SIAM J. SCI. COMPUT.
Vol. 36, No. 2, pp. A588–A608

PRECONDITIONED KRYLOV SUBSPACE METHODS FOR SAMPLING MULTIVARIATE GAUSSIAN DISTRIBUTIONS*

EDMOND CHOW† AND YOUSEF SAAD‡

Abstract. A common problem in statistics is to compute sample vectors from a multivariate Gaussian distribution with zero mean and a given covariance matrix A. A canonical approach to the problem is to compute vectors of the form $y = Sz$, where S is the Cholesky factor or square root of A, and z is a standard normal vector. When A is large, such an approach becomes computationally expensive. This paper considers preconditioned Krylov subspace methods to perform this task. The Lanczos process provides a means to approximate $A^{1/2}z$ for any vector z from an m-dimensional Krylov subspace. The main contribution of this paper is to show how to enhance the convergence of the process via preconditioning. Both incomplete Cholesky preconditioners and approximate inverse preconditioners are discussed. It is argued that the latter class of preconditioners has an advantage in the context of sampling. Numerical tests, performed with stationary covariance matrices used to model Gaussian processes, illustrate the dramatic improvement in computation time that can result from preconditioning.

Key words. preconditioning, sampling, Gaussian processes, covariance matrix, matrix square root, sparse approximate inverse, Krylov subspace methods, Lanczos process

AMS subject classifications. 60G15, 62E17, 65C20, 65F08, 65F10, 65F60

DOI. 10.1137/130920587

1. Introduction.

In a wide variety of probabilistic simulations, it is necessary to compute sample vectors from a multivariate Gaussian distribution with zero mean and a given (and possibly changing) covariance matrix. For large-scale problems, this task is computationally expensive, and despite the availability of several approaches, fast methods are still highly desired since such sampling remains a computational bottleneck in these simulations.

In geostatistical simulations, for example, data points are associated with spatial locations, and the covariance between two data points may be expressed as a function of their spatial locations. Upward of 10^6 locations may be used, leading to a very large covariance matrix. These locations are often the grid points of a regularly spaced two-dimensional (2-D) grid, or may be distributed irregularly over a geographic region. In the common case where the covariance function only depends on the vector joining the two spatial locations, the covariance function is called *stationary*.

Most methods for sampling a Gaussian distribution with a given covariance matrix may be classified into three categories: factorization methods, polynomial methods, and spectral methods. Factorization methods are the best-known. In this paper, we assume that the covariance matrix, which we denote by A, is positive definite. A Gaussian sample $y \sim N(0, A)$ can be computed by a factorization method as $y = Sz$, where $A = SS^T$ and z is a standard normal vector. With this definition of S, it is

*Submitted to the journal's Methods and Algorithms for Scientific Computing section May 10, 2013; accepted for publication (in revised form) January 22, 2014; published electronically April 3, 2014.

http://www.siam.org/journals/sisc/36-2/92058.html

†School of Computational Science and Engineering, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0765 (echow@cc.gatech.edu). This author's work was supported by NSF under grant ACI-1306573.

‡Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 (saad@cs.umn.edu). This author's work was supported by NSF under grant DMS-1216366.

clear that the covariance of y will be the matrix A. As a result, any S that satisfies $A = SS^T$ can be used, for example, the lower triangular Cholesky factor or principal square root of A. Use of the Cholesky factor is most common.

As is well known, however, the Cholesky factorization is computationally expensive for large problems, scaling as $O(n^3)$ for dense $n \times n$ covariance matrices. When the covariance matrix is sparse, a sparse Cholesky factorization could be used [30]. Even sparse matrix Cholesky factorizations can be exceedingly expensive, for example, in situations when the underlying graph of the matrix is based on a three-dimensional (3-D) grid [18].

The second category of methods is based on matrix polynomials. Here, sample vectors of the form $p(A)z$ are computed, where z is again a normal vector of mean zero, and p is a polynomial usually chosen such that $p(A)$ approximates the principal square root of A, i.e., $p(t)$ approximates $\sqrt{t}$ over the spectrum of A. One need not form the matrix $p(A)$ explicitly. Instead, for each z, $p(A)z$ is computed from a sequence of products with the matrix A. Two main types of polynomial methods have been used: those that choose $p(A)$ as an expansion of Chebyshev or other orthogonal polynomials [15, 8], and those that construct the sample from a Krylov subspace; see [13, 16, 31, 12, 21] and the references therein. The latter are called Krylov subspace sampling methods and will be described in detail in section 2.1.

When expanding the square root function, $\sqrt{t}$, in orthogonal polynomials, we first note that in some cases, explicit formulae for the expansion coefficients are available. For example, one can exploit the fact that the function $\sqrt{(1-\xi)/2}$, defined for $|\xi| < 1$, admits a known expansion in terms of Legendre polynomials [26]. From this we can obtain an explicit expansion for $\sqrt{t}$ in the interval $(0, a)$ with $a > 0$ via the change of variable $(1-\xi)/2 = t/a$ and by writing $\sqrt{t} = \sqrt{a}\sqrt{t/a}$. This results in the expansion

$$\sqrt{t} = \sum_{j=0}^{\infty} \frac{-\sqrt{a}}{(2j-1)(2j+3)} P_j\left(1 - 2\frac{t}{a}\right),$$

where P_j is the Legendre polynomial of degree j. Note that the coefficient corresponding to $j = 0$ is positive ($2\sqrt{a}/3$) and all others are < 0. However, because such explicit formulae for the expansion coefficients use an interval $(0, a)$ that includes the origin, the expansion converges slowly and is not appealing in practice. Thus orthogonal polynomial methods usually compute the expansion coefficients by some other means. The approach taken by Fixman [15] is to compute the expansion coefficients by *interpolation*, that is, Chebyshev interpolation points are chosen, and the coefficients are computed by solving the equations that set the value of the interpolant at each point. Due to the discrete orthogonality property of Chebyshev polynomials, the equations are easy to solve. A more elaborate expansion, using a two-step approach in which the function is first approximated by a spline and the resulting function expanded in a basis of orthogonal polynomials, has also been presented [8].

We mentioned above that polynomial approximation methods compute a sample $p(A)z$ without computing $p(A)$. Computing $p(A)z$ for an arbitrary z only requires matrix-vector products with the matrix A and thus has better scaling compared to Cholesky factorization. In addition, the matrix A is not required explicitly, making polynomial methods appropriate in "matrix-free" scenarios, including when A is dense but a fast, sub-$O(n^2)$ algorithm for operating with A is available. Another potential advantage of polynomial methods is that they are approximative rather than exact. This may further reduce the cost of the method since, in many applications, computing a sample with the exact covariance may be unnecessary.

The third category of methods is spectral methods, which exploit the structure of stationary covariance functions to approximate Gaussian samples via fast Fourier transforms [35]. Note also that a stationary covariance function over samples on a regular grid leads to a covariance matrix with block Toeplitz structure. Such matrices may often be embedded in a circulant matrix, allowing FFT methods to be used [11]. Spectral methods are the methods of choice when covariance matrices are highly structured in these ways.

In this paper, we address polynomial methods, which are asymptotically faster than factorization methods and more general than spectral methods. Polynomial approximation methods may be regarded as iterative, with one iteration per polynomial order, and we are motivated to reduce the number of iterations required to achieve a given accuracy. The approach that we take is akin to the concept of "preconditioning" for linear systems. Although the term preconditioning has been used with sampling methods before, the goals of these previous works have been very different; see section 2.3.

This paper presents a method of accelerating polynomial methods for Gaussian sampling. The method uses an approximate factorization of the covariance matrix, which can be called a preconditioner. To motivate the choice of preconditioner, consider that the inverse of a covariance matrix, known as a *precision matrix*, measures the conditional independence between data points. For Gaussian Markov distributions, this precision matrix is sparse. For many other types of Gaussian distributions, the precision matrix shows a strong decay in the size of its elements. These matrices may be well approximated by a sparse matrix, and thus we propose using a sparse approximate inverse preconditioner.

2. Preconditioned polynomial sampling methods.

2.1. Polynomial sampling methods. The basic idea of the Krylov subspace method for approximating $f(A)z$, where f is a certain function, is to find an approximation of $f(A)z$ from the Krylov subspace

$$K_m(A, z) = \text{span}\{z, Az, \ldots, A^{m-1}z\}, \tag{2.1}$$

where z is a standard normal vector. This means that $f(A)z$ is approximated by a vector of the form $p_{m-1}(A)z$, where p_{m-1} is a polynomial of degree $\leq m-1$. Hence Krylov subspace sampling methods and methods based on polynomial expansions are related.

