International Journal for Uncertainty Quantification, 5 (6): 527–544 (2015)

A BLOCK CIRCULANT EMBEDDING METHOD FOR SIMULATION OF STATIONARY GAUSSIAN RANDOM FIELDS ON BLOCK-REGULAR GRIDS

M. Park[*] *& M. V. Tretyakov*

School of Mathematical Sciences, University of Nottingham, Nottingham, NG7 2RD, United Kingdom

Original Manuscript Submitted: 04/06/2015; Final Draft Received: 09/30/2015

We propose a new method for sampling from a stationary Gaussian random field on a grid which is not regular but has a regular block structure, which is often the case in applications. The introduced block-circulant embedding method (BCEM) can outperform the classical circulant embedding method (CEM), which requires a regularization of the irregular grid before its application. Comparison of BCEM vs CEM is performed on typical model problems.

KEY WORDS: *stationary Gaussian random field, irregular grids, sampling techniques, circulant embedding method, symmetric block-toeplitz matrices, block fast-fourier transform*

1. INTRODUCTION

Uncertainties are often modeled using stationary Gaussian fields [1–5]. Efficient generation of samples from stationary Gaussian fields is crucial for using Monte Carlo techniques, which are the backbone of uncertainty quantification simulations, in studying the behavior of systems subject to uncertainties. There are a few numerical techniques for sampling Gaussian random fields on a grid. For instance, one can find a square root of the corresponding covariance matrix using Cholesky's decomposition and then multiplying the square root by a vector of independent Gaussian random variables to simulate a sample. This is an exact method, but it is rarely used in applications due to the high cost of Cholesky's decomposition in high dimensions. Another possibility is the Karhunen-Loeve expansion (see, e.g., [6, 7]), which requires knowledge of eigenvalues and eigenfunctions of the covariance operator for the Gaussian random field. In many cases of practical interest the eigenvalue problem has to be solved numerically, which can be expensive, especially when eigenvalues decay slowly. Also, this method is not exact. The fast and exact method of generating large samples from stationary Gaussian fields on regular grids is the circulant embedding method (CEM) [8–10], which is widely used in various uncertainty quantification (UQ) applications such as groundwater flow simulation [6, 11], weather field forecasting [2], and liquid composite molding processes [12]. The two main drawbacks of CEM are (i) the requirement imposed on the grid to be regular while irregular grids of a block structure naturally appear in many applications (see three typical examples below) and (ii) the need to deal with nonpositive definiteness of circular embedding matrices, which often occur in practical applications. The remedies for the latter were considered in [13–15]; here we deal with the first deficiency of CEM. To this end, we propose a new block-circulant embedding method (BCEM). Let us clarify the matter using the following three examples which come from sampling a random permeability field in groundwater flow simulations.

Example 1.1. *Triangular finite element with a quadrature point located at the barycenter of the triangle.*
Consider generation of a stationary log-normal random permeability field to be used in simulations based on triangular finite elements and the Gaussian quadrature rule of degree 1 within a rectangular domain. Assume that the rectangular domain consists of small rectangles (see Fig. 1) and that there is no overlap of these rectangles. To perform the finite

[*]Correspond to M. Park, E-mail: min.park@nottingham.ac.uk, URL: https://github.com/parkmh

2152–5080/15/$35.00

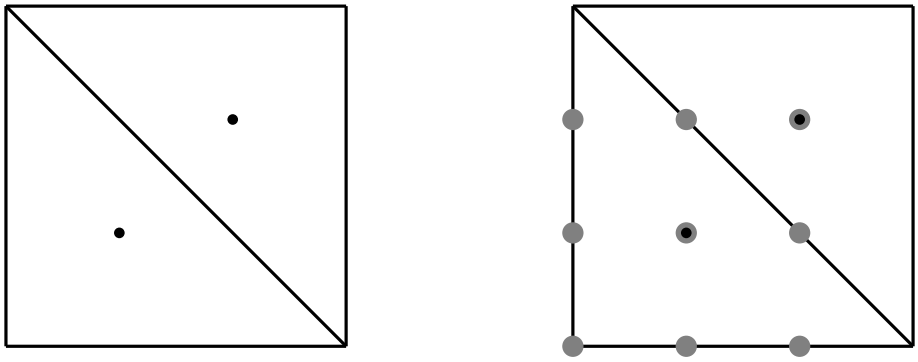

FIG. 1: Left: Locations of nodes in 2D triangular elements. Right: Regular grid nodes (gray) which contain all the original nodes (black). Note that the nodes on the right and top sides of the rectangles belong to the neighboring elements.

element simulation, it is sufficient to have sampled values of the permeability field at the quadrature points only (see the black circles in Fig. 1). The covariance matrix of the corresponding stationary Gaussian random field at all quadrature (black) points is symmetric block-Toeplitz, but the blocks themselves are not symmetric. Hence, in order for the standard CEM to be applicable, seven extra (i.e., artificial from the point of view of sampling permeability values sufficient for the finite element simulation) points should be added to each rectangle (the gray circles in Fig. 1 are regular grid points involving black circles). In contrast to CEM, the new method—BCEM—allows one to sample values at the required points (black circles) without adding extra nodes to the grid, and it does so in a very efficient way, as we will see in the next sections.

Example 1.2. *Cell-centered finite volume discretization in multilevel Monte Carlo (MLMC) computation.*
The multilevel Monte Carlo (MLMC) method is a Monte Carlo technique that can give a substantial reduction of computational complexity in comparison with the standard Monte Carlo method thanks to making use of a hierarchical sampling [16, 17]. In the MLMC algorithm, when computing the difference of quantities on two consecutive grids with mesh sizes h and $2h$, the pair of fine and coarse random samples must come from the same realization of the random field. In the cell-centered finite volume discretization, which uses permeability values at the centers of cells, the locations of coarse random fields do not coincide with nodes on the fine grid (see Fig. 2). In this case there exists a uniform grid with the mesh size $h/2$ containing both fine and coarse points, and hence it is possible to generate

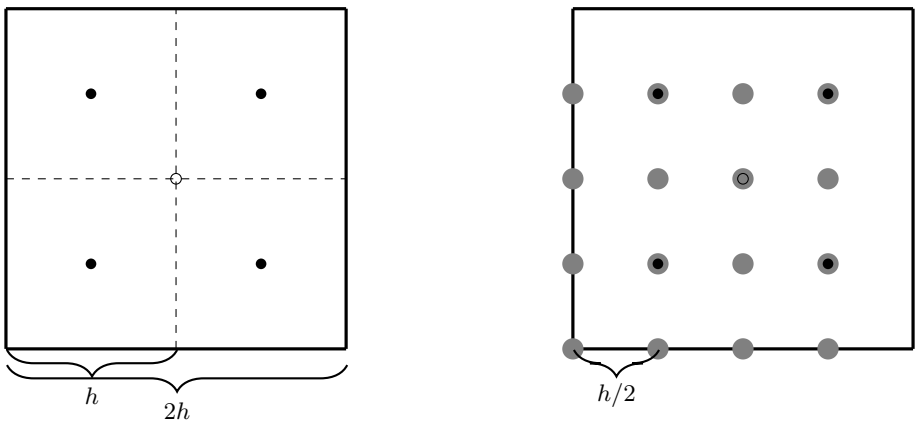

FIG. 2: Left: Location of sampling points on the fine grid (black) of size h and the coarse grid (hollow) of size $2h$ using the cell-centered finite volume discretization in 2D. Right: Uniform grid (gray) which contains both black and hollow points. Note that the nodes on the right and top sides of the rectangles belong to the neighboring elements.

the required pair from the same realization by applying CEM on this finer uniform grid (gray circles in Fig. 2). However, this leads to an increase of both simulation time and memory requirements and, hence, to deterioration of the MLMC performance. Table 1 compares the number of nodes at which the random field actually needed to be sampled for MLMC (which will be the same as the number of nodes used in BCEM) against the ones on the fine, regularized grid required by CEM. In the table, n denotes the total number of blocks used in MLMC. That is, in the d-dimensional space, we have $n = n_1 \sum_{i=1}^{L-1} 2^{d(i-1)}$, where n_1 is the number of blocks on the level 1 and L is the overall number of levels in MLMC. The portion of unused values grows as dimension increases. The benefit of BCEM is that by exploiting the block-regular structure of grids used in MLMC, it allows us to sample at the points used in finite volume simulation without the need to regularize the grid by adding extra points, which can result in substantial savings of both computational time and memory in comparison with applying CEM.

Example 1.3. *Conditional random field generation on block-regular grids.*
The conditional random field generation based on CEM was considered in [18]. In this approach one builds a symmetric matrix of the form

$$R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}, \tag{1}$$

where $R_{11} \in \mathbb{R}^{n_1 \times n_1}$ is a (block) circulant matrix, and $R_{22} \in \mathbb{R}^{n_2 \times n_2}$ is a covariance matrix of field values at locations of measurements. The matrix R can be decomposed as

$$R = \begin{bmatrix} \frac{1}{\sqrt{n_1}} F \Lambda^{1/2} & 0 \\ K & L_{22} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{n_1}} \Lambda^{1/2} F^H & K^H \\ 0 & L_{22}^T \end{bmatrix}, \tag{2}$$

where $K = (1/\sqrt{n_1}) R_{21} F \Lambda^{-1/2}$ and L_{22} is a real matrix such that $L_{22} L_{22}^T = R_{22} - K K^H$. Here F denotes a discrete Fourier transform matrix, and Λ is a diagonal matrix whose entries are eigenvalues of R_{11}. The computational costs of forming Λ, K and KK^H are $\mathcal{O}(n_1 \log n_1)$ flops, $\mathcal{O}(n_2 n_1 \log n_1)$ flops, and $\mathcal{O}(n_2^2 n_1)$ flops, respectively. Note that in practice $n_2 \ll n_1$.

