Multidimensional Systems and Signal Processing, 14, 295–318, 2003

A Krylov Subspace Method for Covariance Approximation and Simulation of Random Processes and Fields☆

MICHAEL K. SCHNEIDER
Laboratory for Information and Decision Systems, MIT

ALAN S. WILLSKY
Laboratory for Information and Decision Systems, MIT

Received July 23, 2001; Revised July 9, 2002; Accepted July 25, 2002

Abstract. This paper proposes a new iterative algorithm for simultaneously computing an approximation to the covariance matrix of a random vector and drawing a sample from that approximation. The algorithm is especially suited to cases for which the elements of the random vector are samples of a stochastic process or random field. The proposed algorithm has close connections to the conjugate gradient method for solving linear systems of equations. A comparison is made between our algorithm's structure and complexity and other methods for simulation and covariance matrix approximation, including those based on FFTs and Lanczos methods. The convergence of our iterative algorithm is analyzed both analytically and empirically, and a preconditioning technique for accelerating convergence is explored. The numerical examples include a fractional Brownian motion and a random field with the spherical covariance used in geostatistics.

Key Words: simulation, covariance matrix approximation, Krylov subspace, conjugate gradient, Lanczos, matrix square root

1. Introduction

This paper addresses the dual problems of simulation and covariance matrix approximation. Specifically, one is given the covariance Λ_x of an l-dimensional zero-mean random vector x. For all of the examples in this paper, the components of the random vector x are samples of a stochastic process or random field. Given the covariance Λ_x, one is interested in computing two quantities. The first is the generation of a zero-mean random vector x' whose covariance $\Lambda_{x'}$ matches the given covariance Λ_x, either exactly or approximately. The second quantity of interest is a low-rank approximation to the covariance. That is, one is interested in computing a reasonably small number of vectors, $a_1, \ldots, a_r$ such that

$$\sum_{i=1}^{r} a_i a_i^T \approx \Lambda_x. \tag{1}$$

☆ This material is based upon work supported by a National Science Foundation Graduate Research Fellowship, by ONR under Grant N00014-00-1-0089 and by AFOSR under Grant F49620-00-0362.

For performing both sets of computations, one would like an algorithm that is as computationally efficient as possible since l may be large, especially in the case of 2-D random fields.

Needs for efficient generation of samples of random processes and fields and low-rank approximations to their covariances can be found in many disciplines, including remote sensing applications (which served as the source of our motivation for this investigation). In particular, because of the huge size of the random fields often arising in such applications, compact covariance approximations are essential, as the computation and even the storage of full covariance matrices are prohibitive. Moreover, in space-time applications, in which uncertainty must be propagated over time, these problems are even more severe [23], [24], and the methods described in this paper offer two alternatives for applications such as these. The first is simply in the direct, low-rank approximation of the covariance of large random fields. An alternative, however, involves the generation of samples of large fields as a component of Monte Carlo methods for such space-time problems. Although directly approximating the covariance is one approach to space and space-time dimensionality problems, Monte Carlo methods are also used [5, and references therein]. In particular, ensemble Kalman filtering is a Monte Carlo method in which a number of samples of the random phenomenon of interest are generated [2], [8], [13], [17]. These samples are then individually propagated through the complex dynamical equations governing the phenomenon in order to estimate how uncertainty propagates over time. Such Monte Carlo techniques require an efficient simulation algorithm such as that described in this paper.

The algorithm developed in this paper for simulation and covariance matrix approximation is derived by exploiting a close connection between simulation and linear least-squares estimation. In particular, the proposed algorithm is based on a recently developed Krylov subspace algorithm for estimation [22]. That algorithm is a modification of the standard conjugate gradient algorithm for solving linear equations that allows for the simultaneous computation of estimates and error variances. The algorithm proposed in this paper holds a similar relationship with respect to conjugate gradient. Specifically, the proposed algorithm is a modification of the standard conjugate gradient algorithm that computes, instead of an estimate, a sample path. Moreover, the algorithm simultaneously computes a low-rank approximation to the given covariance matrix.

Other approaches to simulation and covariance matrix approximation include some variants on the Lanczos algorithm for computing eigendecompositions [10, Chapter 9]. In particular, Xu, Kailath, et al. investigated using a Lanczos algorithm for forming a low-rank approximation to a covariance matrix in the context of subspace tracking [26], [27], [28], [29]. The Lanczos algorithm has also been used to approximate matrix square roots, which can be used for simulation [3], [6], [25] (see Section 3.1). These algorithms, like ours, recursively tridiagonalize a matrix as an intermediate step. The Lanczos methods, then, perform eigendecompositions on the tridiagonal matrices in the process of computing final approximations. Instead, our method makes use of Cholesky factorizations. Thus, our algorithm is more recursive in nature and closer to conjugate gradient than the Lanczos methods. The advantages of our algorithm over Lanczos approaches include a natural stopping criteria, the ability to precondition the problem to accelerate convergence, and a slight efficiency gain. These advantages are expounded upon in subsequent sections.

The remainder of the paper is organized as follows. Section 2 presents our Krylov subspace method for simulation and covariance matrix approximation. The computational complexity of our method is compared against other algorithms in Section 3. The convergence of our method and preconditioning techniques for accelerating it are discussed in Section 4. Section 5 contains numerical examples.

2. A Krylov Subspace Method for Simulation and Covariance Matrix Approximation

Our approach to simulation and covariance matrix approximation is based on the connection between simulation and linear least-squares estimation. In particular, let $p_1^T x, \ldots, p_k^T x$ be unit variance linear functionals of x that whiten x, i.e. $\mathrm{Cov}(p_i^T x, p_j^T x) = \delta_{ij}$. The best linear estimate of x given $p_1^T x, \ldots, p_k^T x$ is then

$$\hat{x}_k(x) = \sum_{i=1}^{k} (b_i)(p_i^T x), \tag{2}$$

where $b_i = \Lambda_x p_i$. Since the $p_i^T x$ are white and unit variance, one can replace them with any other sequence of white unit variance random variables $w_1, w_2, \ldots, w_k$ to generate another random vector with the same second-order statistics, yielding, as in (2):

$$x_k' = \sum_{i=1}^{k} b_i w_i. \tag{3}$$

The covariance of x_k' is

$$\Lambda_{x_k'} = \sum_{i=1}^{k} b_i b_i^T. \tag{4}$$

The matrix $\Lambda_{x_k'}$ is an approximation to Λ_x, and the vector x_k' is a sample drawn from a distribution with covariance $\Lambda_{x_k'}$.

The linear functionals that we use form bases for a type of nested subspaces known as Krylov subspaces. A Krylov subspace of dimension k generated by a matrix Λ_x and vector s is given by

$$\mathrm{span}(s, \Lambda_x s, \ldots, \Lambda_x^{k-1} s) \tag{5}$$

and is denoted by $\mathcal{K}(\Lambda_x, s, k)$. The advantages of using Krylov subspaces are twofold. The first is computational. Since

$$\mathrm{span}(p_1, \ldots, p_k) = \mathrm{span}(s, \Lambda_x s, \ldots, \Lambda_x^{k-1} s), \tag{6}$$

determining p_{k+1} given $p_1, \ldots, p_k$ is not much more difficult than multiplying Λ_x by a vector (more details are provided subsequently). Moreover, computing the product can often be done efficiently using FFTs, wavelets [14], or some other fast transform. The second reason for using Krylov subspaces is that as long as the starting vector is not an eigenvector of Λ_x (or more generally that it does not reside in a proper invariant subspace), the sequence of Krylov subspaces capture more and more of the dominant modes of Λ_x with increasing k [19], [22]. In particular, the algorithm will find a good low-rank approximation to Λ_x as the dimension of the Krylov subspace increases.

