Exploiting Low Rank Covariance Structures for Computing High-Dimensional Normal and Student-t Probabilities

Jian Cao, Marc G. Genton, David E. Keyes [1] and George M. Turkiyyah [2]

October 25, 2019

Abstract

We present a preconditioned Monte Carlo method for computing high-dimensional multivariate normal and Student-t probabilities arising in spatial statistics. The approach combines a tile-low-rank representation of covariance matrices with a block-reordering scheme for efficient Quasi-Monte Carlo simulation. The tile-low-rank representation decomposes the high-dimensional problem into many diagonal-block-size problems and low-rank connections. The block-reordering scheme reorders between and within the diagonal blocks to reduce the impact of integration variables from right to left, thus improving the Monte Carlo convergence rate. Simulations up to dimension 65,536 suggest that the new method can improve the run time by an order of magnitude compared with the non-reordered tile-low-rank Quasi-Monte Carlo method and two orders of magnitude compared with the dense Quasi-Monte Carlo method. Our method also forms a strong substitute for the approximate conditioning methods as a more robust estimation with error guarantees. An application study is provided to illustrate that the new computational method makes the maximum likelihood estimation feasible for high-dimensional skew-normal random fields.

Keywords: Adaptive cross approximation, Block reordering, Hierarchical matrix, Skew-normal random field, Tile-low-rank matrix.

[1] CEMSE Division, Extreme Computing Research Center, King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia.
E-mail: {jian.cao, marc.genton, david.keyes}@kaust.edu.sa
This research was supported by King Abdullah University of Science and Technology (KAUST).
[2] Department of Computer Science, American University of Beirut, Beirut, Lebanon.
E-mail: gt02@aub.edu.lb

1 Introduction

Data used for spatial statistical modeling usually display a certain degree of skewness and heavy-tailedness (e.g., Kim et al. (2004)), for which there is comprehensive literature about skew-elliptical distributions (Genton, 2004; Azzalini and Capitanio, 2014). The majority of such constructed distributions can be written as selection distributions (Arellano-Valle et al., 2006), where the latent conditioning random vector, $\mathbf{U}$, belongs to a fixed set C. Therefore the probability density function of the observed random vector, $\mathbf{V}$, involves $P(\mathbf{U} \in C|\mathbf{V} = \mathbf{v})$, which is difficult to evaluate in high dimensions. The common choice for the density generating function of $(\mathbf{U}^T, \mathbf{V}^T)^T$ often corresponds to the multivariate normal (MVN) distribution or the multivariate Student-t (MVT) distribution. This paper proposes a method to compute the multivariate normal and Student-t probabilities in high dimensions that utilizes the low-rank feature of the spatial covariance matrix and a block-reordering scheme to increase the convergence rate.

Genton et al. (2018) improved the efficiency for computing the MVN probabilities in high dimensions by utilizing hierarchical matrices (Hackbusch, 2015) and proper Quasi-Monte Carlo (QMC) rules. Run times can be reduced by a factor of more than 20 for dimensions larger than 10^4. Cao et al. (2019) combined the hierarchical technique with the conditioning method from Trinh and Genz (2015), which further improved the efficiency by a factor of 10 to 15 but the method introduces approximations which do not provide error estimation. In fact, the univariate conditioning method corresponds to the QMC method with a size-one sampling rule computed at run time. In this paper, we further develop the QMC methods, seeking to reduce the needed QMC sample size through the block-reordering scheme introduced in Cao et al. (2019).

The prevalent algorithm for computing the MVN probabilities is based on the separation-of-variable (SOV) technique (Genz, 1992), which converts the integration region to the unit hypercube. The counterpart for MVT probabilities was later proposed in Genz and Bretz (1999) built on the definition in Equation (3) below. However, this MVT counterpart is less efficient due to the lack of optimized libraries for computing the univariate Student-t probabilities and

quantiles similar to those available for computing MVN probabilities and quantiles. Genz and Bretz (2002) provided a second algorithm for computing MVT probabilities, considering the n-dimensional MVT probability as a scale-mixture of n-dimensional MVN probabilities, shown in Equation (4a), based on which the MVT probabilities are computed as efficiently as the MVN probabilities. In this paper, we develop the low-rank versions of the QMC methods for MVT probabilities.

The remainder of this paper is structured as follows. In section 2, we introduce the SOV technique for MVN and MVT problems and describe dense QMC algorithms for both probabilities. In section 3, we compare two integration-oriented reordering schemes and study the impact on the ranks resulting from the block reordering scheme, which leads to the tile-low-rank (TLR, or block-low-rank) version of the QMC algorithms. In section 4, we compare the dense QMC method, the TLR QMC method, and the preconditioned TLR QMC method with a focus on high-dimensional MVN and MVT probabilities. In section 5, we estimate the parameters for simulated high-dimensional skew-normal random fields as well as fit the skew-normal model to a large wind speed dataset of Saudi Arabia as examples where the methods developed in this paper can be applied. Section 6 concludes the paper.

2 SOV for MVN and MVT Probabilities

The SOV technique transforms the integration region into the unit hypercube, where efficient QMC rules can improve the convergence rate. The SOV of MVN probabilities is based on the Cholesky factor of the covariance matrix (Genz, 1992) and this naturally leads to the second form of SOV for MVT probabilities (Genz and Bretz, 2002). The two forms of the MVT probabilities have been derived in Genz (1992) and Genz and Bretz (2002). In this paper, we summarize the derivations for completeness and clarification of notations.

2.1 SOV for MVN integrations

We denote an n-dimensional MVN probability with $\Phi_n(\mathbf{a}, \mathbf{b}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $(\mathbf{a}, \mathbf{b})$ defines a hyperrectangle-shaped integration region, $\boldsymbol{\mu}$ is the mean vector, and $\boldsymbol{\Sigma}$ is the covariance matrix. The MVN probability has the form:

$$\Phi_n(\mathbf{a}, \mathbf{b}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \int_{\mathbf{a}-\boldsymbol{\mu}}^{\mathbf{b}-\boldsymbol{\mu}} \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}\mathbf{x}^T \boldsymbol{\Sigma}^{-1} \mathbf{x}\right) \mathrm{d}\mathbf{x}. \tag{1}$$

Without loss of generality, we set $\boldsymbol{\mu} = \mathbf{0}$ and denote the n-dimensional MVN probability with $\boldsymbol{\Phi}_n(\mathbf{a}, \mathbf{b}; \boldsymbol{\Sigma})$. We use $\mathbf{C}$ to represent the lower Cholesky factor of $\boldsymbol{\Sigma}$ and c_{ij} to represent the element on the i-th row and j-th column of $\mathbf{C}$. Following the procedure in Genz (1992), we can transform $\Phi_n(\mathbf{a}, \mathbf{b}; \boldsymbol{\Sigma})$ into:

$$\Phi_n(\mathbf{a}, \mathbf{b}; \boldsymbol{\Sigma}) = (e_1 - d_1) \int_0^1 (e_2 - d_2) \cdots \int_0^1 (e_n - d_n) \int_0^1 \mathrm{d}\mathbf{w}, \tag{2}$$

where $d_i = \Phi\{(a_i - \sum_{j=1}^{i-1} c_{ij} y_j)/c_{ii}\}$, $e_i = \Phi\{(b_i - \sum_{j=1}^{i-1} c_{ij} y_j)/c_{ii}\}$, $y_j = \Phi^{-1}\{d_j + w_j(e_j - d_j)\}$, and $\Phi(\cdot)$ is the cumulative distribution function (cdf) of the standard normal distribution.

The integration region is transformed into $[0, 1]^n$ and efficient sampling rules can be applied to simulate $\mathbf{w}$, although the integrand is difficult to compute in parallel because d_i and e_i depend on $\{y_j, j = 1, \ldots, i-1\}$ while y_i depends on d_i and e_i. Only univariate standard normal probabilities and quantile functions are needed, which can be readily obtained with the high efficiency of scientific computing libraries, for example, the Intel MKL. The Cholesky factorization has a complexity of $O(n^3)$ but modern CPUs and libraries have been developed to handle matrices with more than 10,000 dimensions with ease.