BCEM can also be used for generation of random fields conditioned on observations. As with the unconditional sampling discussed above, applications of conditional sampling often deal with grids which are not regular but have a regular block structure (see, e.g., conditional MLMC simulation in [11]). Since BCEM requires a smaller n_1 value as it does in the unconditional case, BCEM in the conditional random field setting can outperform CEM.

BCEM also has the remarkable feature that it is parallelizable, in contrast to the standard CEM which is a serial algorithm [of course, CEM can exploit parallelism of the fast Fourier transform (FFT) but BCEM's main ingredient is also FFT and it can benefit from FFT parallelism as well], i.e., BCEM has a further significant advantage over CEM.

The rest of the paper is organized as follows. In Section 2 we illustrate the idea of BCEM in the case of one-dimensional space. In Section 3 we present a multidimensional BCEM. Computational complexity of BCEM is discussed in Section 4, where some numerical experiments comparing BCEM with the standard CEM show that already in 2D, BCEM can be 3 times faster in sample generation than CEM.

TABLE 1: Comparison of the number of nodes needed by BCEM and CEM in Example 1.2, where n is the total number of blocks used in MLMC

Dimension	BCEM	CEM	CEM/BCEM
1	$(2+1)n$	$4n$	1.3
2	$(2^2+1)n$	$4^2 n$	3.2
3	$(2^3+1)n$	$4^3 n$	7.1

2. ILLUSTRATION OF THE IDEA

To illustrate the idea of BCEM, we start with presenting it in the 1D case.

Consider a uniform grid $\Omega_r = \{x_0, \dots, x_N\}$ on the interval $\Omega = [x_0, x_N]$ with a grid size $h = (x_N - x_0)/N$, and sets of points $S_i = \{s_1^{(i)}, \dots, s_\ell^{(i)}\} \subset \Omega_i = [x_i, x_{i+1}]$ with $s_j^{(i)} = x_i + \delta_j$, where $0 \le \delta_1 < \delta_2 < \cdots < \delta_\ell < h$ (see Fig. 3). Note that δ_j are independent of the index i. The grid $\Omega_s := \bigcup_{i=0}^{N-1} S_i$ is, in general, nonuniform (it is uniform if $\ell = 1$) but it is block uniform, i.e., the distribution of points in each subinterval (in other words, block) Ω_i is the same.

Let $\mathrm{Z}(x)$, $x \in \mathbb{R}$, be a stationary Gaussian random field with zero mean and covariance function $r(x)$. Our aim is to sample from $\mathrm{Z}(x)$ on the grid Ω_s. If Ω_s is not a grid of equispaced points, then the covariance matrix of the field $\mathrm{Z}(x)$ on Ω_s is not Toeplitz. In this case the standard CEM [8–10] cannot be applied to this covariance matrix in order to perform highly efficient computing of its square root with subsequent generation of the required Gaussian field samples. The simplest remedy is to extend the non-uniform grid Ω_s to the uniform grid $\tilde{\Omega}_s$ by adding points (see Fig. 3) and then apply the standard circulant embedding method, but this approach results in a substantial increase of computational costs. In this paper, we propose a different approach which does not need to add points to Ω_s and which is cheaper than the use of the standard CEM on the extended uniform grid $\tilde{\Omega}_s$.

Consider the covariance matrix R of the random vector $\mathrm{Z}(s_j^{(i)})$, $s_j^{(i)} \in \Omega_s$, written in the block matrix form:

$$R = \begin{bmatrix} R_{0,0} & R_{0,1} & R_{0,2} & \cdots & R_{0,N-1} \\ R_{1,0} & R_{1,1} & R_{1,2} & \cdots & R_{1,N-1} \\ R_{2,0} & R_{2,1} & R_{2,2} & \cdots & R_{2,N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{N-1,0} & R_{N-1,1} & R_{N-1,2} & \cdots & R_{N-1,N-1} \end{bmatrix}_{N\ell \times N\ell}, \tag{3}$$

where each block matrix $R_{i,k}$ is defined as

$$R_{i,k} = \left[r(|s_j^{(i)} - s_l^{(k)}|) \right]_{1 \le j,l \le \ell}. \tag{4}$$

Now note that, by construction,

$$R_{i,j} = \begin{cases} R_{k,l} & \text{if } \; j - i = l - k, \\ R_{k,l}^T & \text{if } \; j - i = k - l. \end{cases} \tag{5}$$

Property (5) implies that the covariance matrix R from (3) can be uniquely determined by its first block row and hence it is symmetric and block Toeplitz, having identical blocks along diagonals. Then R can be rewritten as

$$R = \begin{bmatrix} R_{0,0} & R_{0,1} & R_{0,2} & \cdots & R_{0,N-1} \\ R_{0,1}^T & R_{0,0} & R_{0,1} & \cdots & R_{0,N-2} \\ R_{0,2}^T & R_{0,1}^T & R_{0,0} & \cdots & R_{0,N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{0,N-1}^T & R_{0,N-2}^T & R_{0,N-3}^T & \cdots & R_{0,0} \end{bmatrix}. \tag{6}$$

We now illustrate how CEM [8–10] can be extended so that its new version, BCEM, is applicable to the symmetric block-Toeplitz matrix R from (6). To this end, we embed R in the $m\ell \times m\ell$ symmetric block-Toeplitz matrix C for some even integer $m \ge 2N$:

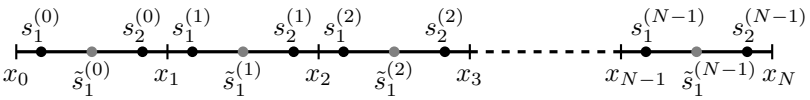

FIG. 3: 1D uniform grid $\Omega_r = \{x_0, \dots, x_N\} \in \Omega = [x_0, x_N]$. Black circles represent the locations of points $s_j^{(i)} \in \Omega_s$, and the combination of black and gray circles correspond to the uniform grid $\tilde{\Omega}_s$.

$$C = \begin{bmatrix} C_0 & C_1 & \cdots & C_{m-1} \\ C_{m-1} & C_0 & \cdots & C_{m-2} \\ \vdots & \vdots & \ddots & \vdots \\ C_1 & C_2 & \cdots & C_0 \end{bmatrix}, \tag{7}$$

where

$$C_k = \left[r(g(|s_i^{(0)} - s_j^{(k)}|)) \right]_{1 \le i,j \le \ell} \tag{8}$$

and

$$g(x) = \begin{cases} x & \text{if } x < mh/2, \\ mh - x & \text{if } x \ge mh/2. \end{cases} \tag{9}$$

Note that $C_i = R_{0,i}$ for $0 \le i \le N-1$ and that C is the covariance matrix for $\mathrm{Z}(x)$ defined in the circular manner on the grid $\Omega_s^E = \bigcup_{i=0}^{m-1} S_i \subset \Omega^E = [x_0, x_m]$, where $x_m = x_0 + mh$ and S_i are defined in the same way as before.

It is not difficult to see that the matrix C has the following properties:

$$C_0 = C_0^T, \tag{10}$$

$$C_k = C_{m-k}^T, \text{ for } 1 \le k \le \frac{m}{2}. \tag{11}$$

The properties (10) and (11) imply that C is a symmetric block-circulant matrix.

Let F_B be the tensor product of a one-dimensional discrete Fourier matrix F_m^1 of order m and an identity matrix I_ℓ of size $\ell \times \ell$:

$$F_B = F_m^1 \otimes \mathrm{I}_\ell = \begin{bmatrix} \mathrm{I}_\ell & \mathrm{I}_\ell & \mathrm{I}_\ell & \cdots & \mathrm{I}_\ell \\ \omega_0 \mathrm{I}_\ell & \omega_1 \mathrm{I}_\ell & \omega_2 \mathrm{I}_\ell & \cdots & \omega_{m-1} \mathrm{I}_\ell \\ \omega_0^2 \mathrm{I}_\ell & \omega_1^2 \mathrm{I}_\ell & \omega_2^2 \mathrm{I}_\ell & \cdots & \omega_{m-1}^2 \mathrm{I}_\ell \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega_0^{m-1} \mathrm{I}_\ell & \omega_1^{m-1} \mathrm{I}_\ell & \omega_2^{m-1} \mathrm{I}_\ell & \cdots & \omega_{m-1}^{m-1} \mathrm{I}_\ell \end{bmatrix}.$$

The matrix C is unitarily block diagonalizable by F_B [10, 19], i.e., there exists $\ell \times \ell$ matrices Λ_k, $k = 0, 1, \ldots, m-1$, such that

$$C = \frac{1}{m} F_B \Lambda F_B^H, \quad \text{where } \Lambda = \begin{bmatrix} \Lambda_0 & 0 & \cdots & 0 \\ 0 & \Lambda_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Lambda_{m-1} \end{bmatrix}. \tag{12}$$

Here H denotes the conjugate transpose. Similarly to the eigenvalue decomposition of a symmetric circulant matrix whose eigenvalues can be calculated by performing a discrete Fourier transform of its first row (or column), the block matrices on the diagonal of Λ can be computed as

$$[C_0 \quad C_1 \cdots C_{m-1}] F_B = [\Lambda_0 \quad \Lambda_1 \cdots \Lambda_{m-1}], \tag{13}$$

or in the component-wise form,

$$\left[C_0^{i,j} \quad C_1^{i,j} \cdots C_{m-1}^{i,j} \right] F_m^1 = \left[\Lambda_0^{i,j} \quad \Lambda_1^{i,j} \cdots \Lambda_{m-1}^{i,j} \right], \text{ where } 1 \le i, j \le \ell. \tag{14}$$

Since the block circulant matrix C is real and symmetric, Λ_k are Hermitian. Furthermore, all the diagonal elements of Λ_k are equal. Therefore, only $\ell(\ell+1)/2 - (\ell-1)$ applications of F_m^1 are required for computing Λ.

