Journal of Applied Mathematics and Stochastic Analysis, **13**:4 (2000), 393-410.

PARALLELIZATION ALGORITHMS FOR MODELING ARM PROCESSES

BENJAMIN MELAMED and SANTOKH SINGH
Rutgers University
Faculty of Management, Department of MSIS
94 Rockafellar Rd., Piscataway, NJ 08854
E-mail: {melamed, santokh}@rbs.rutgers. edu

(Received March, 2000; Revised September, 2000)

AutoRegressive Modular (ARM) processes are a new class of nonlinear stochastic processes, which can accurately model a large class of stochastic processes, by capturing the empirical distribution and autocorrelation function simultaneously. Given an empirical sample path, the ARM modeling procedure consists of two steps: a global search for locating the minima of a nonlinear objective function over a large parametric space, and a local optimization of optimal or near optimal models found in the first step. In particular, since the first task calls for the evaluation of the objective function at each vector of the search space, the global search is a time consuming procedure. To speed up the computations, parallelization of the global search can be effectively used by partitioning the search space among multiple processors, since the requisite communication overhead is negligible.

This paper describes two space-partitioning methods, called **Interleaving** and **Segmentation**, respectively. The speedups resulting from these methods are compared for their performance in modeling real-life data.

Key words: ARM Processes, Autocorrelation Function, Circulant Matrices, Parallel Computation, Partitioning, **GSLO** Algorithm, Optimization, Similarity Classes.

AMS subject classifications: 60K30, 62M10, 90B99.

1. Introduction

AutoRegressive Modular (ARM) processes are a new class of nonlinear stochastic processes. ARM processes can accurately model a large class of nonlinear stochastic processes, by capturing the empirical distribution and autocorrelation function simultaneously [11]. Two subclasses of ARM processes have been studied and exercised on real-life empirical data. The main class, called TES (Transform Expand Sample), has been studied in some detail [7, 8, 3, 4, 5, 9], and a software environment called TEStool, (has been created to support TES modeling [2]. A finite-state variant,

called QTES (Quantized TES) has been developed recently [10], and work is in progress to create a modeling environment to support it as well.

ARM can be used whenever one or more empirical sample paths (measurements) from a stationary stochastic process are available, and the empirical autocorrelation function exhibits significant autocorrelations. Furthermore, since ARM is non-parametric, no assumptions need to be made on the form of the stationary distribution nor on the shape of the autocorrelation function. This flexibility is useful in many application areas. For example, modeling traffic on emerging high speed networks has recently attracted considerable attention. A salient feature of projected major traffic sources is a high degree of burstiness which manifests itself in high autocorrelations. Both compressed video and file transfer are cases in point. In order to obtain good predictions of the performance measures resulting from offering a bursty traffic stream to a server system, it is important to accurately model both the marginal distribution and autocorrelation function, a task that the ARM modeling methodology is designed to carry out for a large variety of stochastic processes.

To this end, [6] proposed a TES fitting algorithm which is a combination of two stages. The first is a brute force search, to be referred to as the *global search*, and the second is a nonlinear optimization, to be referred to as the *local optimization*. The global search has an inherent high time complexity and is the subject of the present paper; the complexity of the local optimization is relatively small. More specifically, the global search requires the evaluation of a nonlinear objective function over a large multidimensional discretized parameter space, consisting of high-dimensional probability vectors (the requisite dimension is around 100). Since the search space grows explosively in the dimension, the global search is time consuming and real-time or near real-time searches are currently impossible with desktop computers. On the other hand, partitioning the search space and distributing the corresponding search subtasks to parallel processors is a natural avenue for speeding up the search. This is due to the fact that each search of the partition subspaces is entirely autonomous, requiring no communication overhead other than task distribution and returning the corresponding results.

This paper investigates two space-partitioning models, called the *Interleaving method* and the *Segmentation method*, and studies their speedup performance, using real-life data. Both methods are algorithmic enumerations of search subspaces that facilitate the global search for an ARM model.

The rest of this paper is organized as follows: Section 2 reviews briefly ARM processes and the associated modeling methodology. Section 3 describes the search space of the global search (commensurate with that used in the **GSLO** algorithm of [6]). Section 4 discusses the parallelization of ARN model fitting. Section 5 is devoted to the development of the two partitioning methods of the search space. Section 6 compares the performance of the two partitioning methods. Finally, Section 7 concludes the paper.

2. Background

This section provides a brief overview of the ARM processes TES and QTES. Throughout the paper, we assume that all stochastic processes are defined over some probability space, $(\Omega, \mathcal{F}, P)$.

2.1 TES Processes

In order to fix the ideas and keep the presentation concise, we shall focus on the main type of TES processes, called TES$^+$; the treatment of other types, such as TES$^-$, is analogous [3, 4, 7, 8]. The construction of TES$^+$ models involves two random sequences in lockstep. The first sequence, called a *background* TES$^+$ process, serves an auxiliary role, and is given by

$$U_n^+ = \begin{cases} U_0, & n = 0 \\ \langle U_{n-1}^+ + V_n \rangle, & n > 0 \end{cases} \tag{1}$$

where U_0 is distributed uniformly on $[0,1)$, $\{V_n\}_{n=1}^{\infty}$ is a sequence of iid random variables, independent of U_0, called the *innovation sequence*, and angular brackets denote the modulo-1 (fractional part) operator $\langle x \rangle = x - \max\{\text{integer } n\colon n \leq x\}$.

The second sequence, called a *foreground* TES$^+$ process and denoted by $\{X_n^+\}_{n=0}^{\infty}$, is a transformed version of a background TES$^+$ process,

$$X_n^+ = D(U_n^+), \tag{2}$$

where D is a measurable transformation from $[0,1)$ to the reals, called a *distortion*. For technical reasons, we require that $\int_0^1 D(x)dx < \infty$.