To see how the structure of Krylov subspaces leads to computational advantages, consider the following approach to reducing a matrix Λ_x to upper Hessenberg form [4, Section 6.6.1]. Note that a matrix is upper Hessenberg if the only non-zero elements are on the subdiagonal and upper triangle, as in

$$\begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}. \tag{7}$$

One can use the Krylov subspace $\mathcal{K}(\Lambda_x, s, k)$ to perform the reduction, as follows. Let A be the matrix whose columns are products of s and Λ_x, i.e.

$$A = \begin{pmatrix} s & \Lambda_x s & \cdots & \Lambda_x^{l-1} s \end{pmatrix}. \tag{8}$$

If s does not reside in a proper invariant subspace of Λ_x, then A is invertible; so, one can write $c = -A^{-1}\Lambda_x^{l-1} s$. Then,

$$\Lambda_x A = A \begin{pmatrix} e_2 & e_3 & \cdots & e_l & -c \end{pmatrix} \tag{9}$$

where e_i is the ith canonical vector (zero accept for a 1 at the ith position). Hence,

$$\Lambda_x A = A \begin{pmatrix} 0 & 0 & \cdots & 0 & -c_1 \\ 1 & 0 & \cdots & 0 & -c_2 \\ 0 & 1 & \cdots & 0 & -c_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -c_l \end{pmatrix} \tag{10}$$

which is upper Hessenberg. Let H denote this matrix. When Λ_x is symmetric (as it is in this paper), it can be further manipulated to yield a tridiagonal matrix. Specifically, one can

perform the QR-decomposition on A to yield $A = QR$. Then,

$$A^{-1}\Lambda_x A = R^{-1}Q^T\Lambda_x QR = H; \tag{11}$$

so,

$$Q^T\Lambda_x Q = RHR^{-1}. \tag{12}$$

R is upper triangular, and H is upper Hessenberg; hence, RHR^{-1} is also upper Hessenberg. Furthermore, Λ_x being symmetric and positive definite implies

$$Q^T\Lambda_x Q = T \tag{13}$$

where T is tridiagonal and positive definite. The reduction in (13) is useful since T can be easily decomposed by Cholesky factorization. The resulting factors can be used to solve, among other things, the realization problem addressed in this paper (the details follow).

One approach to computing the tridiagonalization in (13) is to use a Lanczos iteration [10, Section 9.1]. The Lanczos iteration is derived by noting that (13) implies

$$\Lambda_x Q = QT. \tag{14}$$

Writing

$$T = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_l \\ & & & \beta_l & \alpha_l \end{pmatrix}, \tag{15}$$

one can rewrite (14), in terms of the columns q_k of Q, as the recursion

$$\alpha_k = q_k^T\Lambda_x q_k \tag{16}$$

$$h_k = \Lambda_x q_k - \alpha_k q_k - \beta_k q_{k-1} \tag{17}$$

$$\beta_{k+1} = \|h_k\| \tag{18}$$

$$q_{k+1} = \frac{h_k}{\beta_{k+1}} \tag{19}$$

which is initialized by setting q_1 equal to a starting vector s, $q_0 = 0$, and $\beta_1 = 0$. For $k < l$, the vectors $q_1, \ldots, q_k$ are orthonormal and tridiagonalize Λ_x. Moreover, they form a basis for the Krylov subspace $\mathcal{K}(\Lambda_x, s, k)$.

In this paper, we are interested in another basis for this Krylov subspace. Specifically, denote the $k \times k$ tridiagonal submatrix of T by T_k and its lower bi-diagonal Cholesky factor by L_k. Then, define the vectors p_i by

$$[p_1\ p_2 \cdots p_k] = [q_1\ q_2 \cdots q_k] L_k^{-T}. \tag{20}$$

Since L_k is lower bi-diagonal, the p_i can be efficiently computed from the q_i using a two-step recursion. These p_i are Λ_x-conjugate (i.e. they whiten x) and form a basis for $\mathcal{K}(\Lambda_x, s, k)$. Note also that the vectors p_i are the conjugate search directions generated by the conjugate gradient method [4, Section 6.6] for solving a linear system of the form

$$\Lambda_x u = v. \tag{21}$$

The method is iterative and, at each iteration k, generates a vector u_k^* that minimizes

$$\|\Lambda_x u_k - v\|_{\Lambda_x^{-1}} \tag{22}$$

over all u_k in $\mathcal{K}(\Lambda_x, s, k)$. In terms of the p_i, there is a simple recursion for u_k^*: $u_{k+1}^* = u_k^* + p_k p_k^T v$. The simple form of the recursion follows from the conjugacy of the p_i, the same reason why the realization approximation in (3) and (4) has a simple form.

For the realization algorithm proposed in this paper, the linear functionals used are precisely these p_i associated with the conjugate gradient algorithm. As already mentioned, they can be efficiently computed using the Lanczos iteration combined with a two-step recursion to solve (20). Note also that the vectors $b_i = \Lambda_x p_i$ in (3) and (4) can also be easily computed from $\Lambda_x q_i$, which are computed as part of the Lanczos iteration.

The Lanczos iteration and conjugate gradient methods are known to have numerical problems that can affect the degree of conjugacy among the p_i. These numerical issues can be addressed by using the Lanczos iteration combined with any of a variety of reorthogonalization schemes to ensure Λ_x-conjugacy among the p_i. The simplest of these is full orthogonalization [4, Section 7.4]. This just recomputes each h_k as

$$h_k \coloneqq h_k - [q_1 \cdots q_k]\ [q_1 \cdots q_k]^T h_k \tag{23}$$

between steps (17) and (18). One can also use more complicated reorthogonalization schemes that are more computationally efficient such as selective orthogonalization [16]. A discussion of the details of selective orthogonalization relevant for simulation can be found in [21, Appendix B]. We have found that the quality of simulation is not significantly affected by the type of orthogonalization used.

The presentation of the simulation and covariance matrix approximation algorithm, thus far, has included neither a choice of starting vector s nor a stopping criterion. For the former, we choose s as a sample of a zero-mean Gaussian random vector with identity covariance (thereby guaranteeing, with probability 1, that this initial choice will not be confined to a proper invariant subspace of Λ_x). Although one might expect performance to

be significantly affected by the particular random sample used for s (for example, one could think of s lying almost in a proper invariant subspace), the result presented later in Theorem 4.1 indicates that the convergence rate is independent of the particular random sample used for s. Moreover, this fact has been observed in practice. For a stopping criterion, we consider those used for the Krylov subspace estimation algorithm referenced in the introduction. In particular, the criterion in [22, Section 2.1] can be modified for use with the simulation and covariance matrix approximation algorithm. Specifically, consider

$$\frac{1}{l}\sum_{i=1}^{l}(\Lambda_{r,k})_{ii} \tag{24}$$

where

$$\Lambda_{r,k} = \Lambda_x - \Lambda_{x'_k}. \tag{25}$$

This measures the total difference in the variances between the given random vector x and the approximate simulation x'_k. It is a useful measure of the quality of approximation of x'_k and is easy to update at each iteration. Hence, it can be used as a basis for a stopping criterion of our Krylov subspace algorithm. The complete algorithm for simulation and covariance matrix approximation is summarized, as follows.