Remark 1. *Consider a uniform grid, i.e., a block-regular grid with the size of a regular grid being a multiple of the number of blocks or, in other words, the points in each S_i being uniformly located. Then BCEM is applicable on the uniform grid (recall that CEM works on regular grids only). Since the covariance depends on the distance between points only, the block-circulant matrix C on the uniform grid satisfies the relationship*

$$C_k^{a,b} = C_k^{c,d} \ \text{ if } \ |a-b| = |c-d|.$$

Consequently, Λ_k in (13) are Toeplitz and the number of 1D FFT F_m^1 to compute distinctive values of Λ is equal to ℓ. Thus, in BCEM the block-circulant matrix can be diagonalized by using ℓ FFTs of order m followed by using a Cholesky decomposition of the block-diagonal matrix Λ, whose block entries are of size $\ell \times \ell$. For small ℓ, the overall computational cost is dominated by $\mathcal{O}(\ell m \log_2 m)$. On the other hand, the complexity of CEM is dominated by FFTs of order $m\ell$, which gives the overall cost $\mathcal{O}(m\ell \log_2 m\ell)$. Hence, BCEM can outperform CEM on the uniform grid, where both CEM and BCEM use the same covariance matrix (see also Remark 5).

The symmetricity of C also guarantees the spectral decomposition

$$\Lambda_k = U_k D_k U_k^H, \tag{15}$$

where U_k is unitary and D_k is a real-valued diagonal matrix. The following proposition implies that Λ from (12) can be decomposed with $m/2+1$ applications of the spectral decompositions (15).

Proposition 2.1. *The block-diagonal matrix Λ from (12) has the property*

$$\Lambda_k = \overline{\Lambda_{m-k}}, \ \text{for} \ \ 1 \le k \le \frac{m}{2}, \tag{16}$$

where the bar denotes the matrix with conjugate complex entries.

Proof. Let $\omega_n = \exp((2\pi n/m)i)$ be a root of unity. Then [see (11) and (14)]

$$\begin{aligned}\overline{\Lambda_{m-k}} &= C_0 + \overline{\omega_{m-k}} C_1 + \cdots + \overline{\omega_{m-k}}^{m-1} C_{m-1} \\ &= C_0 + \omega_k C_1 + \cdots + \omega_k^{m-1} C_{m-1} \\ &= \Lambda_k. \end{aligned}$$

□

It follows from (12) and (15) that C has the eigenvalue decomposition

$$C = \frac{1}{m} (F_B U) D (F_B U)^H, \tag{17}$$

where the unitary block-diagonal matrix U and the diagonal matrix D are of the form

$$U = \begin{bmatrix} U_0 & 0 & \cdots & 0 \\ 0 & U_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & U_{m-1} \end{bmatrix} \ \text{and} \ D = \begin{bmatrix} D_0 & 0 & \cdots & 0 \\ 0 & D_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_{m-1} \end{bmatrix}.$$

We note that C is positive semidefinite if and only if $D_k^{i,i} \ge 0$ for each $0 \le k \le m-1$ and $1 \le i \le \ell$.

Assume for the moment that all the eigenvalues of C are non-negative. Let two independent random vectors ξ_1 and ξ_2, each of size m, be normally distributed $\mathcal{N}(O, \mathrm{I}_m)$, i.e., $\mathrm{E}[\xi_i \xi_j^T] = \delta_{ij}\mathrm{I}_m$, where δ_{ij} denotes the Kronecker delta. Set $\eta = U(D/m)^{1/2}(\xi_1 + i\xi_2)$. Then the real and imaginary parts of the vector $\zeta := F_B\eta$ give two independent random vectors ζ_1 and ζ_2 that are both distributed as $N(0, C)$. Since R is embedded in C, the corresponding parts of ζ_1 and ζ_2 are distributed as $\mathcal{N}(O, R)$. Note that the matrix-vector multiplication $F_B\eta$ can be calculated component-wise by ℓ applications of F_m^1.

The algorithm described above depends on positive semidefiniteness of the symmetric block-circulant matrix C. The sufficient conditions for symmetric circulant matrices to have all non-negative eigenvalues were developed for the 1D case in [9] and [8]. Here we extend these conditions to the symmetric block-circulant matrix C from (27). To this end, we introduce a uniform grid $\tilde{\Omega}_s$ such that $\Omega_s \subset \tilde{\Omega}_s$ and consider the covariance matrix $\tilde{R}$ defined on $\tilde{\Omega}_s$. As Ω_s is a subset of $\tilde{\Omega}_s$, R is a submatrix of the matrix $\tilde{R}$. Let a uniform grid $\tilde{\Omega}_s^E$ contain all points of $\Omega_s^E = \bigcup_{i=0}^{m-1} S_i$. Then the symmetric block-circulant matrix C is a submatrix of a symmetric circulant matrix $\tilde{C}$:

$$\tilde{C}^{i,j} = [r(g(|x_i - x_j|))], \tag{18}$$

where the function $g(x)$ is as in (9) and $x_i, x_j \in \tilde{\Omega}_s^E$. Therefore there exists an injection matrix P^T such that

$$C = P^T \tilde{C} P. \tag{19}$$

An injection matrix can be built by eliminating rows of the identity matrix, which correspond to points not in Ω_s^E. For instance,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

is an injection matrix from $\{x_1, x_2, x_3, x_4, x_5\}$ to $\{x_1, x_3, x_5\}$. The relationship (19) leads to the following proposition.

Proposition 2.2. *If $\tilde{C}$ is positive semidefinite, then so is C.*

When the circulant matrix $\tilde{C}$ fails to be positive semidefinite, Wood and Chen [8] suggested to increase the size of $\tilde{C}$ until it becomes positive semidefinite (the so-called padding technique). From the relationship (19) between C and $\tilde{C}$, the same strategy can be used for the matrix C. That is, increase m until C becomes positive semidefinite. Therefore the number of blocks m which is required for C to be positive semidefinite depends on the grid size of the uniform grid $\tilde{\Omega}_s^E$, not on the number of points in Ω_s. Thus the number of paddings needed for BCEM is the same as for CEM (see also Remark 3).

3. MULTIDIMENSIONAL BCEM

In the previous section we illustrated the idea of BCEM in the simpler setting of 1D space. In this section we present multidimensional BCEM, the computational complexity of which is discussed in the next section.

We start with introducing the notation which largely follows [10]. Let $\mathbb{Z}^d$ be the set of d vectors with non-negative integer components, and $\mathbf{0}$ and $\mathbf{1}$ be the d-dimensional vectors with all components equal to 0 and 1, respectively. For $\mathbf{i} = (\mathbf{i}[1], \ldots, \mathbf{i}[d])^T$, $\mathbf{j} = (\mathbf{j}[1], \ldots, \mathbf{j}[d])^T \in \mathbb{Z}^d$, we define addition in $\mathbb{Z}^d$:

$$\mathbf{i} + \mathbf{j} = (\mathbf{i}[1] + \mathbf{j}[1], \ldots, \mathbf{i}[d] + \mathbf{j}[d])^T,$$

and also the product of elements of $\mathbf{i}$:

$$\overline{\mathbf{i}} = \prod_{k=1}^{d} \mathbf{i}[k].$$

For any $\mathbf{j} \in \mathbb{Z}^d$, all components of which are strictly positive, we define the set $\mathcal{I}(\mathbf{j})$:

$$\mathcal{I}(\mathbf{j}) = \{\mathbf{i} \in \mathbb{Z}^d : 0 \le \mathbf{i}[k] \le \mathbf{j}[k] - 1 \text{ for all } 1 \le k \le d\}. \tag{20}$$

Note that the cardinality of $\mathcal{I}(\mathbf{j})$ is equal to $\overline{\mathbf{j}}$.

