PIECEWISE LINEAR TIME SERIES ESTIMATION WITH GRASP

MARCELO C. MEDEIROS, MAURICIO G.C. RESENDE, AND ALVARO VEIGA

ABSTRACT. This paper describes a heuristic to build piecewise linear statistical models with multivariate thresholds, based on a Greedy Randomized Adaptive Search Procedure (GRASP). GRASP is an iterative randomized sampling technique that has been shown to quickly produce good quality solutions for a wide variety of optimization problems. In this paper we describe a GRASP to sequentially split an n-dimensional space in order to build a piecewise linear time series model.

1. INTRODUCTION AND PROBLEM DESCRIPTION

A time series is a sequence of observations of a certain phenomenon over time. The main goal of most statistical forecasting techniques is to analyze the past outcomes of a given series in order to predict its future behavior.

Historically, the most frequently used approaches to time series model building assume that the data under study are generated from a linear Gaussian stochastic process [4]. However, it is well known that real-life systems are usually nonlinear, and certain features, such as limit-cycles, asymmetry, amplitude-dependent frequency responses, jump phenomena, and chaos cannot be correctly captured by linear statistical models. Over the last years, several nonlinear time series models have been proposed in the classical technical literature (see [9] and [5] for a comprehensive review). One model that has found a large number of successful applications is the threshold autoregressive model (TAR), proposed by Tong [7] and Tong and Lim [10]. The TAR model is a piecewise linear process whose central idea is to change the parameters of a linear autoregressive model according to the value of a known variable, called the *threshold variable*. If this variable is a lagged value of the time series, the model is called a self-exciting threshold autoregressive (SETAR) model.

In this paper, we propose a heuristic to estimate SETAR models with multivariate thresholds. This is a generalization of the procedures described in [10] and [11], where the thresholds are monovariate. The proposed method is based on a semi-greedy algorithm, called Greedy Randomized Search Adaptive Procedure (GRASP), proposed by Feo and Resende [2, 3, 6].

The paper is organized as follows. Section 2 gives a general description of threshold models. Section 3 presents the proposed procedure. Section 4 describes briefly the GRASP methodology and presents its application to our particular problem. Section 5 presents some numerical examples illustrating the performance of the proposed model. Concluding remarks are made in Section 6.

Date: February 26, 1999.

Key words and phrases. Nonlinear time series analysis, piecewise linear models, combinatorial optimization, search heuristic, GRASP.

AT&T Labs Research Technical Report: 99.3.1.

2. Threshold Autoregressive Models

The threshold autoregressive model was first proposed by Tong [7] and further developed by Tong and Lim [10] and Tong [8]. The main idea of the TAR model is to describe a given stochastic process by a piecewise linear autoregressive model, where the determination of whether each of the models is active or not depends on the value of a known variable.

A time series y_t is a *threshold process* if it follows the model

$$y_t = \sum_{j=1}^{l} \left[\phi_0^{(j)} + \sum_{i=1}^{p} \phi_i^{(j)} y_{t-i} + \varepsilon_t^{(j)} \right] I^{(j)}(q_t), \tag{1}$$

where $\varepsilon_t^{(j)}$ is a white noise process with zero mean and finite variance $\sigma^{2(j)}$, and the terms $\phi_0^{(j)}, \phi_1^{(j)}, \ldots, \phi_p^{(j)}$ are real coefficients. $I^{(j)}(\cdot)$ is an indicator function, defined by

$$I^{(j)}(q_t) = \begin{cases} 1, & \text{if } q_t \in \mathbb{R}_j; \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where $\mathbb{R}_j = (r_{j-1}, r_j]$ is a partition of the real line $\mathbb{R}$, defined by a linearly ordered subset of the real numbers, $\{r_0, \ldots, r_l\}$, such that $r_0 < r_1 < \ldots < r_l$, where $r_0 = -\infty$ and $r_l = \infty$. Model (1) is composed by l autoregressive linear models, each of which will be active or not depending on the value of q_t. The choice of the threshold variable, q_t, which determines the dynamics of the process, allows a number of possible situations. An important case is when q_t is replaced by y_{t-d}, where the model becomes the self exciting threshold autoregressive model

$$y_t = \sum_{j=1}^{l} \left[\phi_0^{(j)} + \sum_{i=1}^{p} \phi_i^{(j)} y_{t-i} + \varepsilon_t^{(j)} \right] I^{(j)}(y_{t-d}), \tag{3}$$

denoted by the acronym SETAR$(l; p_1, \cdots, p_l)$. The scalar d is known as the delay parameter or the length of the threshold.

In [10], a grid search to estimate SETAR models, based on the Akaike's information criterion [1], was proposed. In [11], Tsay developed a graphical procedure and a statistical test for nonlinearity to estimate the thresholds. Both methodologies consider only thresholds controlled by a single lagged observation of the time series.

3. The Multiple Threshold Autoregressive Model

3.1. **Model Presentation.** As stated in Section 2, the dynamics of a SETAR model are controlled by a partition of the real line $\mathbb{R}$ induced by the parameters r_j. However, in a more general situation, it will be interesting to consider a partition of an n-dimensional space, say $\mathbb{R}^n$. This paper proposes a procedure to estimate SETAR models with evolution controlled by a partition of a multidimensional space induced by h separating hyperplanes. The proposed method can be immediately generalized to a dynamic regression framework.

Consider an n-dimensional Euclidean space and a point $\mathbf{x}$ in that space. A hyperplane is defined by

$$\mathbb{H} = \{\mathbf{x} \in \mathbb{R}^n \,|\, \omega^T \mathbf{x} = \beta\}, \tag{4}$$

where ω is an n-dimensional parameter vector and β is a scalar parameter. Figure 1 shows an example in $\mathbb{R}^2$. The direction of ω determines the orientation of the hyperplane and the scalar term β determines the position of the hyperplane in terms of its distance from the origin.