An important family of distortions is based on empirical time series, $Y = \{Y_n\}_{n=0}^{N}$; these distortions consist of composite transformations of the form

$$D_{Y,\xi}(x) = \hat{H}_Y^{-1}(S_\xi(x)), \quad x \in [0,1). \tag{3}$$

Here, the inner transformation, S_ξ, is a "smoothing" operation, called a *stitching transformation*, parameterized by $0 \leq \xi \leq 1$, and given by

$$S_\xi(y) = \begin{cases} y/\xi, & 0 \leq y \leq \xi \\ (1-y)/(1-\xi), & \xi \leq y < 1. \end{cases} \tag{4}$$

The outer transformation, $\hat{H}_Y^{-1}$, is the inverse of the empirical (histogram) distribution function computed from an empirical time series, $Y = \{Y_n\}$, as

$$\hat{H}_Y^{-1}(x) = \sum_{j=1}^{J} 1_{[\hat{C}_{j-1}, \hat{C}_j)}(x)[l_j + (x - \hat{C}_{j-1})\frac{w_j}{\hat{p}_j}], \quad 0 \leq x < 1, \tag{5}$$

where J is the number of histogram cells, $[l_j, r_j)$ is the support of cell j with width $w_j = r_j - l_j > 0$, $\hat{p}_j$ is the probability estimator of cell j and $\{\hat{C}_i\}_{i=0}^{J}$ is the cdf of $\{\hat{p}_j\}_{j=1}^{J}$, i.e., $\hat{C}_j = \sum_{i=1}^{j} \hat{p}_i$, $1 \leq j \leq J$ ($\hat{C}_0 = 0$ and $\hat{C}_J = 1$).

Because stitching transformations preserve uniformity [7], the inversion method can still be applied to stitched background process $\{S_\xi(U_n^+)\}$. It follows that any foreground TES$^+$ sequence of the form

$$X_n^+ = \hat{H}_Y^{-1}(S_\xi(U_n^+)) \tag{6}$$

is guaranteed to have the empirical (histogram) distribution $\hat{H}_Y$, *regardless* of the innovation density f_V and stitching parameter ξ selected. The choice of a pair (ξ, f_V) will largely determine the dependence structure of (6) and, in particular, its autocorrelation function. Thus, TES modeling decouples the fitting of the empirical distribution from the fitting of the empirical autocorrelation function. Since the former is guaranteed, one can concentrate on the latter.

In practice, TES modeling has the form of a heuristic search for a suitable pair (ξ, f_V), such that the ensuing TES model approximates well the leading empirical autocorrelations and gives rise to sample path realizations which adequately resemble the empirical record. The transition density and autocorrelation functions of TES processes, required for the purpose of modeling, can be computed from fast and accurate formulas developed in [3, 4]. In particular, background TES$^+$ processes are stationary Markovian and uniform on $[0,1)$, and the τ-step transition density of $\{U_n^+\}$ is given by [3, 8]

$$g_\tau^+(y \mid x) = \begin{cases} \sum_{\nu=-\infty}^{\infty} \tilde{f}_V^\tau(i2\pi\nu) e^{i2\pi\nu(y-x)}, & 0 \le y, x < 1 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where $\tilde{f}(s) = \int_0^\infty e^{-sx} f(x)dx$ denotes the Laplace Transform of f. The autocorrelation function of the foreground process $\{X_n^+\}$ is given by [3, 8]

$$\rho_X^+(\tau) = \frac{2}{\sigma_X^2} \sum_{\nu=1}^{\infty} Re[\tilde{f}_V^\tau(i2\pi\nu)] \mid \tilde{D}(i2\pi\nu) \mid^2, \tag{8}$$

where σ_X^2 is the common (finite) variance of the X_n^+.

2.2 QTES Processes

Like its continuous analog, TES, the QTES class consists of background and foreground sequences. In analogy to Section 2.1, we focus on QTES$^+$ processes.

A *background* QTES$^+$ process is a quantized (discretized) variant of a background TES$^+$ process over a finite state space. To this end, we partition the unit circle (of circumference 1) into a finite number of subintervals (called *slices*). Specifically, let $\mathfrak{Z}_M = \{0, 1, \ldots, M-1\}$, for some integer $M > 0$. A quantization of the unit circle, $[0,1)$, is a partition

$$\Gamma(M) = \{\Gamma_m = [m\delta, (m+1)\delta) : m \in \mathfrak{Z}_M\}$$

of the interval $[0,1)$ into M subintervals (slices), Γ_m, each of length $\delta = \delta(M) = 1/M$. The usual integer-valued modulo-M operator from the integers onto $\mathfrak{Z}_M$ will be denoted by $\langle m \rangle_M = m - M\lfloor \frac{m}{M} \rfloor$.

Let C_0 be an integer-valued random variable, distributed uniformly over $\mathfrak{Z}_M$, i.e., $P\{C_0 = m\} = 1/M$, for all $m \in \mathfrak{Z}_M$. Let further, $\{J_n\}_{n=1}^\infty$ be an iid sequence of integer-valued random variables, independent of C_0 (the *innovation sequence*). The common distribution of the J_n is denoted by $f_J(j) = P\{J_n = j\} = \phi_j$, $j \in \mathfrak{Z}_M$. A *background* QTES$^+$ process is defined by

$$C_n^+ = \begin{cases} C_0, & n = 0 \\ \langle C_{n-1}^+ + J_n \rangle_M, & n > 0. \end{cases} \tag{9}$$

Background QTES$^+$ processes of the form (9) are stationary Markovian, and their marginal distributions are uniform on $\mathfrak{Z}_M$, i.e.,

$$P\{C_n^+ = m\} = \frac{1}{M} = \delta, \quad m \in \mathfrak{Z}_M. \tag{10}$$

Their τ-step transition probabilities, denoted by $h_\tau^+(k \mid j) = P\{C_{n+\tau}^+ = k \mid C_n^+ = j\}$, have the representation

$$h_\tau^+(k \mid j) = \frac{1}{M} \sum_{\nu=0}^{M-1} \tilde{f}_J^\tau(i2\pi\nu) e^{i2\pi\nu(k-j)/M}, \tag{11}$$

where

$$\tilde{f}_J(i2\pi\nu) = \sum_{m=0}^{M-1} f_J(m) e^{-i2\pi\nu m/M}, \quad \nu \in \mathfrak{Z}_M, \tag{12}$$

is the discrete Fourier transform of $f_J \equiv \{\phi_m = f_J(m): m \in \mathfrak{Z}_M\}$.