We use 'mvn' to denote the integrand function of Equation (2), whose pseudocode was originally proposed in Genz (1992). Because the 'mvn' function is also the subroutine in other functions of this paper, we summarize it here in algorithm 2.1a. The algorithm returns P, the probability estimate from one sample and $\mathbf{y}$ whose coefficients are described in Equation (2). Keeping $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{C}$ unchanged, the mean and standard deviation of the outputs P from a

Algorithm 2.1a QMC for MVN probabilities

mvn($\mathbf{C}, \mathbf{a}, \mathbf{b}, \mathbf{w}$)
$n \leftarrow \dim(\mathbf{C})$, $s \leftarrow 0$, $\mathbf{y} \leftarrow \mathbf{0}$, and $P \leftarrow 1$
for $i = 1 : n$ **do**
    if $i > 1$ **then**
        $s \leftarrow \mathbf{C}(i, 1 : i-1)\mathbf{y}(1 : i-1)$
    end if
    $a' \leftarrow \frac{a_i - s}{C_{i,i}}$, and $b' \leftarrow \frac{b_i - s}{C_{i,i}}$
    $y_i \leftarrow \Phi^{-1}[w_i\{\Phi(b') - \Phi(a')\}]$
    $P \leftarrow P \cdot \{\Phi(b') - \Phi(a')\}$
end forreturn P and $\mathbf{y}$

set of well designed $\mathbf{w}$, usually conforming to a Quasi-Monte Carlo rule, form the probability and error estimates. In our implementation, we employ the Richtmyer Quasi-Monte Carlo rule (Richtmyer, 1951), where the batch number is usually much smaller than the batch size.

2.2 SOV for MVT integrations

We denote an n-dimensional MVT probability with $T_n(\mathbf{a}, \mathbf{b}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu)$, where ν is the degrees of freedom. Here, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ have the same meanings as in the MVN probability. To simplify the notations, $\boldsymbol{\mu}$ is again assumed to be $\mathbf{0}$. There are two common equivalent definitions for T_n, of which the first one is:

$$T_n(\mathbf{a}, \mathbf{b}; \boldsymbol{\Sigma}, \nu) = \frac{\Gamma(\frac{\nu+n}{2})}{\Gamma(\frac{\nu}{2})\sqrt{|\boldsymbol{\Sigma}|(\nu\pi)^n}} \int_{a_1}^{b_1} \cdots \int_{a_n}^{b_n} \left(1 + \frac{\mathbf{x}^T \boldsymbol{\Sigma}^{-1} \mathbf{x}}{\nu}\right)^{-\frac{\nu+n}{2}} \mathrm{d}\mathbf{x}, \tag{3}$$

where $\Gamma(\cdot)$ is the gamma function. Based on this definition, Genz and Bretz (1999) transformed the integration into the n-dimensional hypercube, where the inner integration limits depend on the outer integration variables. However, the integration needs to compute the cdf and the quantile function of the univariate Student-t distribution at each integration variable. A second equivalent form defines T_n as a scale mixture of the MVN probability, specifically:

$$T_n(\mathbf{a}, \mathbf{b}; \boldsymbol{\Sigma}, \nu) = \frac{2^{1-\frac{\nu}{2}}}{\Gamma(\frac{\nu}{2})} \int_0^\infty s^{\nu-1} e^{-s^2/2} \Phi_n\left(\frac{s\mathbf{a}}{\sqrt{\nu}}, \frac{s\mathbf{b}}{\sqrt{\nu}}; \boldsymbol{\Sigma}\right) \mathrm{d}s, \tag{4a}$$

$$= E\left[\Phi_n\left(\frac{S\mathbf{a}}{\sqrt{\nu}}, \frac{S\mathbf{b}}{\sqrt{\nu}}; \boldsymbol{\Sigma}\right)\right]. \tag{4b}$$

The density of a χ-distribution random variable, S, with degrees of freedom ν, is exactly $\frac{2^{1-\frac{\nu}{2}}}{\Gamma(\frac{\nu}{2})} s^{\nu-1} e^{-s^2/2}$, $s > 0$. Thus, $T_n(\mathbf{a}, \mathbf{b}; \mathbf{\Sigma}, \nu)$ can be also written as Equation (4b). The integrand boils down to the MVN probability discussed in the previous section. Hence, we can apply a Quasi-Monte Carlo rule in the $(n+1)$-dimensional hypercube to approximate this expectation, where only the cdf and the quantile function of the univariate standard normal distribution are involved. It is worth pointing out that considering T_n as a one-dimensional integration of Φ_n and applying quadrature is much more expensive than integrating directly in $(n+1)$ dimensions.

We describe the integrand functions based on the two SOV schemes in algorithm 2.2a and algorithm 2.2b, corresponding to Equation (3) and Equation (4a), respectively. Algorithm 2.2a

Algorithm 2.2a QMC for MVT probabilities based on Equation (3)

mvt$(\mathbf{C}, \mathbf{a}, \mathbf{b}, \nu, \mathbf{w})$
$n \leftarrow \dim(\mathbf{C})$, $s \leftarrow 0$, $ssq \leftarrow 0$, $\mathbf{y} \leftarrow \mathbf{0}$, and $P \leftarrow 1$
for $i = 1 : n$ **do**
    if $i > 1$ **then**
        $s \leftarrow \mathbf{C}(i, 1 : i-1)\mathbf{y}(1 : i-1)$
    end if
    $a' \leftarrow \frac{a_i - s}{\mathbf{C}_{i,i} \cdot \sqrt{\nu + ssq} \cdot (\nu + i)}$ and $b' \leftarrow \frac{b_i - s}{\mathbf{C}_{i,i} \cdot \sqrt{\nu + ssq} \cdot (\nu + i)}$
    $y_i \leftarrow T^{-1}_{\nu+i} \left[w_i \left\{ T_{\nu+i}(b') - T_{\nu+i}(a') \right\} + T_{\nu+i}(a') \right] \cdot \sqrt{\frac{\nu + ssq}{\nu + i}}$
    $P \leftarrow P \cdot \left\{ T_{\nu+i}(b') - T_{\nu+i}(a') \right\}$
    $ssq \leftarrow ssq + y_i^2$
end forreturn P

Algorithm 2.2b QMC for MVT probabilities based on Equation (4a)

mvt$(\mathbf{C}, \mathbf{a}, \mathbf{b}, \nu, w_0, \mathbf{w})$
$\mathbf{a}' \leftarrow \frac{\chi_\nu^{-1}(w_0)}{\sqrt{\nu}} \mathbf{a}$, $\mathbf{b}' \leftarrow \frac{\chi_\nu^{-1}(w_0)}{\sqrt{\nu}} \mathbf{b}$ **return** mvn$(\mathbf{C}, \mathbf{a}', \mathbf{b}', \mathbf{w})$

calls the univariate Student-t cdf and the quantile function with an increasing value of degrees of freedom at each iteration whereas algorithm 2.2b relies on $(w_0, \mathbf{w})$ from an $(n+1)$-dimensional Quasi-Monte Carlo rule and calls the 'mvn' kernel from algorithm 2.1a with the scaled integration limits. We use single-quoted 'mvn' and 'mvt' to denote the corresponding algorithms to distinguish them from the uppercase MVN and MVT used for multivariate normal and Student-t in this paper.

Table 1: Relative error and time of the three algorithms. 'mvt 1', 'mvt 2', and 'mvn' refer to algorithm 2.2a, algorithm 2.2b, and algorithm 2.1a. The covariance matrix is generated from a 2D exponential model, $\exp(-\|\mathbf{h}\|/\beta)$, where $\beta = 0.1$, based on n random points in the unit square. The lower integration limits are fixed at $-\infty$ and the upper limits are generated from $N(\mu_u, 1.5^2)$, where $\mu_u = \max(4, \min(7, \log_4 n))$. ν is set as 10 for the 'mvt' algorithms. The upper row is the average relative estimation error and the lower row is the average computation time over 20 iterations. All three algorithms have the same sample size of $N = 10^4$.

n	16	64	256	1,024	4,096
mvt 1	0.2%	1.2%	8.6%	9.2%	5.8%
	0.6s	*2.6s*	*10.9s*	*46.1s*	*223.1s*
mvt 2	0.0%	0.4%	2.8%	2.9%	1.5%
	0.0s	*0.0s*	*0.2s*	*2.2s*	*32.8s*
mvn	0.0%	0.3%	2.9%	2.7%	1.7%
	0.0s	*0.0s*	*0.2s*	*2.1s*	*37.6s*

A numerical comparison between algorithm 2.2a and algorithm 2.2b is shown in table 1. The counterpart for MVN probabilities (algorithm 2.1a) is included as a benchmark. The table indicates that the first definition as in Equation (3) leads to an implementation slower by one order of magnitude. Additionally, the convergence rate from Equation (3) is also worse than that from Equation (4a). Although the univariate Student-t cdf and quantile function are computed the same number of times as their standard normal counterparts, their computation takes much more time and probably produces lower accuracy due to the lack of optimized libraries. It is also worth noting that the change of the relative error horizontally in table 1 is affected by the changing true probability across the cells. Due to its performance advantage, we refer to algorithm 2.2b as the 'mvt' algorithm from this point on. It has negligible marginal complexity over the 'mvn' algorithm since the only additional step is scaling the integration limits. In fact, its time efficiency is even improved over the 'mvn' algorithm because of the scaling of the integration limits.