Introduce a d-dimensional rectangular parallelepiped,

$$\Omega = \prod_{i=1}^{d} [x_{\mathbf{0}}[i], x_{\mathbf{N}}[i]] \subset \mathbb{R}^d, \tag{21}$$

where $\mathbf{N} = (\mathbf{N}[1], \dots, \mathbf{N}[d])^T \in \mathbb{Z}^d$ and the vector $\mathbf{h}$ with the components $\mathbf{h}[i] = (x_{\mathbf{N}}[i] - x_{\mathbf{0}}[i])/\mathbf{N}[i]$. Further, points $x_{\mathbf{i}_k} = (x_{\mathbf{i}_k}[1], \dots, x_{\mathbf{i}_k}[d])^T$ with $x_{\mathbf{i}_k}[j] = x_{\mathbf{0}}[j] + \mathbf{i}_k[j]\mathbf{h}[j]$ form a regular grid $\Omega_r = \{x_{\mathbf{i}_0}, \dots, x_{\mathbf{i}_{\overline{\mathbf{N}+\mathbf{1}}-1}}\}$ on the rectangular parallelepiped Ω (see Fig. 4). The domain Ω in (21) can be divided into d-dimensional rectangular parallelepipeds as

$$\Omega = \bigcup_{\mathbf{j}_k \in \mathcal{I}(\mathbf{N})} \Omega_{\mathbf{j}_k}, \tag{22}$$

where

$$\Omega_{\mathbf{j}_k} = \prod_{i=1}^{d} [x_{\mathbf{j}_k}[i], x_{\mathbf{j}_k}[i] + \mathbf{h}[i]],\ \mathbf{j}_k \in \mathcal{I}(\mathbf{N}). \tag{23}$$

For the purpose of algorithm development, we use a lexicographic ordering of $\mathbf{j}_k$ with respect to k, i.e., row after row and layer after layer (see Fig. 4).

Consider a stationary Gaussian random field $\mathrm{Z}(x)$, $x \in \mathbb{R}^d$, with zero mean and covariance function $r(x)$. We assume that the problem at hand is such that we need to sample $\mathrm{Z}(x)$ at the nodes $s_i^{(\mathbf{j}_k)}$ defined as follows (see the examples in the Introduction and also Fig. 4):

$$s_j^{(\mathbf{j}_k)} = x_{\mathbf{j}_k} + \delta_j, \tag{24}$$

where $\delta_j = (\delta_j[1], \dots, \delta_j[d])^T$ with $0 \le \delta_j[i] < \mathbf{h}[i]$ for $1 \le i \le d$. Here ℓ is the number of sampling points in each subdomain $\Omega_{\mathbf{j}_k}$. That is, in each $\Omega_{\mathbf{j}_k}$ the points from the set $S_{\mathbf{j}_k} = \{s_1^{(\mathbf{j}_k)}, \dots, s_\ell^{(\mathbf{j}_k)}\}$ are distributed according to the same pattern for all $\mathbf{j}_k \in \mathcal{I}(\mathbf{N})$. Denote the grid

$$\Omega_s = \bigcup_{\mathbf{j}_k \in \mathcal{I}(\mathbf{N})} \Omega_{\mathbf{j}_k}.$$

Note that δ_j are independent of the index vector $\mathbf{j}_k$. The covariance matrix R of $Z(s_i^{(\mathbf{j}_k)})$, $s_i^{(\mathbf{j}_k)} \in \Omega_s$, is block-Toeplitz. In the one-dimensional case it consists of non-Toeplitz blocks of order ℓ (see Section 2). In the d-dimensional case with $d > 1$, it consists of blocks which all have the properties of a correlation matrix in the $d-1$ dimensional space. We emphasize that the matrix R is not Toeplitz and hence CEM is not directly applicable here.

Analogously to CEM, in order to build a block-circulant matrix, we consider an extended domain $\Omega^{\mathrm{E}} = \prod_{i=1}^{d} [x_{\mathbf{0}}[i], x_{\mathbf{m}}[i]]$, where $\mathbf{m} = (\mathbf{m}[1], \dots, \mathbf{m}[d])^T$ with $\mathbf{m}[i] \ge 2\mathbf{N}[i]$ and $x_{\mathbf{m}}[i] = x_{\mathbf{0}}[i] + \mathbf{m}[i]\mathbf{h}[i]$ for $1 \le i \le d$. Figure 4 shows an example of the computation domain Ω with $\mathbf{N} = (4,4)^T$ and the extended domain Ω^E with $\mathbf{m} = (8,8)^T$. Vectors $\mathbf{j}_k \in \mathcal{I}(\overline{\mathbf{m}+\mathbf{1}})$ form the extended regular grid $\Omega_{\mathrm{r}}^E = \{x_{\mathbf{j}_k} = (x_{\mathbf{j}_k}[1], \dots, x_{\mathbf{j}_k}[d])^T \mid \mathbf{j}_k \in \mathcal{I}(\mathbf{m}+\mathbf{1})\} \subset \Omega^E$. There are $\overline{\mathbf{m}+\mathbf{1}}$ regular grid points in the set Ω_{r}^E. The parallelepiped Ω^E can be divided into d-dimensional small parallelepipeds as (see also Fig. 4)

$$\Omega^E = \bigcup_{\mathbf{j}_k \in \mathcal{I}(\mathbf{m})} \Omega_{\mathbf{j}_k}, \tag{25}$$

where $\Omega_{\mathbf{j}_k}$, $\mathbf{j}_k \in \mathcal{I}(\mathbf{m})$, are as in (23).

We now describe BCEM in the d-dimensional case, which is applicable to our block-Toeplitz covariance matrix R. In contrast to the 1D setting, where the covariance function is always even because of its symmetry, further classification of covariance functions is needed in higher dimensional cases. We say that r is *component-wise even* in the ith coordinate if

$$r(x[1], \dots, -x[i], \dots, x[d]) = r(x[1], \dots, x[i] \dots, x[d]) \tag{26}$$

for all $x \in \mathbf{R}^d$; otherwise, we say that r is *uneven component-wise* in some coordinates.

For simplicity of the exposition, let us assume for now that $r(x)$ is component-wise, even in all coordinates. We discuss a modification of BCEM in the uneven case in Remark 2.

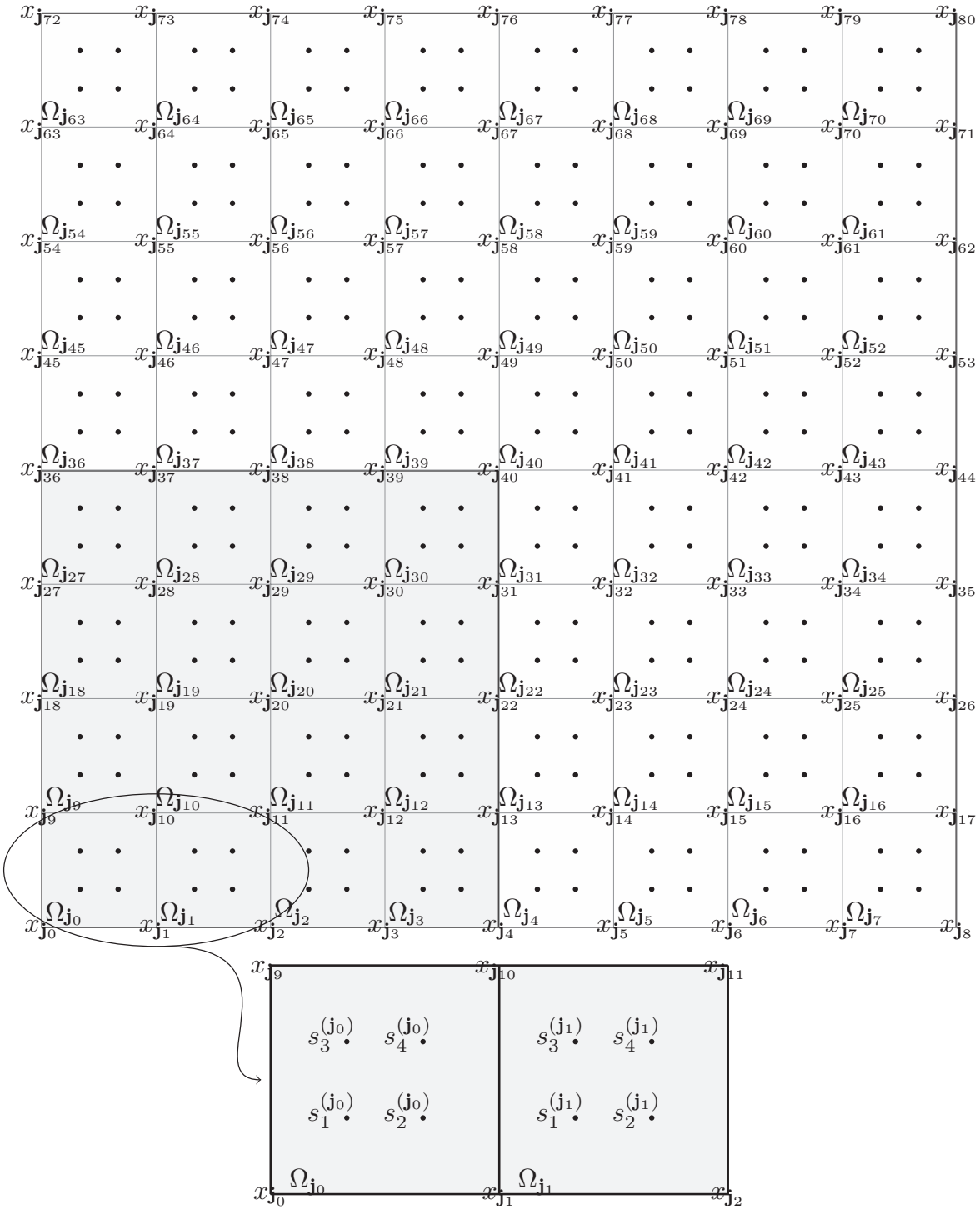

FIG. 4: The 2D uniform grid $\Omega_r^E = \{x_{\mathbf{j}_0} = x_{\mathbf{0}}, x_{\mathbf{j}_1}, \ldots, x_{\mathbf{j}_{\overline{\mathbf{m+1}}-1}} = x_{\mathbf{m}}\}$ on the rectangle $\Omega^E = [x_{\mathbf{0}}[1], x_{\mathbf{m}}[1]] \times [x_{\mathbf{0}}[2], x_{\mathbf{m}}[2]]$ for $\mathbf{m} = (8,8)^T$. Nodes $s_j^{(\mathbf{j}_k)}$ of the block-regular grid Ω_s^E are represented by black circles. The shaded rectangle corresponds to the computation (i.e., the domain of interest for the problem at hand) domain $\Omega = [x_{\mathbf{0}}[1], x_{\mathbf{N}}[1]] \times [x_{\mathbf{0}}[2], x_{\mathbf{N}}[2]]$ for $\mathbf{N} = (4,4)^T$.