The construction of a foreground QTES$^+$ process associates with each state $m \in \mathfrak{Z}_M$ of the modulating process, $\{C_n^+\}$, a random variable with a marginally uniform distribution on the corresponding slice, $\Gamma_m = [m\delta, (m+1)\delta)$. The process $\{Q_n^+\}$ is a continuous extension of $\{C_n^+\}$, defined by

$$Q_n^+ = \sum_{m \in \mathfrak{Z}_m} 1\{C_n^+ = m\} \delta(m + W_n), \tag{13}$$

where the sequence $\{W_n\}$ is iid with uniform marginals on $[0,1)$, so that $\delta(m + W_n)$ is uniform on $\Gamma_m = [m\delta, (m+1)\delta)$. In other words, Q_n^+ is conditionally uniform on Γ_m, given that $C_n^+ = m$. Combining (10) and (13), it follows that $\{Q_n^+\}$ is uniform on $[0,1)$, with τ-step transition density

$$q_\tau^+(y \mid x) = h_\tau^+(I_\Gamma(y) \mid I_\Gamma(x)) \frac{1}{\delta}, \tag{14}$$

where $I_\Gamma(z)$ is the (unique) slice index $k = k(z)$ in the partition Γ, such that $z \in \Gamma_k$.

Let D be a distortion such that $\int_0^1 D(x)dx < \infty$, and let $\{Z_n^+\}$ be the corresponding QTES$^+$ foreground sequence, with elements $Z_n^+ = D(Q_n^+)$. Thus, $\{Z_n^+\}$ has the same marginal distribution as the TES$^+$ foreground sequence, $\{X_n^+\}$. The autocorrelation function of the process $\{Z_n^+\}$ is given by (see [10] for details)

$$\rho_Z^+(\tau) = \frac{\delta^2}{\sigma_Z^2} \sum_{\nu=1}^{M-1} \tilde{f}_J^\tau(i2\pi\nu) \mid \tilde{D}(i2\pi\nu) \mid^2. \tag{15}$$

A QTES$^+$ model can be fitted to empirical data by searching for a pair (ξ, f_J) analogously to the searches implemented in TEStool. Alternatively, a TES$^+$ process can be readily approximated by a QTES$^+$ process, by discretizing f_V in a natural way into a corresponding f_J, where

$$f_J(m) = \int_{\Gamma_m} f_V(x)dx, \quad 0 \le m \le M-1. \tag{16}$$

2.3 The ARM Modeling Methodology

The ARM modeling methodology produces an exact match to the empirical distribution, and simultaneously approximates the empirical autocorrelation function. Recall that a key feature of this methodology is the decoupling of the fitting of these two empirical statistics. In fact, the underlying theory ensures an automatic matching of the empirical distribution [3, 8], a fact that allows the modeling procedure to focus on fitting the empirical autocorrelation function. The latter fit is accomplished by a search for a suitable innovation density, f, and stitching parameter, ξ. More specifically, for a given empirical sequence $Y = \{Y_n\}$, we search for a pair (f, ξ) and the resultant ARM model, such that the empirical autocorrelation function, $\widehat{\rho}_Y$, is adequately approximated by the model autocorrelation function, $\rho_{f,\xi}$, where $f = f_V$ for TES processes, and $f = f_J$ for QTES processes.

In practice, the fitting problem is cast as the following nonlinear optimization problem: Find an optimal point of innovation density and stitching parameter, (f^*, ξ^*), such that

$$(f^*, \xi^*) = \underset{(f,\xi)}{\operatorname{argmin}} \{g(f,\xi) = \sum_{\tau=1}^{T} a_\tau [\rho_{f,\xi}(\tau) - \widehat{\rho}_Y(\tau)]^2\}, \tag{17}$$

where T is the maximal lag of the empirical autocorrelation function. To this end, we select a positive integer K and restrict the search space of f to the parameter space

$$\mathcal{G}_K = \{(\mathbf{P}, \xi): \mathbf{P} \in \mathcal{P}_K, \xi \in [0,1]\}, \tag{18}$$

where $\mathcal{P}_K$ is a set of discrete densities $\mathbf{P} = (P_1, P_2, \ldots, P_K)$, $\sum_{n=1}^{K} P_n = 1$ (see [6] for details). This reduces the optimization problem (17) to finding optimal parameters, $(\mathbf{P}^*, \xi^*) \in \mathcal{G}_K$, such that

$$(\mathbf{P}^*, \xi^*) = \underset{(\mathbf{P},\xi) \in \mathcal{G}_K}{\operatorname{argmin}} \; g_K(\mathbf{P}, \xi), \tag{19}$$

where the objective function g_K has the form

$$g_K(\mathbf{P}, \xi) = \sum_{\tau=1}^{T} a_\tau [\rho_{\mathbf{P},\xi}(\tau) - \widehat{\rho}_Y(\tau)]^2. \tag{20}$$

To solve the optimization problem (19)-(20), reference [6] proposed the so-called **GLSO** algorithm, which consists of two steps:

- **GS** Discretize the parameter space $\mathcal{G}_K$, into a finite number of points Then, for each such point, evaluate the objective function g_K and keep the best B points (those points, $\mathbf{x} \in \mathcal{G}_K$, which give rise to the B smallest values of $g_K(\mathbf{x})$).
- **LO** Using each of these B points as a starting point, find a local minimum of g_K via a nonlinear programming algorithm. Then, select among them that point, $\mathbf{x}^*$, which gives rise to the smallest local minimum, $g_K(\mathbf{x}^*)$.

The computational bottleneck lies in the execution of the **GS** step due to its large size. In fact, the total number, N_{tot}, of points, $\mathbf{x}$, at which the nonlinear function

$g_K(\mathbf{x})$ has to be evaluated is

$$N_{tot} = N_\xi \begin{pmatrix} N_\mathbf{P} + K - 2 \\ N_\mathbf{P} - 1 \end{pmatrix} \tag{21}$$

where $N_\mathbf{P}$ and N_ξ are the number of equidistant values that each P_n and ξ can assume, respectively. If we wish to search over additional ARM classes (e.g., TES$^+$ and TES$^-$ or QTES$^+$ and TES$^-$), then the value of (21) should be doubled. Clearly, with the recommended value of K ($K = 100$), even moderate values of $N_\mathbf{P}$ and N_ξ render the value of N_{tot} in (21) rather large. This calls for speeding up the **GS** step. The present paper focuses on achieving this by means of parallelization in a natural way as follows:

1. Given m processors, $j = 0, 1, \ldots, m-1$, the first processor ($j = 0$) is designated as master.
2. The master processor partitions the search space, $\mathcal{G}_K$, into roughly equal-sized search subspaces, $\mathcal{G}_K(1), \ldots, \mathcal{G}_K(m)$.
3. Each processor j is handed over the corresponding optimization subproblem, namely, the search for the best B solutions of the optimization problem

$$\underset{(\mathbf{P},\xi) \in \mathcal{G}_K(j)}{\operatorname{argmin}} \; g_K(\mathbf{P}, \xi). \tag{22}$$

4. The mB best solution of all processes are returned to the master processor, which selects from their union the best B solutions.