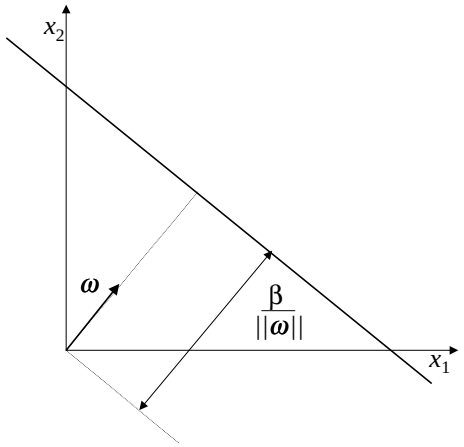

FIGURE 1. Hyperplane defined by $\omega^T \mathbf{x} = \beta$ in $\mathbb{R}^2$.

A hyperplane induces a partition of the space into two regions defined by the halfspaces

$$\mathbb{H}^+ = \{\mathbf{x} \in \mathbb{R}^n | \omega^T \mathbf{x} \geq \beta\} \tag{5}$$

and

$$\mathbb{H}^- = \{\mathbf{x} \in \mathbb{R}^n | \omega^T \mathbf{x} < \beta\}. \tag{6}$$

With h hyperplanes, an n-dimensional space will be split into several polyhedral regions. Each region is defined by the nonempty intersection of the halfspaces (5) and (6) of each hyperplane.

For a given hyperplane, defined by ω and β, denote by $I_{\omega,\beta}(\mathbf{x})$ an indicator function

$$I_{\omega,\beta}(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \in \mathbb{H}^+; \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

The main idea of the proposed procedure is to use (7) to create a multidimensional threshold structure. Suppose that an n-dimensional space is spanned by n lagged values of a given stochastic process y_t, say $\mathbf{x}_t^T = [y_{t-1}, \cdots, y_{t-n}]$, and suppose we have h functions $I_{\omega_i,\beta_i}(\mathbf{x})$, $i = 1, \ldots, h$, each of which defines a threshold. Now consider a time-varying time series model defined as

$$y_t = \phi_t^{(0)} + \sum_{i=1}^{p} \phi_t^{(i)} y_{t-i} + \varepsilon_t. \tag{8}$$

The time evolution of the coefficients $\phi_t^{(j)}$ of (8) is given by

$$\phi_t^{(j)} = \sum_{i=1}^{h} \lambda_{ij} I_{\omega_i,\beta_i}(\mathbf{x}_t) - \gamma_j, \;\; j = 0, \cdots, p, \tag{9}$$

where λ_{ij} and γ_j, $i = 1, \ldots, h$ and $j = 1, \ldots, p$, are real coefficients. Equations (8) and (9) represent a time-varying model with a multivariate threshold structure defined by h separating hyperplanes.

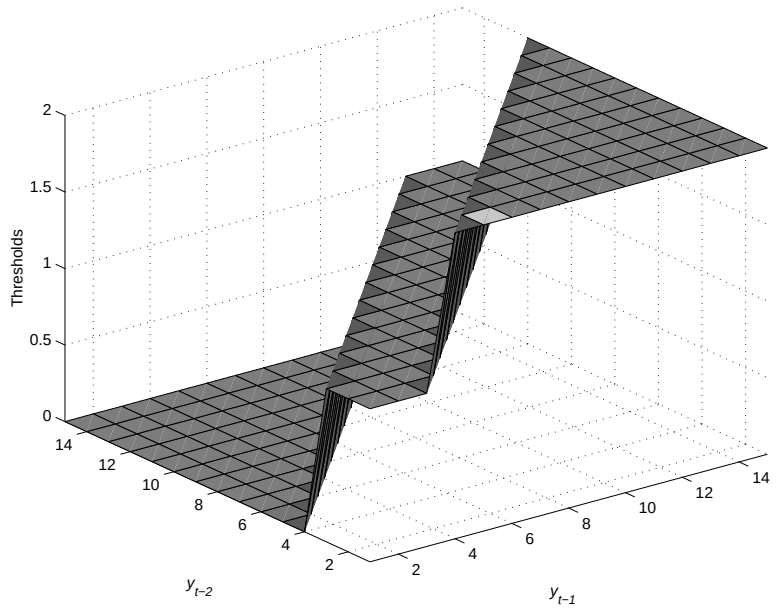

FIGURE 2. Threshold boundaries of model (10).

To illustrate, consider the following example

$$y_t = \begin{cases} 0.2 + 0.8y_{t-1} + \varepsilon_t, & \text{if } |y_{t-1} - y_{t-2}| \geq 0.2; \\ 0.2 - 0.4y_{t-1} + \varepsilon_t, & \text{otherwise.} \end{cases} \tag{10}$$

Clearly, equation (10) has two thresholds: one when $y_{t-1} - y_{t-2} = 0.2$ and another when $y_{t-1} - y_{t-2} = -0.2$. Figure 2 shows the threshold boundaries. The value of parameters ω_i and β_i, $i = 1,2$, are: $\omega_1^T = \omega_2^T = [1,-1]$, $\beta_1 = -0.2$ and $\beta_2 = 0.2$. Equation (9) can be rewritten as

$$\phi_t^{(j)} = \begin{cases} -\gamma_j, & \text{if } y_{t-1} - y_{t-2} \leq -0.2; \\ \lambda_{1j} - \gamma_j, & \text{if } -0.2 < y_{t-1} - y_{t-2} < 0.2; \\ \lambda_{1j} + \lambda_{2j} - \gamma_j, & \text{otherwise.} \end{cases} \tag{11}$$

The values of parameters λ_{ij} and γ_j can be easily determined by matching equations (11) and (10).