Algorithm 2.1 *Krylov Subspace Method for Simulation and Covariance Matrix Approximation.*

1. *Initialize* $x'_0 = 0$, $(\Lambda_{r,k})_{ii} = (\Lambda_x)_{ii}$ *for* $i = 1, \ldots, l$.
2. *Generate a zero mean Gaussian random vector* s *with identity covariance to initialize the Krylov subspace.*
3. *Perform the following operations for each step* k *until*

$$\frac{1}{l}\sum_{i=1}^{l}(\Lambda_{r,k})_{ii}$$

falls below a threshold χ*:*

(a) *Compute the conjugate search direction* p_k *and filtered backprojection* $b_k = \Lambda_x p_k$ *using a reorthogonalized Lanczos iteration, (16)–(19).*

(b) *Generate an independent random number* w_k *and update the simulation*

$$x'_k = x'_{k-1} + b_k w_k \tag{26}$$

and the errors in approximating the variances

$$(\Lambda_{r,k})_{ii} = (\Lambda_{r,k-1})_{ii} - ((b_k)_i)^2 \quad \text{for } i = 1, \ldots, l. \tag{27}$$

3. Comparison of Simulation and Covariance Approximation Algorithms

This section compares our Krylov subspace simulation and covariance approximation algorithm to three other approaches. The focus is on a comparison with the Lanczos algorithms for matrix and matrix square root approximation [25]; however, approaches using Karhunen-Loève bases and FFT methods are also examined. The comparison centers on computational complexity issues.

3.1. *Krylov Subspace Simulation and Covariance Approximation vs. Lanczos Matrix Approximation*

Our Krylov subspace algorithm and the Lanczos algorithm for matrix approximation yield results that would be almost the same in exact arithmetic. Specifically, the covariance matrix approximation generated at step k by our Krylov subspace algorithm is

$$\Lambda_x Q_k T_k^{-1} Q_k^T \Lambda_x \tag{28}$$

where

$$Q_k = (q_1 \cdots q_k) \tag{29}$$

and q_i and T_k are as in Section 2. The Lanczos algorithm for matrix approximation, on the other hand, generates the following approximation at step k:

$$Q_k T_k Q_k^T. \tag{30}$$

To see that these approximations are very similar, consider running the Lanczos tridiagonalization to completion (l steps). For any k, then, one can write

$$\Lambda_x = \begin{pmatrix} Q_k & Q_k^\perp \end{pmatrix} \begin{pmatrix} T_k & E_k^T \\ E_k & T_k^\perp \end{pmatrix} \begin{pmatrix} Q_k & Q_k^\perp \end{pmatrix}^T \tag{31}$$

where the columns of $Q_k{}^\perp$ are the Lanczos vectors generated after step k, $T_k{}^\perp$ is tridiagonal, and E_k is of the form

$$E_k = \begin{pmatrix} 0 & \cdots & 0 & * \\ 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{pmatrix}. \tag{32}$$

Thus,

$$\Lambda_x Q_k T_k^{-1} Q_k^T \Lambda_x = Q_k T_k Q_k^T \; + \; Q_k^\perp E_k Q_k^T \; + \; Q_k E_k^T \left(Q_k^\perp\right)^T + Q_k^\perp E_k T_k^{-1} E_k^T \left(Q_k^\perp\right)^T. \tag{33}$$

So, the difference between our Krylov subspace algorithm approximation in (28) and the Lanczos iteration approximation in (30) is at most rank 3 since E_k is rank one.

The primary advantage of our Krylov subspace algorithm is that it allows for one to recursively update the synthesis and approximation error at each step. In contrast, Lanczos algorithms for matrix or matrix square root approximation typically perform an eigendecomposition of the T_k to form the syntheses and estimate approximation [6], [25], [27]. The eigenvalues of T_k are used to estimate the approximation error, and the square root of Λ_x is approximated by performing a full eigendecomposition of T_k and taking square roots of the eigenvalues.

The recursive structure of our algorithm also results in a modest computational gain. The amount of reduction can be quantified, as follows, by counting the number of floating point operations (flops). Only terms cubic in either k and l are considered. Suppose each Λ_x-vector product requires a number of flops that grows linearly with l. Suppose further that the algorithm is run for k iterations. Then, the computational load of the Krylov subspace simulation and covariance approximation algorithm is dominated by the $2lk^2$ flops required to perform reorthogonalization. The standard Lanczos methods must also perform reorthogonalization and compute eigendecompositions of T_k. Computing eigenvalues at step i to check the approximation quality requires about $30i^2$ flops if symmetric QR is used [10, pp. 420-1]. By step k, this has required $15k^3$ flops. Computing a full eigendecomposition at step k to generate a matrix square root requires $6k^3$ flops. The totals are provided in Table 1. Thus, our Krylov subspace algorithm achieves a modest computational gain of $21k^3$ over standard Lanczos algorithms.

3.2. Karhunen Loève Bases

In terms of mean-squared error, the optimal approach to generating a low-rank approximation to a covariance matrix Λ_x and an associated simulation is to perform a Karhunen-Loève decomposition of Λ_x. However, any implementation of a simulation and covariance matrix approximation algorithm using Karhunen-Loève bases requires a numerical routine for the approximate computation of partial eigendecompositions. One of the most popular iterative routines is also based on the Lanczos algorithm [10, Chapter 9], described previously in Section 2. At step k the Lanczos algorithm will have computed a basis for the Krylov subspace $\mathcal{K}(\Lambda_x, s, k)$, from which one computes approximate eigenvectors by selecting appropriate vectors. Note that while exact Karhunen-Loeve approximation is mean-square optimal, the approximate procedure just described is always inferior in mean-square error (for the same computational effort) to the method described in this paper. This follows from the fact that at iteration k, our approach will project the covariance Λ_x onto the *entire* Krylov subspace $\mathcal{K}(\Lambda_x, s, k)$ whereas the approximate Karhunen-Loève approach projects Λ_x onto a *proper* subspace of $\mathcal{K}(\Lambda_x, s, k)$.

3.3. FFT Methods

One may be able to utilize FFT's for simulation and covariance approximation if the covariance is that of equally spaced samples of a stationary stochastic process or random

Table 1. Flops Required for Simulation and Covariance Approximation.

Method	Flops
Krylov Subspace Simulation and Covariance Approximation	$2lk^2$
Standard Lanczos	$2lk^2+21k^3$

field. The covariance matrix for such a process is Toeplitz [10, Section 4.7]. A Toeplitz matrix is a matrix in which each row (column) is a shifted version of the previous row (column). Since covariance matrices are also symmetric, a Toeplitz covariance matrix will have the form

$$\Lambda_x = \begin{pmatrix} K[0] & K[1] & K[2] & \cdots & K[l-1] \\ K[1] & K[0] & K[1] & \cdots & K[l-2] \\ \vdots & \vdots & \vdots & & \vdots \\ K[l-1] & K[l-2] & K[l-3] & \cdots & K[0] \end{pmatrix} \tag{34}$$

where K is the covariance function. FFT's can be used to simulate such a stochastic process when one can embed the covariance matrix in a larger, circulant, positive-definite matrix. A circulant matrix is a Toeplitz matrix in which each row (column) is a circular shift of the previous row (column) [10, Section 4.7.7]. Thus, a circulant covariance matrix has the following form (when l is even)

$$\Lambda_x = \begin{pmatrix} K[0] & K[1] & \cdots & K\left[\frac{l}{2}-1\right] & K\left[\frac{l}{2}\right] & K\left[\frac{l}{2}-1\right] & \cdots & K[1] \\ K[1] & K[0] & \cdots & K\left[\frac{l}{2}-2\right] & K\left[\frac{l}{2}-1\right] & K\left[\frac{l}{2}\right] & \cdots & K[2] \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ K[1] & K[2] & \cdots & K\left[\frac{l}{2}-1\right] & K\left[\frac{l}{2}\right] & K\left[\frac{l}{2}-1\right] & \cdots & K[0] \end{pmatrix}. \tag{35}$$

The details of the FFT computation are given in Appendix A along with a discussion of how FFT's can be used for simulating certain nonstationary processes that have stationary increments, such as fractional Brownian motion.

Unlike approaches for simulation and covariance approximation using Karhunen Loève bases, FFT methods may be computationally competitive with our Krylov subspace algorithm for certain covariance matrices Λ_x. Specifically, when FFT's can be and are used for simulation, the computation is dominated by the FFT. This can be implemented by an algorithm that is $O(l \log l)$ where l is the dimension of the random vector. Whether this is competitive or not with our Krylov subspace algorithm depends on the problem.