3. The Quantized Search Space

The search space $\mathcal{G}_K$ in Equation (18) is too broad for a practical global search in the **GS** step of the **GSLO** algorithm. Consequently, we quantize $\mathcal{G}_K$ as follows.

The scalar component, ξ, of stitching parameters is quantized simply by selecting a finite number of values in the interval $[0, 1]$, and defining

$$\Xi = \{0 \le \xi_1 \le \ldots \le \xi_L \le 1\}, \tag{23}$$

where L is some selected quantization level, and the ξ_i are usually selected equidistant.

The vector component, $\mathbf{P}$, of probability vectors is restricted to lie in a finite space. To this end, we select an integer $K > 0$ and a quantization level $M > 0$, and define a vector space with integer components

$$\mathcal{H}_{K,M} = \left\{ \mathbf{v} = (j_1, j_2, \ldots, j_K) : 0 \le j_i \le M, 1 \le i \le M, \sum_{i=1}^{K} j_i = M \right\}. \tag{24}$$

Letting now $q = 1/M$, the quantized search space of probability vectors is defined by

$$\mathcal{Q}_K = \{q(j_1, j_2, \ldots, j_K) : (j_1, j_2, \ldots, j_K) \in \mathcal{H}_{K,M}\}. \tag{25}$$

It is convenient to partition $\mathcal{H}_{K,M}$ into subsets of vectors $\mathbf{v}$ with equal element multiplicities $n_i = n_i(\mathbf{v})$, where n_i is the multiplicity of i in $\mathbf{v}$, if i occurs precisely n_i times

in **v**. Then, the cardinality of $\mathcal{H}_{K,M}$ is

$$|\mathcal{H}_{K,M}| = \sum_{\substack{0 \le n_0,\ldots,n_M \le K \\ n_0+\ldots+n_M=K}} \frac{K!}{\prod_{j=0}^{M} n_j!}. \tag{26}$$

Thus, even for a moderately large K, the cardinality of $\mathcal{H}_{K,M}$ is very large. Evidently, the task of enumerating the elements of $\mathcal{H}_{K,M}$ or Q_K and evaluating the objective function for each element in the latter can be rather time consuming.

4. Search Parallelization

To speedup the search over $\mathcal{H}_{K,M}$ we employ m parallel processors, taking advantage of the fact that searches on disjoint subsets do not interact, except when the results are returned. To this end, subdivide $\mathcal{H}_{K,M}$ into subspaces $\mathcal{H}_{K,M}(j)$, $j = 0,1,\ldots,m-1$, such that $\mathcal{H}_{K,M} = \bigcup_{j=0}^{m-1} \mathcal{H}_{K,M}(j)$, and assign processor j to search subspace $\mathcal{H}_{K,M}(j)$. Obviously, for identical processors, the best speedup is attained when $\mathcal{H}_{K,M}$ is partitioned into disjoint pieces of (roughly) equal size. Partitioning means that $\mathcal{H}_{K,M}(i) \cap \mathcal{H}_{K,M}(j) = \phi$ for all $i \neq j$, so that processors do not overlap their searches. The roughly equal size of the $\mathcal{H}_{K,M}(j)$ balances the computational load over all processors. In practice, we take

$$|\mathcal{H}_{K,M}(j)| = \begin{cases} \left\lfloor \dfrac{|\mathcal{H}_{K,M}|}{m} \right\rfloor, & 1 \le j \le m-1 \\ |\mathcal{H}_{K,M}| - (m-1)\left\lfloor \dfrac{|\mathcal{H}_{K,M}|}{m} \right\rfloor, & j = 0, \end{cases} \tag{27}$$

where $\lfloor \cdot \rfloor$ denotes the floor operator, $\lfloor x \rfloor = \max\{\text{integer } n : n \le x\}$.

We implemented the parallelization using the PVM (Parallel Virtual Machine) package in a natural way. Given $m > 1$ processors, $0,\ldots,m-1$, one of them (say, processor 0) is designated the *master*, and the remaining processors, $1,\ldots,m-1$, are designated *slaves*. The master processor is responsible for partitioning $\mathcal{H}_{K,M}$ and distributing the descriptions of subsearches over the $\mathcal{H}_{K,M}(j)$ to the slave processors. Each processor then proceeds to search for the best B models, and the slave processors return their results to the master processor. The latter completes the search by selecting the best B models among the mB returned by all subsearches as well as post-processing the results.

5. Partitioning and Enumeration Methods

The efficacy of parallelization as a speedup approach depends greatly on efficient algorithms for partitioning $\mathcal{H}_{K,M}$ into subspaces $\mathcal{H}_{K,M}(j)$, and on enumerating the members of each $\mathcal{H}_{K,M}(j)$. Let $I_{\mathcal{H}_{K,M}} = \{1,\ldots,|\mathcal{H}_{K,M}|\}$ the be index set of the search space $\mathcal{H}_{K,M}$ under some enumeration, where the cardinality of $\mathcal{H}_{K,M}$ is given by (26), and recall the notation $\langle n \rangle_N = n(\mathrm{mod} N)$. This section presents two methods

of producing balanced partitions and efficient enumerations as per the discussion in Section 4.

5.1 The Interleaving Method

The *Interleaving* method partitions $\mathcal{H}_{K,M}$ into subsets $\mathcal{H}_{K,M}(j)$ given by

$$\mathcal{H}_{K,M}(j) = \{\mathbf{v}(i) \in \mathcal{H}_{K,M}\colon \langle i \rangle_m = j\},\ j = 0, 1, \ldots, m-1. \tag{28}$$

Thus, the corresponding index set $I_{\mathcal{H}_{K,M}(j)}$ consists of the elements $\{j, j+m, j+2m, \ldots\}$. The term 'interleaving' is inspired by this construction.

The Interleaving method has two main advantages. First, it is conceptually extremely simple, once an enumeration method has been selected. And second, it works for any enumeration. Its main disadvantage is the "wasted" enumeration work performed by processor j when fetching the next element of $\mathcal{H}_{K,M}(j)$. To this end, processor j "skips" all indices following $j + im \in I_{\mathcal{H}_{K,M}(j)}$ up until index $j + (i+1)m \in I_{\mathcal{H}_{K,M}(j)}$. Since every such fetch operation (beyond the first one) requires m iterations of the enumeration algorithm, this operation gets even more "wasteful" as the number of parallel processors, m, increases.