An orthogonal basis $v_1, v_2, \ldots, v_m$ of the Krylov subspace K_m can be built by essentially a Gram–Schmidt process. For a general matrix A (not necessarily Hermitian), this is the Arnoldi process. In the Hermitian case, the Arnoldi process simplifies to the Lanczos process which is described in Algorithm 1.

The result of the algorithm, in exact arithmetic, is the system of vectors $V_m = [v_1, v_2, \ldots, v_m]$ which forms an orthonormal basis of the subspace K_m. In addition we use the coefficients α_j, β_j generated by the algorithm to form the tridiagonal matrix

$$T_m = \text{Tridiag}\left[\beta_j, \alpha_j, \beta_{j+1}\right]_{j=1:m} \ . \tag{2.2}$$

This is a tridiagonal matrix whose nonzero entries of row j are $\beta_j, \alpha_j, \beta_{j+1}$, respectively. Note that for row 1 the term β_1 is omitted and for row m the term β_{m+1} is omitted. Letting e_m^T be the mth row of the $m \times m$ identity matrix, we will also denote by $\bar{T}_m$ the $(m+1)\times m$ matrix obtained by appending the row $\beta_{m+1}e_m^T$ to T_m. With this

ALGORITHM 1. LANCZOS PROCESS.

Data: Matrix A, initial vector v_1 with $\|v_1\| = 1$, integer m
Result: Lanczos basis and Lanczos factorization

```
Initialization: β_1 = 0 and v_0 = 0
for j = 1 to m do
    v = A v_j − β_j v_{j−1}
    α_j = v_j^T v
    v = v − α_j v_j
    β_{j+1} = ‖v‖
    if β_{j+1} = 0 then
        Set m = j and Return
    end
    v_{j+1} = v/β_{j+1}
end
```

the following relations are satisfied:

$$AV_m = V_m T_m + \beta_{m+1} v_{m+1} e_m^T \tag{2.3}$$

$$= V_{m+1} \bar{T}_m. \tag{2.4}$$

In particular, the orthogonality of the v_i's implies that

$$V_m^T A V_m = T_m. \tag{2.5}$$

The Krylov subspace method for approximating the matrix function $f(A)z$ starts with the observation that the optimal approximation in K_m (that minimizes the 2-norm of the error) from this subspace is

$$y_m^* = V_m V_m^T f(A) z.$$

This is the orthogonal projection of the exact solution onto the Krylov subspace. Let us now choose the basis V_m such that the first vector of this basis is $v_1 = z/\|z\|_2$. Then we can write the optimal approximation as

$$y_m^* = \beta V_m V_m^T f(A) V_m e_1,$$

where $\beta = \|z\|_2$ and e_1 is the first column of the $m \times m$ identity matrix. If we now take $V_m^T f(A) V_m \approx f(V_m^T A V_m)$, then from (2.5) we obtain the approximation

$$\tilde{y}_m = \beta V_m f(T_m) e_1 \approx y^*. \tag{2.6}$$

The quantity $f(T_m)$ is computed using any method, and its computation is inexpensive because m is assumed to be small.

Various upper bounds for the error $\|f(A)z - \tilde{y}_m\|$ have been developed to analyze convergence; see, for example [13, 12, 37, 22]. Although these bounds often tackled the case when $f(t) = e^t$, extensions to other functions are usually straightforward [21]. Most of these bounds rely on some estimate of the error made in the best uniform approximation of the function f by polynomials of degree m. An interesting result [22] links the error for the case of interest to us, i.e., when $f(t) = \sqrt{t}$, to that of linear systems, i.e., the case when $f(t) = 1/t$, by proving that $\|f(A)z - \tilde{y}_m\| \leq \sqrt{\lambda_{\min}} \|r_m\|$, where $\lambda_{\min}$ is the smallest eigenvalue of A and r_m is the residual vector at the mth

step of the conjugate gradient method for solving $Ay = z$. The norm of this residual is in turn bounded by exploiting known results.

A well-known weakness of these bounds is that they are not sharp enough to explain the improved convergence that preconditioning can bring. For the case when $f(t) = 1/t$, which, as was just mentioned, can serve to study the case $f(t) = \sqrt{t}$, preconditioning generally results in a good clustering of the eigenvalues around one, which in turn will lead to much faster convergence. This improved convergence is not captured by existing error bounds, such as the ones mentioned above, since these rely solely on the largest and smallest eigenvalues, or the condition number, of the matrix. In effect the bounds imply a *linear convergence* whereas a *superlinear convergence* is observed in practice, and this was studied early on by, among others, Axelsson and Lindskog [4] and Van der Sluis and Van der Vorst [38], and again very recently by Axelsson and Karátson [3]. While it is well known that typical convergence bounds fall short of taking advantage of eigenvalue clustering, a few papers have, however, studied this carefully. Among these, the articles [32, 23, 3] analyze the superlinear convergence behavior of Krylov-based methods in the presence of clusters in the spectrum.

For the case addressed in this paper, $f(A) = A^{1/2}$, the corresponding approximation

$$A^{1/2}z \approx \beta V_m {T_m}^{1/2} e_1 \tag{2.7}$$

is based on computing the matrix square root on a much smaller subspace, where it is inexpensive to compute exactly, and then mapping the result to the original space. Implicitly, we have $\tilde{y} = p(A)z$, where $p(A)$ is an approximation to the square root of A. The method differs from the Chebyshev polynomial approximation in that the approximation is not uniform over an interval, but is optimized over the eigenvalues of A. Note that to compute $\tilde{y}$ by (2.6), V_m needs to be stored; no such storage is needed in the Chebyshev polynomial approximation.

In practice, the number of basis vectors m does not need to be chosen beforehand. An approximation $\tilde{y}$ can be computed after each Lanczos iteration, i.e., (2.6) is computed after line 10 of Algorithm 1 using the basis vectors and tridiagonal matrix computed so far. A stopping criterion is applied based on the accuracy of $\tilde{y}$. Such a stopping criterion will be discussed in section 4.1.

We note in passing that when several sample vectors are needed for the same covariance matrix, it is possible to use block-Krylov methods, effectively generating several samples with one block-Krylov subspace [1].

2.2. Preconditioning. When solving linear systems of equations by Krylov subspace methods, one can substantially improve convergence of the iterative process by preconditioning the linear system; see, e.g., [33]. This entails a modification of the original system so that the resulting coefficient matrix has a more favorable eigenvalue distribution which leads to much faster convergence. It is natural to ask whether or not a similar enhancement can be achieved for the problem of approximating $A^{1/2}z$ via Krylov subspaces. This question is addressed in this section. We will see that unlike the linear system case, preconditioning here changes the problem being solved. However, a sample with the desired covariance can still be recovered.

Let $G^TG \approx A^{-1}$ be a preconditioner and thus $GAG^T \approx I$. We assume that G is invertible. Consider using the preconditioned matrix GAG^T, rather than A, in a polynomial sampling method to produce a sample

$$\tilde{w} = p(GAG^T)z \tag{2.8}$$

which is approximately distributed as $N(0, GAG^T)$. Since GAG^T is well conditioned, we expect that only a small number of terms is required in the polynomial approximation to the square root of GAG^T. To construct a sample with the desired covariance, apply G^{-1} to each sample $\tilde{w}$,

$$\tilde{y} = G^{-1}\tilde{w} = G^{-1}p(GAG^T)z,$$

which is approximately distributed as $N(0, A)$ and approximates $G^{-1}(GAG^T)^{1/2}z$. The matrix $S = G^{-1}(GAG^T)^{1/2}$ satisfies

$$SS^T = G^{-1}(GAG^T)^{1/2}(GAG^T)^{1/2}G^{-T} = A$$

as desired, but it is not a Cholesky factor or square root of A. The idea leading to the method described here is that S can be *any* of an infinite number of quantities that satisfies $SS^T = A$. How well the covariance of $\tilde{y}$ approximates A depends on the accuracy of p and not on G. The convergence rate depends on the quality of the approximation $G^TG \approx A^{-1}$. For example, in the extreme case when we have an exact Cholesky factorization at our disposal, $G^TG = A^{-1}$, and we only need to take $p(A) = I$, i.e., $\tilde{y}$ becomes $\tilde{y} = G^{-1}p(GAG^T)z = G^{-1}z$. This corresponds to the standard method based on Cholesky (G^{-1} is the lower triangular Cholesky factor of A). However, *we now have the option of using an approximate factorization instead of an exact one.*

As usual, the preconditioned matrix GAG^T is not formed explicitly. Instead, G and G^T are applied to vectors in these methods. To construct the desired sample, however, we must be able to easily *solve* with G. (The roles of matrix-vector multiplications and solves are reversed if we define the preconditioner GG^T as an approximation to A.) These requirements are more demanding than the usual requirements for preconditioners.