Specifically, consider two different asymptotic scenarios. In each case, suppose the given covariance is that of a set of equally-spaced samples of a random field defined over a compact subset of $\mathbb{R}^d$. In the first case, let the size of the subset grow but keep the sampling density fixed. Then, as l increases, consider the behavior of our Krylov subspace

algorithm. The number of linear functionals of the process, k, needed to meet a desired level of accuracy should grow linearly with l. This is a consequence of the need to use more linear functionals of the process to capture its behavior over the larger region. Since our Krylov subspace algorithm has complexity $O(lk^2)$, and the FFT method, $O(l \log l)$, the FFT method will become more competitive as the region size grows.

Now consider the case where the region size is fixed, but the sampling density increases. Then, as l increases, the number k of linear functionals will remain constant. Instead, one needs different linear functionals that capture the behavior on the refined grid. Our Krylov subspace algorithm will compute the appropriate linear functionals for the grid size. In this case, our Krylov subspace algorithm will be less computationally intensive than the FFT method for large l. The problem sizes at which one method becomes less intensive than the other depend on the specific problem and implementations.

4. Convergence and Preconditionings

Whether the Krylov subspace algorithm for simulation and covariance matrix approximation is applicable for a particular problem depends on how the algorithm converges. In this section, the convergence rate is bounded, and preconditioning strategies for accelerating that rate are discussed.

4.1. Bound on the Convergence Rate

The convergence analysis focuses on two quantities. The first is the initial vector, s. Recall that the initial vector is chosen randomly, according to a Gaussian distribution with zero mean and a covariance proportional to the identity. Note that the initial vector affects the convergence of the method by implicitly determining the linear combination of eigenvectors of Λ_x used to start the generation of the Krylov subspaces. The question addressed by the subsequent bound is whether the convergence rate depends on the initial vector when that vector is chosen randomly. The second quantity on which the convergence analysis focuses is the relative separation of the eigenvalues $\lambda_{x,i}$ of Λ_x. The relative separation is measured by the minimum of $(\lambda_{x,i} - \lambda_{x,i+1})/(\lambda_{x,i+1} - \lambda_{x,l})$ over i. One would expect that the Krylov subspaces will do a better job of capturing the dominant eigenspaces when the relative separation is larger. In particular, we assume that there exists a constant $\lambda_{\text{sep}} > 0$ such that

$$\frac{\lambda_{x,i} - \lambda_{x,i+1}}{\lambda_{x,i+1} - \lambda_{x,l}} \geq \lambda_{\text{sep}} > 0. \tag{36}$$

Given this assumption, one can state the following bound on the total difference between the variances of the given random vector and those of the approximation generated by our Krylov subspace algorithm. The bound is stated without proof since it is a special case of Theorem 3.1 in [22].

THEOREM 4.1: *If there exists a constant λ_{sep} such that (36) holds,*

$$\sum_{j=1}^{l} (\Lambda_x - \Lambda_{x,k})_{jj} \leq \frac{\|s\|^2 \eta \|\Lambda_x\|}{\left(1 - \frac{1}{\gamma^2}\right)\left(1 - \frac{1}{\sqrt{\gamma}}\right)} \gamma^{-k/2} + \sum_{i=k}^{l-1} (i - k + 4) \lambda_{x, \lfloor \frac{i}{4} \rfloor} \tag{37}$$

where γ is given by

$$\gamma \triangleq 1 + 2\left(\lambda_{sep} + \sqrt{\lambda_{sep} + \lambda_{sep}^2}\right) \tag{38}$$

and η is a random variable whose statistics depend only on λ_{sep}.

Note that the bound in Theorem 4.1 consists of two terms. The second term primarily relates to how good an approximation to Λ_x one obtains by retaining only the first k eigenvectors. The first term primarily relates to how well the first k eigenvectors are approximated by the k-dimensional Krylov subspace generated by Λ_x and s. Note that the rate of decay of this term, γ, is independent of the starting vector, s. However, the rate is highly dependent on the relative separation of the eigenvalues as measured by λ_{sep}. The more separated the eigenvalues are, the faster this term in the bound decays.

4.2. Preconditioning

The bound in Theorem 4.1 suggests preconditioning as a method for accelerating convergence. The idea behind preconditioning for simulation and covariance matrix approximation is to pre- and post-multiply the matrix generating the Krylov subspace by B and B^T, respectively, so as to increase the separation of the eigenvalues and, in turn, γ. Essentially, this alters our algorithm in that at step k, instead of computing the best linear estimate of x based on $p_1^T x, \ldots, p_k^T x$ as in (2)–(4), the preconditioned version of our algorithm computes the best linear estimate of x given $p_1^T(Bx), \ldots, p_k^T(Bx)$. The preconditioned form of the algorithm is not significantly more involved computationally than the standard version. Moreover, the preconditioning can be implemented in a manner that only requires one to know, and have a routine for multiplying vectors by, $B^T B$, instead of both B and B^T. The algorithmic details of preconditioning for the Krylov subspace method for simulation and covariance matrix approximation are very similar to those for Krylov subspace estimation [22, Section 4.1] and, thus, are not explicitly given here.

Constructing good preconditioning matrices, $B^T B$, for covariance approximation and approximate simulation is an ongoing area of research, and a thorough discussion is beyond the scope of this paper. One could potentially adapt existing methods for preconditioning eigendecomposition computations [20, Chapter 8] or utilize techniques from spectral analysis [7], [18]. However, one may also be able to use simple strategies such as preconditioners of the form

$$UDU^T \tag{39}$$

where U is an approximation to the eigenvectors of Λ_x, and D is a diagonal matrix whose diagonal elements $(D)_{ii}$ are given by $\alpha\mu^i + 1$ for some constants $0 < \mu < 1$ and $\alpha \gg 1$. The motivation for a preconditioner of this form is as follows. If U were the exact eigenvectors of Λ_x, the eigenvalues of the product of the preconditioner and Λ_x would be given by $(\alpha\mu^i + 1)\lambda_{x,i}$. The geometric decay induced by μ^i for small i would increase the separation of those eigenvalues of the preconditioned system. Since the eigenvectors of Λ_x are not known exactly, 1 is added to the geometric decay to ensure that no eigenvalue is attenuated too much. An example of such a preconditioner is given in Section 5.3.

5. Numerical Examples

The performance of our Krylov subspace algorithm is illustrated in this section with three examples. For each example, three sets of results are provided. First, a high quality sample path generated by the Krylov subspace algorithm is plotted along with a sample path whose statistics exactly match the given covariance. Second, the difference between the true variances and those of the Krylov subspace algorithm's approximation are plotted. Since the difference between the exact and approximate covariances is always positive semi-definite for our method, the variances provide a good measure of the quality of the approximation. Lastly, a plot is presented of the fraction of total mean-squared error reduction

$$\frac{\mathrm{Tr}(\Lambda_{r,k})}{\mathrm{Tr}(\Lambda_x)}, \tag{40}$$

versus approximation rank k, where $\Lambda_{r,k}$ is given by (25). For comparison, the error reduction obtained by the exact Karhunen-Loève approach is also plotted. All results were generated using MATLAB on a Sun workstation with a floating point precision of approximately 2×10^{-16}.

5.1. *Fractional Brownian Motion*

The first example consists of simulating and approximating the covariance matrix of 1024 samples of a fractional Brownian motion (fBm) with a Hurst parameter $H = 3/4$ [15]. The covariance of fBm is given by

$$K_{xx}(s,t) = \frac{1}{2}\left(|t|^{2H} + |s|^{2H} - |t-s|^{2H}\right). \tag{41}$$

Note that fBms have stationary increments, and one can synthesize this fBm exactly with 2048-point FFTs and also generate good finite-rank approximations to the covariance matrix, as discussed in Appendix A. As a result, there is not necessarily a need to use a Krylov subspace method to simulate and approximate the covariance of this fBm.