We first build the block-circulant embedding of the block-Toeplitz matrix R. Consider the block-circulant matrix C of the form

$$C = [C_{\mathbf{j}_a\mathbf{j}_b}]_{0\le a,b\le \overline{\mathbf{m}}-1}\,, \tag{27}$$

where the (i,j)th element of a block $C_{\mathbf{j}_a\mathbf{j}_b}$ is equal to

$$C^{i,j}_{\mathbf{j}_a\mathbf{j}_b} = r(g_{\mathbf{m}}(s_i^{(\mathbf{j}_a)} - s_j^{(\mathbf{j}_b)})), \tag{28}$$

and the vector function $g_{\mathbf{m}}(x) = (g^1_{\mathbf{m}}(x),\dots,g^d_{\mathbf{m}}(x))^T$, $x \in \{x \in \mathbf{R}^d : 0 \le |x[i]| < \mathbf{m}[i]\mathbf{h}[i], i = 1,\dots,d\}$ is defined by

$$g^i_{\mathbf{m}}(x) = \begin{cases} x[i], & \text{if } |x[i]| < \mathbf{m}[i]\mathbf{h}[i]/2, \\ x[i] - \mathbf{m}[i]\mathbf{h}[i], & \text{if } \mathbf{m}[i]\mathbf{h}[i]/2 < x[i] < \mathbf{m}[i]\mathbf{h}[i], \\ x[i] + \mathbf{m}[i]\mathbf{h}[i], & \text{if } \mathbf{m}[i]\mathbf{h}[i]/2 < -x[i] < \mathbf{m}[i]\mathbf{h}[i]. \end{cases} \tag{29}$$

The first block row of the block-circulant matrix C is an $\ell \times \ell\overline{\mathbf{m}}$ matrix C_f of the form

$$C_f = \begin{bmatrix} C_{\mathbf{j}_0\mathbf{j}_0} & C_{\mathbf{j}_0\mathbf{j}_1} & \cdots & C_{\mathbf{j}_0\mathbf{j}_{\overline{\mathbf{m}}-1}} \end{bmatrix}. \tag{30}$$

Also, note that the covariance matrix R can be expressed via blocks $C_{\mathbf{j}_a\mathbf{j}_b}$ as

$$R = [C_{\mathbf{j}_a\mathbf{j}_b}]_{\mathbf{j}_a,\mathbf{j}_b\in\mathcal{I}(\mathbf{N})}. \tag{31}$$

The block-circulant matrix C is block diagonalizable by a block discrete Fourier transform (BDFT) matrix, $F_B = F^d_{\mathbf{m}} \otimes I_\ell$, where $F^d_{\mathbf{m}}$ is a d-dimensional DFT matrix. That is, we have $C = (1/\overline{\mathbf{m}})F_B\Lambda F_B^H$, where $\Lambda = \mathrm{diag}(\Lambda_0,\dots,\Lambda_{\mathbf{m}})$. The blocks on the diagonal of Λ can be found by simply taking BDFT of the first block row [10, 19]. Furthermore, using the fact that F_B is the tensor product involving the identity matrix, we derive the following component-wise computation:

$$[\Lambda^{i,j}_0 \cdots \Lambda^{i,j}_{\overline{\mathbf{m}}-1}] = \mathrm{FFT}_d([C^{i,j}_{\mathbf{j}_0\mathbf{j}_0} \cdots C^{i,j}_{\mathbf{j}_0\mathbf{j}_{\overline{\mathbf{m}}-1}}]), \tag{32}$$

where $1 \le i,j \le \ell$ and FFT_d is the d-dimensional FFT. Note that instead of FFT_1 we will write FFT.

Due to the fact that Λ is Hermitian and all diagonal entries of Λ_k are the same, the required number of FFT_d of size $\mathbf{m}$ (which is equivalent to FFT of order $\overline{\mathbf{m}}$) in (32) is $\ell(\ell+1)/2 - (\ell-1)$.

If Λ is positive-definite, the Cholesky decomposition $\Lambda = LL^H$ exists, where L is a block-diagonal matrix, with each block being a lower triangular matrix. Then we obtain the decomposition $C = (1/\overline{\mathbf{m}})F_BL(F_BL)^H$.

As in the one-dimensional case (see Section 2), let $\xi = \xi_1 + i\xi_2$ be a complex-valued random vector of order $\overline{\mathbf{m}}$ with ξ_1 and ξ_2 being real, normal random vectors such that $\mathrm{E}[\xi_i] = 0$ and $\mathrm{E}[\xi_i\xi_j^T] = \delta_{ij}I$. Set $\widetilde{L} = (1/\overline{\mathbf{m}})^{1/2}L$ and $\eta = \widetilde{L}\xi$. Multiplying the square root of C by ξ, we obtain the complex-valued vector

$$F_B\eta = \zeta = \zeta_1 + i\zeta_2, \tag{33}$$

with the properties $\mathrm{E}[\zeta_1\zeta_1^T] = \mathrm{E}[\zeta_2\zeta_2^T] = C$, and ζ_1 and ζ_2 are independent. Using tensor-product properties of F_B, ζ can be computed in the component-wise manner:

$$[\zeta[i] \quad \zeta[i+\ell]\dots\zeta[i+(\overline{\mathbf{m}}-1)\ell] = \mathrm{FFT}_d([\eta[i] \quad \eta[i+\ell]\dots\eta[i+(\overline{\mathbf{m}}-1)\ell]) \tag{34}$$

for $1 \le i \le \ell$.

To summarize, the new BCEM can be presented in algorithmic form as follows:

Algorithm 1: Block-circulant embedding method (BCEM)

Given $N \in \mathbb{Z}^d, x_{\mathbf{0}} \in \Omega$, and strictly positive valued vector $\mathbf{h} \in \mathbf{R}^d$,

Step 1. Choose a vector $\mathbf{m} \in \mathbb{Z}^d$ such that $\mathbf{m}[i] \geq 2\mathbf{N}[i]$ for all $1 \leq i \leq d$.

Step 2. Compute the first block row of the circulant matrix C as described in (27)–(30).

Step 3. Compute the block-diagonal matrix $\Lambda = \mathrm{diag}(\Lambda_0, \cdots, \Lambda_{\overline{\mathbf{m}}-1})$ using (32).

Step 4. Compute the square root of Λ applying Cholesky decompositions to diagonal blocks of Λ:

$$\Lambda = LL^H, \tag{35}$$

where L is a block-diagonal matrix with lower triangular block of order ℓ.

Step 5. If the Cholesky decomposition fails in *Step 4*, increase $\mathbf{m}[i]$ by one or more and go to *Step 2*.

Step 6. Compute $\widetilde{L} = (1/\overline{\mathbf{m}})^{1/2} L$.

Step 7. Generate a random complex vector of dimension $\overline{\mathbf{m}}\ell$, $\xi = \xi_1 + i\xi_2$, with two independent vectors ξ_1 and ξ_2 being $\mathcal{N}(0, I_{\overline{\mathbf{m}}\ell})$. Compute $\eta = \widetilde{L}\xi$.

Step 8. Compute $z = (\zeta[1], \ldots, \zeta[\overline{\mathbf{m}}\ell])^T$ by applying FFT_d ℓ times as in (34).

This algorithm works for all lengths $\mathbf{m}[i]$, $1 \leq i \leq d$, even and odd, when the covariance function $r(x)$ is component-wise even in all coordinates. But, for efficiency purposes, it is preferable to choose $\mathbf{m}$ in Step 1 so that each of its components is a power of 2 as the algorithm relies on FFT.

Note that if $\ell = 1$, then Ω_r is regular, C is circular, Λ becomes diagonal, instead of block diagonal, and Algorithm 1 degenerates to the standard CEM.