In general, the preconditioner must be designed such that the additional cost of applying the preconditioner in the Lanczos process is more than offset by the reduction in the number of iterations required for convergence. In addition, the cost of constructing the preconditioner must also be low, but this cost can be amortized over several sample vectors that are computed with the same or nearly the same covariance matrix.

We note that if we modify the Krylov subspace sampling algorithm to sample from $N(0, A^{-1})$, then the preconditioning task is somewhat simplified. If the preconditioned matrix used in the iterations is GAG^T, then

$$\begin{aligned} w &\sim N(0, (GAG^T)^{-1}), \\ Gw &\sim N(0, A^{-1}), \end{aligned}$$

that is, it is only necessary to be able to multiply by the factor G, and it is not necessary to be able to solve with G.

2.3. Other preconditioned sampling methods. Schneider and Willsky [34] appear to be the first to propose a preconditioned Krylov subspace sampling method. Their paper addresses the dual problem of simulation, i.e., obtaining sample zero-mean vectors y whose covariance matrix is A, and covariance matrix estimation, i.e., the problem of finding a low-rank approximation to A. Their method also relies on Krylov subspaces but it is very different from what we have presented. Their algorithm starts from a *single* vector v (*unrelated* to z) and proceeds to build the

Krylov subspace $K_m(A, v)$. Let V_m be the Lanczos basis for this subspace and T_m the related matrix in (2.2). The paper [34] then builds the columns of $P_m = V_m L_m^{-T}$ as a new basis for $K_m(A, v)$, where $L_m L_m^T$ is the Cholesky factorization of T_m. As is well known [33, section 6.7], these basis vectors are conjugate directions for A, i.e., we have $P^T AP = I$, and they represent the conjugate gradient vectors of the conjugate gradient method, up to scaling factors. Given an m-dimensional white sampling vector w, with unit variance, the sampling vector to simulate A is taken to be

$$y = AP_m w. \tag{2.9}$$

There is a well known and simple 2-vector recurrence relation [33] to obtain the sequence of the p_i's, and this constitutes the main ingredient that links the conjugate gradient algorithm to the Lanczos process. The paper [34] focuses on this relation and the computational advantages resulting from it. Parker and Fox [27] showed how the sampler presented in [34] could be implemented by a simple addition to the conjugate gradient algorithm.

The rationale behind (2.9) is that in the ideal case when $W = [w_1, w_2, \ldots, w_m]$ is a set of random vectors such that the covariance matrix WW^T is the identity, clearly

$$APWW^T P^T A = APP^T A = AV_m T_m^{-1} V_m^T A. \tag{2.10}$$

This is not exactly equal to the original covariance matrix A, but at this point one is tempted to say that $V_m T_m^{-1} V_m^T$ approximates the inverse of A and so $AV_m T_m^{-1} V_m^T A$ is close to A as desired. However, the *approximation* $A^{-1} \approx V_m T_m^{-1} V_m^T$ *is a poor one in general,* unless m is large enough, close to the rank of A, so approximating A by (2.10) will work only under restricted circumstances in practice. This is explained next.

The gist of the method presented in [34] is based on (2.10), and indeed the second part of their algorithm provides the following low rank approximation to the covariance matrix:

$$B_m = AV_m T_m^{-1} V_m^T A. \tag{2.11}$$

A similar but simpler approximation to the covariance matrix is given by $A_m \equiv V_m T_m V_m^T$, and it was shown in [34] that A_m differs from B_m by a rank-3 matrix. A slightly better result can be obtained by comparing B_m with A_{m+1}. Indeed, it follows from the relation (2.4) that $B_m = V_{m+1} \bar{T}_m T_m^{-1} \bar{T}_m^T V_{m+1}^T$. As can be easily shown, the $(m+1) \times (m+1)$ matrix $\bar{T}_m T_m^{-1} \bar{T}_m^T$ differs from T_{m+1} only in its diagonal entry $(m+1, m+1)$. Specifically, $\beta_{m+1}^2 e_m^T T_m^{-1} e_m$ replaces the diagonal entry α_{m+1}, and as a result we have

$$B_m = A_{m+1} - \eta_m v_{m+1} v_{m+1}^T \quad \text{with} \quad \eta_m \equiv \alpha_{m+1} - \beta_{m+1}^2 e_m^T T_m^{-1} e_m \ .$$

In the end the two approximations are close to each other, differing only by a rank-one matrix.

However, for either of B_m or A_{m+1} to be a good approximation to A, their range, which is the Kryolv subspace $K_{m+1}(A, v)$, must capture all the eigenvectors of A, i.e., it must contain good approximations to these eigenvectors. As is well known, the Lanczos process typically takes many more steps to obtain good approximations for interior eigenvalues of A than for those located at both ends of the spectrum. These

approaches will therefore work only if m is large enough (close to the rank of A). In typical situations when A is not well approximated by a small rank matrix, the process may require many steps to converge. This can easily be verified by an experiment with a diagonal matrix with uniformly distributed eigenvalues in the interval $(0,\ 1]$. Another proof of this for the simpler case of A_m is that

$$\|A - A_m\| = \max_{\|v\|=1} \|(A - A_m)v\| \geq \|(A - A_m)v_m\| = \beta_{m+1},$$

as can be easily shown. What this means is that A_m cannot be close to A unless β_{m+1} is small, an indication that a nearly invariant subspace of A has been captured by $K_m(A, v)$; see [33, Proposition 6.6].

Another problem with this approach is that the vectors v_i, $i = 1, \ldots, m$ must be orthogonal in order for the approximation to work, and various reorthogonalization schemes are advocated for this purpose in [34]. In contrast, when approximating a single sampling vector by (2.7), this is not an issue in that the method works much like a conjugate gradient method for solving the linear system $Ay = z$, which approximates $A^{-1}z$ by $\beta V_m T_m^{-1} e_1$.

In summary, the algorithm presented in [34] is implicitly based on first replacing A by a low-rank approximation B_m with which sampling is done. Since B_m is naturally factored as $B_m = (AV_m L_m^{-T})(AV_m L_m^{-T})^T$, sampling with B_m is easy. However, the underlying approximation $B_m \approx A$ is likely to be inaccurate unless m is large (close to the rank of A). The procedure will also require the use of reorthogonalization schemes. In contrast, our proposed algorithm computes an approximation to $A^{1/2}v$ using (2.7) and exploits preconditioning to reduce the number of required steps.

The method proposed in [34] has difficulties when A has repeated eigenvalues. For example, if A equals the identity matrix, then the method "converges" in one step, but the sample is based on the rank-2 approximation B_1 to the identity matrix. To alleviate this problem, Schneider and Willsky [34] propose preconditioning to *spread out* the spectrum of A. Their preconditioning aims to change the eigenvalue distribution and is therefore more akin to preconditioning eigenvalue problems. Note that the goal of such a preconditioner is just the opposite of that of standard preconditioners for linear systems, which aim at clustering the spectrum. Nevertheless, the preconditioned algorithm itself is exactly the preconditioned conjugate gradient (PCG) algorithm with the addition proposed by Parker and Fox [27]. The preconditioner does not need to be in factored form, as is usual for the PCG algorithm. Schneider and Willsky [34] proposed a preconditioner of the form UDU^T, where U approximates the eigenvectors of A and D is an appropriate diagonal matrix. Constructing general and efficient preconditioners to spread out the spectrum is an open problem.

Parker and Fox [27] also proposed the related method

$$\tilde{y}_{PF} = V_m L_m^{-T} z_m,$$

which samples from $N(0, A^{-1})$. One application of this is for the case when the precision matrix A^{-1} is available and is sparse. In this case, the Krylov subspace sampling method iterates with the precision matrix, giving samples from the desired distribution.