5.2 The Segmentation Method

The *Segmentation* method enjoys a major advantage in that it provides an explicit mapping, $e(n)$, from $I_{\mathcal{H}_{K,M}}$ to $\mathcal{H}_{K,M}$. Since the index domain, $I_{\mathcal{H}_{K,M}}$, is trivial to partition and the resulting index sets $I_{\mathcal{H}_{K,M}(j)}$ are trivial to enumerate (as sets of consecutive integers), the mapping $e(n)$ induces a partition of $\mathcal{H}_{K,M}$ as well as enumerations of the resulting $\mathcal{H}_{K,M}(j)$ in a natural way. Unlike the Interleaving method, the Segmentation method performs efficient enumeration, since the mapping $e(n)$ obviates any need for "skipping". However, enumeration via the Segmentation method is conceptually more complex than in the Interleaving method.

We begin with definitions of various types of circulant matrices (see [1] for more details.)

Definition 1: A $K \times K$ matrix $\mathbf{C}$ is a *circulant* matrix, if its elements satisfy the relation

$$c_{i,j} = c_{1,\langle j-i \rangle_K + 1},\ \ 1 \le i, j \le K, \tag{29}$$

where $c_{i,j}$ is the (i,j)-th element of the matrix $\mathbf{C}$.

It follows that in a circulant matrix, each row can be obtained by a circular right shift operation on the preceding row, so that a circulant matrix has the form

$$\mathbf{C} = \begin{pmatrix} c_1 & c_2 & \dots & c_K \\ c_K & c_1 & \dots & c_{K-1} \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ c_2 & c_3 & \dots & c_1 \end{pmatrix}.$$

Definition 2: The *basic circulant* matrix of order K is the $K \times K$ circulant matrix

$$\mathbf{B}_K = \begin{pmatrix} 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \cdot & \dots & \cdot \\ 1 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}. \tag{30}$$

Definition 3: The *s-circulant* matrix of a circulant matrix $\mathbf{C}$ of order K is the $K \times K$ matrix

$$\mathbf{C}^{(s)} = \begin{pmatrix} c_1 & c_2 & \dots & c_K \\ c_{\langle K-s+1\rangle_K} & c_{\langle K-s+2\rangle_K} & \dots & c_{\langle K-s\rangle_K} \\ c_{\langle K-2s+1\rangle_K} & c_{\langle K-2s+2\rangle_K} & \dots & c_{\langle K-2s\rangle_K} \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ c_{\langle s+1\rangle_K} & c_{\langle s+2\rangle_K} & \dots & c_{\langle s\rangle_K} \end{pmatrix}. \tag{31}$$

The next definition is a notion of decomposition of a positive integer into all possible vectors of positive addends.

Definition 4: For a given integer $M > 0$, a *decomposition set* $\mathfrak{D}_M$ is defined as

$$\mathfrak{D}_M = \left\{ \mathbf{d} = (d_1, \dots, d_{n_\mathbf{d}}) : 0 < d_j \le M, d_j \ge d_{j+1}, \sum_{j=1}^{n_\mathbf{d}} d_j = M \right\}. \tag{32}$$

Thus, a decomposition vector $\mathbf{d} \in \mathfrak{D}_M$ consists of a monotonically decreasing sequence of positive integers that sum to M.

Finally, the last definition groups together circulant matrices that are related via a common decomposition vector, $\mathbf{d}$.

Definition 5: Let $\mathbf{d} \in \mathfrak{D}_M$ be a decomposition vector, and let $K \geq M$ be a given integer. The *Similarity class* $\mathcal{S}_K(\mathbf{d})$ is the set of all $K \times K$ circulant matrices whose non-zero row elements coincide with the elements of $\mathbf{d}$.

Note that the above definition is well-defined due to the 1-1 correspondence

$$(K, \mathbf{d}) \rightarrow \mathcal{S}_K(\mathbf{d}). \tag{33}$$

Henceforth, we let $I_{\mathfrak{D}_M}$ and $I_{\mathcal{S}_K(\mathbf{d})}$ denote index sets corresponding to the sets in the respective subscripts. The selection of the corresponding set enumerations are immaterial, unless otherwise specified.

5.2.1 Generating decomposition vectors

This subsection presents an algorithm for generating and enumerating all decomposition vectors $\mathbf{d} \in \mathfrak{D}_M$, for a given $M > 0$. It will be used later on to enumerate the search space $\mathfrak{H}_{K,M}$ and its subspaces $\mathfrak{H}_{K,M}(j)$, $j = 0, 1, \ldots, m-1$. In the algorithm to follow, $concat(\mathbf{v}, a)$ concatenates a vector $\mathbf{v} = (v_1, \ldots, v_n)$ of arbitrary length $n > 0$ with the scalar a, returning the vector $(v_1, \ldots, v_n, a)$. The procedure below is invoked by $decompose(\mathbf{null}, M, M)$, where **null** denotes the null vector.

Decomposition Vectors Enumeration Algorithm

Input: quantization level $M > 0$
Output: an enumeration of the set $\mathfrak{D}_M$
procedure *decompose*(vector $\mathbf{v}$, integer l, integer r)
  if $l \geq r$
    output $concat(\mathbf{v}, r)$;
  end if
  for $n = 1, \ldots, \min(l, r-1)$
    $decompose(concat(\mathbf{v}, n), n, r-n)$;
  end for
end *decompose*

5.2.2 Construction and enumeration of similarity classes

Similarity classes are fundamental to the enumeration of vectors in search space. In this subsection we construct and enumerate the members of a given similarity class.