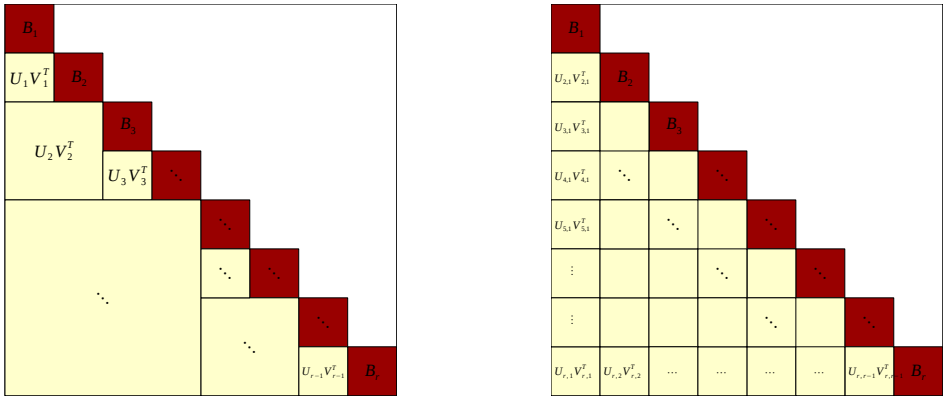


Figure 1: Structures of hierarchical (left) and tile-low-rank (right) matrices.

3 Low-rank Representation and Reordering for MVN and MVT Probabilities

3.1 Overview

For high-dimensional problems, Monte Carlo and quasi Monte Carlo methods are the only practical methods for computing the integrations of Equation (1) and Equation (3) in a general setting. The cost of these computations depends on the product of the number of Monte Carlo samples, N, needed to achieve a desired accuracy and the cost per Monte Carlo sample. Using the standard dense representation of covariance, the computational complexity for each Monte Carlo sample is $O(n^2)$ as can be seen in algorithm 2.1a and algorithm 2.2b. In contrast, the use of hierarchical covariance representations as shown in fig. 1 allows this complexity to be reduced to $O(kn \log n)$ (Genton et al., 2018) where k is a nominal local rank of the matrix blocks and n the overall problem dimension. Using nested bases in the hierarchical representation Boukaram et al. (2019), it is possible to reduce this cost further to an asymptotically optimal $O(kn)$.

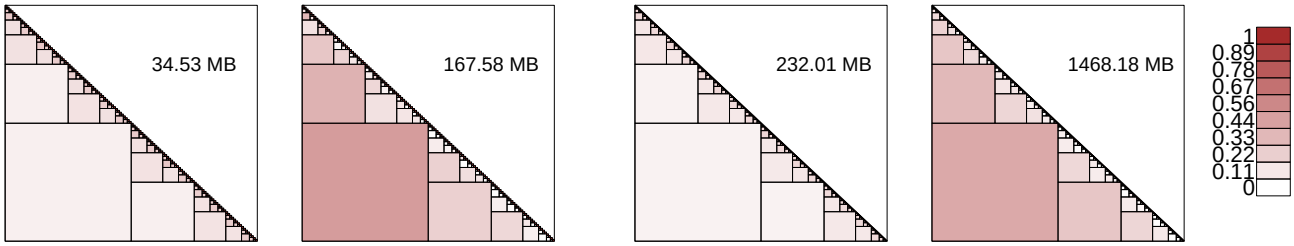
Small local ranks k in the hierarchical representation depend on the separability of the underlying geometry and are directly affected by the ordering of the underlying point set. When the row cluster and the column cluster of an off-diagonal matrix block are well separated spatially, the ranks of these blocks tend to be rather small, growing very weakly with the problem dimension n. When the geometry is a subset of $\mathbb{R}^2$ or $\mathbb{R}^3$, a space-filling curve or a spatial partitioning

method in combination with a space-filling curve may be used for indexing to keep the index distances reasonably consistent with the spatial distances. The point set is then further divided into blocks (clusters) according to these indices to build the hierarchical representation.

The optimal ordering for reducing the cost per Monte Carlo sample however is unfortunately generally not the optimal ordering for reducing the total number of samples N. A proper reordering scheme that takes into account the widths of the integration limits of the MVN and MVT probabilities can have a substantial effect on reducing the variance of the estimates, making the numerical methods far more effective relative to a default ordering Schervish (1984); Genz and Bretz (2009). Trinh and Genz (2015) analyzed ordrering heuristics and found that a univariate reordering scheme, that sorts the variables so that the outermost integration variables have the smallest expected values, significantly increased the estimation accuracy. This heuristic was more effective overall than more expensive bivariate reordering schemes that might further reduce the number of samples needed. In Cao et al. (2019), a block reordering scheme was proposed with the hierarchical matrix representations used in high dimensions. Specifically, within each diagonal block $\mathbf{B}_i$, univariate reordering was applied and the blocks were reordered based on their estimated probabilities using this univariate reordering scheme.

The important point here is that these reordering schemes shuffle the variables based on their integration limits to achieve better convergence for the integration, measured by the number of samples needed. They produce different orders from the geometry-oriented ordering obtained by spatial partitioning methods or space-filling curves. The reordering increases the local ranks k of the hierarchical representation making the per-sample computation more expensive.

In this paper, we seek a better middle ground between the geometry-oriented and the integration-oriented orderings by combining a block reordering scheme with the tile-low-rank representation of covariance illustrated in fig. 1. We also introduce the TLR versions of the QMC algorithms for computing MVN and MVT probabilities.



(a) n=16,384 (b) n=65,536

Figure 2: Increase in local rank and memory footprint of the Cholesky factor of a hierarchical matrix due to integration-oriented ordering. In each subfigure, the left panel is under Morton order while the right panel is under the block reordering. The diagonal block size is $\sqrt{n}$. The storage cost for the lower triangular part of the Cholesky factor is marked in each subfigure. The color for each block is coded by the logarithm of the rank-to-block-size ratio, linearly transformed into $(0, 1)$.

3.2 TLR as a practical representation for MVN and MVT

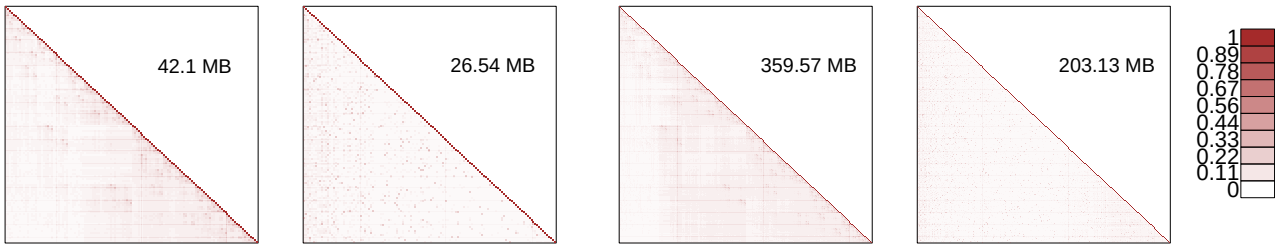
To show the effect of rank increase due to re-ordering, we consider an MVN computation experiment. We use Morton order (Samet, 1990) as the geometry-oriented ordering scheme for building an initial hierarchical covariance matrix, and examine the rank change under the integration-oriented block ordering scheme proposed in (Cao et al., 2019) for MVN probabilities using randomly generated integration limits. The reordering for MVT problems shares the same principle. Specifically, because the expectation of S from Equation (4a) is $\sqrt{2}\Gamma\{(\nu+1)/2\}/\Gamma(\nu/2)$, converging to $\sqrt{\nu}$ quickly as ν increases, Genz and Bretz (2002) proposed substituting S with $\sqrt{\nu}$ and the reordering becomes exactly the same as that for the MVN probability.

The matrices in this experiment for two different problem sizes are shown in fig. 2. The matrices are Cholesky factors of the covariance matrices built with the 2D exponential covariance model, $\exp(-\|\mathbf{h}\|/\beta)$, $\beta = 0.3$, based on a perturbed grid in the unit square as described in section 5.2 and the rank of each block is defined as the number of singular values above an absolute threshold of 10^{-2}.

Figure 2 shows the change in storage costs (indicative of rank increase) introduced to the hierarchical structure with a weak admissibility condition by the block reordering scheme. In the

weak admissibility hierarchical structure, every off-diagonal block touching the main diagonal is represented as $\mathbf{U}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are thin matrices. This representation is beneficial only if the ranks of the blocks are small. However, in practice, the ranks can grow substantially when the underlying geometry is arbitrarily reordered without taking into account spatial proximity. To illustrate the rank change, we list the memory costs and color-code the rank-to-block-size ratio under both a spatially-aware Morton order and random block reordering scheme in fig. 2.

