Statistics and Computing (2021) 31:2
https://doi.org/10.1007/s11222-020-09978-y

Exploiting low-rank covariance structures for computing high-dimensional normal and Student-*t* probabilities

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

Received: 4 March 2020 / Accepted: 25 November 2020

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 hierarchical 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 to wind stochastic generators is provided to illustrate that the new computational method makes the maximum likelihood estimation feasible for high-dimensional skew-normal random fields.

Keywords Block reordering · Hierarchical matrix · Skew-normal random field · Tile-low-rank matrix

1 Introduction

The multivariate normal (MVN) probability appears frequently in statistical applications. For example, the probability density functions of several skew-normal (Genton 2004; Azzalini and Capitanio 2014; Arellano-Valle et al. 2006) and Bayesian probit (Durante 2019) models involve MVN cumulative distribution functions. It is also needed in computing the excursion and contour regions discussed in Bolin and Lindgren (2015). For many of these applications, the MVN probability is regarded as a bottleneck and approximations to the covariance matrix are often applied in high dimensions. The MVN probability is one example of numerical integration, in which the quadrature-based methods are typically not applicable in hundreds of dimensions. The Monte Carlo-based methods are more flexible, but their convergence rate is subject to several factors. In this paper, we aim to reduce the time costs and extend the limits for computing MVN probabilities.

The prevalent algorithm for computing MVN probabilities is based on the separation-of-variable (SOV) technique (Genz 1992), which converts the integration region to the unit hypercube to improve the convergence rate. This method is more robust than its improved variants but has poor scalability, with the costs of $O(n^3)$ for the Cholesky factorization and $O(n^2)$ per MC sample, where n is the MVN problem dimension. State-of-the-art methods for computing MVN probabilities include the hierarchical quasi-Monte Carlo (QMC) method (Genton et al. 2018), the minimax tilting method (Botev 2017), the two-step method (Azzimonti

This research was supported by King Abdullah University of Science and Technology (KAUST).

✉ Jian Cao
jian.cao@kaust.edu.sa

Marc G. Genton
marc.genton@kaust.edu.sa

David E. Keyes
david.keyes@kaust.edu.sa

George M. Turkiyyah
gt02@aub.edu.lb

[1] CEMSE Division, Extreme Computing Research Center, King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia

[2] Department of Computer Science, American University of Beirut, Beirut, Lebanon

Published online: 12 January 2021

and Ginsbourger 2018), and the hierarchical conditioning method (Cao et al. 2019). The hierarchical QMC method reduced the costs per sample through the hierarchical representation (Hackbusch 2015) of the Cholesky factor. Its drawback is its incompatibility with variable reordering and hence its inability to benefit from an improved convergence rate. The minimax tilting method significantly improves the convergence rate with importance sampling but needs to solve an expensive optimization with $O(n)$ parameters for the proposal density. The two-step method decomposes a high-dimensional MVN probability into a low-dimensional one and a high-dimensional residual, which is only applicable to orthant MVN probabilities that have constant upper and lower integration limits. Cao et al. (2019) used the hierarchical representation and the conditioning technique that samples the integrand only once to achieve high computation efficiency. However, this hierarchical conditioning method only provides crude probability estimates without any error estimation.

This paper builds on the original SOV method in Genz (1992) and introduces a variant that has better performance than the hierarchical QMC method in Genton et al. (2018). Specifically, we combine the SOV method with the tile-low-rank (TLR) representation (Weisbecker 2013; Mary 2017; Akbudak et al. 2017), which improves efficiency from two aspects. First, the TLR representation is compatible with block-wise variable reordering and hence benefits from a higher convergence rate. Secondly, the memory footprint of the Cholesky factor under the TLR representation can be smaller than that under the hierarchical representation, indicating lower costs per QMC sample. In this paper, we only compare our methods with the hierarchical QMC method in Genton et al. (2018) because the other three state-of-the-art methods are not directly based on the SOV algorithm. As another extension, we propose an iterative version of the original block reordering in Cao et al. (2019) that further improves the convergence rate and performs the Cholesky factorization simultaneously. The corresponding algorithm for multivariate Student-t (MVT) probabilities is also developed. Finally, we demonstrate the capability of our methods in tens of thousands of dimensions with two maximum likelihood estimation (MLE) studies based on simulated data and a wind dataset.

The remainder of this paper is structured as follows. In Sect. 2, we review the SOV algorithm (Genz and Bretz 2009) for MVN and MVT problems and describe the dense QMC algorithms for both probabilities. In Sect. 3, we show that the TLR representation is more aligned with block-wise variable reordering than hierarchical representations. Additionally, an improved version of the block reordering from Cao et al. (2019) is proposed. In Sect. 4, we compare the dense QMC method, the hierarchical QMC method, and the TLR QMC methods with a focus on high-dimensional MVN and MVT probabilities. In Sect. 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 to demonstrate the usage of our methods. Finally, Sect. 6 concludes the paper. The execution times in this paper are measured on a 4-core Intel Core i7 CPU with 64 GB memory without parallelization.

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 SOV for MVT probabilities have been derived in Genz (1992) and Genz and Bretz (2002). In this paper, we summarize the derivations for completeness.

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:

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

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 $\mathbf{L}$ to represent the lower Cholesky factor of $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^\top$ and l_{ij} to represent the element on the i-th row and j-th column of $\mathbf{L}$. 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 $d_i = \Phi\{(a_i - \sum_{j=1}^{i-1} l_{ij} y_j)/l_{ii}\}$, $e_i = \Phi\{(b_i - \sum_{j=1}^{i-1} l_{ij} y_j)/l_{ii}\}$, $y_j = \Phi^{-1}\{d_j + w_j(e_j - d_j)\}$, 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 Eq. (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 Eq. (2). Keeping $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{L}$ unchanged, the mean and standard deviation of the outputs P from a 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.

Algorithm 2.1a QMC for MVN probabilities

mvn($\mathbf{L}$, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{w}$)
$n \leftarrow \dim(\mathbf{L})$, $s \leftarrow 0$, $\mathbf{y} \leftarrow \mathbf{0}$, and $P \leftarrow 1$
for $i = 1 : n$ **do**
  if $i > 1$ **then**
    $s \leftarrow \mathbf{L}(i, 1 : i-1)\mathbf{y}(1 : i-1)$
  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 for
return P and $\mathbf{y}$

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, $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ is the scale matrix. 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:

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

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, $T_n(\mathbf{a}, \mathbf{b}; \boldsymbol{\Sigma}, \nu)$ can be also written as Eq. (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.

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

mvt_sov($\mathbf{L}$, $\mathbf{a}$, $\mathbf{b}$, ν, $\mathbf{w}$)
$n \leftarrow \dim(\mathbf{L})$, $s \leftarrow 0$, $ssq \leftarrow 0$, $\mathbf{y} \leftarrow \mathbf{0}$, and $P \leftarrow 1$
for $i = 1 : n$ **do**
  if $i > 1$ **then**
    $s \leftarrow \mathbf{L}(i, 1 : i-1)\mathbf{y}(1 : i-1)$
  end if
  $a' \leftarrow \frac{a_i - s}{\mathbf{L}_{i,i} \cdot \sqrt{\nu + ssq} \cdot (\nu+i)}$ and $b' \leftarrow \frac{b_i - s}{\mathbf{L}_{i,i} \cdot \sqrt{\nu + ssq} \cdot (\nu+i)}$
  $y_i \leftarrow T_{\nu+i}^{-1}\left[w_i\left\{T_{\nu+i}(b') - T_{\nu+i}(a')\right\} + T_{\nu+i}(a')\right] \cdot \sqrt{\frac{\nu+ssq}{\nu+i}}$
  $P \leftarrow P \cdot \left\{T_{\nu+i}(b') - T_{\nu+i}(a')\right\}$
  $ssq \leftarrow ssq + y_i^2$
end for
return P

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

mvt_scale($\mathbf{L}$, $\mathbf{a}$, $\mathbf{b}$, ν, 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($\mathbf{L}$, $\mathbf{a}'$, $\mathbf{b}'$, $\mathbf{w}$)

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

Table 1 Relative error and time of the three algorithms

n	16	64	256	1,024	4,096
mvt_sov	0.0%	0.2%	0.7%	1.4%	4.2%
	0.7s	*3.0s*	*13.3s*	*58.7s*	*283.1s*
mvt_scale	0.0%	0.0%	0.2%	0.4%	1.3%
	0.0s	*0.0s*	*0.2s*	*2.0s*	*40.8s*
mvn	0.0%	0.0%	0.1%	0.4%	1.2%
	0.0s	*0.0s*	*0.2s*	*2.0s*	*40.1s*

'mvt_sov', 'mvt_scale', and 'mvn' refer to Algorithms 2.2a, 2.2b, and 2.1a. The upper row is the average relative estimation error and the lower row is the average computation time over 20 replicates. The covariance matrix is generated from a 2D exponential kernel, $\exp(-\|\mathbf{h}\|/\beta)$, where $\mathbf{h}$ is the distance vector, based on n locations on a perturbed grid in the unit square. The lower integration limits are $-\infty$ and the upper limits are independently generated from $N(5.5, 1.25^2)$. The degrees of freedom ν for MVT probabilities are 10. The Monte Carlo sample size is 10^4

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.