Remark 2. *Algorithm 1 is applicable when the covariance function is component-wise even [see (26)]. Although the covariance function is even by definition, i.e., $r(-x) = r(x)$, it could be component-wise uneven, e.g.,*

$$r(x) = \exp(-x^T Ax) \quad \textit{with} \quad A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix},\ x \in \mathbb{R}^2$$

is uneven. In this case, the matrix C defined by the vector function g_m in (29) usually does not have block-circulant structure because of conflicting definitions at the points x with $x[i] = \mathbf{m}[i]\mathbf{h}[i]/2$, if r is uneven in the ith coordinate. Two adjustments to make the CEM work in uneven cases were suggested in [8], which can be applied to BCEM by modifying Algorithm 1 as follows:

If r is uneven in the ith coordinate, either

(a) choose $\mathbf{m}[i]$ to be an odd integer, e.g., a power of three;

or

(b) still choose $\mathbf{m}[i]$ to be an even integer and define $C_{\mathbf{j}_k}$ using (28) and (29), except put $r(x) = 0$ for all x such that $|x[i]| = \mathbf{m}[i]\mathbf{h}[i]/2$ for some i.

In either case, the resulting matrix C has a block-circulant structure, and thus Algorithm 1 can be seamlessly extended to the uneven case with the aforementioned modifications.

Remark 3. *As mentioned earlier, the matrix C is often negative definite in practical applications of CEM and BCEM. Following [8], we increase the matrix C in Algorithm 1 (see its Step 5) until it becomes positive semidefinite (the padding technique). The padding technique is universal and usually efficient when the correlation length of a random field is in a range from small to medium relative to the size of a computational domain and the field is not too smooth. Otherwise, the use of the padding technique could be very expensive. There are two recently developed alternatives to padding (a cutoff of the circulant matrix [13, 14] and smoothing window circulant embedding [15]), which can deal with the problem of negative definiteness of circulant matrices effectively. The techniques from [13–15] are as applicable to BCEM as they are for CEM.*

The equispaced FFT is highly parallelizable in high dimensions, and its highly scalable implementations are proposed in [20, 21]. This could be beneficial in the standard CEM because its computation is dominated by the FFT. Still equipped with the parallelism of the FFT, BCEM can be further parallelized in a natural way because the applications of FFT_d in Step 3 and Step 8 of Algorithm 1 can be performed separately and simultaneously. Moreover, block-diagonal matrix operations in Step 4 and Step 7 can be performed separately and simultaneously. Therefore, the overall BCEM algorithm contains two-level parallelism, giving us a significant advantage over the standard CEM.

As we see in the next section, BCEM can be faster than CEM, both in taking square roots of the corresponding circulant matrices (performed, of course, only once per the whole Monte Carlo simulation) and in sampling the random field required in each Monte Carlo run. The latter is usually more important in Monte Carlo-type simulations.

4. COMPUTATIONAL COMPLEXITY OF BCEM

In this section we analyze the computational complexity of BCEM. To this end, we use the same convention as in Golub and Van Loan [22] for counting the number of floating point operations: $5m\log_2 m$ flops for FFT of size m and $n^3/3$ flops for the Cholesky decomposition of a matrix of order n.

Step 3 of Algorithm 1 is the initial factorization of the block-circulant matrix C by taking BDFT of its first block row, which can be computed using the ordinary DFT in (32) at the cost

$$\text{cost}_1 = \left(\frac{\ell(\ell+1)}{2} - (\ell-1)\right)(5\overline{\mathbf{m}}\log_2\overline{\mathbf{m}})\ \text{flops}. \tag{36}$$

Here we took into account that each Λ_k is Hermitian and its diagonal elements have the same value.

In Step 4, the square root operation on the block-diagonal matrix Λ with $\overline{\mathbf{m}}$ blocks of order ℓ can be performed on each block separately using the Cholesky decomposition method. In Proposition 2.1, we proved in the one-dimensional case that Λ has pairs of complex conjugate blocks, Λ_k and Λ_{m-k}, which allows us to compute the square root of Λ_k and use its complex conjugate as a square root of its complex conjugate pair Λ_{m-k}. This is based on the periodicity and conjugate symmetry of FFT. Hence, Proposition 2.1 can be extended to the higher-dimensional cases. Then the matrix Λ can be decomposed at the cost

$$\text{cost}_2 = \prod_{i=1}^{d}\left(\frac{\mathbf{m}[i]}{2}+1\right)\frac{\ell^3}{3}\ \text{flops}. \tag{37}$$

Remark 4. *Note that if the nodes of $S_{\mathbf{j}_k}$ are regularly (uniformly) distributed in $\Omega_{\mathbf{j}_k}$ for all $\mathbf{j}_k \in I(\mathbf{m})$, which is often the case in applications (see, e.g., Example 1.1), then all blocks on the diagonal of Λ are block Toeplitz (Toeplitz). Toeplitz matrix and block-Toeplitz matrix can be decomposed using Schur's algorithm [23] and block Schur's algorithm [24], respectively, which have $O(\ell^2)$ complexity as opposed to $O(\ell^3)$ for the standard Cholesky decomposition. Making use of Schur's algorithms can reduce the cost of Algorithm 1.*

In Step 1, computing a realization of ζ requires block-diagonal matrix-vector multiplication $\widetilde{L}\xi$, and ℓ applications of FFT of order $\overline{\mathbf{m}}$ in (34) at the cost

$$\text{cost}_3 = \ell^2\overline{\mathbf{m}}\ \text{flops} \tag{38}$$

and

$$\text{cost}_4 = \ell(5\overline{\mathbf{m}}\log_2\overline{\mathbf{m}})\ \text{flops}, \tag{39}$$

respectively.

To conclude, the cost of BCEM is $\mathcal{O}(\ell^3\overline{\mathbf{m}} + \ell^2\overline{\mathbf{m}}\log_2\overline{\mathbf{m}})$ flops. In practical applications of BCEM (see examples in the Introduction) the size of blocks ℓ is relatively small while the number of blocks $\overline{\mathbf{m}}$ is large. Recall that BCEM is designed for block-regular grids Ω_s. Its main computational advantage in comparison with CEM (which is designed for regular grids) comes from the fact that the use of CEM in the case of simulations on a block-regular grid Ω_s requires regularization of Ω_s, i.e., adding a significant number of extra nodes which BCEM does not need. Hence BCEM works on a grid with a smaller number of nodes than CEM and needs to generate random vectors ζ of smaller size than CEM (and hence makes fewer calls to a random number generator to sample ξ).

Remark 5. *It can be shown that the use of BCEM on a regular grid split in blocks of size ℓ can be more effective in sampling the random field than CEM but it is computationally more expensive in the matrix decomposition than CEM. The latter can be overcome by exploiting the fact that BCEM is parallelizable in comparison with CEM. Thus BCEM can be more effective than CEM, even in the case of regular grids for which CEM is designed.*

We now compare computational the complexity of BCEM and CEM using the three examples from the Introduction and the following exponential covariance function [cf. (28)]:

$$r(\mathbf{x}) = \sigma^2 \exp\left(-\frac{\|\mathbf{x}\|_1}{0.3}\right), \tag{40}$$

where $\|\cdot\|_1$ means L_1 norm. We note that the circulant matrix C [cf. (30), (28)] of the size $\mathbf{m} = 2\mathbf{N}$ formed by (40) is always positive definite (see, e.g., [9]). This means, in particular, that Step 1 (i.e., padding) of Algorithm 1 is not needed in this case. For simplicity, we consider the domain Ω to be the unit square in the examples.

Example 4.1. *Triangular finite element with a quadrature point located at the barycenter of the triangle.*

In Example 1.1 (see Fig. 1), each rectangular block contains nine uniform grid nodes. Hence the order of the circulant matrix used by CEM is $9\overline{\mathbf{m}}$, where $\mathbf{m} = 2\mathbf{N}$ and $\overline{\mathbf{m}}$ is the number of rectangular blocks in the extended domain Ω^{E}. Then the matrix decomposition cost for CEM is $45\overline{\mathbf{m}}\log_2 9\overline{\mathbf{m}}$ flops, and generation of each realization of the random field requires another $45\overline{\mathbf{m}}\log_2 9\overline{\mathbf{m}}$ flops.

Here BCEM uses only two points in each rectangular block, so the order of the block-circulant matrix is $2\overline{\mathbf{m}}$. Substituting $\ell = 2$ into (36) and (37), the total matrix decomposition cost for BCEM is $10\overline{\mathbf{m}}\log_2\overline{\mathbf{m}} + (\mathbf{m}[1]/2 + 1)(\mathbf{m}[2]/2 + 1)(8/3)$ flops. Each realization of the random field is generated at the cost of $4\overline{\mathbf{m}} + 10\overline{\mathbf{m}}\log_2\overline{\mathbf{m}}$ flops [see (38) and (39)].

Figure 5 shows the floating point operations required by the two algorithms. One can see that BCEM is more effective in both procedures and that the complexity of BCEM grows at roughly the same rate as for CEM. Compared to CEM, BCEM reduces the matrix decomposition cost and the generation cost approximately in 2.5 times and 4 times, respectively. The improvement in computational efficiency is due to the fact that BCEM works with just 2/9 of nodes that CEM uses to build the circulant matrix. This also means that BCEM requires 4.5 times less memory than CEM.