We now compare the rank change caused by changing the ordering from the spatially-oriented strategy to an integration-oriented strategy in the TLR representation. fig. 3 shows the effect of reordering on the TLR Cholesky factor. The average rank of the off-diagonal blocks in the



(a) n=16,384 (b) n=65,536

Figure 3: Change in local rank and memory footprint of the Cholesky factor of TLR matrix due to integration-oriented ordering. In each subfigure, the left panel is under Morton order while the right panel is under the block reordering. The diagonal block size is $\sqrt{n}$. The storage cost for the lower triangular part of the Cholesky factor is marked in each subfigure. The color for each block is coded by the logarithm of the rank-to-block-size ratio, linearly transformed into $(0, 1)$.

TLR structure even decreased when applying the block reordering scheme, which shuffles the diagonal blocks. This is because the block reordering scheme only shuffles the off-diagonal blocks and does not affect the overall low-rank feature of the TLR covariance matrix. Additionally, if we remove the first i rows and columns of a Cholesky factor, it becomes the Cholesky factor of the Schur complement (Zhang, 2006) of the bottom-right block of the covariance matrix, which is also the covariance matrix for the $(i+1)$-th to the n-th variables, conditional on the first to the i-th variables. Under Morton order, the indices usually begin at the corner of the geometry and slowly moves into the center while under the block reordering scheme, blocks of variables

more centered in the geometry are likely to have smaller indices. If we condition on the first i variables and examine the overall correlation among the other $(n-i)$ variables, it is smaller if the first i variables are more centered. Hence, the overall magnitudes of the entries of the TLR Cholesky factor are smaller under the block reordering scheme. When two TLR matrices have similar low-rank behavior with singular values decrease equally fast, the one with smaller magnitude entries has lower ranks overall if truncated to an absolute accuracy level. Therefore, the TLR structure creates a synergy with the block reordering scheme.

The memory footprint of the TLR structure roughly increases at $O(n^{3/2})$ because the tile size is $m = \sqrt{n}$ and the average rank for the off-diagonal blocks only grows weakly with n. Although its asymptotic complexity is not optimal, it has proved sufficient for reasonably high dimensions because of its smaller constants. There are also two practical benefits compared with the weak admissibility hierarchical structure. First, fast approximation algorithms, for example, the adaptive cross approximation (ACA) (Bebendorf, 2011), can be more reliably applied under TLR due to its lower ranks. Second, the regularity of the flat structure of TLR benefits more directly from modern hardware architectures.

3.3 Reordering schemes and TLR factorizations

The block reordering scheme was proposed in Cao et al. (2019) and shown to improve the estimation accuracy of the conditioning method with lower complexity than the univariate or bivariate reordering scheme introduced in Trinh and Genz (2015). In this paper, we introduce a recursive version of the block reordering scheme computed during Cholesky factorization. The recursive block reordering can be also viewed as the block version of the univariate reordering scheme in Trinh and Genz (2015).

Algorithm 3.3a describes the original block reordering scheme proposed in Cao et al. (2019) while algorithm 3.3b is the recursive version that produces the Cholesky factor. We use $\mathbf{\Sigma}_{i,j}$ to represent the (i,j)-th size-m block of $\mathbf{\Sigma}$. Similar notations are also used for $\mathbf{a}$ and $\mathbf{b}$. When $i \neq j$,

$\mathbf{\Sigma}_{i,j}$ is stored in the low-rank format. The blue lines in algorithm 3.3b mark the matrix operations that are also in the block Cholesky factorization. If we ignore the cost for steps 5 and 9, the complexity of algorithm 3.3b is the same as the Cholesky factorization. Although the complexity for accurately computing Φ_m and the truncated expectations is high, the univariate conditioning method (Trinh and Genz, 2015), with a complexity of $O(m^3)$, can provide an estimate for both that is indicative enough. Algorithm 3.3a ignores the correlation between the m-dimensional blocks and also uses the univariate conditioning method for approximating Φ_m. Therefore, the block reordering scheme has a total complexity of $O(nm^2)$ but requires a succeeding Cholesky factorization while the recursive block reordering has additional complexity of $O(n^2m)$ over the Cholesky factorization but produces the Cholesky factor simultaneously.

Algorithm 3.3a Block reordering

1: bodr($\mathbf{\Sigma}, \mathbf{a}, \mathbf{b}, m$)
2: $r = n/m$
3: **for** $j = 1 : r$ **do**
4: $\quad p_l \approx \Phi_m(\mathbf{a}_l, \mathbf{b}_l; \mathbf{\Sigma}_{l,l})$
5: **end for**
6: **for** $j = 1 : r$ **do**
7: $\quad \tilde{j} = \text{argmin}_l(p_l), l = j, \ldots, r$
8: $\quad \mathbf{p}[j \rightleftharpoons \tilde{j}]$ and block-wise $\mathbf{\Sigma}[j \rightleftharpoons \tilde{j}, j \rightleftharpoons \tilde{j}]$, $\mathbf{a}[j \rightleftharpoons \tilde{j}]$, $\mathbf{b}[j \rightleftharpoons \tilde{j}]$
9: **end for**

The truncated product and subtraction operations, $\odot$ and $\ominus$, indicate the corresponding matrix operations which involve truncation to smaller ranks to maintain required accuracy. $\mathbf{\Sigma}_{i_1,j} \odot \mathbf{\Sigma}_{j_1,j}^T$ and $\mathbf{\Sigma}_{i,j} \odot \mathbf{\Sigma}_{j,j}^{-T}$ have complexities of $O(mk^2)$ and $O(m^2k)$ respectively, where m is the tile size and k is the local rank. The $\ominus$ operation uses ACA truncated at an absolute tolerance to keep the result low-rank. For the studies in section 4 and section 5, we set the tolerance to 10^{-5}. Prior to the TLR Cholesky factorization, we construct the TLR covariance matrix with ACA given the covariance kernel, the underlying geometry and the indices of variables. Therefore, the total memory needed for computing MVN and MVT probabilities is $O(kn^2/m)$.

Algorithm 3.3b Block reordering during Cholesky factorization

rbodr($\mathbf{\Sigma}, \mathbf{a}, \mathbf{b}, m$)
$r = n/m$
for $j = 1 : r$ **do**
  for $l = j : r$ **do**
    $p_l \approx \Phi_m(\mathbf{a}_l, \mathbf{b}_l; \mathbf{\Sigma}_{l,l})$
  end for
  $\tilde{j} = \text{argmin}_l(p_l), l = j, \ldots, r$
  Block-wise $\mathbf{\Sigma}[j \rightleftharpoons \tilde{j}, j \rightleftharpoons \tilde{j}]$, $\mathbf{a}[j \rightleftharpoons \tilde{j}]$, $\mathbf{b}[j \rightleftharpoons \tilde{j}]$
  $\mathbf{y}_j \approx E_m[\mathbf{Y} | \mathbf{Y} \sim N_m(\mathbf{0}, \mathbf{\Sigma}_{j,j}), \mathbf{Y} \in (\mathbf{a}_j, \mathbf{b}_j)]$
  $\mathbf{\Sigma}_{j,j} = \text{Cholesky}(\mathbf{\Sigma}_{j,j})$
  for $i = j + 1 : r$ **do**
    $\mathbf{\Sigma}_{i,j} = \mathbf{\Sigma}_{i,j} \odot \mathbf{\Sigma}_{j,j}^{-T}$
    $\mathbf{a}_i = \mathbf{a}_i - \mathbf{\Sigma}_{i,j} \odot \mathbf{y}_j$, $\mathbf{b}_i = \mathbf{b}_i - \mathbf{\Sigma}_{i,j} \odot \mathbf{y}_j$
  end for
  for $j_1 = j + 1 : r$ **do**
    for $i_1 = j + 1 : r$ **do**
      $\mathbf{\Sigma}_{i_1,j_1} = \mathbf{\Sigma}_{i_1,j_1} \ominus \mathbf{\Sigma}_{i_1,j} \odot \mathbf{\Sigma}_{j_1,j}^T$
    end for
  end for
end for

3.4 Preconditioned TLR QMC algorithms

Algorithm 3.4a and algorithm 3.4b describe the TLR versions of the 'mvn' and 'mvt' algorithms. To distinguish them from the dense 'mvn' and 'mvt' algorithms, we expand the storage structure of $\mathbf{C}$, the TLR Cholesky factor, as the interface of the TLR algorithms. The definitions of $\mathbf{B}_i$, $\mathbf{U}_i$, and $\mathbf{V}_i$ are shown in fig. 1, where $\mathbf{U}_i$ and $\mathbf{V}_i$ are indexed in column-major order.