A numerical comparison between Algorithms 2.2a and 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 Eq. (3) leads to an implementation slower by one order of magnitude. Additionally, the convergence rate from Eq. (3) is also worse than that from Eq. (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, likely because of the lack of optimized libraries, and produces lower accuracy. 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.

3 Low-rank representation and reordering for MVN and MVT probabilities

3.1 Overview

More flexible than quadrature methods, Monte Carlo (MC) procedures provide several viable options for computing MVN and MVT probabilities. The cost of these computations depends on the product of the number of MC samples, N, needed to achieve a desired accuracy and the cost per MC sample. Under the standard dense representation of covariance, the computational complexity for each sample is $O(n^2)$ as shown in Algorithms 2.1a and 2.2b. Genton et al. (2018) proposed using the hierarchical representation for the Cholesky factor, illustrated in Fig. 1, which reduced the complexity per sample to $O(kn \log n)$, where k is a nominal local rank of the matrix blocks. Using nested bases in the hierarchical representation (Boukaram et al. 2019), it is possible to reduce this cost further to an asymptotically optimal $O(kn)$.

In this paper, we assume the covariance matrix is generated from a set of spatial locations and a covariance kernel. Small local ranks in off-diagonal blocks are obtained when the row cluster and the column cluster are well separated spatially, growing only 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 spatial locations and the corresponding variables are then further divided into blocks (clusters) according to these indices to build the hierarchical representation. We also use the terms 'cluster' to align with the literature of hierarchical matrices; see Hackbusch (2015) for more details.

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 ordering 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 under 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. This has less impact on the local ranks of the hierarchical structure than the reordering schemes discussed in Trinh and Genz (2015).

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 to achieve the desired accuracy. They produce different orders from the geometry-oriented ordering obtained by spatial partitioning methods

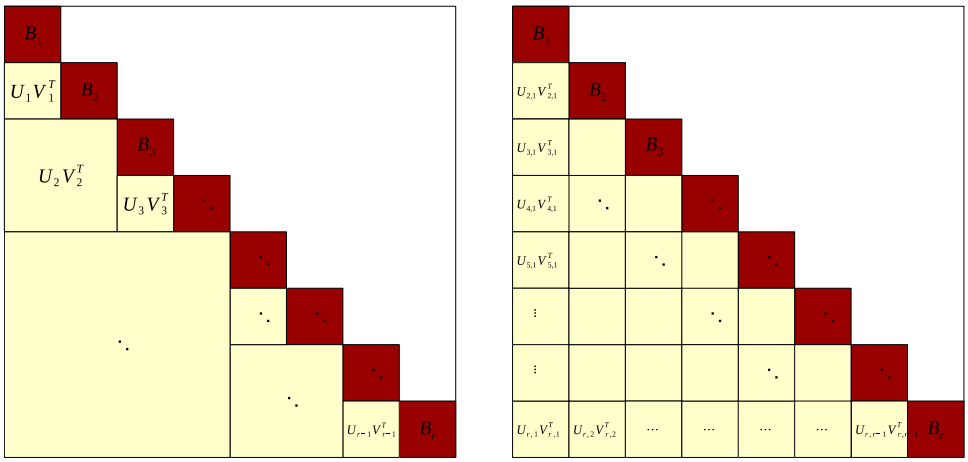


Fig. 1 Structures of hierarchical (left) and tile-low-rank (right) matrices

or space-filling curves. The reordering increases the local ranks k of the hierarchical representation or broader low-rank representations, 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 TLR representation of covariance illustrated in Fig. 1. We also introduce the TLR versions of the QMC algorithms for computing MVN and MVT probabilities.

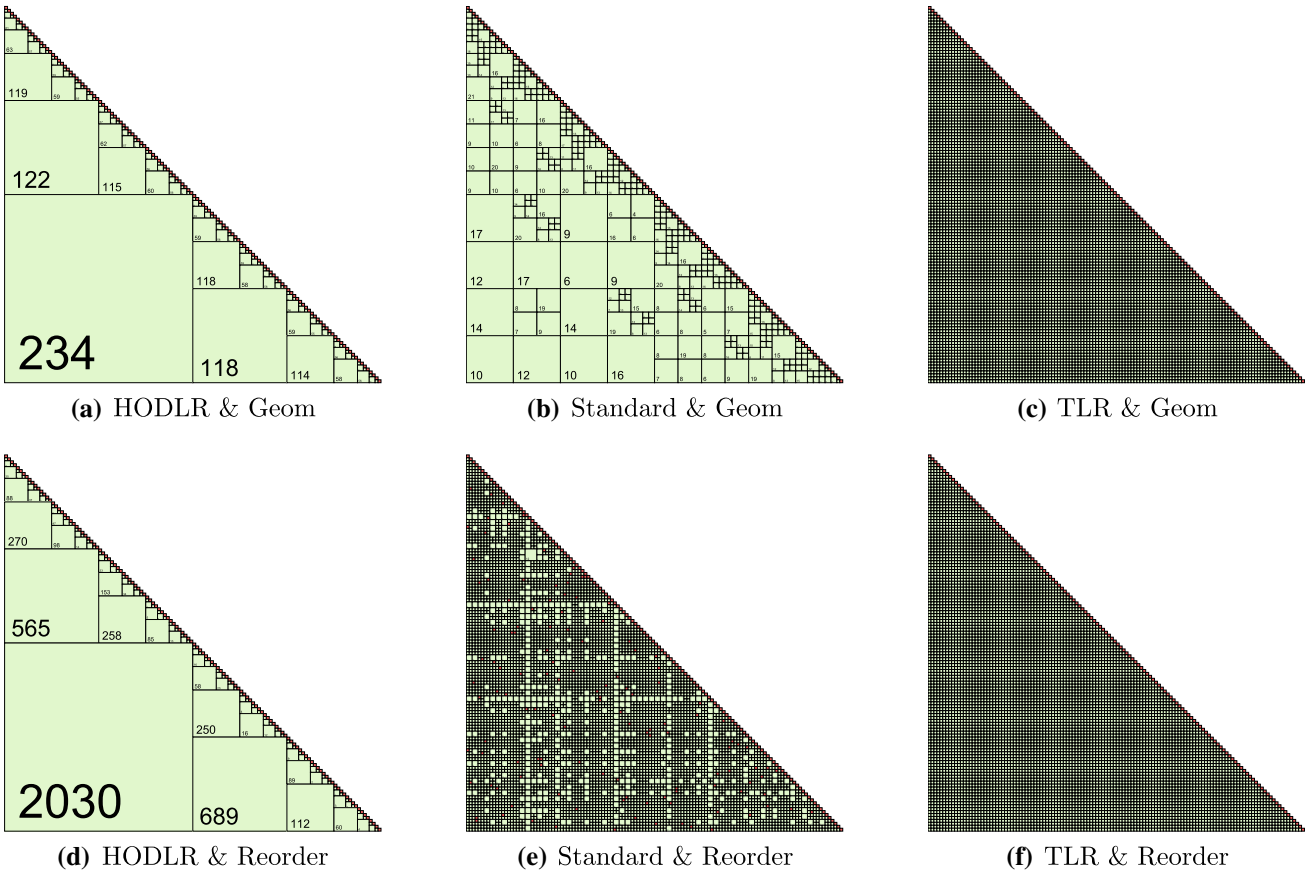
3.2 TLR as a practical representation for MVN and MVT

To show the TLR structure has good compatibility with the block-reordering scheme introduced in Cao et al. (2019), we consider an MVN problem whose integration limits are independent from the geometry. The geometry is a 128×128 grid in the unit square whose locations are initially indexed with the geometrical clustering method provided in HLIBpro v2.8 (Börm et al. 2003; Kriemann 2005; Grasedyck et al. 2008). The partitioning of each cluster is cardinality balanced, i.e., the two child clusters have equal number of indices and the minimum cluster size is set to 128. This geometrical indexing is used as the benchmark for measuring the efficiency of three low-rank structures discussed in Fig. 2. Assuming that the integration limits are independent and identically distributed, the block reordering is equivalent to shuffling the 128 clusters previously computed. A cluster-wise shuffle of the first indexing is used for measuring the compatibility of the low-rank representations with the integration-oriented ordering. Figure 2 describes the approximation of the Cholesky factors of the two covariance matrices using the hierarchical structures under the weak and the standard admissibility conditions as well as the TLR structure while Table 2 lists the corresponding time costs and memory footprints of the factorization.

We focus on the Cholesky factor instead of the covariance matrix because the former's memory footprint has a linear relationship with the cost per sample in the computation of MVN probabilities. For each low-rank structure, the covariance matrix is constructed and factorized using the fixed-precision truncation with an absolute error of 10^{-4}. It is worth mentioning that this truncation accuracy is unnecessarily high for the TLR structure and the hierarchical structure under the standard admissibility condition while necessary for the Cholesky factorization of the hierarchical structure under the weak admissibility condition (HODLR). Therefore, in practice, we expect higher efficiency than what Fig. 2 indicates when using the former two low-rank structures. In contrast, the fixed-rank truncation is less suitable for our purpose because the local ranks have large variability. Choosing a high fixed rank would waste memory while using a low fixed rank could cause factorization failure. Furthermore, the fixed-precision truncation with a relative error is empirically found to be less efficient than that with an absolute error, generating higher memory footprints at the accuracy threshold for a successful factorization.