However, the problem of simulating and approximating the covariance of a fBm provides a good example of how our Krylov subspace algorithm could be used to simulate a non-stationary process. The example also illustrates the algorithm's power in obtaining near-optimal low-rank covariance approximations.

Figure 1 presents the results. Part (a) of the figure shows sample paths generated using the exact FFT method and using 50 iterations of the Krylov subspace method. Note that the sample path generated using the Krylov subspace method, which represents a rank-50 approximation to a 1024-sample process, looks qualitatively quite similar to the sample path using exact fBm statistics. That this is the case is actually not surprising if we examine part (b) of this figure which shows that the differences between the exact and approximate variances are small. There are two other interesting features of the variance differences. One is that, they are uniformly small. The other is that they consist mostly of high frequency oscillations indicating that higher frequency modes (and, hence, the least important ones) are the ones left out of the approximation. Again, this is expected since, as indicated in part (c), the Krylov subspace approach is picking linear functionals that are almost as good as picking the optimal ones, namely the eigenvectors. Also, note that the Krylov subspace approach does much better than the FFT-based approach outlined in Appendix A for approximating the covariance matrix at any specified rank.[1]

5.2. *Windowed Cosine Covariance*

For the second example, the given random vector consists of 1024 samples in the unit interval of a stationary process whose covariance function, $K_{xx}(\tau)$, is a Gaussian-windowed cosine:

$$K_{xx}(\tau) = e^{-\frac{\tau^2}{2}}\cos(2\pi\tau). \tag{42}$$

This process is interesting because one can not exactly simulate it efficiently using FFTs despite the fact that the process is stationary. As discussed in Appendix A, efficient simulation using FFTs can only be performed when the 2048×2048 circulant embedding matrix for the given 1024×1024 covariance matrix is positive semi-definite. However, the circulant embedding matrix for this example has a substantial number of negative eigenvalues, as indicated in Figure 2.

Results using our Krylov subspace algorithm are plotted in Figure 3. Sample paths are plotted in part (a). The exact sample path is generated by forming a square root of the covariance matrix. The approximate synthesis is generated using only 14 iterations of our Krylov subspace algorithm, which yields a rank-14 approximation to this 1024-sample process. At this rank, both syntheses have similar structure. This is expected because the differences between the approximate and exact sample path variances are small, as indicated in part (b) of the figure. Not many iterations are needed because the eigenvalues of the covariance matrix for this process decay rapidly, and the Krylov subspace approach is near optimal, as indicated in part (c) of the figure.

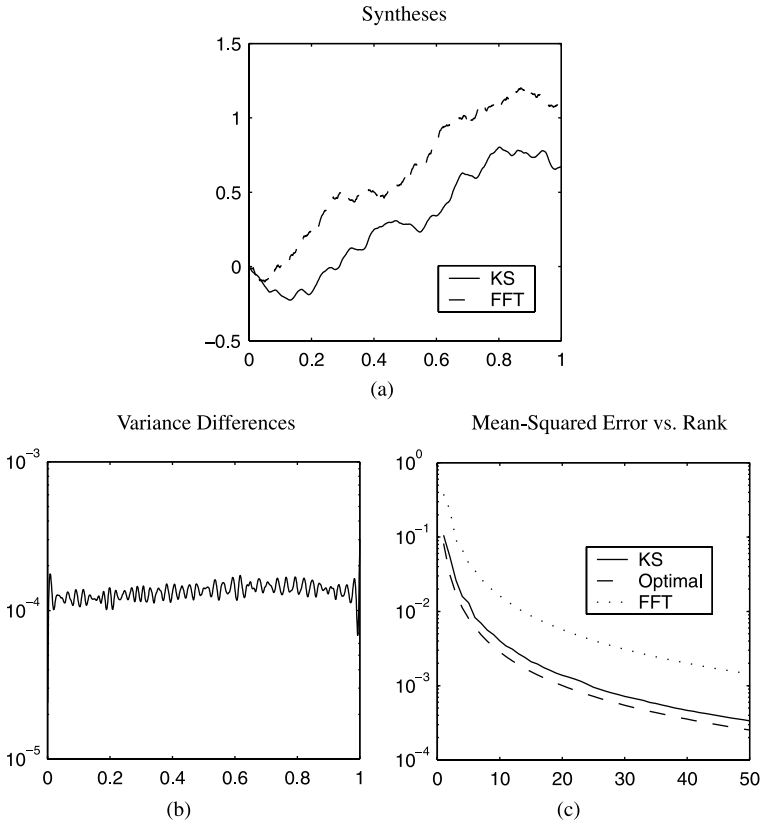

Figure 1. These results are for simulating 1024 samples of a fBm with Hurst parameter $H = 3/4$. Part (a) shows sample paths generated with FFTs (an exact method) and with 50 iterations of our Krylov subspace (KS) algorithm. The difference between the variances are plotted in part (b). Note that the true fBm variances are given by $t^{3/2}$ as t ranges from zero to one. The fraction of error reduction obtained by each method as a function of approximation rank is plotted in part (c). The optimal (KL) results are plotted for comparison.

5.3. Two-Dimensional Spherical Covariance

Lastly, consider simulating and approximating the covariance matrix of a two-dimensional isotropic random field with radial covariance

$$K_{xx}(\tau) = \begin{cases} 1 - \frac{3}{2}|\tau| + \frac{1}{2}|\tau|^3 & 0 \le |\tau| \le 1 \\ \\ 0 & \text{otherwise.} \end{cases} \tag{43}$$

This covariance function is known as the spherical covariance function in the geostatistical community [5], [12]. Partly due to its potential practical application, and partly, its rich

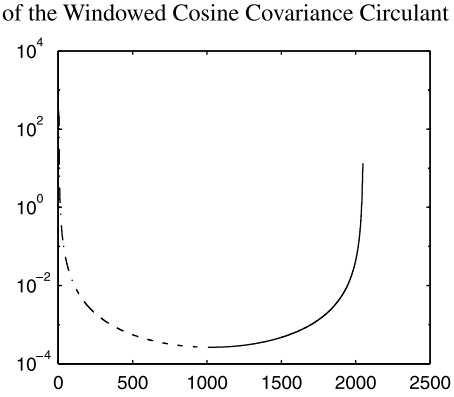

Figure 2. The plot shows the eigenvalues of the 2048-point circulant embedding for the covariance matrix of 1024 samples of a process with a windowed cosine covariance given by (42). There are both positive and negative eigenvalues. The curve plots the magnitudes. Those plotted with a solid line are negative; those, with a dashed line, positive.

structure, the spherical covariance has been used by several researchers to characterize algorithms for simulation and covariance matrix approximation [5], [9], [11]. One can consider using two-dimensional FFTs to simulate samples of a field with spherical covariance, as discussed in Appendix A. As in the previous example, however, the associated two-dimensional circulant embedding matrix is not necessarily positive semi-definite. This will happen, in particular, when the samples are taken from a square grid that does not include the unit square. In order to demonstrate the performance of our Krylov subspace algorithm on a two-dimensional problem for which FFT-based methods do not apply, this section considers a problem on a 33×33 grid covering $[0, 32/45] \times [0, 32/45]$. The 64×64 two-dimensional circulant embedding of the covariance matrix of these samples has several negative eigenvalues as illustrated in Figure 4.