Algorithm 3.4a TLR QMC for MVN probabilities

tlrmvn($\mathbf{B}, \mathbf{U}, \mathbf{V}, \mathbf{a}, \mathbf{b}, \mathbf{w}$)
$\mathbf{y} \leftarrow \mathbf{0}$, and $P \leftarrow 1$
for $i = 1 : r$ **do**
  if $i > 1$ **then**
    for $j = i : r$ **do**
      $\mathbf{\Delta} = \mathbf{U}_{j,i-1}(\mathbf{V}_{j,i-1}^T \mathbf{y}_{i-1})$
      $\mathbf{a}_j = \mathbf{a}_j - \mathbf{\Delta}$, $\mathbf{b}_j = \mathbf{b}_j - \mathbf{\Delta}$
    end for
  end if
  $(P', \mathbf{y}_i) \leftarrow \text{MVN}(\mathbf{B}_i, \mathbf{a}_i, \mathbf{b}_i, \mathbf{w}_i)$
  $P \leftarrow P \cdot P'$
end forreturn P

Algorithm 3.4b TLR QMC for MVT probabilities

1: tlrmvt($\mathbf{B}, \mathbf{U}, \mathbf{V}, \mathbf{a}, \mathbf{b}, \nu, w_0, \mathbf{w}$)

2: $\mathbf{a}' \leftarrow \frac{\chi_\nu^{-1}(w_0)}{\sqrt{\nu}}\mathbf{a}$, $\mathbf{b}' \leftarrow \frac{\chi_\nu^{-1}(w_0)}{\sqrt{\nu}}\mathbf{b}$ **return** TLRMVN($\mathbf{B}, \mathbf{U}, \mathbf{V}, \mathbf{a}', \mathbf{b}', \mathbf{w}$)

Similar to algorithm 3.3b, we use subscripts to represent the size-m segment of $\mathbf{a}$, $\mathbf{b}$, $\mathbf{y}$, and $\mathbf{w}$. The two algorithms compute the integrand given one sample $\mathbf{w}$ in the n-dimensional unit hypercube. In our implementation, the Richtmyer rule (Richtmyer, 1951) is employed for choosing $\mathbf{w}$. 'tlrmvn' is called by 'tlrmvt', where the additional inputs, ν and w_0, have the same meaning as those in algorithm 2.2b. The TLR structure reduces dense matrix-vector multiplication to low rank matrix-vector multiplication when factoring the correlation between blocks into the integration limits. The two algorithms can be either preconditioned by the block reordering or the recursive block reordering. We examine the performance of the TLR QMC algorithms in section 4.

4 Numerical Simulations

Table 2 gathers the performance of the dense (Genz, 1992) and the TLR QMC methods for computing MVN and MVT probabilities, measured on a workstation with 50 GB memory and 8 Xeon(R) E5-2670 CPUs. Methods are assessed over 20 simulated problems for each combination of problem dimension n and correlation strength β. The highest dimension in our experiment is 2^{16}. Higher dimensions should be still feasible for the preconditioned TLR QMC methods but the construction of the TLR Cholesky factor will be more time-consuming and a lower truncation level for the ACA algorithm is needed to guarantee the positive definiteness of the TLR covariance matrix. $\beta = 0.3$, 0.1, and 0.03 correspond to an effective range of 0.90, 0.30, and 0.09, the first of which is considered long given that the underlying geometry lies in the unit square. The tile size m for the TLR QMC methods is set as $\sqrt{n}$. The time for the block reordering preconditioner is listed within the time costs of the 'rtlrmvn' and 'rtlrmvt' methods while the time for constructing the covariance matrix and computing the Cholesky factor is not

included. The QMC sample size is set at $N = 10^4$ for the methods without the block reordering preconditioner while at $N = 10^3$ for the two preconditioned QMC methods. The sampling follows the Richtmyer rule in the unit hypercube. Table 2 shows that the preconditioned TLR QMC methods achieve a lower estimation error in most cases while reducing the time cost by one and two orders of magnitude compared with the TLR QMC methods without preconditioning

Table 2: Performance of the three methods under weak, medium, and strong correlations. 'mvn' and 'mvt' are the dense QMC methods, 'tlrmvn' and 'tlrmvt' are the TLR QMC methods, and 'rtlrmvn' and 'rtlrmvt' are the TLR QMC methods preconditioned with the block reordering scheme. The covariance matrix, integration limits, and the degrees of freedom are generated the same way as in table 1. The upper row is the average relative estimation error and the lower row is the average computation time over 20 replicates.

$\beta = 0.3$						
n	mvn	tlrmvn	rtlrmvn	mvt	tlrmvt	rtlrmvt
1,024	2.4% *2.1s*	2.5% *1.0s*	1.0% *0.1s*	2.3% *2.1s*	3.3% *1.1s*	1.6% *0.1s*
4,096	1.2% *45.0s*	1.2% *9.5s*	1.1% *0.9s*	1.6% *43.9s*	1.6% *8.9s*	1.2% *0.8s*
16,384	1.0% *1233.6s*	0.9% *60.2s*	0.8% *5.5s*	1.0% *1236.1s*	1.4% *58.4s*	1.3% *5.3s*
65,536	*NA*	2.2% *304.4s*	1.7% *31.4s*	*NA*	2.5% *301.4s*	1.7% *31.2s*
$\beta = 0.1$						
n	mvn	tlrmvn	rtlrmvn	mvt	tlrmvt	rtlrmvt
1,024	2.4% *2.0s*	2.3% *0.9s*	1.2% *0.1s*	2.6% *1.9s*	3.2% *0.9s*	1.6% *0.1s*
4,096	1.7% *40.2s*	1.8% *5.7s*	1.2% *0.5s*	1.8% *40.0s*	2.3% *5.9s*	1.5% *0.5s*
16,384	1.4% *1216.7s*	1.3% *43.8s*	0.9% *3.8s*	1.3% *1220.3s*	1.3% *44.6s*	1.0% *3.8s*
65,536	*NA*	3.8% *299.9s*	3.6% *29.4s*	*NA*	4.1% *285.6s*	2.4% *28.0s*
$\beta = 0.03$						
n	mvn	tlrmvn	rtlrmvn	mvt	tlrmvt	rtlrmvt
1,024	0.7% *1.9s*	0.6% *0.9s*	0.4% *0.1s*	0.9% *2.0s*	2.1% *0.9s*	0.8% *0.1s*
4,096	1.1% *39.5s*	1.2% *5.7s*	0.5% *0.4s*	1.1% *39.6s*	1.1% *5.8s*	0.7% *0.4s*
16,384	0.8% *1210.8s*	0.9% *36.7s*	0.4% *2.6s*	0.9% *1214.4s*	1.1% *37.2s*	0.6% *2.4s*
65,536	*NA*	4.5% *228.8s*	2.1% *18.8s*	*NA*	2.9% *233.6s*	1.7% *19.5s*

and the dense QMC methods respectively. It is worth noting that the relative error can be affected by the magnitude of the true probabilities. The average relative error decreases from $n = 1{,}024$ to $n = 4{,}096$ because the true probability increases due to our design for generating the upper integration limits. Cao et al. (2019) and Trinh and Genz (2015) generated their upper integration limits from $U(0, n)$ while Genton et al. (2018) simulated from a univariate Gaussian distribution, which is also used in this paper. Although both methods led to 'normal-ranged' true probabilities, the former method produced more uninformative integration variables, for example, the variables whose integration regions contain $(-10, 10)$, which made the problem less challenging since the uninformative variables can be ignored to simplify the problem to a much lower dimension. Intuitively, wider-spread integration limits work in favor of the block reordering scheme whereas, in contrast, any reordering scheme would become ineffective if all integration limits are equal. Since table 2 is based on problems with relatively concentrated integration limits, we expect even more accurate results for other simulated problems.

5 Application to Stochastic Generators

5.1 A skew-normal stochastic generator

Stochastic generators model the space-time dependence of the data in the framework of statistics and aim to reproduce the physical process that is usually emulated through a system of partial differential equations. The emulation of the system requires tens of variables and a very fine grid in the spatio-temporal domain, which is extremely time-and-storage demanding (Castruccio and Genton, 2016). For example, the Community Earth System Model (CESM) Large ENSemble project (LENS) required ten million CPU hours and more than four hundred terabytes of storage to emulate one initial condition (Jeong et al., 2018). Castruccio and Genton (2016) found statistical models could form efficient surrogates for reproducing the physical processes in climate science and concluded that extra model flexibilities would facilitate the modeling on a finer scale; see Castruccio and Genton (2018) for a recent account.

The MVN and MVT methods developed in this paper allow to consider more complexity in the construction of stochastic generators. A significant improvement in flexibility is to introduce skewness since the majority of the statistical models used nowadays are Gaussian-based, i.e. they rely on a symmetric distribution. Generally speaking, there are three ways of introducing skewness to an elliptical distribution, all of which involve the cdf of the distribution. The first is through reformulation, which multiplies the elliptical probability density function (pdf) by its cdf. The second method introduces skewness via selection. Assuming $(\mathbf{X}^T, \mathbf{Y}^T)^T$ have a joint multivariate elliptical distribution, $\mathbf{X}|\mathbf{Y} > \boldsymbol{\mu}$, where $\boldsymbol{\mu}$ is an n-dimensional vector, has a skew-elliptical distribution. Arellano-Valle and Genton (2010) studied a general class of skewed reformulations and introduced its link to the selection representation. The third method is defined by the stochastic representation, specifically, $\mathbf{Z} = \mathbf{X} + |\mathbf{Y}|$, where $\mathbf{X}$ and $\mathbf{Y}$ are two independent elliptical random vectors. Zhang and El-Shaarawi (2010) studied the skew-normal random field based on this construction assuming a general correlation structure for $\mathbf{Y}$, because of which a direct maximum likelihood estimation is almost impossible. Instead, $\mathbf{Y}$ was taken as a latent random variable and the EM algorithm was applied. In the M-step, the conditional expectations of $\mathbf{X}$ were computed through the Markov chain Monte Carlo method. Thus, the cost for maximizing the likelihood is expectedly high.