The selected covariance kernel, $\exp(-\|\mathbf{h}\|/0.3)$, has a range parameter $\beta = 0.3$ and an effective range of 0.9, indicating a strong correlation in the unit square. Overall, stronger correlation presents more challenge in terms of Cholesky factorization because the covariance matrix becomes more numerically singular. The exponential kernel corresponds to the Matérn kernel with a smoothness parameter $\nu = 0.5$ and the correlation for small $\|\mathbf{h}\|$ (relative to β) increases asymptotically to one when ν increases, for which the singularity of the Matérn kernel generally grows with ν if the given locations are in a compact domain. Therefore, smooth kernels are prone to having singularity issues, in which case either increasing the truncation accuracy or adding a nugget effect can enhance the factorization.

The admissibility condition measures the ratio of the distance between two clusters to a weighted average of their



(a) HODLR & Geom **(b)** Standard & Geom **(c)** TLR & Geom

(d) HODLR & Reorder **(e)** Standard & Reorder **(f)** TLR & Reorder

Fig. 2 Partition and local ranks of a 16,384-dimensional covariance matrix represented with three low-rank structures. Red denotes dense blocks and green is used for low-rank blocks whose ranks are the numbers inside. For each structure, two indexing methods are compared, namely a geometrical indexing (Geom) and the block reordering (Reorder). HODLR and STD are the hierarchical structures under the weak and the standard admissibility conditions, respectively. TLR is the tile-low-rank structure. The covariance matrix is built with a grid in the unit square and the exponential covariance kernel, $\exp(-\|\mathbf{h}\|/0.3)$, where $\mathbf{h}$ is the vector connecting two locations. (Color figure online)

Table 2 Cholesky factorization times and memory footprints for a 16,384-dimensional covariance matrix under a geometrical indexing (Geom) and block reordering (Reorder)

	factorize (Geom)	factorize (Reorder)	memory (Geom)	memory (Reorder)
HODLR	5.7*s*	46.0*s*	102MB	409MB
STD	0.6*s*	2.8*s*	60MB	82MB
TLR	4.5*s*	2.9*s*	88MB	67MB

HODLR and STD are the hierarchical structures under the weak and the standard admissibility conditions, respectively. TLR is the tile-low-rank structure. The covariance matrix is built with a grid in the unit square and the exponential covariance kernel, $\exp(-\|\mathbf{h}\|/0.3)$, where $\mathbf{h}$ is the vector connecting two locations

diameters. Hence, a big ratio indicates good separability and a small local rank, whereas the HODLR structure has large blocks in the low-rank format whose ratios are close or even equal to zero, leading to high memory footprint. Additionally, its increase is also the largest when switching from the geometrical indexing to the block reordering because a shuffle to the leaf clusters has stronger impact on the blocks whose row and column clusters consist of a larger number of leaf clusters. Contrarily, the memory footprint of using the TLR structure even decreases when switching to the block reordering. This may appear surprising, but an explanation is given from two perspectives:

1. The overall local ranks in the TLR covariance matrix do not change. The block reordering shuffles the leaf clusters, which only rearranges the off-diagonal blocks since the block size in the TLR representation is equal to the size of leaf clusters.
2. Secondly, the magnitude of the Schur complement decreases faster with the block column index under the

block reordering. The Schur complement at block column i, $\boldsymbol{\Sigma}_{i:r,i:r} - \boldsymbol{\Sigma}_{i:r,1:i-1}\boldsymbol{\Sigma}^{-1}_{1:i-1,1:i-1}\boldsymbol{\Sigma}^{\top}_{i:r,1:i-1}$, is the conditional covariance matrix for variables in clusters i to r given variables in clusters 1 to $i-1$. We argue that for general spatial kernels, the conditional covariance matrix tends to have smaller magnitude, e.g., the Frobenius norm, when the locations of the conditioning variables are more scattered. This is treated as a heuristic without proof since it is not the focus of the paper.

Therefore, the block reordering maintains the local ranks of the TLR covariance matrix while increasing the magnitude decay of its Cholesky factor, which collectively reduce the memory footprint of the TLR Cholesky factor under the fixed-precision truncation with an absolute error. The combination of geometrical indexing and the standard admissibility condition produces the smallest memory footprint and factorization time. However, it is unable to benefit from an improved convergence rate. Under the block reordering, the standard admissibility condition produces a fine block partition that approaches the TLR structure and has a higher memory footprint than the TLR structure. The reason is that not satisfying the standard admissibility condition does not guarantee that the block's dense representation is more economical than its low-rank representation, and in fact, the opposite is true for many dense blocks.

We conclude that under the block reordering, the TLR structure has the highest efficiency among the three low-rank structures discussed above and is only slightly less efficient than the combination of geometrical indexing and the standard admissibility condition. Additionally, the TLR structure is conceptually simpler than the other two and hence, more likely to benefit from parallel 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 at a lower cost than the univariate or bivariate reordering scheme introduced in Trinh and Genz (2015). In this paper, we improve the original block-reordering scheme by ordering the clusters of variables iteratively. The new iterative block reordering, similar to the block version of the univariate reordering scheme in Trinh and Genz (2015), enjoys a higher convergence rate and produces the Cholesky factor simultaneously.

Algorithm 3.3a describes the original block-reordering scheme proposed in Cao et al. (2019) while Algorithm 3.3b is the iterative version that produces the Cholesky factor. We use $\boldsymbol{\Sigma}_{i,j}$ to represent the (i,j)-th size-m block of $\boldsymbol{\Sigma}$. Similar notations are also used for $\mathbf{a}$ and $\mathbf{b}$. The symbol $\rightleftharpoons$ indicates the switching of coefficients, rows, or columns. Variables can be overwritten by themselves after computations for performance benefits. When $i \neq j$, $\boldsymbol{\Sigma}_{i,j}$ is stored in the low-rank format. The blue lines in Algorithm 3.3b mark the matrix operations that are also in the TLR Cholesky factorization (Akbudak et al. 2017). If we ignore the cost for steps 5 and 9, the complexity of Algorithm 3.3b is the same as the TLR 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 size-m clusters 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 iterative block reordering has additional complexity of $O(n^2m)$ over the TLR Cholesky factorization but produces the Cholesky factor simultaneously.

Algorithm 3.3a Block reordering

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

Algorithm 3.3b Block reordering during Cholesky factorization

1: rbodr($\boldsymbol{\Sigma}$, $\mathbf{a}$, $\mathbf{b}$, m)
2: $r = n/m$
3: **for** $j = 1 : r$ **do**
4: $\quad$ **for** $l = j : r$ **do**
5: $\quad\quad \mathbf{p}[l] \approx \Phi_m(\mathbf{a}_l, \mathbf{b}_l; \boldsymbol{\Sigma}_{l,l})$
6: $\quad$ **end for**
7: $\quad \tilde{j} = \text{argmin}_l(\mathbf{p}[l]), l = j, \ldots, r$
8: $\quad$ Block-wise $\boldsymbol{\Sigma}[j \rightleftharpoons \tilde{j}, j \rightleftharpoons \tilde{j}]$, $\mathbf{a}[j \rightleftharpoons \tilde{j}]$, $\mathbf{b}[j \rightleftharpoons \tilde{j}]$
9: $\quad \mathbf{y}_j \approx E_m[\mathbf{Y}|\mathbf{Y} \sim N_m(\mathbf{0}, \boldsymbol{\Sigma}_{j,j}), \mathbf{Y} \in (\mathbf{a}_j, \mathbf{b}_j)]$
10: $\quad \boldsymbol{\Sigma}_{j,j} = \text{Cholesky}(\boldsymbol{\Sigma}_{j,j})$
11: $\quad$ **for** $i = j + 1 : r$ **do**
12: $\quad\quad \boldsymbol{\Sigma}_{i,j} = \boldsymbol{\Sigma}_{i,j} \odot \boldsymbol{\Sigma}^{-\top}_{j,j}$
13: $\quad\quad \mathbf{a}_i = \mathbf{a}_i - \boldsymbol{\Sigma}_{i,j} \odot \mathbf{y}_j, \mathbf{b}_i = \mathbf{b}_i - \boldsymbol{\Sigma}_{i,j} \odot \mathbf{y}_j$
14: $\quad$ **end for**
15: $\quad$ **for** $j_1 = j + 1 : r$ **do**
16: $\quad\quad$ **for** $i_1 = j + 1 : r$ **do**
17: $\quad\quad\quad \boldsymbol{\Sigma}_{i_1,j_1} = \boldsymbol{\Sigma}_{i_1,j_1} \ominus \boldsymbol{\Sigma}_{i_1,j} \odot \boldsymbol{\Sigma}^{\top}_{j_1,j}$
18: $\quad\quad$ **end for**
19: $\quad$ **end for**
20: **end for**

The truncated operations, $\odot$ and $\ominus$, indicate that matrix product and matrix subtraction are followed by truncation to smaller ranks while maintaining the required accuracy, given that the block-wise ranks may generally expand as a result of the operations. Here, $\boldsymbol{\Sigma}_{i_1,j} \odot \boldsymbol{\Sigma}_{j_1,j}^{\top}$, and $\boldsymbol{\Sigma}_{i,j} \odot \boldsymbol{\Sigma}_{j,j}^{-\top}$ 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 Sects. 4 and 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 $O(kn^2/m)$.