3.2. **Estimation of the Parameters.** Equations (8) and (9) can be cast in matrix notation

$$y_t = \phi_t^T \mathbf{y}_t + \varepsilon_t, \tag{12}$$

where $\mathbf{y}_t^T = [1, y_{t-1}, y_{t-2}, \ldots, y_{t-p}]$, and $\phi_t^T = \left[\phi_t^{(0)}, \phi_t^{(1)}, \ldots, \phi_t^{(p)}\right]$. Defining $I_t^T = \left[I_{\omega_1,\beta_1}(\mathbf{x}_t), I_{\omega_2,\beta_2}(\mathbf{x}_t), \cdots, I_{\omega_h,\beta_h}(\mathbf{x}_t)\right]$,

$$\Lambda = \begin{bmatrix} \lambda_{01} & \lambda_{02} & \ldots & \lambda_{0h} \\ \lambda_{11} & \lambda_{12} & \ldots & \lambda_{1h} \\ \lambda_{21} & \lambda_{22} & \ldots & \lambda_{2h} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{p1} & \lambda_{p2} & \ldots & \lambda_{ph} \end{bmatrix}, \qquad \gamma = \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_p \end{bmatrix},$$

equation (9) can be rewritten as

$$\phi_t = \Lambda I_t - \gamma. \tag{13}$$

Denoting $\Theta = [\Lambda,\ -\gamma]$ and $\Omega_t^T = [I_t,\ \mathbf{1}]$, where $\mathbf{1}^T = [\underbrace{1,\ 1,\ 1,\ \cdots,\ 1}_{h}]$, equation (12) becomes

$$y_t = (\Theta\Omega_t)^T \mathbf{y}_t + \varepsilon_t = \Omega_t^T \Theta^T \mathbf{y}_t + \varepsilon_t. \tag{14}$$

Applying the **vec**[1] operator to both sides of equation (14), and using the property[2] that $\mathbf{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\mathbf{vec}(\mathbf{B})$, we obtain

$$y_t = \left(\mathbf{y}_t^T \otimes \Omega_t^T\right) \mathbf{vec}\left(\Theta^T\right) + \varepsilon_t. \tag{15}$$

Equation (15) is a linear regression model to which the ordinary least squares estimator can be applied, obtaining

$$\mathbf{vec}(\hat{\Theta}) = \left[\sum_{i=1}^{N} \left(\mathbf{y}_t^T \otimes \Omega_t^T\right)^T \left(\mathbf{y}_t^T \otimes \Omega_t^T\right)\right]^{-1} \sum_{i=1}^{N} \left(\mathbf{y}_t^T \otimes \Omega_t^T\right)^T y_t \tag{16}$$

Sometimes in practice, the matrix $\left[\sum_{i=1}^{N} \left(\mathbf{y}_t^T \otimes \Omega_t^T\right)^T \left(\mathbf{y}_t^T \otimes \Omega_t^T\right)\right]$ does not have an inverse and a pseudo-inverse should be calculated by a singular-value decomposition algorithm.

The problem now is to estimate parameters ω_i and β_i, $i = 1, \ldots, h$. As stated earlier in this section, these parameters define a hyperplane in an n-dimensional space. If we have N observations of $\mathbf{x}_t$, we must consider hyperplanes that separate the observed points. In fact, we only need to consider the hyperplanes defined by combinations of those points. Since in an n-dimensional space, we need n distinct points to define a hyperplane. Hence, if we have N points, there are $\frac{N!}{n!(N-n)!}$ possible hyperplanes to search. One way would be to search all the possible combinations of hyperplanes and choose the combination that minimizes the sum of squared errors. Of course, for most practical problems this is infeasible. In the next section we propose a procedure based on GRASP to choose the set of h hyperplanes with small sum of squared errors.

4. A GRASP FOR PIECEWISE LINEAR MODELS

A GRASP is a multi-start iterative randomized sampling technique, with each GRASP iteration consisting of two phases, a construction phase and a local search phase. The best overall solution is kept as the result.

The construction phase of GRASP is essentially a randomized greedy algorithm, where a feasible solution is iteratively constructed, one element at a time. At each construction iteration, the choice of the next element to be added to the solution is determined by ordering all candidate elements in a candidate list with respect to a greedy function. This function measures the (myopic) benefit of selecting each element. The heuristic is adaptive because the benefits associated with every element are updated at each iteration of the construction

[1] Let $\mathbf{A}$ be a $(m \times n)$ matrix with $(m \times 1)$ columns $\mathbf{a}_i$. The **vec** operator transforms $\mathbf{A}$ into an $(mn \times 1)$ vector by stacking the columns of $\mathbf{A}$.

[2] $\otimes$ denotes the Kronecker product. Let $\mathbf{A} = (a_{ij})$ and $\mathbf{B} = (b_{ij})$ be $(m \times n)$ and $(p \times q)$ matrices, respectively. The $(mp \times nq)$ matrix $\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}$ is the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$.

```
procedure grasp( )
1     do k = 1, ··· , MaxIter →
2           ConstructGreedyRandomizedSolution( );
3           LocalSearch( );
4           UpdateSolution( );
5     end do;
6     return(BestSolutionFound);
end grasp;
```

FIGURE 3. GRASP construction procedure.

phase to reflect the changes brought on by the selection of the previous element. The probabilistic component of a GRASP is characterized by randomly choosing one of the best candidates in the list, but not necessarily the top candidate. The list of best candidates is called the *restricted candidate list* (RCL). This choice technique allows for different solutions to be obtained at each GRASP iteration, but does not necessarily compromise the power of the adaptive greedy component of the method.