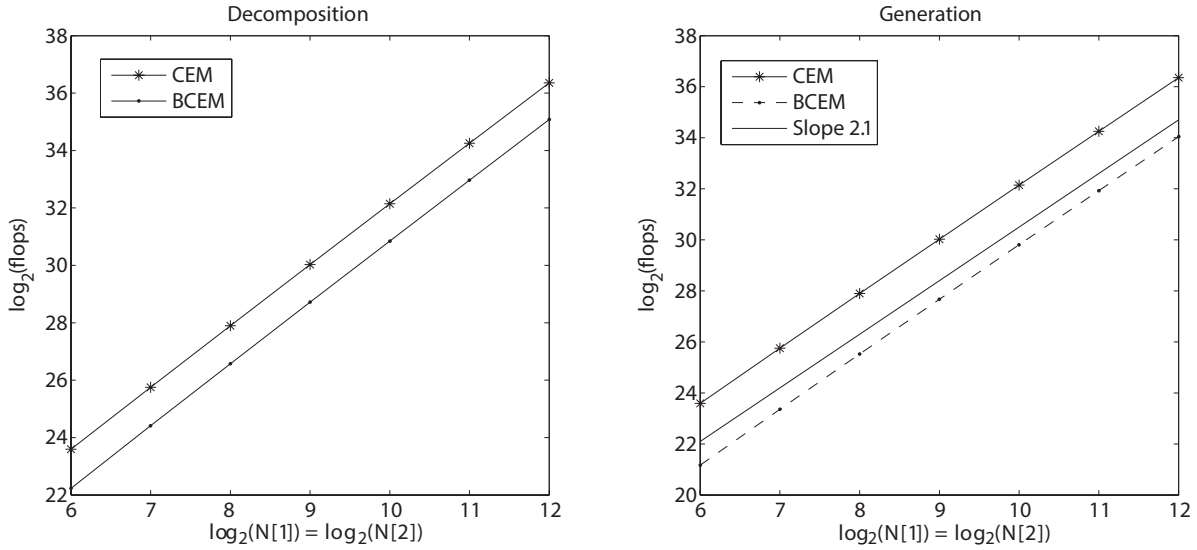

FIG. 5: The floating point operations required for the matrix decomposition and random field generation stages of CEM and BCEM in Example 4.1.

To compare the performance of BCEM and CEM further, we generated samples of the random field by these two methods on an Intel Xeon E5-2450, 96GB RAM computer using MATLAB R2014a. Figure 6 shows the average computational time of generation of a single realization of the random field by both methods and how it increases with increasing $\mathbf{N}$. Table 2 gives the CPU time and the speedup in generating a random vector using BCEM against those using CEM. For both methods, the CPU time increases with increase of $\mathbf{N}$ at about the same rate as the theoretical rate shown in Fig. 5 (right). We see that BCEM is about 4.3–4.5 times faster than CEM, which is close to the theoretical cost estimation in Fig. 5.

Example 4.2. *Cell-centered finite volume discretization in multilevel Monte Carlo (MLMC) computation.*

In Example 1.2, BCEM uses 5 out of 16 of the uniform nodes required for CEM in each individual block to generate random variables located at the centers of both the fine and coarse cells. That is, CEM should generate random variables at the extra 11 nodes that are not used in the finite-volume discretization and are not used by BCEM. Then memory requirement for CEM and BCEM is $16\overline{\mathbf{m}}$ and $5\overline{\mathbf{m}}$, respectively, which makes BCEM more attractive when the number of blocks is large.

Whereas the matrix decomposition and sampling costs in CEM both require $80\overline{\mathbf{m}}\log_2 16\overline{\mathbf{m}}$ flops, the computational costs of the matrix decomposition and sampling in BCEM are $55\overline{\mathbf{m}}\log_2\overline{\mathbf{m}}+(\mathbf{m}[1]/2+1)(\mathbf{m}[2]/2+1)(125/3)$ flops and $25\overline{\mathbf{m}}+25\overline{\mathbf{m}}\log_2\overline{\mathbf{m}}$ flops, respectively [see (36)–(39) with $\ell=5$]. Note that the total costs are dominated by $\overline{\mathbf{m}}\log_2\overline{\mathbf{m}}$. Hence, for large $\overline{\mathbf{m}}$, the ratio of the matrix decomposition in CEM to one in BCEM is close to

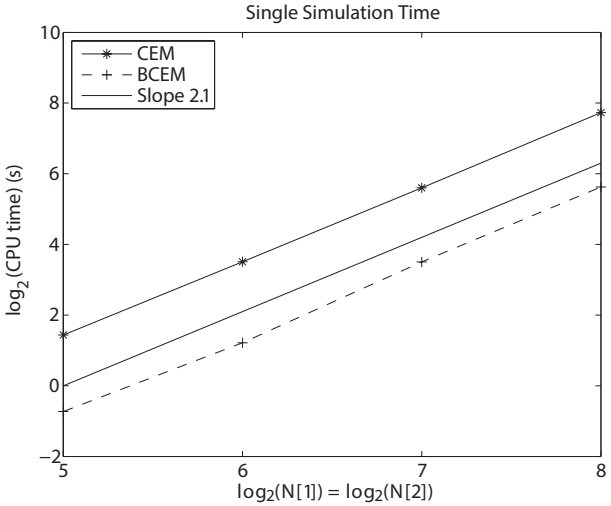

FIG. 6: Average time required to simulate a single realization of the random field in Example 4.1 computed using 1000 independent samples.

TABLE 2: Average time required to simulate a single realization of the random field in Example 4.1 computed using 1000 independent samples

	CPU Time (s)		
N	**BCEM**	**CEM**	**Speedup**
32	0.60	2.71	4.5
64	2.32	11.39	4.9
128	11.34	48.55	4.3
256	49.37	212.21	4.3

$80/55 \approx 1.45$. For the sample generation cost, the ratio is close to $80/25 \approx 3.2$. These theoretical computational costs are shown in Fig. 7.

Figure 8 gives the CPU times for the random field generation. We see that the actual computational cost increases with increase of $\mathbf{N}$, similarly to the theoretical one as in Fig. 7. Table 3 demonstrates that BCEM is nearly 3 times faster than CEM as we expected from Table 1 and Fig. 7. Also note that BCEM is highly parallelizable, so the computation cost can be further reduced using parallel algorithms.

Example 4.3. *Conditional random field generation on block-regular grids.*

Consider the conditional random field generation based on CEM introduced earlier in Example 1.3. For our experiments, we use the same block-regular grid as in Example 4.1 and choose synthetic 10 observation data. The coordinates of 10 points and observed values are provided in Table 4.

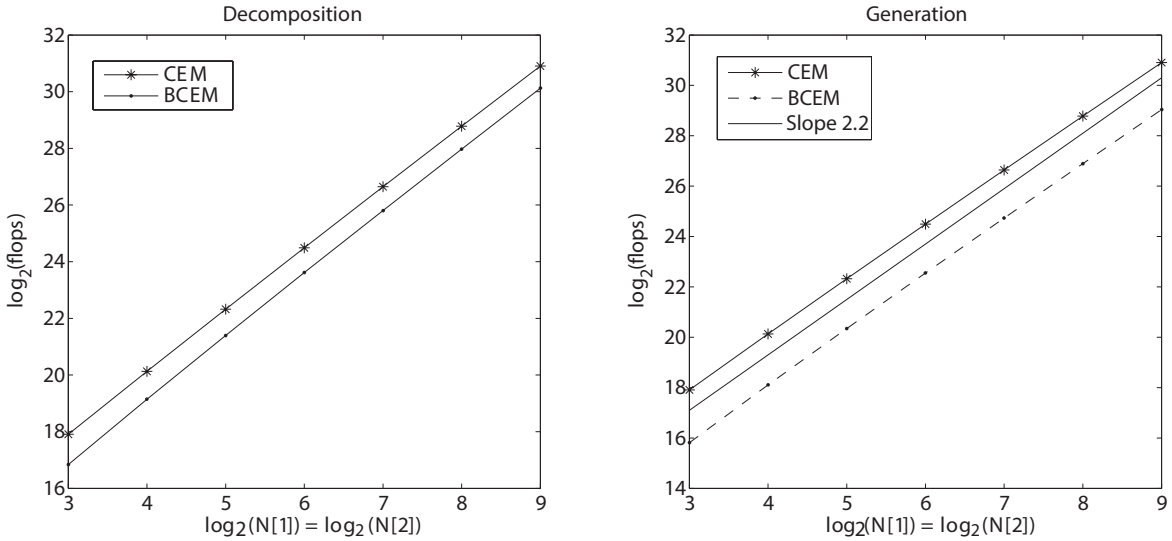

FIG. 7: The floating point operations required for the matrix decomposition and random field generation stages of CEM and BCEM in Example 4.2.

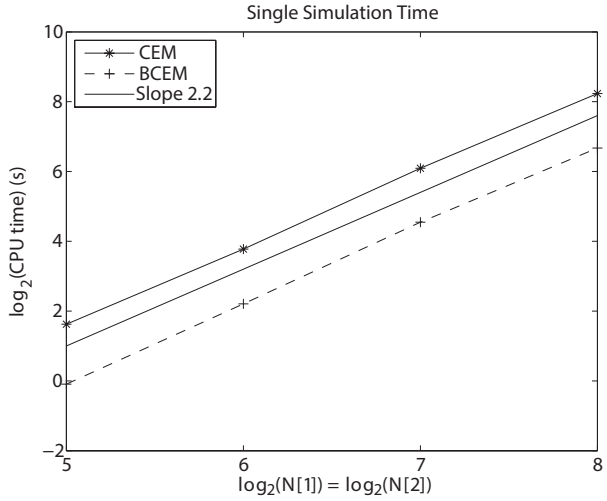

FIG. 8: Average time required to simulate a single realization of the random field in Example 4.2 computed using 1000 independent samples.