3.4 Preconditioned TLR QMC algorithms

Algorithms 3.4a and 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 $\mathbf{L}$, the TLR Cholesky factor, as the interface of the TLR algorithms. The definitions of $\mathbf{B}_i$, $\mathbf{U}_{i,j}$, and $\mathbf{V}_{i,j}$ are shown in Fig. 1.

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), recommended by Genz and Bretz (2009), is employed for choosing $\mathbf{w}$. Here, 'tlrmvn' is called by 'tlrmvt,' where the additional inputs, ν and w_0, bear 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 size-m clusters into the integration limits. The TLR structure reduces the complexity of matrix vector multiplication, hence the cost per MC sample, at the step of block updating the integration limits (Lines 6 and 7 in Algorithm 3.4a). The TLR QMC is a variant of the SOV algorithm from Genz (1992) that belongs to the same category as the hierarchical QMC (Genton et al. 2018). Algorithms 3.4a and 3.4b can be either preconditioned by the block reordering or the iterative block reordering. We examine the performance of the TLR QMC algorithms in Sect. 4.

Algorithm 3.4a TLR QMC for MVN probabilities

1: tlrmvn($\mathbf{B}$, $\mathbf{U}$, $\mathbf{V}$, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{w}$)
2: $\mathbf{y} \leftarrow \mathbf{0}$, and $P \leftarrow 1$
3: **for** $i = 1 : r$ **do**
4: **if** $i > 1$ **then**
5: **for** $j = i : r$ **do**
6: $\boldsymbol{\Delta} = \mathbf{U}_{j,i-1}(\mathbf{V}_{j,i-1}^T \mathbf{y}_{i-1})$
7: $\mathbf{a}_j = \mathbf{a}_j - \boldsymbol{\Delta}$, $\mathbf{b}_j = \mathbf{b}_j - \boldsymbol{\Delta}$
8: **end for**
9: **end if**
10: $(P', \mathbf{y}_i) \leftarrow \text{MVN}(\mathbf{B}_i, \mathbf{a}_i, \mathbf{b}_i, \mathbf{w}_i)$
11: $P \leftarrow P \cdot P'$
12: **end for**
13: **return** P

Algorithm 3.4b TLR QMC for MVT probabilities

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

4 Numerical simulations

Table 3 summarizes the performance of the dense (Genz 1992), the hierarchical (Genton et al. 2018), and the TLR QMC methods for computing MVN and MVT probabilities. Methods are assessed over 20 simulated problems for each combination of problem dimension n and correlation strength β. $\beta = 0.3$, 0.1, and 0.03 correspond to the effective ranges of 0.90, 0.30, and 0.09, respectively, representing strong, medium, and weak correlation strengths in the unit square. The tile size m for the TLR QMC methods is set as $\sqrt{n}$ for the optimal complexity per sample, $O(n^{3/2})$. Overall, smaller blocks are more easily represented in the low-rank format, i.e., having lower local ranks, but a partition that is too fine may compromise the memory savings while bigger blocks can lead to higher local ranks and higher memory footprint for the diagonal blocks. For ease of comparison, the diagonal block size for the hierarchical QMC methods is also set as $\sqrt{n}$. We apply a fixed-precision truncation of 10^{-4} to $\beta = 0.3$ and of 10^{-3} to the other correlation strengths to guarantee the success of Cholesky factorization while enhancing computation efficiency. It is worth mentioning that the error caused by the truncation to a lower rank is typically invisible compared with that from the Monte Carlo integration since the difference between the estimates of the same MVN/MVT problem in Table 3 is well explained by their Monte Carlo standard errors. The sample size is $N = 10^4$ for the methods without any preconditioner and $N = 10^3$ for the four preconditioned methods to highlight their computation times for reaching the same accuracy. The listed time in Table 3 covers only the integration algorithm, not including the construction of the covariance matrix, the block reordering, and the Cholesky factorization. The computation time for Cholesky factorization is indicated in Table 2, generally smaller than that of the Monte Carlo integration with 10^4 samples by more than one order of magnitude. The iterative block reordering has the same order of complexity, $O(n^{5/2})$, as the (TLR) Cholesky factorization when $m = \sqrt{n}$ while the block reordering has the same complexity as the first iteration of the iterative block reordering. The complexity for constructing the covariance matrix is negligible, one order smaller than the Cholesky fac-

Table 3 Performance of the dense, hierarchical, and TLR methods for computing MVN/MVT probabilities

n	mvn	hmvn	tlrmvn	rtlrmvn	rrtlrmvn	mvt	hmvt	tlrmvt	rtlrmvt	rrtlrmvt
$\beta = 0.3$ (strong correlation)										
1024	0.5%	0.5%	0.5%	0.4%	0.4%	0.7%	1.5%	1.6%	1.7%	1.6%
	2.5s	*1.1s*	*1.7s*	*0.1s*	*0.1s*	*2.6s*	*1.1s*	*2.0s*	*0.2s*	*0.2s*
4096	1.1%	1.1%	1.0%	0.9%	1.0%	1.0%	1.6%	1.4%	1.1%	1.2%
	42.4s	*9.7s*	*14.1s*	*1.4s*	*1.3s*	*41.1s*	*7.9s*	*12.7s*	*1.2s*	*1.2s*
16384	2.2%	2.3%	2.2%	2.0%	1.8%	4.5%	4.3%	4.2%	2.8%	2.4%
	1233.5s	*56.2s*	*93.9s*	*8.2s*	*8.2s*	*1198.3s*	*49.9s*	*90.0s*	*7.8s*	*7.8s*
65536	N.A.	5.8%	4.7%	3.5%	2.6%	N.A.	13.5%	13.7%	6.3%	5.9%
	N.A.	*309.5s*	*596.2s*	*50.8s*	*51.2s*	*N.A.*	*294.9s*	*594.7s*	*49.9s*	*50.4s*
$\beta = 0.1$ (medium correlation)										
1024	0.4%	0.4%	0.4%	0.4%	0.3%	0.5%	0.6%	0.5%	0.6%	0.7%
	2.4s	*1.0s*	*1.6s*	*0.1s*	*0.1s*	*2.4s*	*0.8s*	*1.6s*	*0.1s*	*0.1s*
4096	1.3%	1.3%	1.4%	1.0%	1.2%	1.1%	1.1%	1.3%	1.2%	1.2%
	38.5s	*5.2s*	*9.3s*	*1.0s*	*1.0s*	*38.4s*	*5.0s*	*9.9s*	*1.1s*	*1.0s*
16384	4.4%	4.3%	4.1%	4.1%	3.5%	3.1%	3.6%	2.8%	3.4%	3.2%
	1188.2s	*37.0s*	*78.7s*	*6.6s*	*6.6s*	*1176.8s*	*36.3s*	*79.6s*	*6.6s*	*6.6s*
65536	N.A.	34.4%	31.1%	12.0%	11.6%	N.A.	17.2%	17.0%	7.7%	7.4%
	N.A.	*286.8s*	*562.9s*	*47.8s*	*49.5s*	*N.A.*	*254.7s*	*531.6s*	*44.9s*	*45.4s*
$\beta = 0.03$ (weak correlation)										
1024	0.1%	0.2%	0.2%	0.1%	0.1%	0.2%	0.2%	0.2%	0.4%	0.4%
	2.4s	*0.9s*	*1.5s*	*0.1s*	*0.1s*	*2.3s*	*0.8s*	*1.6s*	*0.1s*	*0.1s*
4096	0.7%	0.7%	0.8%	0.5%	0.5%	0.7%	0.7%	0.8%	0.7%	0.8%
	38.1s	*4.9s*	*8.9s*	*0.9s*	*0.9s*	*37.7s*	*4.4s*	*9.1s*	*0.9s*	*0.9s*
16384	3.5%	3.6%	4.0%	2.8%	2.4%	2.6%	2.4%	2.3%	1.7%	1.6%
	1118.5s	*29.4s*	*64.5s*	*5.4s*	*5.4s*	*1097.4s*	*27.5s*	*65.0s*	*5.5s*	*5.4s*
65536	N.A.	67.0%	75.3%	13.7%	14.2%	N.A.	13.5%	13.5%	8.8%	8.1%
	N.A.	*201.1s*	*450.2s*	*37.9s*	*37.9s*	*N.A.*	*203.6s*	*459.8s*	*39.0s*	*38.6s*

'mvn' and 'mvt' are the dense QMC methods, 'hmvn' and 'hmvt' are the hierarchical QMC methods, 'tlrmvn' and 'tlrmvt' are the TLR QMC methods, 'r' indicates the block-reordering preconditioner, and 'rr' indicates the iterative block-reordering preconditioner. The upper row is the average relative estimation error and the lower row is the average computation time over 20 replicates. The covariance matrix is generated from a 2D exponential kernel, $\exp(-\|\mathbf{h}\|/\beta)$, where $\mathbf{h}$ is the distance vector, based on n locations on a perturbed grid in the unit square. The lower integration limits are $-\infty$ and the upper limits are independently generated from $N(5.5, 1.25^2)$. The degrees of freedom ν for MVT probabilities are 10. The QMC sample size for preconditioned methods is 10^4 and 10^3 for others

torization. The highest dimension in our experiment is 2^{16}. Considerations for higher dimensions include the truncation precision required for Cholesky factorization and the sample size needed to reach the desired accuracy.