As is the case for many deterministic methods, the solutions generated by a GRASP construction are not guaranteed to be locally optimal with respect to simple neighborhood definitions. Hence, it is almost always beneficial to apply a local search to attempt to improve each constructed solution. Normally, a local optimization procedure such as a two-exchange is employed. While such procedures can require exponential time from an arbitrary starting point, empirically their efficiency significantly improves as the initial solution improves. Through the use of customized data structures and careful implementation, an efficient construction phase can be created which produces good initial solutions for efficient local search. The result is that often many GRASP solutions are generated in the same amount of time required for the local optimization procedure to converge from a single random start. Furthermore, the best of these GRASP solutions is generally significantly better than the solution obtained by a local search from a random starting point.

Figure 3 illustrates a generic GRASP in pseudo-code. The GRASP takes as input parameters for setting the maximum number of GRASP iterations and the seed for the random number generator. The GRASP iterations are carried out in lines 1–5. Each GRASP iteration consists of the construction phase (line 2), the local search phase (line 3) and, if necessary, the incumbent solution update (line 4).

When applied to the piecewise linear time series modeling problem, the construction phase consists of sequentially choosing hyperplanes until the maximum number of hyperplanes is reached. The greedy function proposed orders the possible hyperplanes with respect to the sum of squared errors of the fitted data. The local search implemented in this GRASP is a two-exchange heuristic, where a hyperplane that is in the solution is substituted by another that is not in the solution. Because the number of possible hyperplanes can grow very fast as a function of the number of observed points, we build an outer loop where in each outer iteration we consider only a random subset of the possible hyperplanes. Figure 4 illustrates the procedure in pseudo-code.

The main procedure takes as input parameters for setting the maximum number of outer loop and GRASP iterations, the maximum number of hyperplanes at each outer loop iteration and the number of hyperplanes in the solution set. It is important to stress that the algorithm supposes that the number of separating hyperplanes, the variables that compose

```
procedure gplts( )
1     BestSolutionFound=OLS;
2     do k = 1, ··· , MaxOuterIter →
3           PermuteData( );
4           RandomSelectHyperplanes( );
5           Grasp( );
6           UpdateSolution( );
7     end do;
8     return(BestSolutionFound);
end gptls;
```

FIGURE 4. Main loop procedure.

```
procedure Grasp( )
1     do k = 1, ··· , MaxInnerIter →
2           SelectαAtRandom( );
3           ConstructGreedyRandomizedSolution( );
4           LocalSearch( );
5           UpdateSolution( );
6     end do;
7     return(BestSolutionFound);
end Grasp;
```

FIGURE 5. GRASP procedure.

the vector $\mathbf{x}_t$ in (9) and the order p of the autoregressive model (8)are known. After setting the best solution to the ordinary least squares solution (OLS) (line 1), at each outer iteration (lines 2–7), the procedure permutes the data sets to avoid any bias in the random number generator and randomly selects a subset C of *MaxHyper* possible hyperplanes to be used in the GRASP procedure. In line 6, if it is necessary, the solution is updated. Figure (5) shows the pseudo-code for the GRASP procedure to the piecewise linear model building.

We next describe the each one of the components in detail.

4.1. **Construction Phase.** In the construction phase each possible hyperplane is ordered according to a cost function. In the context of time series analysis, the chosen cost function is the sum of squared errors. To capture the adaptive component, at each time a hyperplane is chosen the remaining hyperplanes are reordered to reflect the benefits of the selection of the previous hyperplanes.

The random component of this GRASP sequentially selects at random the hyperplanes from the restricted candidate list (RCL) until the maximum number of hyperplanes is reached. Let $\alpha \in [0,1]$ be a given parameter and $SSE(hp)$ the cost of selecting a given hyperplane from the set of all possible alternatives C, then the RCL is defined as

$$RCL = \{hp \in C \mid SSE(hp) \leq \underline{hp} + \alpha(\overline{hp} - \underline{hp})\} \tag{17}$$

where $\underline{hp} = \min\{SSE(hp) \mid hp \in C\}$ and $\overline{hp} = \max\{SSE(hp) \mid hp \in C\}$.

In this implemention at each inner iteration the parameter α is a randomly chosen from a uniform distribution between 0 and 1, as seen in line 2 of Figure 5. Figure 6 illustrates the

```
procedure ConstructGreedyRandomizedSolution( )
1     s = ∅;
3     do k = 1, ··· , MaxHyperSolution →
4           hp̄ = max{SSE(hp) | hp ∈ C};
5           hp̲ = min{SSE(hp) | hp ∈ C};
6           RCL = {hp ∈ C | SSE(hp) ≤ hp̲ + α(hp̄ − hp̲)};
7           SelectHyperAtRandom(RCL);
8           s = s⋃{hp};
9           AdaptCost( );
9     end do;
end ConstructGreedyRandomizedSolution;
```

FIGURE 6. Construction phase.

```
procedure LocalSearch( )
1     do s is not locally optimal in N(s) →
2           FindBetterSolution( );
3           ReplaceHyperplanes( );
4     end do;
end LocalSearch;
```

FIGURE 7. Local search phase.

construction phase of the GRASP used in this paper. The construction procedure receives as parameters the maximum number of hyperplanes that compose the solution, the RCL parameter α and the subset of possible hyperplanes.

4.2. **Local Search.** For a given problem, a local search algorithm works in a iterative fashion by successively replacing the current solution by a better solution in the neighborhood of the current solution with respect to some cost function. It terminates when there is no better solution found in the neighborhood.

The local search implemented in this GRASP is a 2-exchange local search, where a hyperplane that it is in the solution set is replaced by another hyperplane that is not in the solution set. Figure 7 shows the pseudo-code of the local search procedure. The considered neighborhood $N(s)$ is the set of all hyperplanes in C that are not in the solution set s.

5. EXPERIMENTAL RESULTS

In this section, we report experimental results with an implementation of the GRASP described in this paper. The experiments were done using simulated data sets with known optimal solution. The experiments were done on a Silicon Graphics Challenge computer (20 MIPS 196 MHz R10000 processors with 6.144 Gbytes of main memory). The algorithms were programmed in MatLab® and translated into C++ with the MatCom® compiler. We generated 500 observations of four stochastic processes[3].