A preconditioned Krylov subspace algorithm is applied to this simulation and covariance approximation problem. Recall from Section 4.2 that one strategy for preconditioning is to use an approximation of the eigenvectors to form a matrix that separates the eigenvalues of Λ_x. Since Λ_x is stationary, the elements of the Fourier basis are approximate eigenvectors. Thus, one can consider a preconditioner that has the following form. First, one zero-pads the image to create a 64×64 image. Then, one performs a two-dimensional DFT, multiplies the resulting spectrum by the transfer function G_p, and then performs an inverse DFT. Finally, one selects the portion of the result corresponding to the original 33×33 image. The transfer function G_p is chosen to be

$$G_p(f) = 50|f|^2(0.4)^{|f|} + 1 \tag{44}$$

where $f \in \{-31, \ldots, 32\} \times \{-31, \ldots, 32\}$ is a two-dimensional frequency vector. The geometric decay in the first term of (44), $(0.4)^{|f|}$, will spread out the eigenvalues. The factor $|f|^2$ prevents low-frequencies from being shaped by the geometric decay. This is

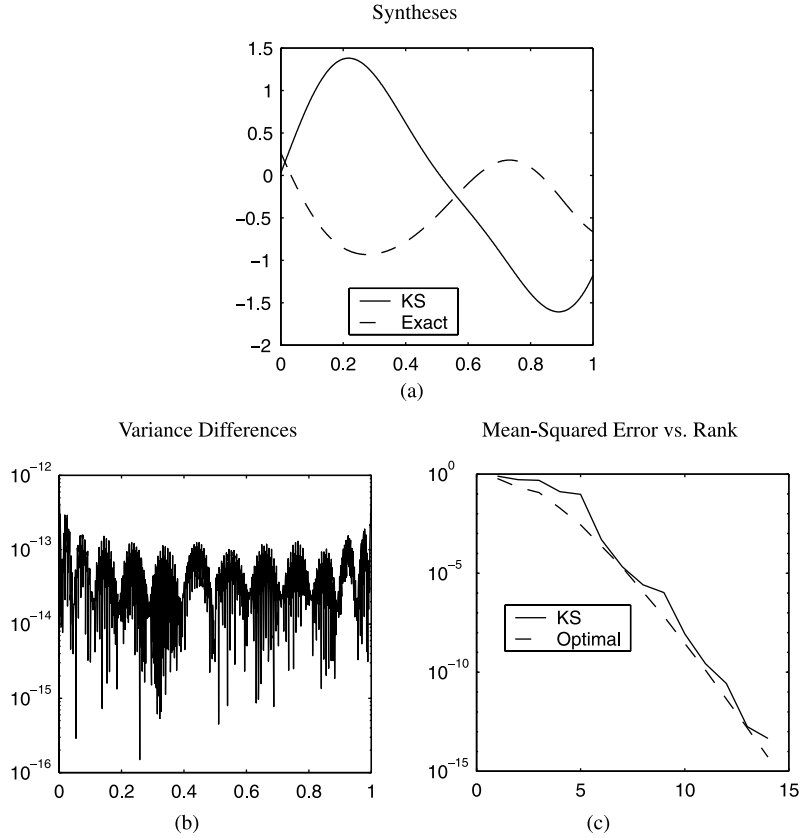

Figure 3. These results are for simulating 1024 samples of a process with a windowed cosine covariance given by (42). Part (a) shows sample paths generated with a matrix square root computation (an exact method) and with 14 iterations of our Krylov subspace (KS) algorithm. The difference between the variances are plotted in part (b). The true variance of the stationary process is 1. The fraction of error reduction obtained by each method as a function of approximation rank is plotted in part (c). The optimal (KL) results are plotted for comparison.

done because the low-frequency modes are significantly affected by the image boundaries, and, thus, the corresponding eigenvalues cannot be effectively separated with this type of Fourier preconditioner. The second term of (44) introduces a shift so that the preconditioner doesn't have a null space.

Exact and approximate syntheses are pictured in parts (a) and (b) of Figure 5. The exact synthesis is generated by computing a matrix square-root as for the windowed-cosine covariance example. The approximate synthesis in part (a) is generated with 53 iterations of the preconditioned Krylov subspace algorithm. Note that the rank-53 approximate simulation of this 33×33 random field is much smoother than the exact one. For moderate quality simulations, the Krylov subspace method tends to compute a very smooth one, as long as most of the power in the field is at low frequencies. This may or

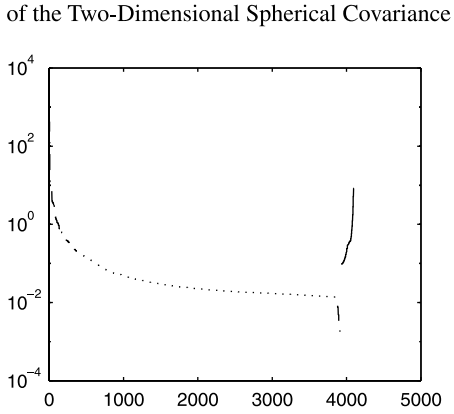

Figure 4. The plot shows the eigenvalues of the circulant embedding for the covariance matrix of samples on a 33×33 grid of a process with a two-dimensional spherical covariance given by (43). There are both positive and negative eigenvalues. The curve plots the magnitudes. Those plotted with a solid line are negative; those, with a dashed line, positive.

may not be desirable for certain applications. However, for many applications, the ability of the Krylov subspace algorithm to pick a close to optimal mean-squared-error and, hence, smooth simulation is desirable. Note that the differences in variances, plotted in part (c) are, for the most part, uniformly low. They ripple in the interior, indicating high-frequency terms have not been approximated as accurately in the simulation. The variance differences are also high at the edges and corners, which are apparently difficult to simulate with finite-rank approximations.

Part (d) of Figure 5 shows how the preconditioned and un-preconditioned Krylov subspace algorithms compare to the optimal (KL) approach to low-rank covariance matrix approximation. Note that the un-preconditioned algorithm is close to optimal but not as close as for the other examples. At very low ranks, the preconditioner does not provide much improvement. This is because the most important modes to capture at these ranks are low-frequency ones that are influenced by edge effects, which are not taken into account in the Fourier-based preconditioner. At middle and higher frequencies, the preconditioned algorithm performs better than the un-preconditioned one. The conclusion is that one can improve performance of the Krylov subspace algorithm by using a simple preconditioner, and there is the potential to improve performance further with more sophisticated preconditioners.

6. Conclusions

This paper has presented a Krylov subspace method for simulation and covariance matrix approximation and compared it to three other methods, namely, one based on Karhunen-Loéve expansions, another based on FFTs, and a third involving Lanczos algorithms for