The three methods have equivalent forms in the one-dimensional case but extend differently into higher dimensions. The first method is flexible but provides little information on the underlying stochastic process. The second method has a clear underlying model and its pdf is usually more tractable than that from the third method but the choice for $\mathbf{Y}$ is usually not obvious, especially when $\mathbf{X}$ is in high dimensions. In the third method, the parameterization is usually more intuitive and the model can be also applied in spatial statistics as a random field. However, the pdf is a summation of a number of terms exponentially growing with the number of locations n, which renders the model difficult to scale. Weighing an intuitive stochastic representation against the pdf complexity, we modify the third construction method based on the $\mathscr{C}$ random

vector properties introduced in Arellano-Valle et al. (2002). A $\mathscr{C}$ random vector can be written as the Hadamard product of two independent random vectors, representing the sign and the magnitude respectively. When $\mathbf{Y}$ is a $\mathscr{C}$ random vector and $\mathbf{X}$ is independent from $\mathbf{Y}$, $G(\mathbf{X}, \mathbf{Y})|\mathbf{Y} > 0$ has the same distribution as $G(\mathbf{X}, |\mathbf{Y}|)$ for any function $G(\cdot)$ (Arellano-Valle et al., 2002). Similar to these authors, our model assumes a stochastic representation where the matrix-vector multiplication that models the dependence structure among the skewness components follows the absolute value operation:

$$\mathbf{Z}^* = \xi \mathbf{1}_n + \mathbf{A}\mathbf{X} + \mathbf{B}|\mathbf{Y}|, \tag{5}$$

where $\xi \in \mathbb{R}$ is the location parameter, $\{X_i | i = 1, \ldots, n\} \cup \{Y_i | i = 1, \ldots, n\}$ are independent and identically distributed standard normal random variables. Hence, $\mathbf{A}\mathbf{X} + \mathbf{B}|\mathbf{Y}|$ has the same distribution as $\mathbf{A}\mathbf{X} + \mathbf{B}\mathbf{Y}|\mathbf{Y} > 0$ since we can choose $G(\mathbf{X}, \mathbf{Y})$ to be $\mathbf{A}\mathbf{X} + \mathbf{B}\mathbf{Y}$. The pdf of $\mathbf{Z}^*$ avoids the 2^n-term summation, which was the hinge in Zhang and El-Shaarawi (2010), making the pdf computation more scalable.

The marginal representation shown in Equation (5) is difficult to extend to the multivariate skewed-t version because the sufficient condition for the equivalence between the conditional representation and the marginal representation is that $\mathbf{X}$ and $\mathbf{Y}$ are independent (Arellano-Valle et al., 2002). However, the sum of independent Student-t random variables does not necessarily lead to another Student-t random variable. Another issue with this representation is the difficulty to generalize as a random field. When $\mathbf{A}$ is a Cholesky factor, $\mathbf{A}\mathbf{X}$ coincides with the classical Gaussian random field but it is not obvious that $\mathbf{B}|\mathbf{Y}|$ can be derived from any well-defined random field. However, for stochastic generators, the model is usually simulated on a fixed spatial domain without the need for prediction at unknown locations and therefore, Equation (5) may serve as the surrogate model for a physical system. In general, this stochastic representation has better-rounded properties due to its advantage in estimation, simulation, and flexibility. Specifically,

- the pdf avoids the summation of 2^n terms as in the model $\mathbf{A}\mathbf{X} + |\mathbf{B}\mathbf{Y}|$, which makes the

pdf estimable;

- the marginal representation in Equation (5) allows for more efficient simulation compared with conditional representations;

- the correlation structure between the skewness components $\mathbf{B}|\mathbf{Y}|$ has full flexibility controlled by $\mathbf{B}$, which can adapt to different datasets for model fitting.

Considering the reasons above, we simulate $\mathbf{Z}^*$ based on the skew-normal distribution without tapping into any skewed Student-t counterpart for the simulation study and use the same model as a stochastic generator for the Saudi wind speed dataset that has more than 18,000 spatial locations.

5.2 Estimation with simulated data

We construct $\mathbf{A}$ and $\mathbf{B}$ before simulating $\mathbf{Z}^*$, where $\mathbf{A}$ controls the correlation strength of the symmetric component while $\mathbf{B}$ adjusts the level of skewness and the correlation between the skewness component. To have a parsimonious model, $\mathbf{A}$ is assumed to be the lower Cholesky factor of a covariance matrix constructed from the 2D exponential kernel, $\sigma_1^2 \exp(-\|\mathbf{h}\|/\beta_1)$, $\beta_1 > 0$, and $\mathbf{B}$ takes the form of a covariance matrix from the kernel, $\sigma_2^2 \exp(-\|\mathbf{h}\|/\beta_2)$, $\beta_2 > 0$, where $\mathbf{h}$ is the vector connecting the two spatial variables' locations. We choose the form of a covariance matrix instead of a Cholesky factor for $\mathbf{B}$ out of two reasons. Numerically, the row sum of a Cholesky factor usually increases with the row index, which produces a large difference between the sum of the first row and that of the last row when the dimension is high. This would cause the coefficients of $\mathbf{Z}^*$ to have a varying order of magnitude. Secondly, due to the first reason, the likelihood would depend on the ordering or the indexing scheme of the random variables in $\mathbf{Z}^*$. Unlike the Cholesky factor, the row sums of a spatial covariance matrix usually have similar magnitudes and the likelihood function becomes independent from the indexing scheme when $\mathbf{B}$ is a covariance matrix. The pdf of $\mathbf{Z}^*$ can be derived based on the results in

Arellano-Valle et al. (2002) to be:

$$2^n \phi_n(\mathbf{z} - \xi \mathbf{1}_n, \mathbf{A}\mathbf{A}^T + \mathbf{B}\mathbf{B}^T) \Phi_n\{-\boldsymbol{\infty}, (\mathbf{I}_n + \mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{A}^{-1}(\mathbf{z} - \xi \mathbf{1}_n); (\mathbf{I}_n + \mathbf{C}^T\mathbf{C})^{-1}\}, \quad (6)$$

where $\mathbf{C} = \mathbf{A}^{-1}\mathbf{B}$. $O(n^3)$ matrix operations are performed multiple times for computing the covariance matrices, which is prohibitive under the dense representation. However, the TLR representation can closely approximate $\mathbf{A}\mathbf{A}^T$ and $\mathbf{B}$ due to the separable underlying geometry and the 2D exponential covariance model. The subsequent Cholesky factorization, matrix multiplication, and matrix inversion can be performed at adequate accuracy and the complexity can be reduced by one order of magnitude. For each $n = 4^r, r = 4, 5, 6, 7$, we generate the geom-

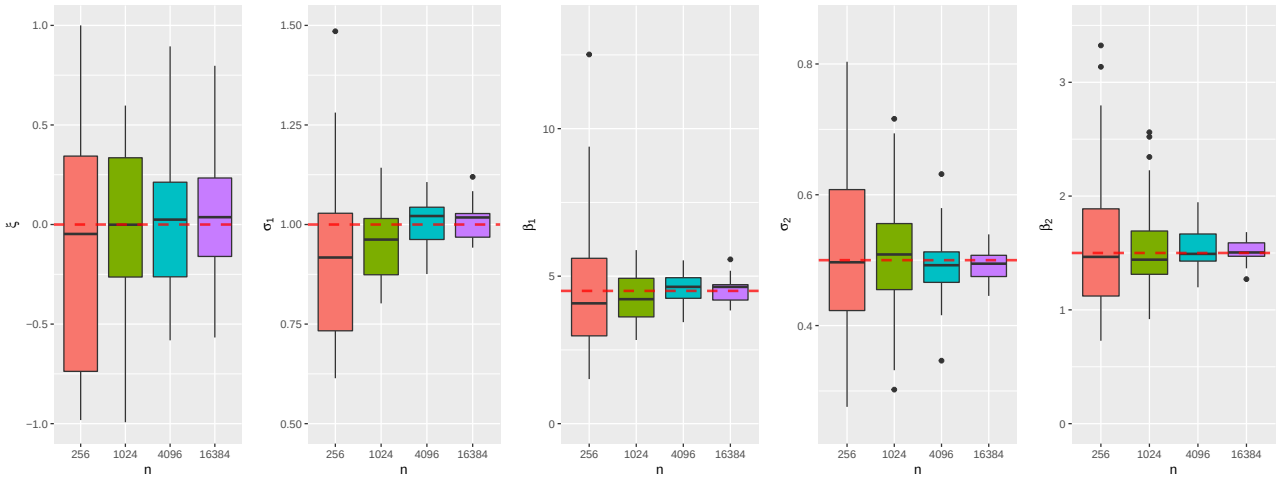


Figure 4: Boxplots of 30 estimation results. Each estimation is based on one realization of the n-dimensional skew-normal model. The red dashed line marks the true value used for generating random vectors from the skew-normal model.