We ran the GRASP described in Section 4 for each of the models where at each outer iteration we randomly select a subset of 250 hyperplanes out of all the possible hyperplanes.

[3] http://www.research.att.com/~mgcr/data/plts.tar.gz

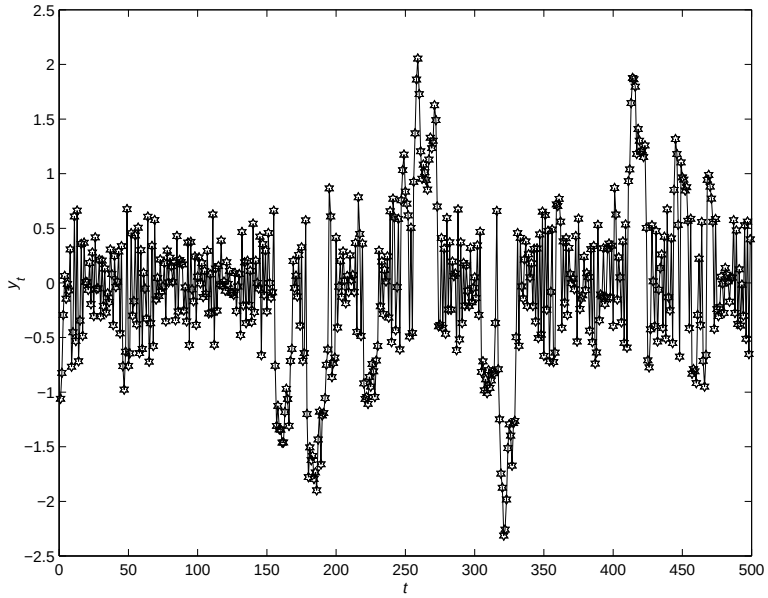

FIGURE 8. Time series generated by (18).

The number of outer and inner iterations were, respectively, 20 and 5. We ran the algorithm using 6 differentrandom number generator seeds (330000, 330001, 330002, 330003, 330004, and 330005). The results are summarized in

Tables 1– 3. Table 1 shows solution quality. For each model the table shows the value of the best solution found by our procedure, the value of the known optimal solution (obtained when the correct parameters of the model are estimated), and the percentage relative error (difference between best solution and optimal solution divided by the optimal solution times 100). Table 2 shows for each model the minimum, maximum, and average (over the six runs) number of outer and inner iterations, as well as minimum, maximum, and average values of the best solutions found by our procedure. Table 3 shows solution times. For each model the table shows minimum, maximum, and average times to find the best solution, and minimum, maximum, and average total running times.

The first generated time series is very simple, defined as

$$y_t = \begin{cases} 0.95y_{t-1} + \varepsilon_t, & \text{if } |y_{t-1}| \geq 0.7; \\ -0.95y_{t-1} + \varepsilon_t, & \text{otherwise.} \end{cases} \tag{18}$$

The random term ε_t is a normally distributed white noise process with zero mean and variance 0.0625. Figure 8 shows the simulated data. In this case there are two thresholds. One when $y_{t-1} = 0.7$ and another when $y_{t-1} = -0.7$. As the thresholds are unidimensional, the total number of possible values for them equals 500, the total number of points. The order of model (18) is 1 and $\mathbf{x}_t = y_{t-1}$.

Figure 9 shows the scatter plot of y_t and $\hat{y}_t$ versus y_{t-1}. The proposed procedure correctly identifies the position of the thresholds of model (18). As shown in Tables 1– 3 the algorithm finds the optimal solution in few iterations.

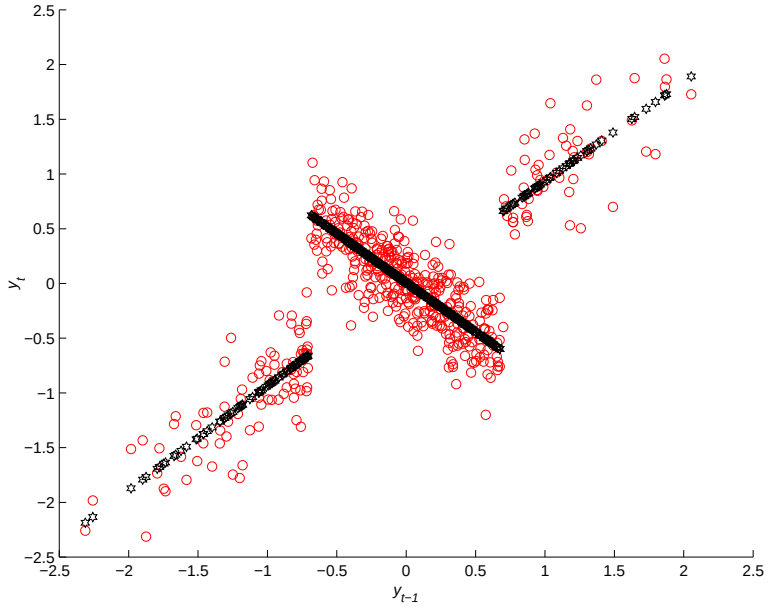

FIGURE 9. Scatter plot of y_t versus y_{t-1} and $\hat{y}_t$ versus y_{t-1}. The circles are the true values of y_t and the stars are the estimated $\hat{y}_t$

.