We point out that when the inverse of the covariance matrix is modeled rather than the covariance matrix itself, a number of other sampling techniques become available. Gibbs sampling [17], for example, computes samples from $N(0, A^{-1})$ by successively sampling from one-dimensional (1-D) Gaussian distributions where the variances are the diagonal entries of A^{-1}. This method converges slowly (it is similar

in structure to Gauss–Seidel iterations) but accelerated and "multigrid" versions have been developed [19].

The concept of preconditioning in sampling also arises when rational Krylov subspaces $K_m((A-\xi I)^{-1}, z)$ rather than standard Krylov subspaces $K_m(A, z)$ are used, and when rational functions are used to approximate the matrix square root or its inverse. The main idea here is to reuse converged eigenvector information from a previously built Krylov subspace; see, e.g., [28, 36, 2].

3. Sparse inverse preconditioning. In the previous section, a method for preconditioning a polynomial sampling method was described. The preconditioner has the constraint that it must be in factored form, $A^{-1} \approx G^T G$, where A is the covariance matrix, and that it must be efficient to apply G, G^T, as well as G^{-1}. This is because (2.8) requires a product with GAG^T at each step of the Lanczos, or other polynomial method, and the final step requires a product with G^{-1}. These constraints go beyond the normal requirements for preconditioners for solving linear systems.

Approximate triangular factorizations of A or A^{-1} satisfy the above constraints. Thus a possible choice of preconditioner is the class of incomplete Cholesky (IC) factorizations, where $LL^T \approx A$. However, these preconditioners suffer from two potential problems. First, the IC factorization may not be computable, even if A is positive definite. Second, it is not clear how to obtain the preconditioner when A is dense. It is possible to apply IC to a sparsified version of A, but a sparsified A has usually lost its positive definiteness.

In this section, we propose using the factorized sparse approximate inverse (FSAI) preconditioner [24], which overcomes the above problems. The specific choice of preconditioner, however, should depend on the covariance matrix. It turns out that FSAI preconditioners, for reasons to be shown, are particularly effective for covariance matrices commonly used to model Gaussian processes [29]. We briefly describe some of these covariance matrices next, before describing the FSAI preconditioner itself.

3.1. Covariance matrices.

3.1.1. Sparse covariance matrices. Given n sample locations, often in 2-D or 3-D, a covariance function $k(r)$ defines a $n \times n$ stationary covariance matrix, where the (i, j)th entry is $k(r_{ij})$ and where r_{ij} is the distance between sample locations i and j. The sample locations may be chosen to be distributed regularly along a grid, or may be distributed irregularly over a domain.

The function

$$k(r) = \left(1 - \frac{r}{l}\right)_+^j \tag{3.1}$$

is one member of a class of *piecewise polynomial* covariance functions. The subscript plus denotes the positive part, i.e., $x_+ = x$ if $x > 0$, and $x_+ = 0$ otherwise. Thus, the function has compact support and the resulting covariance matrix is sparse. Because of sparsity, such covariance functions are ideal to use with iterative methods. The parameters l and j define the characteristic length scale and differentiability of the Gaussian process, respectively. All entries of the covariance matrix are nonnegative, and the matrix is also positive definite by construction for certain values of j.

3.1.2. Dense covariance matrices. We now describe three covariance functions that are not compact and thus lead to dense covariance matrices. The *exponential* covariance function is

$$k(r) = \exp(-r/l),$$

where l is a characteristic length scale parameter. The *Gaussian radial basis function* (RBF) is

$$k(r) = \exp\left(-\frac{r^2}{2l^2}\right).$$

This covariance function, also known as the *squared exponential* function, is very widely used.

The *Matérn* covariance function is

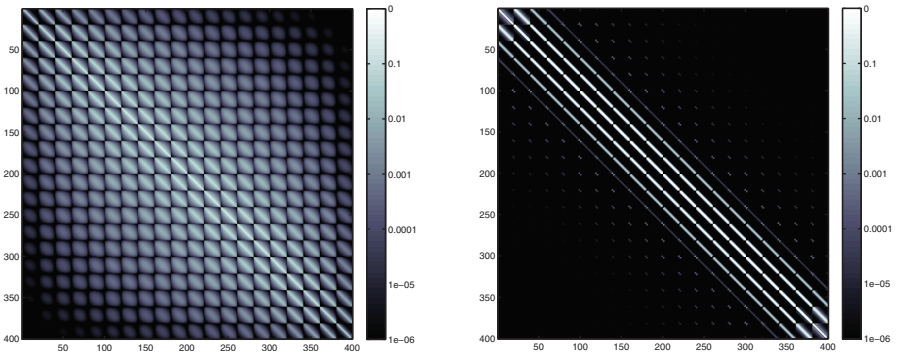
$$k(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}r}{l}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}r}{l}\right),$$

where K_ν is the modified Bessel function of the second kind. The Matérn function is parameterized by ν and is very flexible. For $\nu = 1/2$, we have the exponential covariance function, and for $\nu = \infty$, we have the Gaussian RBF. The Matérn covariance matrices are more ill-conditioned for larger values of ν.

When using iterative methods which require multiplying a dense covariance matrix by a vector, there exist fast algorithms, faster than $O(n^2)$, that do not involve explicitly forming the matrix, e.g., [20, 25].

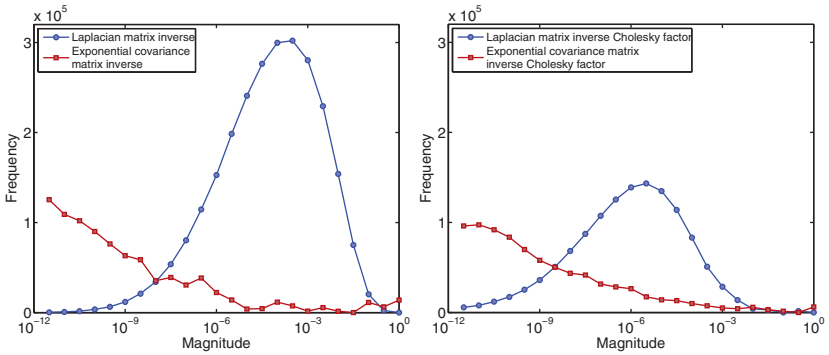
3.1.3. Inverses of covariance matrices. For (1-D) spatial data, the inverse of the exponential covariance matrix is sparse (it is tridiagonal in some matrix ordering) and corresponds to a Markov process. In higher dimensions, the inverse is no longer sparse, but the size of its entries decay very rapidly from the diagonal. For many types of problems, it is typical for entries in the inverse to decay rapidly from the diagonal [10]; however, the decay is particularly rapid for the covariance matrices described above.

The size of the entries in the inverse matrix is illustrated in Figure 1 for an exponential covariance matrix ($l = 1/2$) with 400 sample locations on a 20×20 grid. This is compared to the matrix from the finite difference discretization of the Laplacian operator on the same grid. (Although the Laplacian is often used as a model of a precision matrix, we use the Laplacian for comparison because preconditioners for the Laplacian are well understood.) The number of large or numerically significant



(a) Inverse Laplacian matrix (b) Inverse exponential covariance matrix

FIG. 1. *Magnitude of entries in inverse matrices. White indicates large values; black indicates small values.*



(a) Inverse (b) Inverse Cholesky factor

FIG. 2. *Histogram of magnitude of entries in inverse matrices. The inverse of the exponential covariance matrix contains much smaller entries than the inverse of the Laplacian matrix, although the largest values are approximately of the same size.*

entries in the inverse of the exponential covariance matrix is much smaller than the number in the inverse Laplacian matrix. This is illustrated quantitatively with a histogram in Figure 2 for the inverse and, anticipating the FSAI preconditioner to be advocated, for the Cholesky factor of the inverse. Results for other covariance matrices and with different parameter values are qualitatively similar to that for the exponential covariance matrix shown here.

These observations motivate using a preconditioner that is a sparse approximation to the inverse of the covariance matrix.

3.2. FSAI preconditioners. An FSAI preconditioner has the form $G^T G \approx A^{-1}$, where G is constrained to be sparse. The matrix G is lower triangular and approximates the lower triangular Cholesky factor of A^{-1}. Sparse approximate inverses can be very effective when A^{-1} has a strong decay in the size of its entries as observed for the covariance matrices in the previous subsection.