Considering only the methods without any preconditioner, the low-rank methods are more scalable than the dense methods and the time difference already reaches two orders of magnitude when $n = 16{,}384$. The hierarchical methods are more efficient than the TLR methods although Table 2 indicates slightly higher memory footprint for the hierarchical Cholesky factor under the weak admissibility condition. This is because the hierarchical methods involve fewer but larger matrix-matrix multiplications compared with the TLR methods, beneficial from modern optimized Level 3 BLAS functions. However, this weakness of the TLR methods is amenable to the future optimized multiplication routine between TLR and dense matrices. Furthermore, hierarchical methods are more sensitive to strong correlation, demanding higher truncation precision and hence, higher local ranks when the covariance matrix approaches singularity. After using the (iterative) block-reordering preconditioner, the TLR methods reach even lower estimation error with one-tenth of the previous sample size, shortening the computation time by up to one order of magnitude compared with the hierarchical methods. Additionally, the iterative block reordering has slightly bigger improvement on the Monte Carlo convergence rate than the non-iterative one, which is seen more clearly in Fig. 3. It is worth mentioning that the block reorderings are more effective when the MVN/MVT problem is more asymmetric. An extreme scenario where reorderings

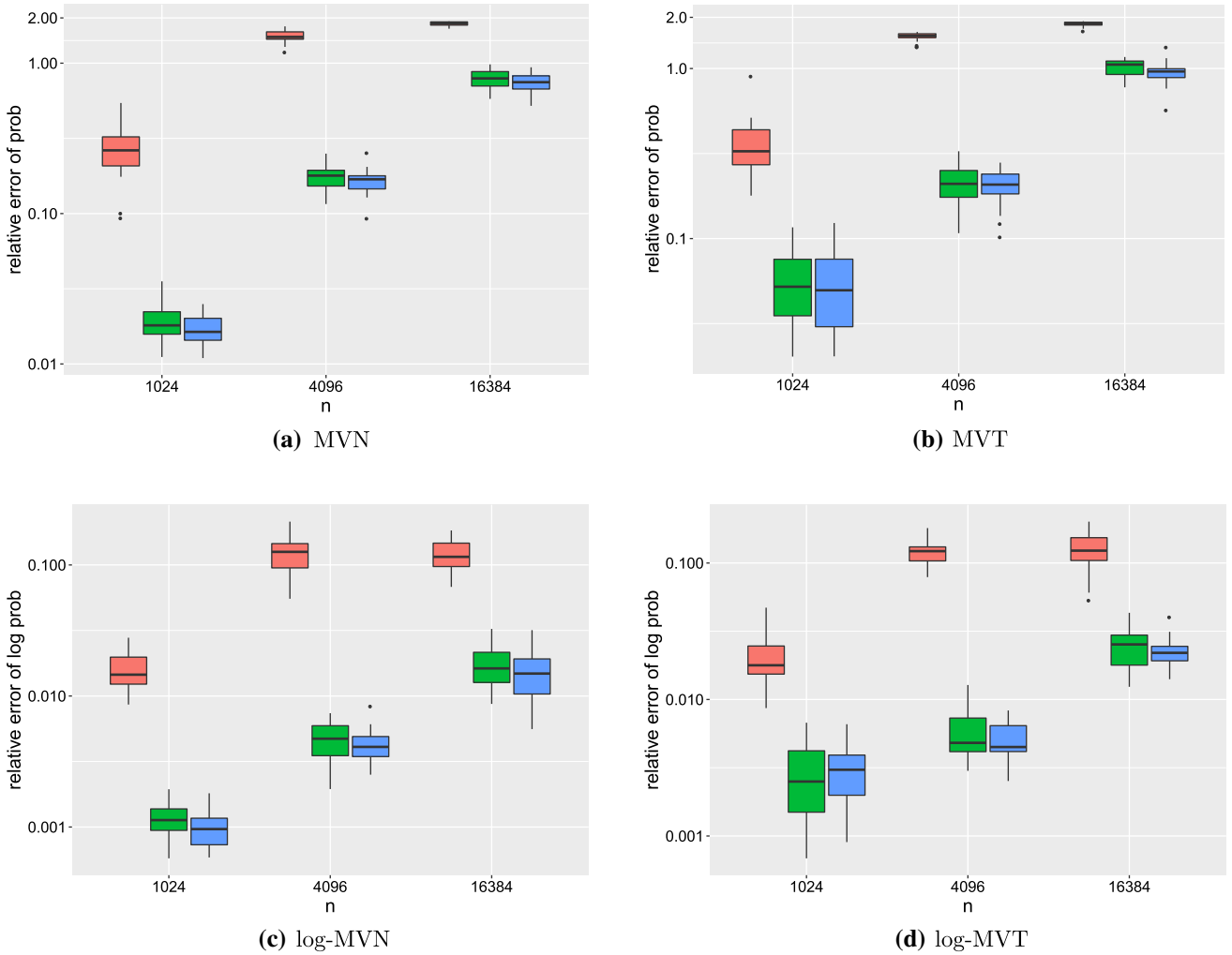


Fig. 3 Relative error for probabilities and log-probabilities. For each n, the three boxplots, from left to right, correspond to the TLR method, the TLR with the block reordering method, and the TLR with the iterative block-reordering method. The relative error for log-probabilities is based on 10 estimations of the same replicate. Each boxplot consists of 20 replicates. The covariance matrix is generated from a 2D exponential kernel, $\exp(-\|\mathbf{h}\|/\beta)$, where $\mathbf{h}$ is the distance vector, based on n locations on a perturbed grid in the unit square. The lower integration limits are $-\infty$ and the upper limits are independently generated from $N(4.0, 1.5^2)$. The degrees of freedom ν for MVT probabilities are 10. The QMC sample size is 10^4

become ineffective is that the correlation is constant and the integration limits are the same across the variables.

MVN and MVT probabilities much smaller than those in Table 3 may appear in high-dimensional applications (Botev 2017). For example, the model and data used in Sect. 5.3 produce a likelihood smaller than 10^{-40}. Overall, the convergence rate decreases if the integration region is pushed towards the tail while keeping the covariance structure unchanged, for which standard scientific workstations may ultimately fail to reach the desired accuracy using a reasonably large sample size N. For example, when $n = 65{,}536$ and $\beta = 0.03$, the MVN methods without block reordering are unable to control the relative error with 10^4 QMC samples, rendering even the most significant digit of the probability estimate unreliable. In Fig. 3, we keep the lower integration limits unchanged but use smaller and more dispersed upper integration limits to visualize one example where 10^4 QMC samples are insufficient for keeping the relative errors low. The dense and hierarchical methods are not included because the non-preconditioned methods should have the same error level when using the same QMC sample size. Figure 3 shows that all methods have a relative error close to or greater than one in 16, 384 dimensions. Nonetheless, the relative error grows more slowly with n for the preconditioned methods, with the iterative block reordering slightly more effective than the non-iterative one. A second observation is that the relative errors of the estimated log-probabilities are on a much smaller magnitude, indi-

cating right-skewness in the distribution of the MVN/MVT probability estimates. Therefore, we may still trust the magnitude of the probability estimates when the relative error is approaching one. The relative errors listed in this paper are the ratios of the Monte Carlo standard errors to the means of the Monte Carlo estimates, and for log-probabilities whose errors are not directly available, we estimate the same problem 10 times to provide replicates of the estimation.

5 Application to stochastic generators

5.1 A skew-normal stochastic generator

One area that benefits from the methods developed in this paper is the likelihood estimation for the statistical models whose probability density function (PDF) involves the MVN/MVT (cumulative) probabilities, and in this section, we use a skew-normal stochastic generator to demonstrate this advantage. 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 become 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 significance of the MVN and MVT methods in this context is an improvement in flexibility by introducing skewness since the majority of statistical models are elliptical. 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 density function by its CDF. The second method introduces skewness via selection, i.e., $(\mathbf{X}^\top, \mathbf{Y}^\top)^\top$ are jointly elliptical and $\mathbf{X} \mid \mathbf{Y} > \boldsymbol{\mu}$ is a skewed elliptical distribution, where $\boldsymbol{\mu}$ is the skewness parameter. Arellano-Valle and Genton (2010) studied the link between PDF reformulation and selection distributions. The third method introduces skewness through constructing a stochastic representation, typically, $\mathbf{Z} = \mathbf{X} + |\mathbf{Y}|$, 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 the third construction and used the Monte Carlo EM (Levine and Casella 2001) algorithm for model selection to avoid the intractable likelihood function.

We use the third method to construct a skew-normal stochastic generator, given its intuitive representation and simplicity for simulation. However, models in this category have intractable PDFs when $\mathbf{Y}$ has a non-trivial covariance structure. To avoid this complexity, we use a diagonal covariance structure for $\mathbf{Y}$ while adding a coefficient matrix to $|\mathbf{Y}|$, whose PDF is tractable based on the properties of $\mathscr{C}$ random vectors developed in Arellano-Valle et al. (2002). Specifically, 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 $\mathbf{Y}$, $G(\mathbf{X}, \mathbf{Y}) \mid \mathbf{Y} > \mathbf{0}$ has the same distribution as $G(\mathbf{X}, |\mathbf{Y}|)$ for any function $G(\cdot)$. We propose the following stochastic generator that has the flexibility of modeling the correlation between the elliptical component and the skewness component:

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

Here, $\xi \in \mathbb{R}$ is the location parameter, $\mathbf{X}$ and $\mathbf{Y}$ are independent standard normal random vectors, and $\mathbf{A}$ and $\mathbf{B}$ are parameter matrices. Choosing $G(\mathbf{X}, \mathbf{Y})$ to be $\mathbf{AX} + \mathbf{BY}$, the PDF of $\mathbf{Z}^*$ is tractable and is written explicitly in Eq. (6) with further parameterizations for $\mathbf{A}$ and $\mathbf{B}$.