The second simulated model is

$$y_t = \begin{cases} 0.2 + 0.8y_{t-1} + \varepsilon_t, & \text{if } |y_{t-1} - y_{t-2}| \geq 0.2; \\ 0.2 - 0.4y_{t-1} + \varepsilon_t, & \text{otherwise,} \end{cases} \tag{19}$$

where ε_t is a normally distributed white noise process with zero mean and variance 0.01. Figure 10 shows the simulated data. In this case there are two bidimensional thresholds. The first one when $y_{t-2} - y_{t-1} = 0.2$ and the second one when $y_{t-2} - y_{t-1} = -0.2$. The order of the model is 1 and $\mathbf{x}_t^T = [y_{t-1}, y_{t-2}]$. The total number of possible hyperplanes is $500!/2!498! = 124750$.

Figure 11 shows the estimated hyperplanes by the proposed procedure. As shown in Tables 1– 3 the results are very good.

The third simulated model is described as

$$y_t = \begin{cases} 0.5 + 0.8y_{t-1} + \varepsilon_t, & \text{if } y_{t-2} \leq 0 \text{ and } y_{t-2} - y_{t-1} > 0; \\ 0.5 - 0.8y_{t-1} + \varepsilon_t, & \text{if } y_{t-2} \leq 0 \text{ and } y_{t-2} - y_{t-1} \leq 0; \\ -0.5 + 0.8y_{t-1} + \varepsilon_t, & \text{if } y_{t-2} > 0 \text{ and } y_{t-2} - y_{t-1} > 0; \\ 0.5 - 0.8y_{t-1} + \varepsilon_t, & \text{otherwise.} \end{cases} \tag{20}$$

The random term in (20) is a normally distributed zero mean white noise with unit variance. The dynamics of model (20) is controlled by two hyperplanes, defined by $y_{t-2} = 0$ and $y_{t-2} - y_{t-1} = 0$. Figures 12 and 13 show, respectively, the simulated data and the estimated hyperplanes.

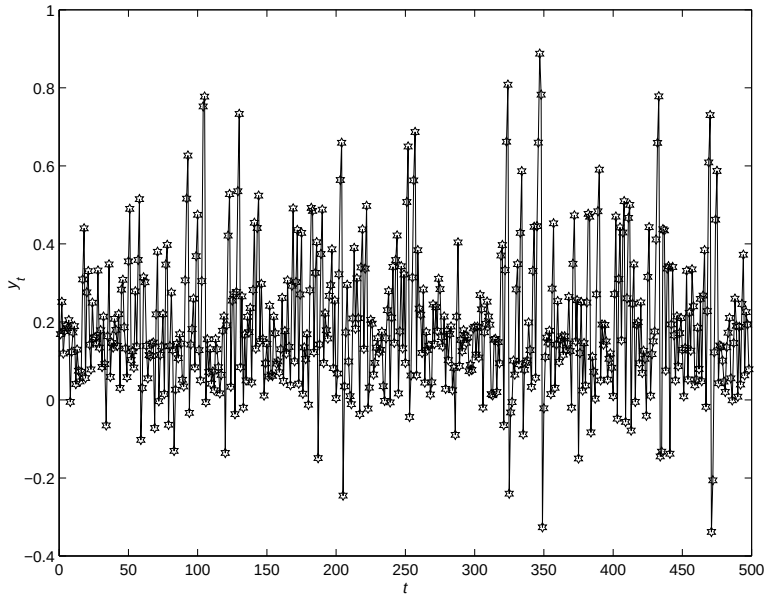

FIGURE 10. Time series generated by (19).

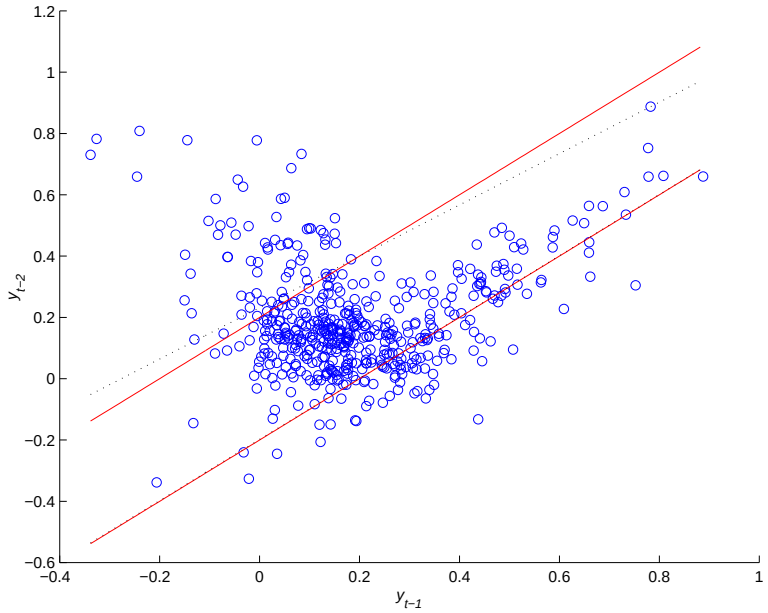

FIGURE 11. Scatter plot of y_t versus y_{t-1} with the true (solid line) and estimated (dashed line) separating hyperplanes.

The last simulated model is described as

$$y_t = \begin{cases} 0.5 + 0.8y_{t-1} + \varepsilon_t, & \text{if } y_{t-3} - y_{t-2} \leq 0 \text{ and } y_{t-2} - y_{t-1} > 0; \\ 0.5 - 0.8y_{t-1} + \varepsilon_t, & \text{if } y_{t-3} - y_{t-2} \leq 0 \text{ and } y_{t-2} - y_{t-1} \leq 0; \\ -0.5 + 0.8y_{t-1} + \varepsilon_t, & \text{if } y_{t-3} - y_{t-2} > 0 \text{ and } y_{t-2} - y_{t-1} > 0; \\ 0.5 - 0.8y_{t-1} + \varepsilon_t, & \text{otherwise,} \end{cases} \tag{21}$$