For given $\mathbf{d} = (d_1, \ldots, d_{n_\mathbf{d}}) \in \mathfrak{D}_M$ and $K \geq M$, we now show how to construct the similarity class $\mathcal{S}_K(\mathbf{d})$; this construction includes an enumeration of the elements of $\mathcal{S}_K(\mathbf{d})$. Observe that the vectors in $\mathcal{S}_K(\mathbf{d})$ correspond to all possible arrangements, in K positions, of elements from a set consisting of the elements of $\mathbf{d}$ and zero. Consequently,

$$|\mathcal{S}_K(\mathbf{d})| = \frac{(K-1)!}{\prod_{i=0}^{M} m_i!}, \tag{34}$$

where for $i > 0$, m_i is the multiplicity of the i-th element in the vector $\mathbf{d}$, and $m_0 = K - \sum_{j=1}^{M} m_j$, i.e., the multiplicity of 0. We can now define similarity classes and

enumerate their elements (circulants) as follows:

$$\mathcal{S}_K(\mathbf{d}) = \left\{ \mathbf{C}(s) = d_1 \mathbf{I}_K + \sum_{j=2}^{n_\mathbf{d}} d_j \mathbf{B}_K^{(s+j-3)}: \ s = 1,2,\ldots, \mid \mathcal{S}_K(\mathbf{d}) \mid \right\}, \tag{35}$$

where $\mathbf{I}_K$ is the $K \times K$ identity matrix, and $\mathbf{B}_K^{(s)}$ is the s-circulant matrix (see Definition 3) of the basic circulant matrix $\mathbf{B}_K$ (see Definition 2).

We are now in a position to make a key observation, namely, that the search space $\mathcal{H}_{K,M}$ can be represented as the collection of all row vectors from circulants matrices, as the circulant matrices range over all similarity classes. Formally, for integers $M > 0$ and $K \geq M$, we can write

$$\mathcal{H}_{K,M} = \{ \cup R(\mathbf{C}): \ \mathbf{C} \in \mathcal{S}_K(\mathbf{d}), \ \mathbf{d} \in \mathcal{D}_M \}, \tag{36}$$

where R is the operator that unravels a matrix $\mathbf{C}$ into the set of all its row vectors.

5.2.3 Enumerating search spaces

We are now in a position to enumerate the search space $\mathcal{H}_{K,M}$ and its component subspaces, $\mathcal{H}_{K,M}(j)$, by exhibiting the mapping from an index $i \in I_{\mathcal{H}_{K,M}}$ to a corresponding vector $\mathbf{v}(i) \in \mathcal{H}_{K,M}$, with the aid of the representation (36) and the enumeration of $\mathcal{D}_M$ from Section 5.2.1. The enumeration mapping $i \rightarrow \mathbf{v}(i)$ will be constructed in a three-step algorithm as follows.

Search Space Enumeration Algorithm

1. **Select a similarity class.**
   Given an index $i \in I_{\mathcal{H}_{K,M}}$, map i to the index $n(i) \in I_{\mathcal{D}_M}$ that satisfies

$$\sum_{j=0}^{n(i)-1} \mid \mathcal{S}_K(\mathbf{d}(j)) \mid \ < i \leq \sum_{j=0}^{n(i)} \mid \mathcal{S}_K(\mathbf{d}(j)) \mid . \tag{37}$$

   The index $n(i)$ specifies an index of a decomposition vector, $\mathbf{d}(n(i))$, in the enumeration of Section 5.2.1, as well as the corresponding similarity class, $\mathcal{S}_K(\mathbf{d}(n(i)))$, given in Equation (35), with the convention $\mathcal{S}_K(\mathbf{d}(0)) = \Phi$.
2. **Select a circulant matrix from the similarity class.**
   Map the pair $(i, n(i))$ to the index $s = s(i,n) \in I_{\mathcal{S}_K(\mathbf{d}(n(i)))}$, given by

$$s(i,n) = \left\lfloor \frac{K + i - 1 - \sum_{j=0}^{n(i)-1} \mid S_K(\mathbf{d}(j)) \mid}{K} \right\rfloor. \tag{38}$$

   The index $s(i,n)$ specifies a circulant matrix $\mathbf{C}(s(i,n))$ in the enumeration of $\mathcal{S}_K(\mathbf{d}(n(i)))$, given in Equation (35).
3. **Select a row vector from the circulant matrix.**
   Map the index $s = s(i,n)$ to the vector $\mathbf{v}(i)$ with elements $v_j(i)$, given by

$$v_j(i) = c_{l(i),j}(s), \tag{39}$$

where

$$l(i) = \langle i + K - 1 \rangle_K + 1. \tag{40}$$

Thus, the vector $\mathbf{v}(i)$ coincides with the vector in row $l(i)$ of the circulant matrix $\mathbf{C}(s(i,n))$. More specifically,

$$v_j(i) = c_{l(i),j}(s) = \begin{cases} \sum_{k=2}^{n_{\mathbf{d}(k)}} d_k(n(i)) \; b_{l(i),j}^{(s+k-3)}, & l(i) \neq j \\ M - \sum_{k=2}^{n_{\mathbf{d}(k)}} d_k(n(i)), & l(i) = j \end{cases} \tag{41}$$

where $d_k(n(i))$ is the k-th element of the decomposition vector $\mathbf{d}(n(i))$, $n_{\mathbf{d}(k)}$ is the number of elements in the decomposition vector $\mathbf{d}(k)$, and $b_{l(i),j}^{(s+k-3)}$ is the $(l(i),j)$-th element of the matrix $\mathbf{B}_K^{(s+k-3)}$, defined as the $(s+k-3)$-th circulant matrix (Definition 3) of the basic circulant matrix $\mathbf{B}_K$ (Definition 2).

We remark that the enumeration algorithm above need not be used to enumerate all vectors, $\mathbf{v}(i)$. Since the enumeration yields successive row vectors within each circulant matrix, it suffices to use the circulant property to enumerate vectors that belong to the same circulant matrix. More specifically, if the successive indices i and $i+1$ in $I_{\mathfrak{H}_{K,M}}$ map to row vectors of the same circulant matrix, then $\mathbf{v}(i+1)$ can be obtained recursively from $\mathbf{v}(i)$ from the relation

$$v_j(i+1) = v_{\langle j-1 \rangle_K}(i), \quad j = 1,2,\ldots,K. \tag{42}$$

Equation (42) results in a considerable computational saving, as compared to the preceding full-blown search space enumeration algorithm.

5.3 Partitioning into Search Subspaces

Partitioning the search space $\mathfrak{H}_{K,M}$ into load-balanced search subspaces, $\mathfrak{H}_{K,M}(j)$, is straight-forward. To this end, just partition the index set $I_{\mathfrak{H}_{K,M}}$ into subsets $I_{\mathfrak{H}_{K,M}}(j)$, $j = 0,1,\ldots,m-1$; this induces a partition of $\mathfrak{H}_{K,M}$ into subsets $\mathfrak{H}_{K,M}(j)$ in a natural way. Thus, the first index in each subset $I_{\mathfrak{H}_{K,M}}(j)$, $j = 0,1,\ldots,m-1$, is

$$\langle j-1 \rangle_m \left\lfloor \frac{|\,\mathfrak{H}_{K,M}\,|}{m} \right\rfloor + 1. \tag{43}$$

Note that all the subspaces are of equal size, except possibly for $\mathfrak{H}_{K,M}(0)$, which may be slightly larger. However, the magnitude of the imbalance does not exceed m. Thus, Equation (43) provides for an optimal load balancing.