As a trade-off of the tractable PDF, Eq. (5) may have certain drawbacks for modeling skewed distributions. Firstly, its extension to other skewed elliptical distributions, including the skewed MVT distribution, is not straightforward because the sum of independent elliptical distributions may no longer belong to the same distribution family. Secondly, $\mathbf{Z}^*$ is typically not a random field for most parameterizations of $\mathbf{A}$ and $\mathbf{B}$, which makes the inference at new locations difficult. Fortunately, for stochastic generators, the model is usually simulated on a fixed spatial domain without the need for prediction at unknown locations. In this section, we show that the preconditioned TLR QMC method is more suitable for estimating the parameters of Eq. (5) than two other state-of-the-art methods and that the fitted skew-normal model is a more realistic stochastic generator for the wind data in Saudi Arabia than the Gaussian random field.

5.2 Estimation with simulated data

In the spirit of parsimony, $\mathbf{A}$ is assumed to be the lower Cholesky factor from an exponential kernel, $\sigma_1^2 \exp(-\|\mathbf{h}\|/\beta_1)$ and $\mathbf{B}$ is assumed as the covariance matrix from another exponential kernel, $\sigma_2^2 \exp(-\|\mathbf{h}\|/\beta_2)$, both generated from n pre-specified locations on the 2D plane. Hence, there are five parameters in total, $(\xi, \sigma_1, \beta_1, \sigma_2, \beta_2)$. The assumption on $\mathbf{A}$ makes $\mathbf{AX}$ a MVN distribution, in fact, Gaussian random field, with an exponential covariance structure while $\mathbf{B}$

is parameterized as a covariance matrix, not a Cholesky factor, for two reasons. Firstly, the row sums of a lower Cholesky factor have great variability in their magnitudes, causing different levels of skewness among the random variables in $\mathbf{Z}^*$. Secondly, due to the first reason, the likelihood would depend on the ordering of the random variables. When $\mathbf{B}$ is a covariance matrix, the row sums usually have similar magnitudes and the likelihood function becomes independent from the ordering within $\mathbf{Z}^*$. The PDF of $\mathbf{Z}^* = \mathbf{z}$ can be derived based on the properties of $\mathscr{C}$ random vectors:

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

with $\mathbf{C} = \mathbf{A}^{-1}\mathbf{B}$.

To simulate $\mathbf{Z}^*$, we first generate n locations on a perturbed grid that expands with n. Specifically, for $n = 4^r$ and $r = 4, 5, 6, 7$, a regular grid in $\mathbb{R}^2$ of dimensions $2^r \times 2^r$ is first generated with the unit distance of $1/15$. Then independent disturbances uniformly distributed in $(0, 0.8/15)$ are added to all locations on the grid in both axis orientations to form the perturbed grid, based on which $\mathbf{A}$ and $\mathbf{B}$ are constructed with $(\sigma_1 = 1.0, \beta_1 = 0.3)$ and $(\sigma_2 = 1.0, \beta_2 = 0.3)$, respectively. Notice that the ordering of the n locations does not affect the probabilistic distribution of $\mathbf{Z}^*$. Finally, ξ is assumed zero without loss of generality and $\mathbf{Z}^*$ is generated based on Eq. (5). For each realization of $\mathbf{Z}^*$, $\mathbf{z} \in \mathbb{R}^n$, we use the Controlled Random Search (CRS) with local mutation (Kaelo and Ali 2006), a global optimization algorithm without gradient usage, to estimate the five parameter values that maximizes Eq. (6). In the optimization, the maximum number of iterations is 1,000, the searching ranges for $\{\xi, \sigma_1, \beta_1, \sigma_2, \beta_2\}$ are $\{(-1.0, 1.0), (0.1, 2.0), (0.01, 0.9), (0.0, 1.0), (0.01, 0.3)\}$, respectively, and the initial values are the lower bounds of the searching ranges. For each n, thirty independent realizations are generated, producing a total of thirty estimation results, which are combined into boxplots in Fig. 4. Overall, the estimation appears asymptotically unbiased and converges to the true values as the dataset dimension n increases. The outliers indicate that there is sometimes a local maximum, with big (σ_1, β_1) and small σ_2, and hence, the fitted skew-normal model is close to a Gaussian random field.

As benchmarks, we consider two other state-of-the-art methods frequently used when the likelihood is not tractable or computationally demanding, namely the Monte Carlo EM (MCEM) (Levine and Casella 2001) and the variational Bayes (SGVB) algorithms (Kingma and Welling 2013). In MCEM, sampling techniques, represented by Markov chain Monte Carlo (MCMC) methods, are used in the E-step to approximate the intractable expectation, in our case $E_{\mathbf{Y}|\mathbf{Z}^*,\boldsymbol{\theta}}[\log f_{\mathbf{Z}^*,\mathbf{Y}|\boldsymbol{\theta}}(\mathbf{z}, \mathbf{y})]$, through sampling from the posterior distribution of the latent random vector, $f_{\mathbf{Y}|\mathbf{Z}^*,\boldsymbol{\theta}}(\mathbf{y})$. Here, we use $f(\cdot)$ and $\boldsymbol{\theta}$ as the general notations for the density function and model parameters, respectively. However, sampling with MCMC can be very expensive and furthermore, this sampling procedure should ideally be repeated in each iteration during optimization, for which only the estimation with $n = 256$ observed locations is performed for our comparison, whose estimated parameters are included in Fig. 5. We use the Hamiltonian Monte Carlo method (Hoffman and Gelman 2014) with a burn-in size of 3,000 and a total sample size of 4,000 to sample $f_{\mathbf{Y}|\mathbf{Z}^*,\boldsymbol{\theta}}(\mathbf{y})$ and compute $E_{\mathbf{Y}|\mathbf{Z}^*,\boldsymbol{\theta}}[\log f_{\mathbf{Z}^*,\mathbf{Y}|\boldsymbol{\theta}}(\mathbf{z}, \mathbf{y})]$, which amounts to the logarithm of the MVN PDF. In the M-step, we use the BFGS (Nocedal 1980) algorithm to optimize the five parameters until convergence. We set an upper limit of 10 iterations between the E-steps and the M-steps, under which the EM algorithm typically cannot reach convergence. To compensate for this, the initial parameter values are set to their true values, under which the MCEM's estimation quality is comparable to that of the maximum likelihood estimators, but its computation cost is significantly higher as discussed later in this section.

The variational Bayes algorithm maximizes the variational lower bound as a surrogate of the marginal probability based on an approximate posterior distribution, often variable-wise independent, of the latent random vector $(\mathbf{Y} \mid \mathbf{Z}^*, \boldsymbol{\theta})$; see Kingma and Welling (2013) for more details. Its error from the true marginal probability is the Kullback–Leibler (KL) divergence between the approximate and the true distributions of $(\mathbf{Y} \mid \mathbf{Z}^*, \boldsymbol{\theta})$, changing with $\boldsymbol{\theta}$ and hence, the optimal parameter values for the marginal probability can be different from those for the variational lower bound. As in Kingma and Welling (2013), we approximate $f_{\mathbf{Y}|\mathbf{Z}^*,\boldsymbol{\theta}}(\mathbf{y})$ with $q_{\mathbf{Y}}(\mathbf{y}) = \prod_{i=1}^n \phi(y_i - \tilde{\xi}_i; \tilde{\sigma}_i^2)$ and repeat the simulation study above by maximizing the variational lower bound:

$$-D_{KL}\left\{q_{\mathbf{Y}}(\mathbf{y}) \middle\| f_{\mathbf{Y}|\boldsymbol{\theta}}(\mathbf{y})\right\} + \int \log\{f_{\mathbf{Z}^*|\mathbf{Y},\boldsymbol{\theta}}(\mathbf{z})\} q_{\mathbf{Y}}(\mathbf{y}) \mathrm{d}\mathbf{y}, \quad (7)$$

where $\tilde{\boldsymbol{\xi}}$ and $\tilde{\boldsymbol{\sigma}}$ are optimization parameters of length n and D_{KL} is the KL divergence. The KL divergence in Eq. (7) is tractable while the integration part is evaluated with Monte Carlo samples of $\mathbf{Y}$, which has the same order of complexity as the dense QMC method for MVN probabilities. Given a sample set of $\mathbf{Y}$, Eq. (7) can be differentiated with respect to its parameters $(\xi, \sigma_1, \beta_1, \sigma_2, \beta_2, \tilde{\boldsymbol{\xi}}, \tilde{\boldsymbol{\sigma}})$ explicitly, allowing faster convergence to the local maximum. Similarly to the MCEM algorithm, we also use the gradient-based BFGS algorithm for the optimization with two sets of initial values for $(\xi, \sigma_1, \beta_1, \sigma_2, \beta_2)$, one at their true values and one randomly chosen from their searching ranges, to show that the true values may not be the global minimizer of Eq. (7). The initial values for $\tilde{\boldsymbol{\xi}}$ and $\tilde{\boldsymbol{\sigma}}$ are $\mathbf{0}$ and $\mathbf{1}$, respectively. In