There are two major techniques for computing sparse approximate inverses in factored form: FSAI [24] and stabilized AINV [5]. Both techniques guarantee that a positive definite preconditioner can be computed if A is positive definite.

Let L be the exact Cholesky factor of A. This L is unknown, and it is only used for the derivation of FSAI. The FSAI method for computing G is based on minimizing the Frobenius norm

$$\|I - GL\|_F^2 = \operatorname{tr}\left((I - GL)(I - GL)^T\right)$$

with the constraint that G has a given lower triangular sparsity pattern $\mathcal{S}_L = \{(i,j) \mid g_{ij} \neq 0\}$. By setting to zero the partial derivatives of the above trace with respect to the nonzero entries in G, we are led to solve

$$(\tilde{G}A)_{ij} = I_{ij}, \quad (i,j) \in \mathcal{S}_L,$$

where $\tilde{G} = D^{-1}G$ and D is the diagonal of L, since $(L^T)_{ij}$ for $(i,j) \in \mathcal{S}_L$ is a diagonal matrix. Since D is not known in general, it is chosen such that the diagonal of GAG^T is all ones. Thus G can be computed without knowing the Cholesky factor L.

Each row i of $\tilde{G}$ can be computed independently by solving a small linear system subproblem involving the symmetric positive definite matrix $A_{\mathcal{J},\mathcal{J}}$, where $\mathcal{J}$ is the index set $\{j \mid (i,j) \in \mathcal{S}_L\}$ (different for each row i). An important point is that if A is dense, as is the case for some covariance matrices, G can still be constructed inexpensively as long as it is sparse. Indeed, in this case, not all entries of A are utilized, and not all entries of A need to be computed for constructing the preconditioner, which is useful if A does not need to be formed to compute matrix-vector products with the matrix.

AINV computes a sparse approximate inverse triangular factorization using an incomplete conjugation (A-orthogonalization) applied to the unit basis vectors. A dropping procedure during the computation is used to restrict the inverse factor to be sparse. Because dropping is performed on the inverse factor rather than on A, it is possible to guarantee the existence of the factorization.

In principle, either FSAI or AINV could be used as preconditioners for the Krylov subspace sampling method. However, we choose FSAI because the subproblems, one for each row i, can be solved independently and in parallel. Further, for stationary covariance functions with sample locations on a regular grid and a natural ordering for the matrix A, the FSAI subproblems corresponding to interior locations are identical and only need to be solved once. This greatly reduces the computation and the storage required for the preconditioner as well as the original matrix A. (However, we did not exploit this in our numerical tests.)

3.3. Sparsity patterns. A sparsity pattern for the approximate inverse factor G needs to be specified for the FSAI algorithm. Typically, increasing the number of nonzeros in G increases the accuracy of the approximation. However, increasing the number of nonzeros in G also increases the cost of constructing and applying the preconditioner. The problem is to find a sparsity pattern that gives good accuracy at low cost.

For irregular sparse matrices A, the structure of powers of A (possibly sparsified) has been proposed and tested as sparsity patterns of sparse approximate inverses [9]. The triangular parts of such patterns may be used for sparse approximate inverse triangular factors.

For sample locations on a regular grid, consider the problem of choosing a sparsity pattern for an individual row of G. Row i of G corresponds to sample location i, and the nonzero elements chosen for row i correspond to a subset of sample locations numbered i and lower (since G is lower triangular). The pattern selected for a row of G is called a *stencil*, and the stencil size is the number of nonzeros in the stencil. For regular grid problems, the same stencil can be used for each row or sample location.

Consider, for example, a 7×7 regular grid of sample locations, leading to a 49×49 matrix A. For several covariance functions, Figure 3 shows row 25 of the exact lower triangular Cholesky factor of A^{-1} as a function on the grid. (In the natural ordering, where grid points are numbered left to right and then top to bottom, location 25 is at the center of the 7×7 grid.) We can choose the stencil using the heuristic that the nonzero pattern of G should correspond to large entries in the inverse Cholesky factor. We choose stencils this way for the regular grid problems in section 4, i.e., based on a small grid where the Cholesky factor of A^{-1} can be computed exactly.

An observation from Figure 3 is that the location of large values of the inverse Cholesky factor on the grid is different for different types of covariance matrices. This implies that different sparsity patterns should ideally be used for different covariance matrices. In particular, the best approximate inverse sparsity pattern for a Laplacian matrix is generally not the best sparsity pattern for covariance matrices.

0.0301	0.0494	0.0645	0.0675	0.0490	0.0280	0.0109
0.0383	0.0676	0.0987	0.1140	0.0683	0.0335	0.0120
0.0419	0.0836	0.1492	0.2230	0.0789	0.0292	0.0090
0.0315	0.0762	0.1935	0.5539	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

(a) Laplacian

0.0022	0.0027	0.0043	0.0057	0.0055	0.0040	0.0041
0.0036	0.0114	0.0342	0.0628	0.0555	0.0324	0.0166
0.0089	0.0438	−0.0506	−0.4890	−0.2767	−0.0706	−0.0107
0.0091	0.0238	−0.6190	1.4544	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

(b) Piecewise Polynomial, $l = 6.5$, $j = 3$

0.0005	0.0006	0.0019	0.0033	0.0031	0.0018	0.0014
0.0011	0.0098	0.0353	0.0682	0.0607	0.0358	0.0197
0.0103	0.0500	−0.0623	−0.5774	−0.3355	−0.0997	−0.0428
−0.0101	0.0050	−0.7472	1.7114	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

(c) Exponential, $l = 1/2$

0.0526	−0.1021	0.1885	−0.2774	0.0000	0.0000	0.0000
−0.1021	0.1983	−0.3661	0.5388	0.0000	0.0000	0.0000
0.1885	−0.3661	0.6758	−0.9947	0.0000	0.0000	0
−0.2774	0.5388	−0.9947	1.4640	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

(d) Gaussian, $l = 1/7$

0.0045	−0.0149	0.0498	−0.1266	−0.0269	0.0038	−0.0006
−0.0163	0.0477	−0.1397	0.3143	0.0483	−0.0078	0.0013
0.0542	−0.1435	0.3728	−0.7424	−0.0639	0.0118	−0.0018
−0.1389	0.3390	−0.7787	1.3580	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

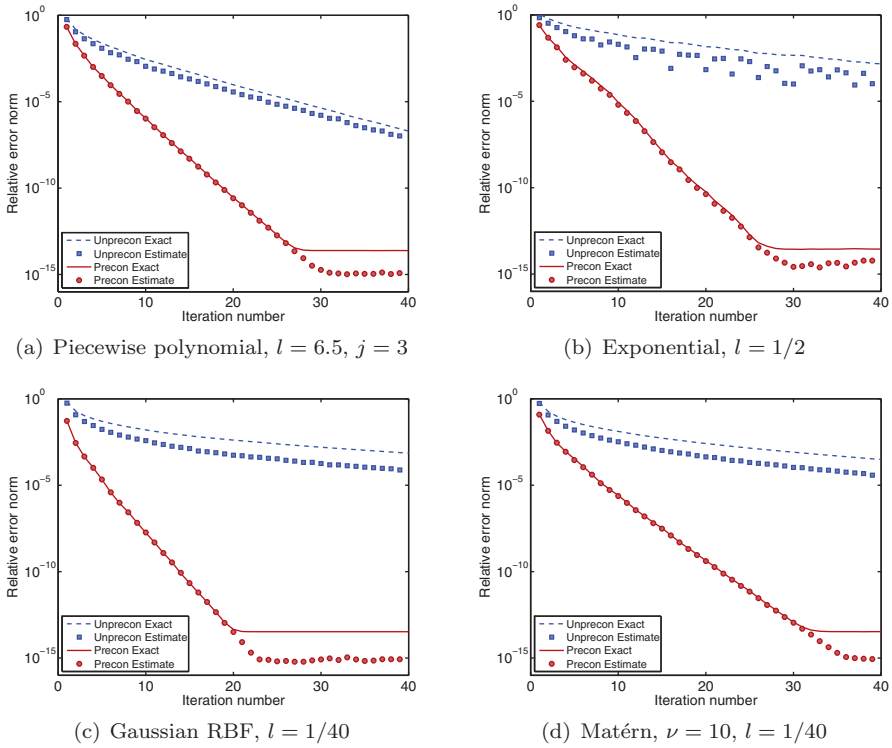
(e) Matérn, $\nu = 10$, $l = 1/7$

FIG. 3. *For a* 7×7 *grid of sample locations, row* 25 *of the lower triangular Cholesky factor of* A^{-1} *plotted as a function of the grid locations.*

4. Numerical tests. In this section, test results are presented for the preconditioned Krylov subspace sampling method for various covariance matrices using the FSAI preconditioner. Each table in this section reports the number of iterations required for convergence and timings for computing a sample vector. The test platform is composed of 2 Intel Xeon X5680 processors (6 cores each at 3.33 GHz) and 24 GB of memory. The computation of the FSAI preconditioner was parallelized using 12 threads.