TABLE 3: Average time required to simulate a single realization of the random field in Example 4.2 computed using 1000 independent samples

N	CPU time (s)		
	CEM	**BCEM**	**Speedup**
32	0.94	3.08	3.3
64	4.62	13.71	3.0
128	23.37	68.28	2.9
256	101.82	301.24	3.0

TABLE 4: Cartesian coordinates of 10 points in the domain $\Omega = [0,\ 1]^2$ and synthetic observation values, z

$x[1]$	$x[2]$	z	$x[1]$	$x[2]$	z
0.2512	0.7264	0.6746	0.1400	0.1922	1.2083
0.5363	0.2909	4.7737	0.2682	0.6228	1.2606
0.7349	0.5463	2.2704	0.4017	0.1244	–3.2401
0.3480	0.9252	0.6082	0.4705	0.2330	–2.3023
0.5128	0.2914	5.1420	0.9608	0.4613	–1.0330

Recall that decomposition of the matrix R in (1) is computationally dominated by the cost of forming K in (2), which requires $\mathcal{O}(n_2 n_1 \log n_1)$ flops, where n_1 is the size of a (block) circulant matrix and n_2 is the number of observations. Note that $n_2 \ll n_1$ and n_2 = 10 in this example.

In order to compare efficiency, we measured the CPU times (in seconds) obtained by forming the matrices K using the two algorithms. One can see from Fig. 9 and Table 5 that BCEM achieves a speedup factor of 6 over CEM in forming the matrices K. We also note that BCEM requires 4.5 times less memory than CEM.

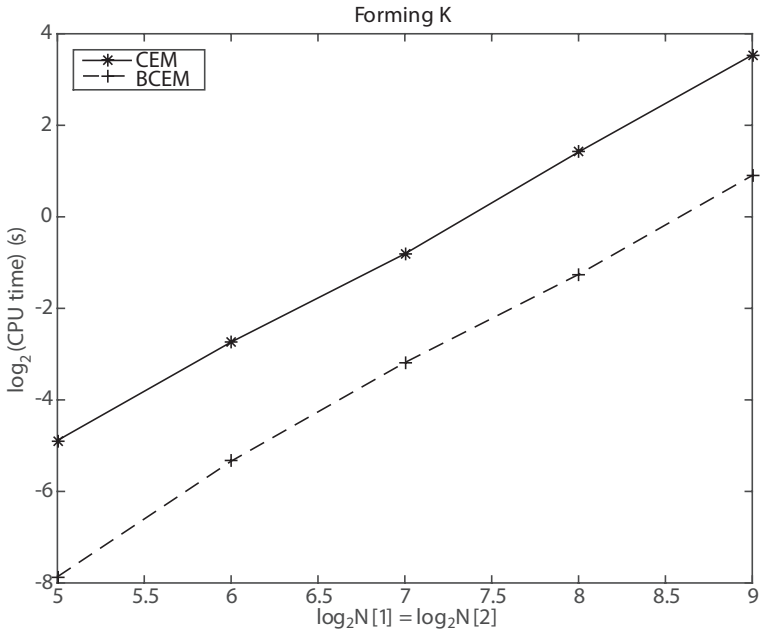

FIG. 9: CPU times for forming the matrix K.

TABLE 5: CPU time Average time required to simulate a single realization of the random field in Example 4.2 computed using 1000 independent samples

	CPU time (s)		
N	**CEM**	**BCEM**	**Speedup**
32	0.0338	0.0043	7.9
64	0.1500	0.0247	6.0
128	0.5710	0.1100	5.1
256	2.6700	0.4180	6.3
512	11.6000	1.8700	6.2

We have compared BCEM and CEM on the 2D examples here. It is not difficult to see (cf. Table 1) that in 3D cases BCEM can outperform CEM even more dramatically.

Remark 6. *The MATLAB codes for BCEM used for Examples 4.1–4.3 are available at https://github.com/parkmh/bcempaper.*

ACKNOWLEDEGMENT

This work was partially supported by the EPSRC through Grant No. EP/K031430/1.

REFERENCES

1. Delhomme, J. P., Spatial variability and uncertainty in groundwater flow parameters: A geostatistical approach, *Water Resour. Res.*, 15(2):269–280, 1979.
2. Gel, Y., Raftery, A. E., Gneiting, T., Tebaldi, C., Nychka, D., Briggs, W., Roulston, M. S., and Berrocal, V. J., Calibrated probabilistic mesoscale weather field forecasting: The geostatistical output perturbation method, *J. Am. Stat. Assoc.*, 99:575–590, 2004.
3. Marheineke, N. and Wegner, R., Fiber dynamics in turbulent flows: General modeling framework, *SIAM J. Appl. Math.*, 66:1703–1726, 2006.
4. Skordos, A. A. and Sutcliffe, M. P. F., Stochastic simulation of woven composites forming, *Compos. Sci. Technol.*, 68:283-296, 2008.
5. Zhang, F., Cosson, B., Comas-Cardona, S., and Binetruy, C., Efficient stochastic simulation approach for RTM process with random fibrous permeability, *Compos. Sci. Technol.*, 71:1478–1485, 2010.
6. Ghanem, R. G. and Spanos, P. D., *Stochastic Finite Elements: A Spectral Approach*, Springer, New York, 1991.
7. Le Maître, O. and Knio, O., *Spectral Methods for Uncertainty Quantification with Applications to Computational Fluid Dynamics*, Springer, New York, 2010.
8. Wood, A. T. A. and Chan, G., Simulation of stationary Gaussian processes in $[0, 1]^d$, *J. Comput. Graph. Stat.*, 3:409–432, 1994.
9. Dietrich, C. R. and Newsam, G. N., Fast and exact simulation of stationary gaussian processes through circulant embedding of the covariance matrix, *SIAM J. Sci. Comput.*, 18:1088–1107, 1997.
10. Chan, G. and Wood, A. T. A., Simulation of stationary Gaussian vector fields, *Stat. Comput.*, 9(4):265–268, 1999.
11. Park, M. and Cliffe, K. A., Conditional multilevel Monte Carlo simulation of groundwater flow in the Culebra Dolomite at the Waste Isolation Pilot Plant (WIPP) site, arXiv:1402.5257, 2014.
12. Verleye, B., Nuyens, D., Walbran, A., and Gan, M., Uncertainty quantification in liquid composite moulding processes, in *Proc. of FPCM11 Conf.*, Auckland, Australia, pp. 265–271, July, 2012.
13. Stein, M., Fast and exact simulation of fractional Brownian surfaces, *J. Comput. Graph. Stat.*, 11:587–599, 2002.

14. Gneiting, T., Ševčíková, H., Percival, D. B., Schlather, M., and Jiang, Y., Fast and exact simulation of large Gaussian lattice systems in $\mathbb{R}^2$: Exploring the limits, *J. Comput. Graph. Stat.*, 15:1–19, 2006.
15. Helgason, H., Pipiras, V., and Abry, P., Smoothing windows for the synthesis of Gaussian stationary random fields using circulant matrix embedding, *J. Comput. Graph. Stat.*, 23:616–635, 2014.
16. Cliffe, K. A., Giles, M. B., Scheichl, R., and Teckentrup, A. L., Multilevel Monte Carlo methods and applications to elliptic PDEs, *Comput. Visual. Sci.*, 14:3–15, 2011.
17. Barth, A., Schwab, C., and Zollinger, N., Multi-level Monte Carlo finite element method for elliptic PDEs with stochastic coefficients, *Numer. Math.*, 119:123–161, 2011.
18. Dietrich, C. R. and Newsam, G. N., A fast and exact method for multidimensional Gaussian stochastic simulations: Extension to realizations conditioned on direct and indirect measurements, *Water Resour. Res.*, 32(6):1643–1652, 1996.
19. Davis, P. J., *Circulant Matrices*, 2nd ed., AMS Chelsea Publishing, Providence, RI, 1979.
20. Frigo, M. and Johnson, S. G., FFTW: An adaptive software architecture for the FFT, in *Proc. of IEEE Int. Conf. on Acoustics*, Vol. 3 of Speech and Signal Processing (ICASSP), pp. 1381–1384, 1998.
21. Eleftheriou, M., Fitch, B., Rayshubskiy, A., Ward, T., and Germain, R., Performance measurements of the 3D FFT on the Blue Gene/L supercomputer, in *Euro-Par 2005 Parallel Processing: 11th Int. Euro-Par Conf.*, Lisbon, Portugal, Aug. 30–Sept. 2, 2005, J. Cunha and P. Medeiros (Eds.), Vol. 3648, Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 795–803, 2005.
22. Golub, G. H. and van Loan, C. F., *Matrix Computations*, 3rd ed., John Hopkins University Press, Baltimore, MD, 1996.
23. Stewart, M., Cholesky factorization of semidefinite Toeplitz matrices, *Linear Algebra Appl.*, 254:497–525, 1997.
24. Gallivan, K. A. and Thirumalai, S., High performance algorithm for Toeplitz and block Toeplitz matrices, *Linear Algebra Appl.*, 241-243:343–388, 1996.