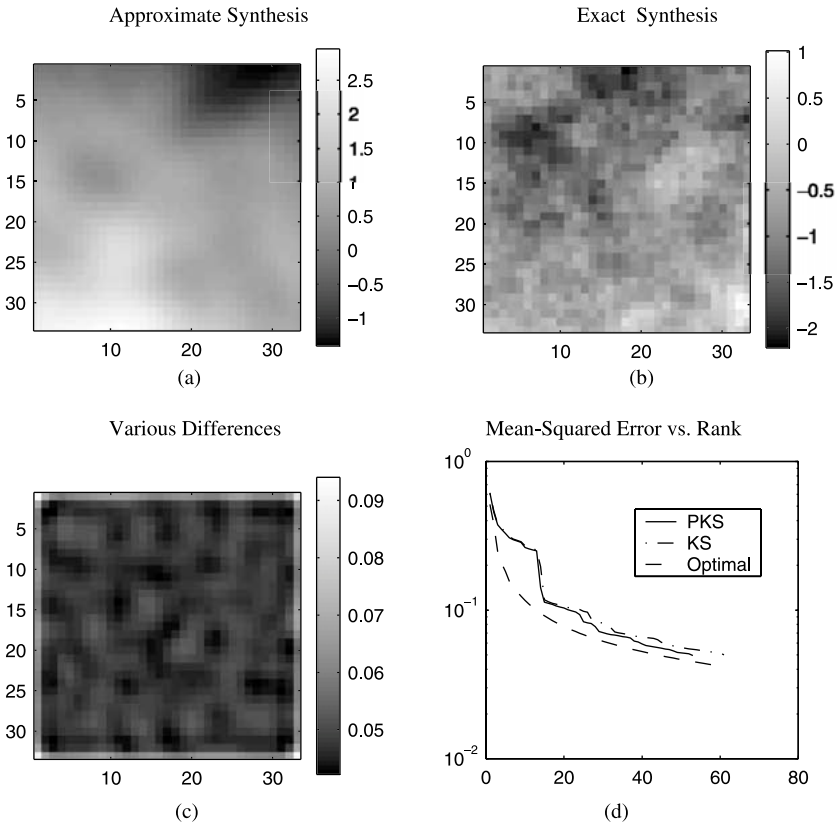

Figure 5. These results are for simulating samples on a 33 × 33 grid of a process with a two-dimensional spherical covariance given by (43). Part (a) shows a sample field generated with 14 iterations of a preconditioned Krylov subspace (PKS) algorithm, and part (b) shows a sample field generated with a matrix square root computation (an exact method). The difference between the variances are imaged in part (c). The true variance of the stationary process is 1. The fractions of error reduction obtained by both the preconditioned (PKS) and unpreconditioned (KS) Krylov subspace algorithm are plotted in part (d) as a function of rank. The optimal (KL) results are plotted for comparison.

matrix and matrix square root approximation. Our proposed Krylov subspace algorithm can efficiently synthesize a process and compute low-rank approximations to covariance matrices provided that covariance matrix-vector products can be efficiently implemented. Compared to methods using Karhunen-Loéve expansions, our method is always superior for the same computational effort. FFT-based methods may be superior in some cases but are either inapplicable or ineffective in others, including the examples shown in this paper. The comparison between our algorithm and Lanczos methods for matrix and matrix square root approximation includes no experimental results but has shown analytically that our algorithm provides certain advantages. In particular, each step of our recursive algorithm

produces an approximation to the covariance of interest, the difference between the variances of the approximation and the exact covariance, and a sample drawn from the approximation. Each step iteratively refines the sample and the approximate covariance by adding new components determined by the Krylov subspace iteration, while the computation of differences in the variances provides the basis for a simple stopping criterion. The Krylov subspace method for simulation and covariance approximation can also be preconditioned to accelerate convergence; although, developing a systematic methodology for creating preconditioners is left for future research.

Appendix A
A FFT Method for Simulation and Covariance Approximation

Using FFTs for simulation of stationary stochastic processes is similar to using FFTs for Λ_x-vector multiplication, i.e. convolution. First, one embeds Λ_x in a circulant embedding matrix C that need be no larger than $2(l-1) \times 2(l-1)$. Specifically, let

$$\Lambda_x = \begin{pmatrix} K[0] & K[1] & K[2] & \cdots & K[l-1] \\ K[1] & K[0] & K[1] & \cdots & K[l-2] \\ \vdots & \vdots & \vdots & & \vdots \\ K[l-1] & K[l-2] & K[l-3] & \cdots & K[0] \end{pmatrix} \tag{45}$$

where K is the covariance function. Then, the $2(l-1) \times 2(l-1)$ circulant embedding matrix is given by

$$C = \begin{pmatrix} K[0] & K[1] & \cdots & K[l-2] & K[l-1] & K[l-2] & \cdots & K[1] \\ K[1] & K[0] & \cdots & K[l-3] & K[l-2] & K[l-1] & \cdots & K[2] \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ K[1] & K[2] & \cdots & K[l-2] & K[l-1] & K[l-2] & \cdots & K[0] \end{pmatrix}. \tag{46}$$

Since C is circulant, it is diagonalized by the DFT matrix F. That is, one can write

$$C = F^*GF \tag{47}$$

where G is a diagonal matrix, whose elements are given by computing the FFT of the first row of C. Interpreting the product Cz as the circular convolution of z with the "impulse response" corresponding to the first row of C, one can think of the diagonal elements of G as defining the corresponding frequency response or transfer function. From (46) and (47), one then has that

$$\Lambda_x = SF^*GFS^T \tag{48}$$

where

$$S = (I\ 0) \tag{49}$$

selects the first l components of a vector. The factorization in (48) amounts to zero-padding (multiplication by S^T), performing a $2(l-1)$ FFT (multiplication by F), multiplying by the transfer function G, performing an inverse FFT (multiplication by F^*), and finally selecting the first l components of the result (multiplication by S).

It would appear that one could synthesize a process by simply multiplying $SF^*\sqrt{G}F$ by a random vector w that has identity covariance because the resulting product has covariance

$$\mathrm{Cov}(SF^*\sqrt{G}Fw) = SF^*\sqrt{G}FF^*\sqrt{G}FS = \Lambda_x, \tag{50}$$

as desired. However, the covariance Λ_x may not admit a positive semi-definite embedding, *i.e.* the circulant matrix C, and hence G, may not be positive semi-definite even if Λ_x is, in which case the square root of G doesn't exist. For those processes which do admit positive semi-definite embeddings, however, FFT-based methods are efficient. The following theorem provides sufficient conditions for a Toeplitz covariance matrix to have a positive semi-definite circulant embedding [5, Theorem 2]. The statement is written in terms of the covariance function $K[i] = (\Lambda_x)_{1(i+1)}$.

THEOREM A.1: *If the values of the covariance function of an l-point random vector,* $K[0], K[1], \ldots, K[l-1]$*, form a sequence that is convex*[2]*, decreasing, and nonnegative, then the associated* $2(l-1) \times 2(l-1)$ *circulant matrix is positive semi-definite.*

As an example, consider the situation where the vector to be synthesized, x, consists of the increments between regularly spaced samples of fractional Brownian motion (fBm) [15] for Hurst parameter $H \in (1/2, 1)$. The increments process is stationary with covariance function

$$K[m] = \frac{\sigma^2\delta^{2H}}{2}\left(|m+1|^{2H} + |m-1|^{2H} - 2|m|^{2H}\right) \tag{51}$$

where δ is the sampling interval and σ^2 is a constant appearing in the definition of fBm [1].

COROLLARY A.1: *The covariance matrix of l increments of fractional Brownian motion can be embedded in a* $2(l-1) \times 2(l-1)$ *positive semi-definite circulant matrix.*

Proof: One can verify, as follows, that the covariance of the increments, $K[m]$, defined in (51), satisfies the conditions of Theorem A.1, i.e. is convex, decreasing, and nonnegative.

To verify that $K[m]$ is nonnegative, note that

$$K[m] = \frac{\sigma^2\delta^{2H}}{2}\left(\left(|m+1|^{2H} - |m|^{2H}\right) - \left(|m|^{2H} - |m-1|^{2H}\right)\right). \tag{52}$$

Since $2H > 1$, $(|m+1|^{2H} - |m|^{2H}) \geq (|m|^{2H} - |m-1|^{2H})$; so, $K[m] \geq 0$. To verify that $K[m]$ is convex, view m as a continuous parameter greater than zero, and note that

$$\frac{dK}{dm} = \frac{\sigma^2\delta^{2H}}{2}\left(2H|m+1|^{2H-1} \pm 2H|m-1|^{2H-1} - 2(2H)|m|^{2H-1}\right)$$

$$\begin{aligned}\frac{d^2K}{dm^2} &= \frac{\sigma^2\delta^{2H}}{2}\Big(2H(2H-1)|m+1|^{2H-2} + 2H(2H-1)|m-1|^{2H-2} \\ &\qquad -2(2H)(2H-1)|m|^{2H-2}\Big) \\ &= \frac{\sigma^2\delta^{2H}}{2}\Big(2H(2H-1)|m+1|^{2H-2} + 2H(2H-1)|m-1|^{2H-2} \\ &\qquad -2H(2H-1)|2^{1/(2H-2)}m|^{2H-2}\Big).\end{aligned} \tag{53}$$

Since $-1 < 2H-2 < 0$, $2^{1/(2H-2)} < 1$. This implies that $|m+1|^{2H-2} > |2^{1/(2H-2)}m|^{2H-2}$; so, $d^2K/dm^2 > 0$, and $K[m]$ is convex. That K is decreasing follows from the fact that K is convex, nonnegative, and asymptotically approaching zero.