Results for sparse covariance matrices using a piecewise polynomial covariance function are shown in section 4.2. Results for dense covariance matrices using exponential, Gaussian RBF, and Matérn covariance functions are shown in section 4.3. However, in section 4.1 we first present the stopping criterion used for these four types of covariance matrices.

4.1. Stopping criterion. To detect convergence of the Krylov subspace sampling method so that the Lanczos iterations can be stopped, we use an estimate of a



(a) Piecewise polynomial, $l = 6.5$, $j = 3$

(b) Exponential, $l = 1/2$

(c) Gaussian RBF, $l = 1/40$

(d) Matérn, $\nu = 10$, $l = 1/40$

FIG. 4. *Convergence of the Krylov subspace sampling method with and without preconditioning for four covariance matrices of size* 1600×1600. *The graphs show that the error norm estimate closely matches the exact error norm, especially in the preconditioned cases when convergence is fast.*

relative error norm. Iteration j of the Lanczos process leads to an approximation

$$\tilde{y}_j \approx G^{-1}(GAG^T)^{1/2}z,$$

where we recall that A is the covariance matrix, $G^TG \approx A^{-1}$ is the preconditioner, and z is a standard normal vector. We define the relative error norm as

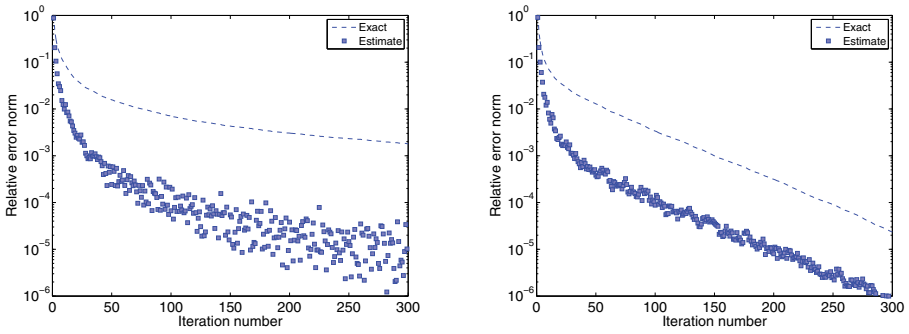
$$e_j = \frac{\|\tilde{y}_j - G^{-1}(GAG^T)^{1/2}z\|_2}{\|G^{-1}(GAG^T)^{1/2}z\|_2}.$$

In practice, this quantity cannot be computed. Instead, we estimate the relative error norm using

$$\tilde{e}_j = \frac{\|\tilde{y}_{j+1} - \tilde{y}_j\|_2}{\|\tilde{y}_{j+1}\|_2}.$$

When convergence is fast, this estimate closely matches the exact relative error norm, since most of the absolute error remaining is reduced in the current step. We use this estimate for both the preconditioned and unpreconditioned cases.

Figure 4 plots the exact and estimated relative error norm of the Krylov subspace sampling method for four covariance matrices, with and without preconditioning. The covariance matrices were constructed using a small, regular 40×40 grid of sample



(a) Krylov subspace sampling method

(b) Krylov subspace sampling method with reorthgonalization

FIG. 5. *Krylov subspace sampling method with and without reorthogonalization, with no preconditioning, for an ill-conditioned matrix of size* 1000×1000. *Without reorthgonalization, convergence slows down.*

locations (matrices are 1600×1600) so that the exact relative error norm could be easily computed and compared. The results verify that the relative norm estimate is suitable for use in a stopping criterion, especially in the preconditioned cases when convergence is fast. The error norm estimate slightly underestimates the exact error norm in the unpreconditioned cases. These problems are smaller versions of those used in the next two subsections. See these subsections for details of the preconditioner used in each case.

The proposed sampling method is based on the Lanczos process, and like any algorithm based on it, a practical concern is the loss of orthogonality between the basis vectors, especially when A is ill-conditioned and when many iterations are used. For our method, loss of orthogonality can be addressed by standard reorthogonalization or restarting techniques. These are discussed specifically for sampling methods in [14, 22, 27]. It is important to understand, however, that the conjugate gradient algorithm for solving linear systems (case $f(t) = 1/t$) can be viewed as a modification of the Lanczos algorithm without reorthogonalization, and yet the algorithm typically converges without a problem, and reorthogonalization is never performed in practice. Our experiments show that in this regard the behavior of our algorithm (case $f(t) = \sqrt{t}$) is similar. While it is beyond the scope of this paper to study the behavior of our algorithm in the presence of rounding, we will illustrate the effect of loss of orthogonality, with a test on a 1000×1000 diagonal matrix with diagonal values 1.05^k, $k = 1, \ldots, 1000$, i.e., with eigenvalues distributed geometrically between 1.05 and approximately 10^{21}. Figure 5 shows the convergence of the sampling method with and without reorthogonalization. Without reorthogonalization, convergence slows down. In both cases, since convergence is very slow (without preconditioning), the estimated error norm is underestimated. Note that addressing loss of orthogonality is an issue separate from the preconditioning ideas presented in this paper. The other numerical tests in this paper did not use reorthogonalization because preconditioning typically resulted in a relatively small number of iterations.

Finally, for the case of low-rank covariance matrices, we point out that approximating A by $V_m T_m V_m^T$ (Lanczos) or $AV_m T_m^{-1} V_m^T A$ [34] will not work well without a good orthogonality level of the vectors v_i, and so reorthogonalization is mandatory for these methods. Our algorithm works more like a conjugate gradient algorithm to

TABLE 1

Piecewise polynomial covariance matrices with sample locations on a regular 1000×1000 *grid: Iteration counts and timings (in seconds) for the Krylov subspace sampling method. Matrices with different values of the parameter* l *are tested, and the average number of nonzeros per row for these matrices is also shown. The preconditioner factor* G *contains at most* 3 *nonzeros per row.*

		Unpreconditioned		Preconditioned		
l	nnz/row	Iterations	Iter time	Iterations	Setup time	Iter time
2.5	21.0	11	1.50	6	0.21	1.35
4.5	68.7	22	5.28	10	0.23	3.12
6.5	136.2	34	13.46	12	0.25	5.39
8.5	223.4	47	30.22	13	0.29	8.70
10.5	345.9	61	53.62	15	0.34	13.38

approximate $f(A)z$, separately for each z, and orthogonality is not essential in the same way that orthogonality is not essential for the conjugate gradient algorithm.

4.2. Tests on sparse covariance matrices. Sparse covariance matrices were generated using the piecewise polynomial covariance function (3.1), with parameter $j = 3$, and with sample locations on a regular 1000×1000 grid with spacing 1, giving a matrix with 10^6 rows and columns. (A "natural" or rowwise ordering of the sample locations was used for this and all covariance matrices with regularly spaced sample locations.) Values of l were chosen between 2.5 and 10.5. The matrices have more nonzeros and are more ill-conditioned for larger values of l.

Table 1 shows the iteration counts and timings for the Krylov subspace sampling method with these covariance matrices. In this and the following tables, *setup time* denotes the time for constructing the FSAI preconditioner, and *iter time* denotes the time for computing a sample vector using the Krylov subspace method. All timings are reported in seconds. The Lanczos iterations were stopped when the estimated relative error norm $\tilde{e}$ fell below 10^{-6}.

Results are shown with and without preconditioning. In the preconditioned case, a very sparse FSAI preconditioner was used, with G containing at most 3 nonzeros per row. In contrast, the covariance matrix has 21.0 to 345.9 nonzeros per row for the range of l tested. The results show that preconditioning reduces the iteration count for convergence as well as the computation time compared to the unpreconditioned case. The time for constructing and applying such a sparse preconditioner is very small. It is also observed that as l increases, the iteration count also increases and that the benefit of preconditioning is greater for larger l.