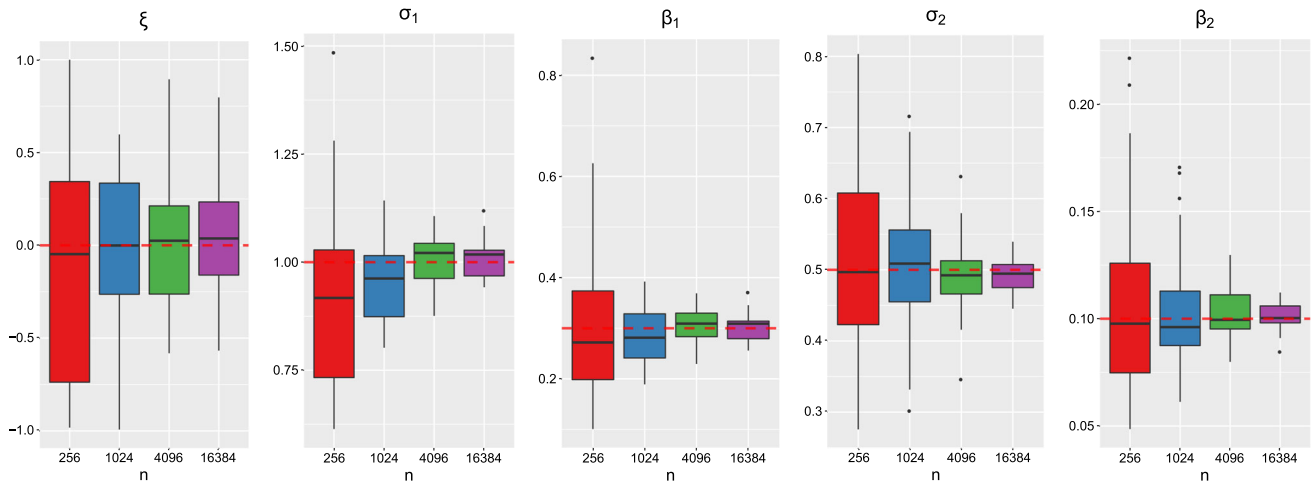


Fig. 4 Boxplots of 30 estimation results from the maximum likelihood estimators. Each estimation is based on one realization of the n-dimensional skew-normal model. The red dashed line marks the true value used for generating random vectors from the skew-normal model. (Color figure online)

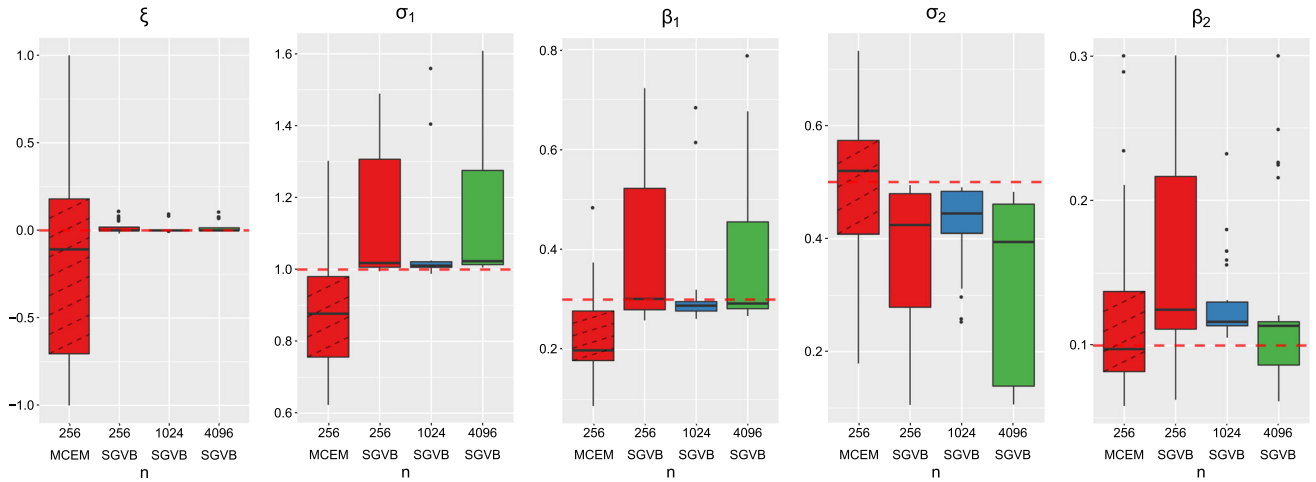


Fig. 5 Boxplots of 30 estimation results from the Monte Carlo EM (MCEM) estimators (shaded with dashed lines) and the variational Bayes (SGVB) estimators. Each estimation is based on one realization of the n-dimensional skew-normal model. The red dashed line marks the true value used for generating random vectors from the skew-normal model. (Color figure online)

principle, a different starting point can be also added to the MCEM algorithm, but this may not be necessary since the experiment for MCEM is confined to $n = 256$ due to high computation costs and its estimation bias is already visible from Fig. 5. Thirty estimation results for each n are summarized in Fig. 5, indicating bias of the variational Bayes estimators as well. Furthermore, the estimation is not necessarily improved after increasing n, raising doubts on the consistency of the estimators.

Table 4 compares the computation costs of the MLE with dense or TLR matrix representations, the MCEM algorithm and the SGVB algorithm. In terms of per optimization iteration, the MLE using the TLR matrix representation ('MLE-TLR') is the fastest, but both MLE methods cannot utilize the numerical gradient because the MVN probability estimates are not sufficiently accurate, for which they may need more iterations to converge. Notice that matrix operations, specifically those $O(n^3)$ matrix operations required by Eq. (6), account for the major cost of the MLE methods, and hence, the 'MLE-TLR' method is further amenable to optimized linear algebra libraries for TLR matrices. The MCEM algorithm overall has the highest computation cost due to the MCMC sampling in the E-step, for which its applicability is limited to small dataset dimensions. The SGVB algorithm converges fastest among all and its cost can be further reduced by low-rank matrix representations. However, it may oversimplify the model, causing estimation bias and inconsistency. We proceed with the 'MLE-TLR' method for the study of wind data in Saudi Arabia because of its computation feasibility and higher estimation quality.

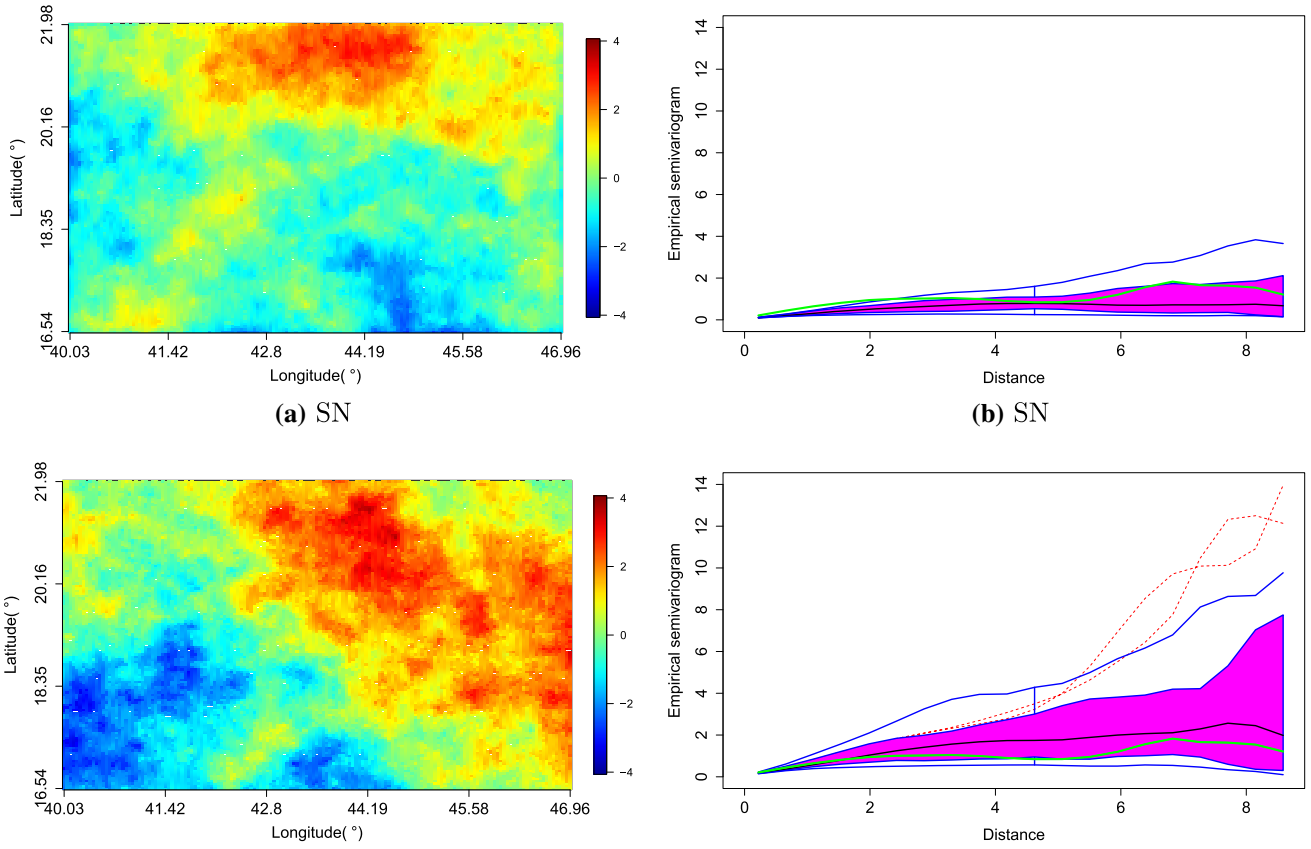
Table 4 Computation time (seconds) per iteration in the maximum likelihood estimation