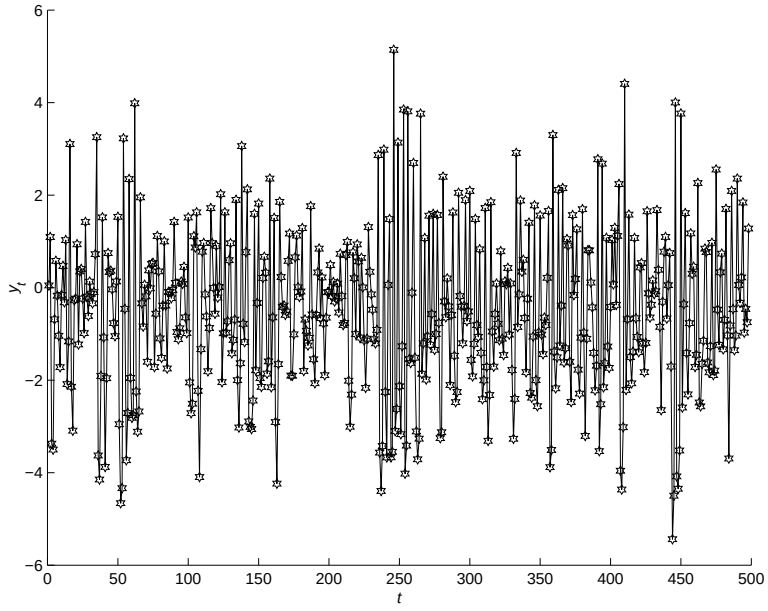

FIGURE 12. Time series generated by (20).

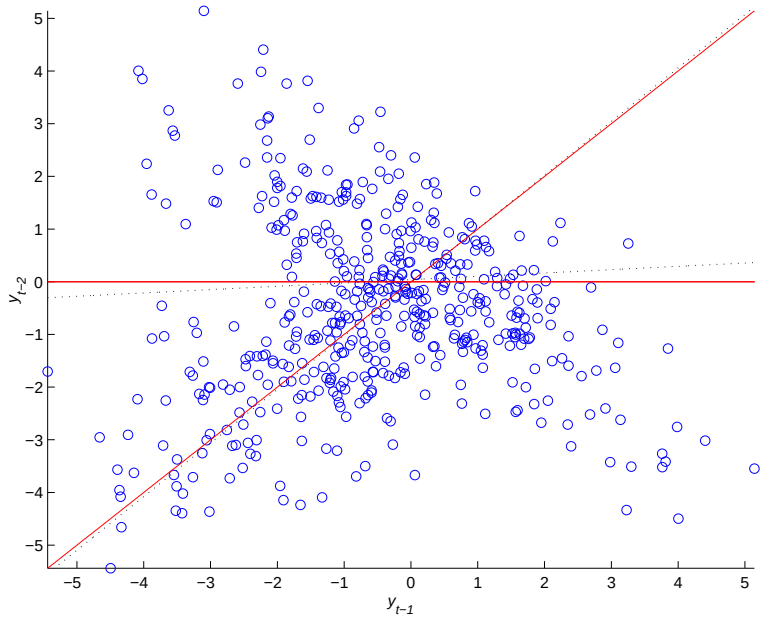

FIGURE 13. Scatter plot of y_t versus y_{t-1} with the true (solid line) and estimated (dashed line) separating hyperplanes.

where ε_t is a zero mean normally distributed white noise with unit variance. As in the previous case, the dynamics of model (21) is controlled by two separating hyperplanes defined by $y_{t-3} - y_{t-2} = 0$ and $y_{t-2} - y_{t-1} = 0$. In this case $\mathbf{x}_t^T = [y_{t-1}, y_{t-2}, y_{t-3}]$. The simulated

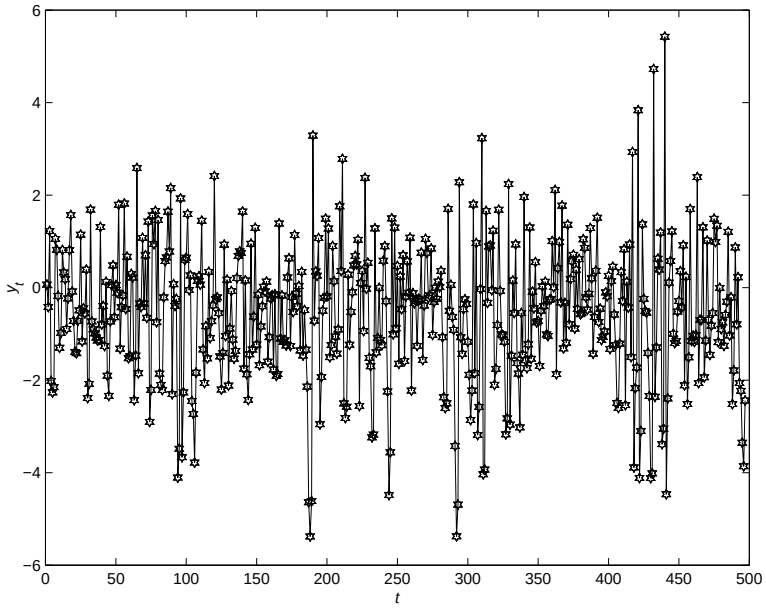

FIGURE 14. Time series generated by (21).

TABLE 1. Experimental results: Solution quality.

Model	Best soln	Optimal soln	% error
I	31.05	31.05	0
II	4.99	4.92	1.43
III	497.60	488.40	1.88
IV	549.79	534.85	2.79

TABLE 2. Experimental results: Iterations and best solutions.