A sparse piecewise polynomial covariance matrix using an irregular distribution of sample locations was also tested. This test problem consists of 205,761 sample locations on a square domain, $[0, 1] \times [0, 1]$, corresponding to nodes of a finite element triangulation of the domain. Figure 6 shows a small example of the distribution of the sample locations.

The ordering of the rows and columns of the covariance matrix and thus of the preconditioner affects the preconditioner quality. We computed a reverse Cuthill–McKee (RCM) ordering [18] of the sparse matrix associated with the finite element triangulation and used this to reorder the covariance matrix. We found this ordering to be slightly more effective than minimum degree orderings sometimes proposed for sparse approximate inverse preconditioners [6].

Table 2 shows the iteration counts and timings for this covariance matrix with irregular sample locations. For this problem, the iterations were stopped using a

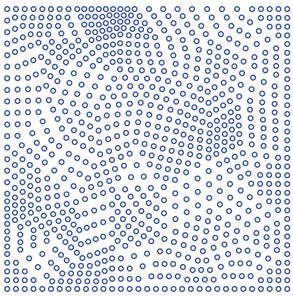


FIG. 6. *Sample locations distributed irregularly over a square domain. This is a small version of the actual problem used.*

TABLE 2

Piecewise polynomial covariance matrix with 205,761 *irregular sample locations: Iteration counts and timings (in seconds) for the Krylov subspace sampling method. Matrices with different values of the parameter l are tested, and the average number of nonzeros per row for these matrices is also shown. The preconditioner factor G contains on average* 3.99 *nonzeros per row.*

		Unpreconditioned		Preconditioned		
l	nnz/row	Iterations	Iter time	Iterations	Setup time	Iter time
.0050	17.2	22	0.605	11	0.048	0.494
.0075	37.9	38	1.739	16	0.054	0.917
.0100	68.0	55	3.396	20	0.057	1.423

relative error norm tolerance of 10^{-9}. For the preconditioner factor G, the sparsity pattern is the lower triangular part of the matrix corresponding to the finite element triangulation, giving G an average of 3.99 nonzeros per row. The results again show that preconditioning reduces the iteration count and computation time for computing a sample vector.

The iterative timings are also much lower than timings for computing a sample via sparse Cholesky factorization. A Cholesky factorization, using approximate minimum degree ordering and computed using 12 threads, required 4.23, 7.75, and 8.34 seconds for the cases $l = 0.0050$, 0.0075, and 0.100, respectively.

4.3. Tests on dense covariance matrices. Dense covariance matrices were generated using the exponential, Gaussian RBF, and Matérn covariance functions. We used sample locations on a $M \times M$ regular grid over the domain $[0, 1] \times [0, 1]$, but sample locations on an irregular grid were also used in the Matérn case. The largest grid tested was $M = 160$, corresponding to a 25600×25600 dense covariance matrix. For all dense covariance matrices, the Lanczos iterations were stopped using an estimated relative error norm tolerance of 10^{-6}.

Table 3 shows iteration counts and timings for exponential covariance matrices of increasing size and with parameter $l = 1/2$. It is observed that iteration counts increase with problem size. Preconditioning is shown to dramatically decrease the iteration count and computation time, particularly for larger problems, and even more so than for the sparse covariance matrices. Here, the factor G contains not more than six nonzeros per row, which minimizes the iteration time. We also note that the setup time for the preconditioner is very small. For dense matrices, the FSAI algorithm is particularly efficient if the entire matrix A is available, since sparse matrix indexing is not needed to construct the subproblem matrices $A_{\mathcal{J},\mathcal{J}}$.

TABLE 3

Exponential covariance matrix on a regular $M \times M$ grid: Iteration counts and timings (in seconds) for the Krylov subspace sampling method. The preconditioner factor G contains at most 6 *nonzeros per row.*

	Unpreconditioned		Preconditioned		
M	Iterations	Iter time	Iterations	Setup time	Iter time
40	74	0.189	13	0.00032	0.023
70	122	1.949	17	0.00092	0.192
100	148	7.418	20	0.00163	0.895
130	191	29.553	24	0.00330	2.810
160	252	81.509	26	0.00478	7.283

TABLE 4

Gaussian RBF covariance matrix on a regular $M \times M$ grid: Iteration counts and timings (in seconds) for the Krylov subspace sampling method. The preconditioner factor G contains at most 22 *nonzeros per row.*

	Unpreconditioned		Preconditioned		
M	Iterations	Iter time	Iterations	Setup time	Iter time
40	108	0.516	9	0.00219	0.021
70	115	1.736	9	0.00709	0.138
100	119	5.781	9	0.01207	0.430
130	121	14.806	9	0.01852	1.154
160	122	34.045	9	0.02085	2.649

We note that for a dense 25600×25600 matrix (corresponding to the 160×160 grid), the Cholesky factorization required 40.8 seconds (corresponding to 139 Gflops/s). In comparison, the preconditioned Krylov subspace sampler required 7.283 seconds for this problem, as shown in Table 3.

Table 4 shows iteration counts and timings for Gaussian RBF covariance matrices of increasing size. For this problem, we chose the parameter $l = 1/M$, which appears to not significantly alter the conditioning of the problem as the problem size increases. For this problem, the optimal number of nonzeros per row of G is 22, much larger than for the exponential covariance matrix. However, the main observation is that again preconditioning greatly reduces the cost of computing a sample vector.

In Table 5 we investigate the effect of the number of nonzeros in G on convergence for the Gaussian RBF problem using a large 160×160 grid. Recall that we refer to the maximum number of nonzeros per row of G as the stencil size. As mentioned, the minimum iteration time is attained at stencil size of approximately 22. For larger stencil sizes, the iteration count is no longer significantly reduced and the additional cost of applying the preconditioner increases the iteration time.

Similar improvements due to preconditioning are shown in Table 6 for Matérn covariance matrices. Here, we vary the parameter ν from 2 to 30, which affects matrix conditioning. Once again, sample locations on a regular 160×160 grid are used. The preconditioner factor G used a 10-point stencil for its sparsity pattern. Increasing the stencil size to 24 does not reduce the iteration count when $\nu = 2$ but does reduce the iteration count to 7 for $\nu = 30$. Here, the setup time is 0.01882 seconds and the iteration time is 1.816 seconds—a more than tenfold improvement over the unpreconditioned case.

Finally, we consider dense covariance matrices using irregular sample locations. Table 7 shows the iteration counts and timings for Matérn covariance matrices using

TABLE 5

Gaussian RBF covariance matrix on a regular 160×160 *grid: Iteration counts and timings (in seconds) for the Krylov subspace sampling method. The preconditioner stencil size is varied from* 3 *to* 24*. The unpreconditioned case is also shown in the last row.*

Preconditioner stencil size	Iterations	Setup time	Iter time
3	50	0.00324	13.320
6	28	0.00480	7.382
8	21	0.00724	5.651
10	19	0.00890	5.464
13	14	0.00977	4.031
15	14	0.01085	4.184
17	12	0.01234	3.583
20	10	0.01768	2.983
22	9	0.02085	2.649
24	9	0.02472	2.711
Unprecon	122	–	34.045

TABLE 6

Matérn covariance matrix with parameter ν *on a regular* 160×160 *grid: Iteration counts and timings (in seconds) for the Krylov subspace sampling method. The preconditioner factor* G *contains at most* 10 *nonzeros per row.*

	Unpreconditioned		Preconditioned		
ν	Iterations	Iter time	Iterations	Setup time	Iter time
2	29	7.001	7	0.00840	2.005
6	48	11.772	8	0.00842	2.610
10	62	17.267	9	0.00852	2.445
14	71	20.250	10	0.00915	3.137
18	78	19.211	11	0.00909	3.644
22	83	24.533	12	0.00812	4.034
26	87	23.488	13	0.00874	4.331
30	91	25.973	13	0.00821	4.003

TABLE 7

Irregular Matérn covariance matrix with parameter ν *with* 13041 *irregular sample locations: Iteration counts and timings (in seconds) for the Krylov subspace sampling method. The preconditioner factor* G *contains on average* 60.8 *nonzeros per row.*

	Unpreconditioned		Preconditioned		
ν	Iterations	Iter time	Iterations	Setup time	Iter time
2	43	3.244	3	0.23863	0.326
6	93	6.513	5	0.24322	0.515
10	124	9.283	6	0.24759	0.665
14	152	12.848	7	0.22196	0.692
18	170	13.747	8	0.22161	0.821
22	181	16.261	9	0.22999	0.884
26	194	16.574	9	0.24543	0.931
30	197	18.074	10	0.24414	1.211

13041 irregular sample locations over a unit square domain, where the sample locations are nodes of a finite element triangularization. RCM ordering based on the finite element mesh was used to order the rows and columns of the covariance matrix. Letting $\tilde{A}$ denote the sparse matrix associated with this discretization, the sparsity

pattern we choose for G is the sparsity pattern of the lower triangular part of $\tilde{A}^6$, which reduces iteration time compared to other powers. The matrix G contains 60.8 nonzeros per row on average. This is a much denser preconditioner than used for the Matérn covariance matrix with regularly spaced sample locations. We attribute this partially to the fact that the optimal sparsity pattern cannot be selected as precisely.