6. Comparison of the Interleaving and Segmentation Methods

This section compares the speed of the two partitioning methods, Interleaving and Segmentation, in terms of elapsed time on Unix machines with the Solaris 2 operating system running the parallelization platform PVM version 3.0. Because the parallelization was run on public computers with variable load, the comparison here is statistical in the sense that we ran three replications of each problem and measured the average elapsed times.

Number of innovation atoms	Partitioning method	Number of processors 1	2	3	4
40	Interleaving	223	181	151	124
	Segmentation	230	179	145	115
60	Interleaving	904	564	393	314
	Segmentation	911	535	340	269
80	Interleaving	2692	2199	1792	1496
	Segmentation	2699	2101	1667	1365

Table 1: Comparison of average elapsed time (in seconds) in the Interleaving and Segmentation methods for laser data

Number of innovation atoms	Partitioning method	Number of processors 1	2	3	4
40	Interleaving	462	406	380	363
	Segmentation	469	400	368	350
60	Interleaving	1357	1202	1001	879
	Segmentation	1365	1150	905	790
80	Interleaving	3365	2611	2015	1653
	Segmentation	3380	2381	1701	1251

Table 2: Comparison of average elapsed time (in seconds) in the Interleaving and Segmentation methods for football data

Number of innovation atoms	Partitioning method	Number of processors 1	2	3	4
40	Interleaving	435	389	345	325
	Segmentation	438	378	333	312
60	Interleaving	1328	1055	886	806
	Segmentation	1331	987	799	732
80	Interleaving	3299	2279	1933	1661
	Segmentation	3307	2056	1698	1336

Table 3: Comparison of average elapsed time (in seconds) in the Interleaving and Segmentation methods for financial data

Comparisons were made for three sets of empirical data:

1. Time series of 1000 data points of laser (random) brightness.
2. Time series of 210 data points of the bit rate (picture size) of compressed video, using the H.261 compression standard. The video scene depicted a

few seconds of a football game.

3. Time series of 168 data points of security prices, adjusted for inflation.

The experiments conducted varied the number of processors ($m = 1, 2, 3, 4$), and the number of innovation atoms ($K = 40, 60, 80$) in the fitted QTES models. The average elapsed time (of 3 replications per setting of m and K) are displayed in Table 1-3, for the laser data, football data and financial data, respectively.

A pictorial representation of each table is depicted in the corresponding Figures 1-3.

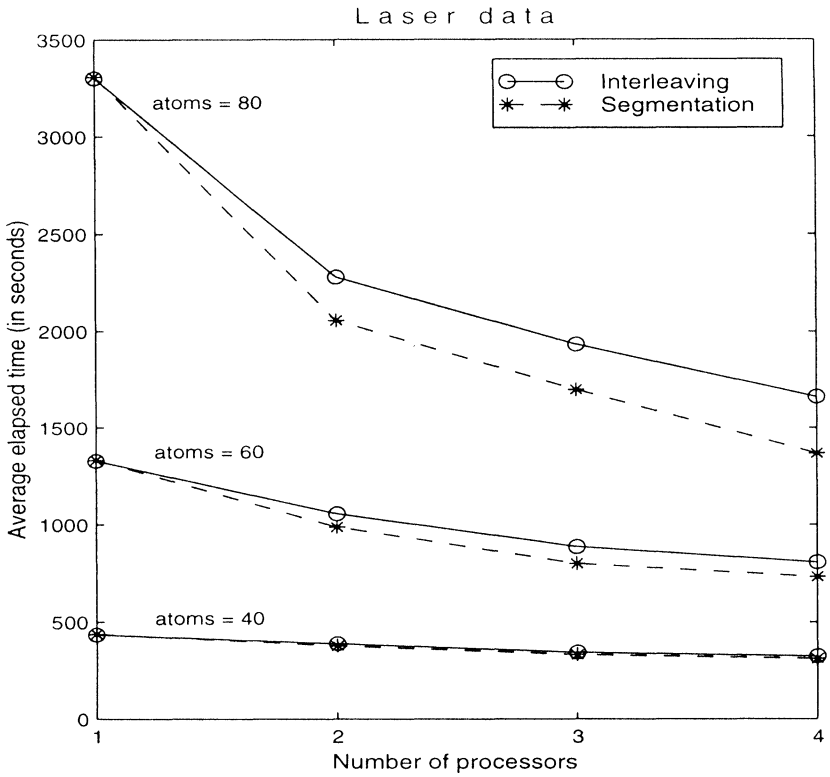

Figure 1: Comparison of the Interleaving and Segmentation methods for laser data

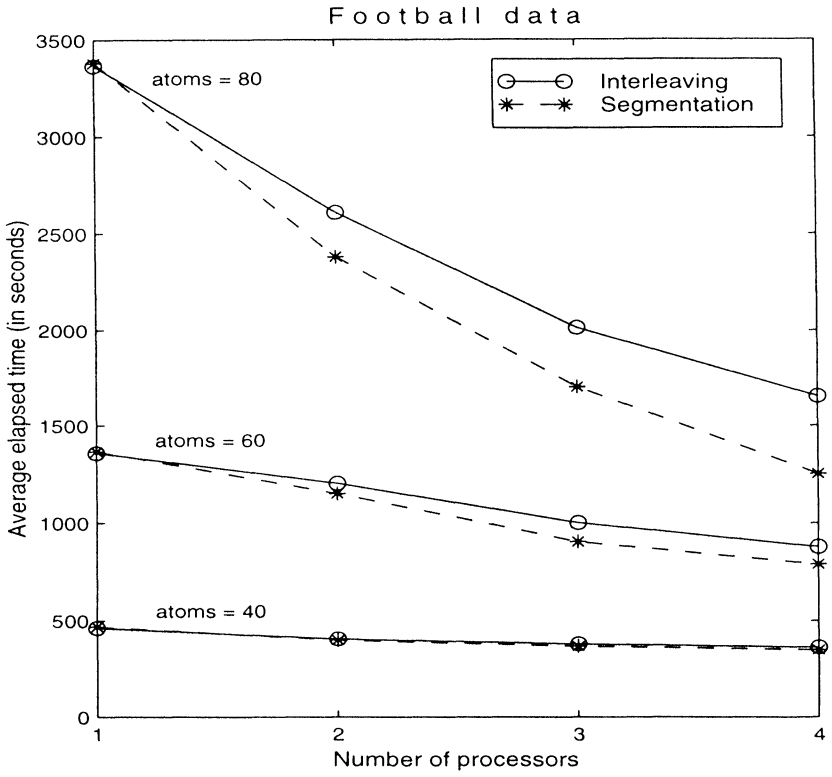

Figure 2: Comparison of the Interleaving and Segmentation methods for football data

