 6 (2003) 215–226.
[21] I. Banicescu, R.L. Carino, Addressing the stochastic nature of scientific computations via dynamic loop scheduling, Electron. Trans. Numer. Anal. 21 (2005) 66–80.
[22] D. Van Dijk, Smooth transition models: extension and outlier robust inference, Ph.D. Thesis, Erasmus University, Rotterdam, Netherlands, 1999.