Finally, although not commonly done, one can also consider using FFT methods to form low-rank approximations to covariance matrices. One method for doing this is to pick out terms from the expansion in (48). That is, one forms a rank $\min(r, l)$ approximation

$$\Lambda_x \approx \sum_{j=1}^{r} (f_{i_j}S^T)^T g_{i_j} (f_{i_j}S^T) \tag{54}$$

where $g_{i_1} \geq g_{i_2} \geq \cdots \geq g_{i_{2(l-1)}}$ are the ordered elements of the diagonal of G and $(f_{i_j}S^T)$ are the corresponding Fourier vectors truncated by the selection matrix S. Hence, one can use FFTs to compute both simulations and covariance matrix approximations. However, this approximation, while very efficient to compute, does not correspond to the optimal Karhunen-Loeve approximation and, in fact, may be far from it. Moreover, as illustrated in Section 5.1 this FFT-based approximation will often be inferior (in the sense of mean-squared error) to our Krylov subspace-based approximation of the same rank.

Notes

1. Note that although the FFT method can be used to obtain exact simulations of the fBm process (Appendix A), truncating this transform to keep only a finite number of terms (which is the natural approximation procedure outlined in the Appendix) does not correspond to mean-square optimal approximations based on the exact Karhunen-Loeve decomposition.
2. A convex sequence is one such that for any two integers $m < n$, $\lambda K[m] + (1 - \lambda)K[n] \geq K[i]$ for all $\lambda \in [0,1]$ and $m \leq i \leq n$.

References

1. R.J. Barton and H.V. Poor, "Signal Detection in Fractional Gaussian Noise," *IEEE Transactions on Information Theory*, vol. 34, no. 5, 1988, pp. 943–959.
2. G. Burgers, P.J.V. Leeuwen, and G. Evensen, "Analysis Scheme in the Ensemble Kalman Filter," *Monthly Weather Review*, vol. 126, 1998, pp. 1719–1724.
3. C. Cabos, "Evaluation of Matrix Functions with the Block Lanczos Algorithm," *Computers and Mathematics with Applications*, vol. 33, no. 1/2, 1997, pp. 45–57.
4. J.W. Demmel, *Applied Numerical Linear Algebra*, Philadelphia: SIAM, 1997.
5. C.R. Dietrich and G.N. Newsam, "Fast and Exact Simulation of Stationary Gaussian Processes Through Circulant Embedding of the Covariance Matrix," *SIAM Journal of Scientific Computation*, vol. 18, no. 4, 1997, pp. 1088–1107.
6. V. Druskin and L. Knizhnerman, "Extended Krylov Subspaces: Approximation of the Matrix Square Root and Related Functions," *SIAM Journal on Matrix Analysis and Applications*, vol. 19, no. 3, 1998, pp. 755–771.
7. N. Dunford and J.T. Schwartz, *Linear Operators*, New York: Wiley Interscience Publishers, 1988.
8. G. Evensen, "Sequential Data Assimilation with a Nonlinear Quasi-Geostrophic Model Using Monte Carlo Methods to Forecast Error Statistics," *Journal of Geophysical Research*, vol. 99, no. C5, 1994, pp. 10143–10162.
9. A.B. Frakt, "Internal Multiscale Autoregressive Processes, Stochastic Realization, and Covariance Extension," Ph.D. thesis, MIT, 1999.
10. G. Golub and C.V. Loan, *Matrix Computations*, Baltimore: Johns Hopkins University Press, 1996.
11. W.W. Irving, "Multiscale Stochastic Realization and Model Identification with Applications to Large-Scale Estimation Problems," Ph.D. thesis, MIT, 1995.
12. A.G. Journel and C.T. Huijbregts, *Mining Geostatistics*, New York: Academic Press, 1978.
13. C.L. Keppenne, "Data Assimilation into a Primitive-Equation Model with a Parallel Ensemble Kalman Filter," *Monthly Weather Review*, vol. 128, 2000, pp. 1971–1981.
14. S. Mallat, G. Papanicolau, and Z. Zhang, "Adaptive Covariance Estimation of Locally Stationary Processes," *Annals of Statistics*, vol. 26, no. 1, 1998, pp. 1–47.
15. B.B. Mandelbrot and J.W.V. Ness, "Fractional Brownian Motions, Fractional Noises and Applications," *SIAM Review*, vol. 10, no. 4, 1968, pp. 422–437.
16. B.N. Parlett and D.S. Scott, "The Lanczos Algorithm with Selective Orthogonalization," *Mathematics of Computation*, vol. 33, no. 145, 1979, pp. 217–238.
17. R.H. Reichle, D.B. McLaughlin, and D. Entekhabi, "Hydrologic Data Assimilation with the Ensemble Kalman Filter," *Monthly Weather Review*, 2001, in press.
18. W. Rudin, *Functional Analysis*, 2nd edition, New York: McGraw-Hill, 1991.
19. Y. Saad, "On the Rates of Convergence of the Lanczos and the Block-Lanczos Method," *SIAM Journal of Numerical Analysis*, vol. 17, no. 5, 1980, pp. 687–706.
20. Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester: Manchester University Press, 1992.

21. M.K. Schneider, "Krylov Subspace Estimation," Ph.D. thesis, Massachusetts Institute of Technology, 2001.
22. M.K. Schneider and A.S. Willsky, "Krylov Subspace Estimation," *SIAM Journal on Scientific Computing*, vol. 22, no. 5, 2001, pp. 1840–1864.
23. R. Todling and S.E. Cohn, "Suboptimal Schemes for Atmosperic Data Assimilation Based on the Kalman Filter," *Monthly Weather Review*, vol. 122, 1994, pp. 2530–2557.
24. R. Todling, S.E. Cohn, and N.S. Sivakumaran, "Suboptimal Schemes for Retrospective Data Assimilation Based on the Fixed-Lag Kalman Smoother," *Monthly Weather Review*, vol. 126, 1998, pp. 2274–2286.
25. H.A.V.D. Vorst, "An Iterative Solution Method for Solving $f(A)x = b$, Using Krylov Subspace Information Obtained for the Symmetric Positive Definite Matrix A," *Journal of Computational and Applied Mathematics*, vol. 18, 1987, pp. 249–263.
26. G. Xu, Y. Cho, and T. Kailath, "Application of Fast Subspace Decomposition to Signal Processing and Communication Problems," *IEEE Transactions on Signal Processing*, vol. 42, no. 6, 1994, pp. 1453–1461.
27. G. Xu and T. Kailath, "Fast Estimation of Principal Eigenspace Using Lanczos Algorithm," *SIAM Journal on Matrix Analysis and Applications*, vol. 15, no. 3, 1994, pp. 974–994.
28. G. Xu and T. Kailath, "Fast Subspace Decomposition," *IEEE Transactions on Signal Processing*, vol. 42, no. 3, 1994, pp. 539–551.
29. G. Xu, H. Zha, G. Golub, and T. Kailath, "Fast Algorithms for Updating Signal Subspaces," *IEEE Transactions on Circuits and Systems–II: Analog and Digital Signal Processing*, vol. 41, no. 8, 1994, pp. 537–549.