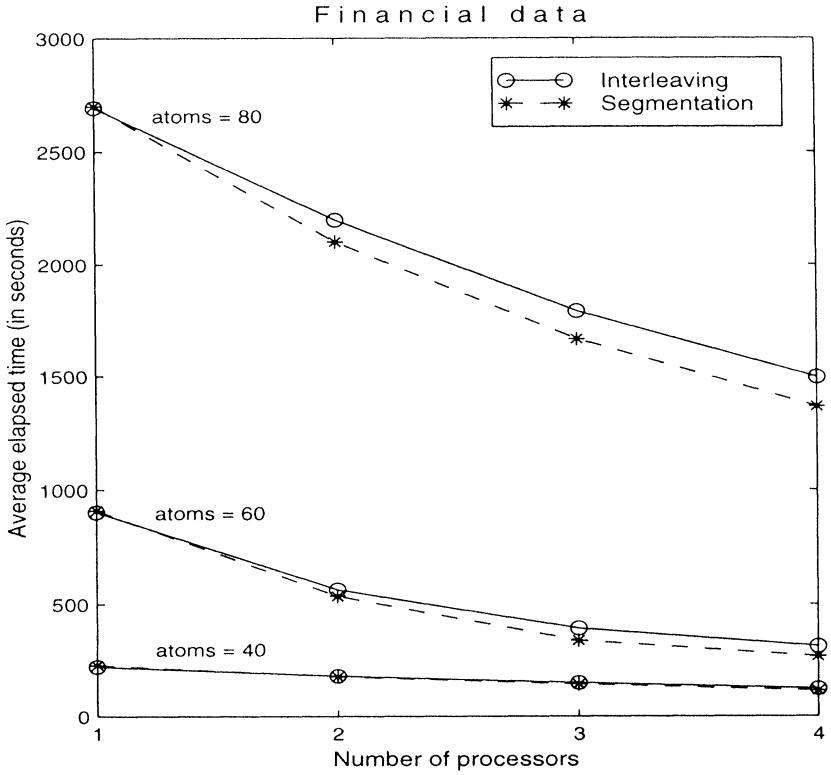

Figure 3: Comparison of the Interleaving and Segmentation method for financial data

Tables 1-3 and Figures 1-3 consistently show that the Segmentation method is faster than the Interleaving method, when more than one processor is used. In fact, the difference in speedup became more pronounced as the load increased due to a corresponding increase in the number of innovation atoms. The superiority of the Segmentation method lies in the effectiveness of the mapping from indices to vectors, employed by this method. This enables each processor to start directly from the first vector of the subspace assigned to it for global search. Furthermore, the fast Segmentation algorithm makes it possible to quickly enumerate the first and subsequent vectors belonging to each subspace, directly from vector indices. In contrast, the Interleaving method for each subspace requires the enumeration of all vectors, including those which do not belong to the subspace.

The Segmentation method was, however, a bit slower when only one processor was employed (i.e., when no parallelization took place). In this case, the two methods give rise to identical enumerations, but the Segmentation method has more overhead. Thus, it is advisable to employ the Interleaving method when parallelization is not available ($m = 1$), and the Segmentation method when it is ($m > 1$).

7. Conclusion

This paper has presented two partitioning methods, Interleaving and Segmentation, and the corresponding enumeration procedures for the quantized search space employed in the parallelization of ARM modeling. It compared these methods via three search examples drawn from various application domains. The examples confirm that parallelization is highly efficacious in speeding up the search. The low communication overhead among processors renders the speedup in the global search nearly linear in the number of processors. Finally, the Segmentation method has been demonstrated to be faster than the Interleaving method when parallelization is available, and that its speed advantage increases as the size of the space search grows.

Acknowledgement

This work was supported by DARPA grant AO-F316, Contract #N6601-97-1-8913.

References

[1] Davis, P.J., *Circulant Matrices*, John Wiley 1979.

[2] Hill, J.R., and Melamed, B., TEStool: A visual interactive environment for modeling autocorrelated time series, *Perf. Eval.* **24**:1&2 (1995), 3-22.

[3] Jagerman, D.L. and Melamed, B., The transition and autocorrelation structure of TES processes Part I: General theory, *Stoch. Models* **8**:2 (1992), 192-219.

[4] Jagerman, D.L. and Melamed, B., The transition and autocorrelation structure of TES processes Part II: Special cases, *Stoch. Models* **8**:3 (1992), 499-527.

[5] Jagerman, D.L. and Melamed, B., The spectral structure of TES processes, *Stoch. Models* **10**:3 (1994), 599-618.

[6] Jelenkovic, P.R. and Melamed, B., Algorithmic modeling of TES processes, *IEEE Trans. on Auto. Contr.* **40**:7 (1995), 1305-1312.

[7] Melamed, B., TES: A class of methods for generating autocorrelated uniform variates, *ORSA J. on Compu.* **3** (1991), 317-329.

[8] Melamed, B., An overview of TES processes and modeling methodology, In: *Perf. Eval. of Compu. and Commun. Sys.* (ed. by L. Donatiello and R. Nelson), *Lecture Notes in Compu. Sci.*, Springer Verlag (1993), 359-393.

[9] Melamed, B. and Hill, J.R., A survey of TES modeling applications, *SIMULATION* **64**:6 (1995), 353-370.

[10] Melamed, B., Ren, Q. and Sengupta, B., The QTES/PH/1 queue, *Perf. Eval.* **26** (1996), 1-20.

[11] Melamed, B., ARM processes and modeling methodology, *Stoch. Models* **15**:5 (1999), 903-929.