Model	Iterations			Best solution		
	min	max	avg	min	max	avg
I	(8,1)	(16,4)	(10,2)	31.05	31.05	31.05
II	(2,1)	(18,1)	(9.70,1)	4.99	5.43	5.23
III	(7,1)	(20,1)	(12.50,1)	497.60	514.77	506.83
IV	(1,1)	(17,1)	(9.17,1)	549.79	560.70	554.48

time series is illustrated in Figure 14. Figure 15 shows the scatter plot of the sequence y_t generated by the optimal solution and the sequence generated by the best solution found. Except for a few points, the estimated solution correctly captures the threshold structure of the model. As shown in Tables 1– 3 and in Figure 15, the results are very encouraging.

6. CONCLUSION

This paper presents a new approach to modeling threshold processes, based on a linear model with time-varying parameters. We show that this formulation is closely related to the SETAR models with the advantage that it incorporates linear multivariate thresholds.

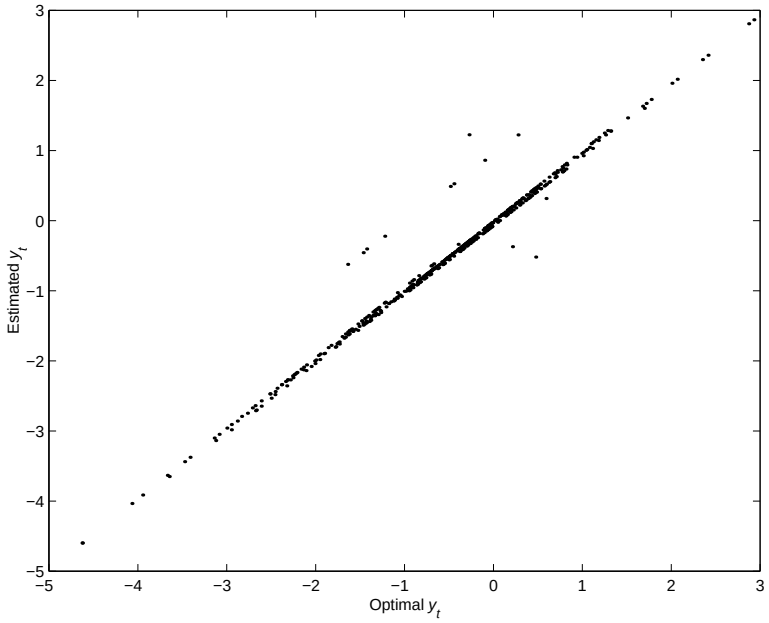

FIGURE 15. Scatter plot of the estimated y_t versus the optimal y_t.

TABLE 3. Experimental results: Solution times.

Model	Time to best (secs)			Total time (secs)		
	min	max	avg	min	max	avg
I	9.16	676.17	423.23	870.00	918.00	902.10
II	35.54	1451.29	770.55	1689.00	1724.40	1752.20
III	51.38	853.60	514.92	862.80	928.20	890.90
IV	9.69	721.45	368.17	858.60	909.00	879.60

A GRASP has been developed to estimate the parameters of the model. Experimental results show that the proposed algorithm performs well in estimating the locations and orientations of the separating hyperplanes. However, the procedure was not designed to estimate neither the number of hyperplanes nor the threshold variables. An interesting extension of the proposed algorithm would be to find the optimal number of hyperplanes and the threshold variables.

ACKNOWLEDGEMENTS

The authors would like to thank Geraldo Veiga for useful discussions and suggestions.

REFERENCES

1. H. Akaike, *Information theory and an extension of the maximum likelihood principle,* Second International Symposium on Information Theory (Budapest) (B. N. Petrov and F. Czáki, eds.), Akademiai Kiadó, 1973, pp. 267–281.
2. T. Feo and M. G. C. Resende, *A probabilistic heuristic for a computationally difficult set covering problem,* Operations Research Letters **8** (1989), 67–71.

3. T.A. Feo and M.G.C. Resende, *Greedy randomized adaptive search procedures*, Journal of Global Optimization **6** (1995), 109–133.
4. G. M. Jenkins G. E. P.Box and G.C. Reinsel, *Time series analysis, forecasting and control*, third ed., Prentice Hall, San Francisco, 1994.
5. C. W. J. Granger and T. Teräsvirta, *Modelling nonlinear economic relationships*, Oxford University Press, Oxford, 1990.
6. M.G.C. Resende, *Greedy randomized adaptive search procedures (GRASP)*, Encyclopedia of Optimization (C. Floudas and P.M. Pardalos, eds.), Kluwer Academic Publishers, 1999, To appear.
7. H. Tong, *On a threshold model*, Pattern Recognition and Signal Processing (Amsterdam) (C. H. Chen, ed.), Sijthoff and Noordhoff, 1978.
8. ———, *Threshold models in non-linear time series analysis*, Lecture Notes in Statistics, vol. 21, Springer-Verlag, Heidelberg, 1983.
9. ———, *Non-linear time series: A dynamical systems approach*, Oxford Statistical Science Series, vol. 6, Oxford University Press, Oxford, 1990.
10. H. Tong and K. S. Lim, *Threshold autoregression, limit cycles and cyclical data (with discussion)*, Journal of the Royal Statistical Society B **42** (1980), 245–292.
11. R. S. Tsay, *Testing and modeling threshold autoregressive processes*, Journal of the American Statistical Society **84** (1989), 431–452.

(M. C. Medeiros) DEPARTMENT OF ELECTRICAL ENGINEERING, CATHOLIC UNIVERSITY OF RIO DE JANEIRO, RIO DE JANEIRO, RJ, BRAZIL.
E-mail address: mcm@ele.puc-rio.br

(M. G. C. Resende) INFORMATION SCIENCES RESEARCH, AT&T LABS RESEARCH, FLORHAM PARK, NJ 07932 USA.
E-mail address: mgcr@research.att.com

(A. Veiga) DEPARTMENT OF ELECTRICAL ENGINEERING, CATHOLIC UNIVERSITY OF RIO DE JANEIRO, RIO DE JANEIRO, RJ, BRAZIL.
E-mail address: alvf@ele.puc-rio.br