	$n = 256$	$n = 1{,}024$	$n = 4{,}096$	$n = 16{,}384$
MCEM	4,209s	N.A.	N.A.	N.A.
SGVB	0.11s	2.7s	68s	1206s
MLE-Dense	(0.31s, 0.28s)	(3.0s, 1.7s)	(118s, 19s)	(10,784s, 541s)
MLE-TLR	(0.81s, 0.67s)	(6.5s, 3.9s)	(48s, 17s)	(533s, 104s)

Both Monte Carlo EM (MCEM) and Variational Bayes (SGVB) optimize the approximated likelihood in each iteration while MLE-Dense and MLE-TLR optimize the exact likelihood of Eq. (6) with dense and TLR matrix representations, respectively. For the paired times in MLE-Dense and MLE-TLR, the left is the average time per iteration while the right is the time for computing the MVN probability

Table 5 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.



(a) SN **(b)** SN

(c) GRF **(d)** GRF

Fig. 6 Heatmap based on one simulation and the functional boxplot of the empirical semivariogram based on 100 simulations. 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. (Color figure online)

Table 6 Empirical moments and BIC comparison

	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

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 simulations

5.3 Estimation with wind data from Saudi Arabia

The dataset we use for modeling is the daily wind speed over a region in the Kingdom of Saudi Arabia on August 5, 2013, produced by the WRF model (Yip 2018), 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 ranges from 40.034 to 46.960 and the latitude ranges 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 Eq. (5). It is worth noting that $\mathbf{B}|\mathbf{Y}|$ has a negative skewness under our parameterization.

Same as in Sect. 5.2, the likelihood function is Eq. (6) and the optimization parameters are $(\xi, \sigma_1, \beta_1, \sigma_2, \beta_2)$. To highlight the contribution of the skewness flexibility, we compare the skew-normal model with the classical Gaussian random field, which is also a special case of the former with $\sigma_2 = 0$. Thus, the Gaussian random field has three optimization parameters (σ_1, β_1, ξ). The initial parameter values, searching ranges, and optimized values are summarized in Table 5, where certain lower bounds are above zero to prevent singularity. We first compare the two fitted models with the functional boxplots (Sun and Genton 2011) of the empirical semivariogram based on 100 simulations, which is shown in Fig. 6. The skew-normal model has significantly smaller band width than the Gaussian random field while both cover the empirical semivariogram of the original data. Two heatmaps of the fitted models are also shown in Fig. 6, but their distinction is not as obvious as in the functional boxplots. Next, based on the same 100 simulations, we compare the quantile intervals of the empirical moments in Table 6, where we also include the BIC values to indicate that the skew-normal model is a better fit. Except for variance, the two models have similar quantile intervals that contain the moments of the wind dataset, but since the empirical moments ignore the correlation between the spatial locations, they may not measure the fitting quality in a comprehensive manner.

6 Conclusion

In this paper, we first summarized the SOV methods from Genz (1992) and Genz and Bretz (1999) for MVN and MVT probabilities. Two definitions of the MVT probability were compared, and one was shown to have better numerical properties than the other. Next, we demonstrated that the TLR structure (Weisbecker 2013; Mary 2017; Akbudak et al. 2017) is more aligned with variable reordering than the HODLR structure used in Genton et al. (2018) as well as the hierarchical structure under the standard admissibility condition, allowing it to benefit from both a reduced cost per sample and an improved convergence rate. Additionally, we introduced an iterative version of the block reordering proposed in Cao et al. (2019) that further improves the convergence rate and produces the TLR Cholesky factor simultaneously. A third contribution is the observation that when the estimation errors are of the same magnitude as the probability estimates, we can still trust the magnitudes of estimates to a certain extent, e.g., for the maximum likelihood estimation of a high-dimensional skew-normal model.

Acknowledgements The authors thank Prof. Georgiy Stenchikov at KAUST for providing the WRF data and the two anonymous reviewers for valuable comments that improved this manuscript.

References

Akbudak, K., Ltaief, H., Mikhalev, A., Keyes, D.: Tile low rank Cholesky factorization for climate/weather modeling applications on manycore architectures. In: International Supercomputing Conference, pp. 22–40. Springer (2017)

Arellano-Valle, R., del Pino, G., San Martín, E.: Definition and probabilistic properties of skew-distributions. Stat. Probab. Lett. **58**, 111–121 (2002)

Arellano-Valle, R.B., Branco, M.D., Genton, M.G.: A unified view on skewed distributions arising from selections. Can. J. Stat. **34**, 581–601 (2006)

Arellano-Valle, R.B., Genton, M.G.: Multivariate unified skew-elliptical distributions. Chil. J. Stat. **1**, 17–33 (2010)

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

Azzimonti, D., Ginsbourger, D.: Estimating orthant probabilities of high-dimensional Gaussian vectors with an application to set estimation. J. Comput. Graph. Stat. **27**, 255–267 (2018)

Bolin, D., Lindgren, F.: Excursion and contour uncertainty regions for latent Gaussian models. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **77**, 85–106 (2015)

Börm, S., Grasedyck, L., Hackbusch, W.: Introduction to hierarchical matrices with applications. Eng. Anal. Boundary Elem. **27**, 405–422 (2003)

Botev, Z.I.: The normal law under linear restrictions: simulation and estimation via minimax tilting. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **79**, 125–148 (2017)

Boukaram, W., Turkiyyah, G., Keyes, D.: Hierarchical matrix operations on GPUs: matrix-vector multiplication and compression. ACM Trans. Math. Softw. **45**, 3:1–3:28 (2019)

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

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

Castruccio, S., Genton, M.G.: Principles for statistical inference on big spatio-temporal data from climate models. Stat. Probab. Lett. **136**, 92–96 (2018)

Durante, D.: Conjugate Bayes for probit regression via unified skew-normal distributions. Biometrika **106**, 765–779 (2019)

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

Genton, M.G., Keyes, D.E., Turkiyyah, G.: Hierarchical decompositions for the computation of high-dimensional multivariate normal probabilities. J. Comput. Graph. Stat. **27**, 268–277 (2018)

Genz, A.: Numerical computation of multivariate normal probabilities. J. Comput. Graph. Stat. **1**, 141–149 (1992)

Genz, A., Bretz, F.: Numerical computation of multivariate t-probabilities with application to power calculation of multiple contrasts. J. Stat. Comput. Simul. **63**, 103–117 (1999)

Genz, A., Bretz, F.: Comparison of methods for the computation of multivariate t probabilities. J. Comput. Graph. Stat. **11**, 950–971 (2002)

Genz, A., Bretz, F.: Computation of Multivariate Normal and t Probabilities, vol. 195. Springer, Berlin (2009)

Grasedyck, L., Kriemann, R., Le Borne, S.: Parallel black box $\mathcal{H}$-LU preconditioning for elliptic boundary value problems. Comput. Vis. Sci. **11**, 273–291 (2008)

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

Hoffman, M.D., Gelman, A.: The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. J. Mach. Learn. Res. **15**, 1593–1623 (2014)

Jeong, J., Castruccio, S., Crippa, P., Genton, M.G., et al.: Reducing storage of global wind ensembles with stochastic generators. Ann. Appl. Stat. **12**, 490–509 (2018)

Kaelo, P., Ali, M.: Some variants of the controlled random search algorithm for global optimization. J. Optim. Theory Appl. **130**, 253–264 (2006)

Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2013). arXiv preprint arXiv:1312.6114

Kriemann, R.: Parallel-matrix arithmetics on shared memory systems. Computing **74**, 273–297 (2005)

Levine, R.A., Casella, G.: Implementations of the Monte Carlo EM algorithm. J. Comput. Graph. Stat. **10**, 422–439 (2001)

Mary, T.: Block low-rank multifrontal solvers: complexity, performance, and scalability, Ph.D. thesis (2017)

Nocedal, J.: Updating quasi-Newton matrices with limited storage. Math. Comput. **35**, 773–782 (1980)

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

Schervish, M.J.: Algorithm AS 195: Multivariate normal probabilities with error bound. J. R. Stat. Soc.: Ser. C (Appl. Stat.) **33**, 81–94 (1984)

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

Sun, Y., Genton, M.G.: Functional boxplots. J. Comput. Graph. Stat. **20**, 316–334 (2011)

Trinh, G., Genz, A.: Bivariate conditioning approximations for multivariate normal probabilities. Stat. Comput. **25**, 989–996 (2015)

Weisbecker, C.: Improving multifrontal solvers by means of algebraic block low-rank representations, Ph.D. thesis (2013)

Yip, C.M.A.: Statistical characteristics and mapping of near-surface and elevated wind resources in the Middle East, Ph.D. thesis, King Abdullah University of Science and Technology (2018)

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