etry in the $[0, 2^{r-4}] \times [0, 2^{r-4}]$ square, mimicking an expanding domain. The spatial locations are on a perturbed grid, where the grid's length unit is $1/15$ and the perturbation is uniformly distributed within $(-0.4/15, 0.4/15)^2$. $\mathbf{A}$ and $\mathbf{B}$ are constructed based on the covariance kernel and the simulated geometry. The likelihood function is the pdf of $\mathbf{Z}^*$ shown in Equation (6) and the optimization seeks to find the parameter values that maximize the likelihood when $\mathbf{z}$ is fixed. In each run, the geometry is regenerated and only dense representations are used for the

simulation of $\mathbf{Z}^*$ to avoid inducing extra estimation error. As the dimension n becomes higher, the likelihood becomes extremely small so that we have to extract the eleven-bit exponent in the double-precision representation and scale the likelihood back to the normal range when computing the P in algorithm 2.1a, algorithm 3.4a, and algorithm 2.2b. The optimization is built on the Controlled Random Search (CRS) with local mutation algorithm (Kaelo and Ali, 2006) from the NLopt library (Johnson, 2014). The true values for $(\xi, \sigma_1, \beta_1, \sigma_2, \beta_2)$ are shown in fig. 4 and their searching ranges are $(-1.0, 1.0)$, $(0.1, 2.0)$, $(0.01, 0.9)$, $(0.0, 1.0)$, and $(0.01, 0.3)$ respectively.

The initial values for the four parameters are set equal to the lower limits of their searching ranges and the stopping condition is an absolute convergence level of 10^{-3} in terms of the log-likelihood. The boxplots for the four chosen dimensions each consisting of 30 estimations are shown in fig. 4. Overall, the estimation improves as the dataset dimension n increases. The outliers may indicate that there is a local maximum, where σ_1 and β_1 are large and σ_2 is small, on a similar magnitude level with the global maximum. In this case, the estimation result is closer to a Gaussian random field.

In the application study, we found that the likelihood was at extreme values when the dimension n was high because the order of magnitude cumulated through the multiplication of one-dimensional probabilities as shown in Equation (2). As a result, the exponent of the likelihood value exceeded what double-precision numbers could accommodate. In case of overflow, we extracted and cumulated the exponent after each multiplication step described on Line 9 of algorithm 2.1a. It is also possible to have $\Phi(b') - \Phi(a')$ smaller than what the double-precision allows so that the quantile function on Line 8 of algorithm 2.1a will produce invalid values. These invalid likelihood values were usually produced by unreasonable parameter value inputs given the observed $\mathbf{z}$ and we treated these likelihood values as a large number, which worked well for our estimation study.

5.3 Estimation with wind data from Saudi Arabia

The dataset we use for modeling is the daily wind speed in the Kingdom of Saudi Arabia on August 5th, 2013, produced by the WRF model, which numerically predicts the weather system based on partial differential equations on the mesoscale and features strong computation capacity to serve meteorological applications (Skamarock et al., 2008). The dataset has an underlying geometry with 155 longitudinal and 122 latitudinal bands. Specifically, the longitude increases from 40.034 to 46.960 and the latitude increases from 16.537 to 21.979, both with an incremental size of 0.045. Before fitting the skew-normal model, we subtract the wind speed at each location with its mean over a six-year window (six replicates in total) to increase the homogeneity across the locations. The vectorized demeaned wind speed data is used as the input dataset, $\mathbf{Z}^*$, for the maximum likelihood estimation. The dataset has a skewness of -0.45 and is likely to benefit from the skewness flexibility introduced by the model in Equation (5). It is worth noting that $\mathbf{B}|\mathbf{Y}|$ has a negative skewness under our parameterization for $\mathbf{B}$ although all its coefficients are non-negative.

The likelihood function is described in Equation (6), where the parameterization of $\mathbf{A}$ and $\mathbf{B}$ also remains unchanged. The optimization involves five parameters, namely ξ, σ_1, β_1, σ_2, and β_2, whose searching ranges, initial values, and optimized values are listed in table 3. Since the likelihood requires the inverse of $\mathbf{A}$ as shown in Equation (6), we set the lower limit of σ_1 to 0.1 to avoid the singularity. The correlation-strength parameters β_1 and β_2 can theoretically

Table 3: Parameter specifications and estimations based on the skew-normal (SN) model and the Gaussian random field (GRF)

	ξ	σ_1	β_1	σ_2	β_2
Range	$(-2, 2)$	$(0.1, 2.0)$	$(0.1, 5.0)$	$(0.0, 2.0)$	$(0.01, 1.0)$
Initial Value	0.000	1.000	0.100	1.000	0.010
SN	-1.211	1.028	4.279	0.419	0.065
GRF	0.338	1.301	4.526	N.A.	N.A.

be close to zero but setting a lower limit above zero can avoid boundary issues. The absolute convergence level is set at 10^{-3} and the optimization produces the results shown in table 3, which has a log-likelihood of 11,508. We compare the optimized skew-normal model with the optimized classical Gaussian random field, which is also a simplified version of Equation (5), where σ_2 is fixed at zero: $\mathbf{Z}^* = \xi\mathbf{1} + \mathbf{AX}$. The estimation of the Gaussian random field thus involves three parameters, (σ_1, β_1, ξ), for which the optimization setups are the same as those for the skew-normal model. The estimated parameter values are also summarized in table 3, which has a log-likelihood of 10,797. The functional boxplots (Sun and Genton, 2011) of the empirical semivariogram based on 100 runs of the fitted skew-normal model and the Gaussian random field are shown in fig. 5. The skew-normal model has significantly smaller band width than the Gaussian random field, although both cover the semivariogram of the original data. The BIC values of the two models and the quantile intervals of the empirical moments based on the same 100 replicates are illustrated in table 4. The BIC values strongly indicate that the skew-normal

Table 4: Empirical moments and BIC comparison. SN denotes the skew-normal model and GRF denotes the Gaussian random field. The intervals represent the 5% to 95% quantile intervals based on 100 replicates.

	Mean	Variance	Skewness	Kurtosis	BIC
Wind data	0.042	0.932	-0.445	2.873	N.A.
SN	$(-1.079, 1.360)$	$(0.308, 1.054)$	$(-0.644, 0.449)$	$(2.274, 3.595)$	-22986
GRF	$(-1.644, 1.911)$	$(0.612, 2.594)$	$(-0.717, 0.489)$	$(2.116, 3.705)$	-21565

model is a better fit than the Gaussian random field. This can be also seen from the variance quantile intervals and the functional boxplots of the empirical semivariogram, where the former has smaller variance and its semivariogram is more aligned with that of the Saudi wind data. The empirical moments ignore the connection between the spatial locations, thus may not be a comprehensive surrogate for the fitting quality. Except for the one for the variance, the two models are not significantly different in terms of the other three quantile intervals but in general, the moments' quantile intervals for the skew-normal model are tighter, which indicates a better