5. Conclusion. The methods presented in this paper compute Gaussian samples having a target covariance matrix A, in matrix polynomial form, i.e., the computed samples are of the form $p(A)z$ for a given vector z, where p is a polynomial. The Lanczos algorithm is used to compute a good approximation of this type to vectors $A^{1/2}z$. The main goal of the paper was to show how to precondition the process. As was argued, standard preconditioners based on Cholesky factorizations have a few drawbacks and so we advocated the use of approximate inverse preconditioners instead. Among these we had a preference for FSAI, since it can be computed very efficiently even if A is dense. In the past, approximate inverse preconditioners have not been very effective preconditioners for solving linear systems of equations, mainly because the inverses of the corresponding matrices do not always have a strong decay property. The picture for covariance matrices is rather different. These matrices are often dense and their inverse Cholesky factors, which are approximated for preconditioning, show a good decay away from the diagonal and can thus can be well approximated at a minimal cost. For various dense covariance matrices of size 25600×25600, we showed that sparse approximate inverse preconditioning can reduce the iteration time by at least a factor of 10. Such preconditioners can also be used for other calculations involving Gaussian processes, not just sampling [7].

Acknowledgments. The authors wish to thank Jie Chen and Le Song for helpful discussions. The authors are also grateful to two anonymous reviewers whose comments contributed to improving this paper.

REFERENCES

[1] T. Ando, E. Chow, Y. Saad, and J. Skolnick, *Krylov subspace methods for computing hydrodynamic interactions in Brownian dynamics simulations*, J. Chem. Phys., 137 (2012), p. 064106.

[2] E. Aune, J. Eidsvik, and Y. Pokern, *Iterative numerical methods for sampling from high dimensional Gaussian distributions*, Statist. Comput., 23 (2013), pp. 501–521.

[3] O. Axelsson and J. Karátson, *Reaching the superlinear convergence phase of the CG method*, J. Comput. Appl. Math., 260 (2014), pp. 244–257.

[4] O. Axelsson and G. Lindskog, *On the rate of convergence of the preconditioned conjugate gradient method*, Numer. Math., 48 (1986), pp. 499–523.

[5] M. Benzi, J. K. Cullum, and M. Tůma, *Robust approximate inverse preconditioning for the conjugate gradient method*, SIAM J. Sci. Comput., 22 (2000), pp. 1318–1332.

[6] M. Benzi and M. Tůma, *Orderings for factorized sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1851–1868.

[7] J. Chen, *A deflated version of the block conjugate gradient algorithm with an application to Gaussian process maximum likelihood estimation*, Preprint ANL/MCS-P1927-0811, Argonne National Laboratory, Argonne, IL, 2011.

[8] J. Chen, M. Anitescu, and Y. Saad, *Computing $f(a)b$ via least squares polynomial approximations*, SIAM J. Sci. Comput., 33 (2011), pp. 195–222.

[9] E. Chow, *A priori sparsity patterns for parallel sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1804–1822.

[10] S. Demko, W. F. Moss, and P. W. Smith, *Decay rates for inverses of band matrices*, Math. Comp., 43 (1984), pp. 491–499.

[11] C. R. Dietrich and G. N. Newsam, *Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix*, SIAM J. Sci. Comput., 18 (1997), pp. 1088–1107.

[12] V. Druskin and L. Knizhnerman, *Krylov subspace approximation of eigenpairs and matrix functions in exact and computer arithmetic*, Numer. Linear Algebra Appl., 2 (1995), pp. 205–217.
[13] V. L. Druskin and L. A. Knizhnerman, *Two polynomial methods of calculating functions of symmetric matrices*, USSR Comput. Math. Math. Phys., 29 (1989), pp. 112–121.
[14] M. Eiermann and O. Ernst, *A restarted Krylov subspace method for the evaluation of matrix functions*, SIAM J. Numer. Anal., 44 (2006), pp. 2481–2504.
[15] M. Fixman, *Construction of Langevin forces in the simulation of hydrodynamic interaction*, Macromolecules, 19 (1986), pp. 1204–1207.
[16] E. Gallopoulos and Y. Saad, *Efficient solution of parabolic equations by polynomial approximation methods*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 1236–1264.
[17] S. Geman and D. Geman, *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*, IEEE Trans. Pattern Anal. Mach. Intell., 6 (1984), pp. 721–741.
[18] A. George and J. W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
[19] J. Goodman and A. D. Sokal, *Multigrid Monte-Carlo method—conceptual foundations*, Phys. Rev. D, 40 (1989), pp. 2035–2071.
[20] L. Greengard and J. Strain, *The fast Gauss transform*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 79–94.
[21] N. J. Higham, *Functions of Matrices: Theory and Computation*, SIAM, Philadelphia, 2008.
[22] M. Ilić, I. W. Turner, and D. P. Simpson, *A restarted Lanczos approximation to functions of a symmetric matrix*, IMA J. Numer. Anal., 30 (2010), pp. 1044–1061.
[23] T. Kerkhoven and Y. Saad, *On acceleration methods for coupled nonlinear elliptic systems*, Numer. Math., 60 (1992), pp. 525–548.
[24] L. Y. Kolotilina and A. Y. Yeremin, *Factorized sparse approximate inverse preconditionings* I. *Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.
[25] R. Krasny and L. Wang, *Fast evaluation of multiquadric RBF sums by a Cartesian treecode*, SIAM J. Sci. Comput., 33 (2011), pp. 2341–2355.
[26] N. N. Lebedev, *Special Functions and Their Applications*, Dover, New York, 1972.
[27] A. Parker and C. Fox, *Sampling Gaussian distributions in Krylov spaces with conjugate gradients*, SIAM J. Sci. Comput., 34 (2012), pp. B312–B334.
[28] M. Popolizio and V. Simoncini, *Acceleration techniques for approximating the matrix exponential operator*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 657–683.
[29] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA, 2006.
[30] H. Rue, *Fast sampling of Gaussian Markov random fields*, J. Roy. Statist. Soc. Ser. B, 63 (2001), pp. 325–338.
[31] Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228.
[32] Y. Saad, *Theoretical error bounds and general analysis of a few Lanczos-type algorithms*, in Proceedings of the Cornelius Lanczos International Centenary Conference, J. D. Brown, M. T. Chu, D. C. Ellison, and R. J. Plemmons, eds., SIAM, Philadelphia, 1994, pp. 123–134.
[33] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[34] M. K. Schneider and A. S. Willsky, *A Krylov subspace method for covariance approximation and simulation of random processes and fields*, Multidimens. Syst. Signal Process., 14 (2003), pp. 295–318.
[35] M. Shinozuka and C.-M. Jan, *Digital simulation of random processes and its applications*, J. Sound Vibration, 25 (1972), pp. 111–128.
[36] D. P. Simpson, I. W. Turner, and A. N. Pettitt, *Fast Sampling from a Gaussian Markov Random Field using Krylov Subspace Approaches*, Technical report, School of Mathematical Sciences, Queensland University of Technology, Brisbane, Australia, 2008.
[37] J. van den Eshof, A. Frommer, T. Lippert, K. Schilling, and H. A. van der Vorst, *Numerical methods for the QCD overlap operator*. I. *Sign-function and error bounds*, Comput. Phys. Comm., 146 (2002), pp. 203–224.
[38] A. van der Sluis and H. A. van der Vorst, *The rate of convergence of conjugate gradients*, Numer. Math., 48 (1986), pp. 543–560.