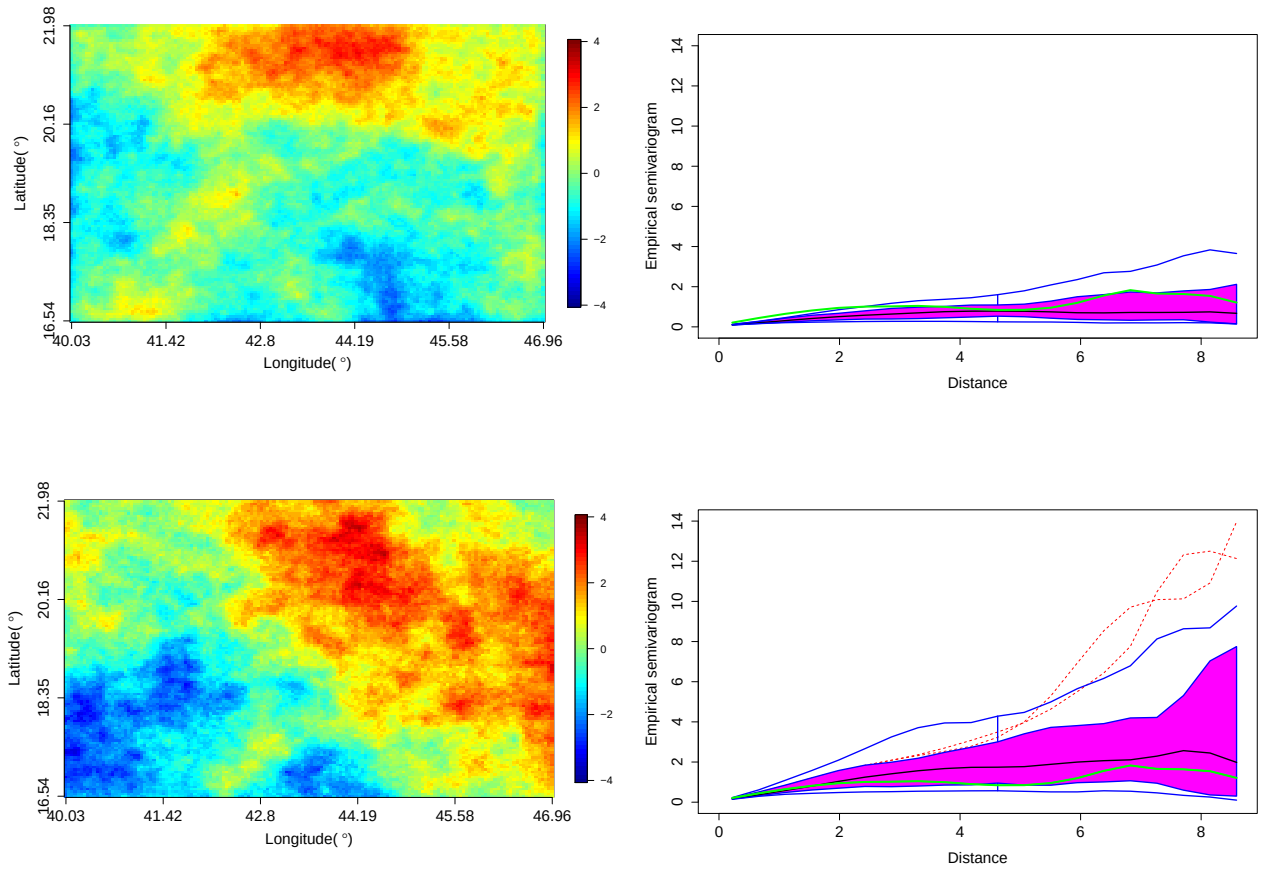


Figure 5: The heatmap based on one replicate and the functional boxplot of the empirical semivariogram based on 100 replicates. Top to bottom are the fitted skew-normal model and the Gaussian random field. The green curve denotes the empirical semivariogram based on the wind speed data. The distance is computed as the Euclidean distance in the longitudinal and latitudinal coordinate system.

fit.

6 Conclusion and Discussion

In this paper, we presented preconditioned TLR versions of the Quasi-Monte Carlo methods in Genz (1992) and Genz and Bretz (2002) suitable for computing MVN and MVT probabilities in very high dimensions. The preconditioning uses a block reordering scheme and results in substantial improvement in performance. Consequently, the new methods can reduce the computation time by an order of magnitude compared with methods using hierarchical matrices (Genton et al., 2018) and two orders of magnitude compared with dense Quasi-Monte Carlo

methods (Genz, 1992; Genz and Bretz, 2002) while keeping the estimation error at the same level.

As an application of our proposed methods, we generated random vectors from a skew-normal model described in Equation (5) of dimensions up to 2^{14} and performed maximum likelihood estimation for five parameters simultaneously. The results showed significant estimation improvement when the dimension of the random vector increased, highlighting the need for efficient computation of high-dimensional MVN and MVT probabilities in modern data-rich applications. In another application, we fitted the same model to wind data from Saudi Arabia that has a resolution of 155×122. The BIC of the skew-normal model was significantly smaller than that of the classical Gaussian random field and the moments, as well as the semivariogram of the simulated skew-normal random vector, resembled those of the dataset closer.

Acknowledgements

The authors thank Prof. Stenchikov at KAUST for providing the WRF data.

References

Arellano-Valle, R., Del Pino, G., and San Martín, E. (2002), "Definition and probabilistic properties of skew-distributions," *Statistics & Probability Letters*, 58, 111–121.

Arellano-Valle, R. B., Branco, M. D., and Genton, M. G. (2006), "A unified view on skewed distributions arising from selections," *Canadian Journal of Statistics*, 34, 581–601.

Arellano-Valle, R. B. and Genton, M. G. (2010), "Multivariate unified skew-elliptical distributions," *Chilean Journal of Statistics*, 1, 17–33.

Azzalini, A. and Capitanio, A. (2014), *The Skew-Normal and Related Families*, vol. 3, Cambridge University Press.

Bebendorf, M. (2011), "Adaptive cross approximation of multivariate functions," *Constructive Approximation*, 34, 149–179.

Boukaram, W., Turkiyyah, G., and Keyes, D. (2019), "Hierarchical Matrix Operations on GPUs: Matrix-Vector Multiplication and Compression," *ACM Transactions on Mathematical Software*, 45, 3:1–3:28.

Cao, J., Genton, M. G., Keyes, D. E., and Turkiyyah, G. M. (2019), "Hierarchical-block conditioning approximations for high-dimensional multivariate normal probabilities," *Statistics and Computing*, 29, 585–598.

Castruccio, S. and Genton, M. G. (2016), "Compressing an ensemble with statistical models: An algorithm for global 3D spatio-temporal temperature," *Technometrics*, 58, 319–328.

— (2018), "Principles for statistical inference on big spatio-temporal data from climate models," *Statistics & Probability Letters*, 136, 92–96.

Genton, M. G. (2004), *Skew-elliptical Distributions and Their Applications: A Journey Beyond Normality*, CRC Press.

Genton, M. G., Keyes, D. E., and Turkiyyah, G. (2018), "Hierarchical decompositions for the computation of high-dimensional multivariate normal probabilities," *Journal of Computational and Graphical Statistics*, 27, 268–277.

Genz, A. (1992), "Numerical computation of multivariate normal probabilities," *Journal of Computational and Graphical Statistics*, 1, 141–149.

Genz, A. and Bretz, F. (1999), "Numerical computation of multivariate t-probabilities with application to power calculation of multiple contrasts," *Journal of Statistical Computation and Simulation*, 63, 103–117.

— (2002), "Comparison of methods for the computation of multivariate t probabilities," *Journal of Computational and Graphical Statistics*, 11, 950–971.

— (2009), *Computation of Multivariate Normal and t Probabilities*, vol. 195, Springer Science & Business Media.

Hackbusch, W. (2015), *Hierarchical Matrices: Algorithms and Analysis*, vol. 49, Springer.

Jeong, J., Castruccio, S., Crippa, P., Genton, M. G., et al. (2018), "Reducing storage of global wind ensembles with stochastic generators," *The Annals of Applied Statistics*, 12, 490–509.

Johnson, S. G. (2014), "The NLopt nonlinear-optimization package," `http://github.com/stevengj/nlopt`.

Kaelo, P. and Ali, M. (2006), "Some variants of the controlled random search algorithm for global optimization," *Journal of Optimization Theory and Applications*, 130, 253–264.

Kim, H., Ha, E., and Mallick, B. (2004), "Spatial prediction of rainfall using skew-normal processes," in *Skew-Elliptical Distributions and Their Applications: A Journey Beyond Normality*, ed. Genton, M. G., CRC Press, pp. 279–289.

Richtmyer, R. D. (1951), "The evaluation of definite integrals, and a quasi-Monte-Carlo method based on the properties of algebraic numbers," Tech. rep., Los Alamos Scientific Lab.

Samet, H. (1990), *The Design and Analysis of Spatial Data Structures*, vol. 85, Addison-Wesley Reading, MA.

Schervish, M. J. (1984), "Algorithm AS 195: Multivariate normal probabilities with error bound," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 33, 81–94.

Skamarock, W. C., Klemp, J. B., Dudhia, J., Gill, D. O., Barker, D. M., Duda, M. G., Huang, X.-Y., Wang, W., and Powers, J. G. (2008), *A Description of the Advanced Research WRF Version 3*, vol. 113, NCAR.

Sun, Y. and Genton, M. G. (2011), "Functional boxplots," *Journal of Computational and Graphical Statistics*, 20, 316–334.

Trinh, G. and Genz, A. (2015), "Bivariate conditioning approximations for multivariate normal probabilities," *Statistics and Computing*, 25, 989–996.

Zhang, F. (2006), *The Schur complement and its applications*, vol. 4, Springer Science & Business Media.

Zhang, H. and El-Shaarawi, A. (2010), "On spatial skew-Gaussian processes and applications," *Environmetrics*, 21, 33–47.