Received: 5 March 2024 | Revised: 18 May 2024 | Accepted: 24 May 2024

DOI: 10.1002/env.2868

RESEARCH ARTICLE

WILEY

On the impact of spatial covariance matrix ordering on tile low-rank estimation of Matérn parameters

Sihan Chen[1] | **Sameh Abdulah**[2] | **Ying Sun**[1,2] | **Marc G. Genton**[1,2]

[1]Statistics Program, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia

[2]Extreme Computing Research Center (ECRC), King Abdullah University of Science and Technology, Thuwal, Saudi Arabia

Correspondence
Sihan Chen, Statistics Program, King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia.
Email: sihan.chen@kaust.edu.sa

Funding information
King Abdullah University of Science and Technology

Abstract

Spatial statistical modeling involves processing an $n \times n$ symmetric positive definite covariance matrix, where n denotes the number of locations. However, when n is large, processing this covariance matrix using traditional methods becomes prohibitive. Thus, coupling parallel processing with approximation can be an elegant solution by relying on parallel solvers that deal with the matrix as a set of small tiles instead of the full structure. The approximation can also be performed at the tile level for better compression and faster execution. The tile low-rank (TLR) approximation has recently been used to compress the covariance matrix, which mainly relies on ordering the matrix elements, which can impact the compression quality and the efficiency of the underlying solvers. This work investigates the accuracy and performance of location-based ordering algorithms. We highlight the pros and cons of each ordering algorithm and give practitioners hints on carefully choosing the ordering algorithm for TLR approximation. We assess the quality of the compression and the accuracy of the statistical parameter estimates of the Matérn covariance function using TLR approximation under various ordering algorithms and settings of correlations through simulations on irregular grids. Our conclusions are supported by an application to daily soil moisture data in the Mississippi Basin area.

KEYWORDS

covariance function, Gaussian process, ordering algorithms, statistical modeling, tile low-rank (TLR) approximation

1 | INTRODUCTION

Spatial statistics is an important branch of statistics that has applications in various research fields, for instance, environmental science (Sun et al., 2015), geography (Kim et al., 2023), economics (Elhorst et al., 2021), epidemiology (Moraga & Montes, 2011), and neuroscience (Ombao et al., 2008), to name but a few. A common way to model spatial data is to consider them as realizations of a Gaussian random field. Suppose we have n spatial locations $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n \in \mathbb{R}^d$ for some $d \in \mathbb{Z}^+$, with $n \in \mathbb{Z}^+$. Denote the observations at these n locations by $\boldsymbol{Z} = \{Z(\boldsymbol{s}_1), \ldots, Z(\boldsymbol{s}_n)\}^\top$. We assume that the distribution of these observations is jointly Gaussian with mean $\mathbb{E}\{Z(\boldsymbol{s})\} = \mu(\boldsymbol{s})$ and covariance parametrized by $\boldsymbol{\theta} \in \mathbb{R}^q$

[Correction added on 11 September 2024, after first online publication: The copyright line was changed.]

for some $q \in \mathbb{Z}^+$: $\mathrm{Cov}\{Z(\boldsymbol{s}_1), Z(\boldsymbol{s}_2)\} = C(\boldsymbol{s}_1, \boldsymbol{s}_2; \boldsymbol{\theta})$. There are many valid methods established for modeling the covariance function $C(\boldsymbol{s}_1, \boldsymbol{s}_2; \boldsymbol{\theta})$; see, for example, Gneiting et al. (2007), Wiens et al. (2020), Chen et al. (2021) and Porcu and White (2022).

Maximum Likelihood Estimation (MLE) is an essential technique for parameter estimation in geospatial data modeling. It hinges on maximizing a likelihood function, which measures how accurately the model reflects observed data. The MLE methodology entails constructing an $n \times n$ covariance matrix, $\boldsymbol{\Sigma}(\boldsymbol{\theta})$, pivotal in defining the correlations between observations at different spatial locations across single or multiple time slots. Under the setting of a Gaussian random field with a single time slot, we form the covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ by letting

$$\boldsymbol{\Sigma}_{i,j}(\boldsymbol{\theta}) = C(\boldsymbol{s}_i, \boldsymbol{s}_j; \boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{\Sigma}_{i,j}(\boldsymbol{\theta})$ denotes the (i,j)th entry of $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ for any $1 \le i,j \le n$. Thus, we have the following expression of the Gaussian log-likelihood function for the observation vector $\boldsymbol{Z}$:

$$\ell(\boldsymbol{\theta}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}(\boldsymbol{\theta})| - \frac{1}{2}\boldsymbol{Z}^\top \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\boldsymbol{Z}. \tag{2}$$

Here $|\boldsymbol{\Sigma}(\boldsymbol{\theta})|$ denotes the determinant of $\boldsymbol{\Sigma}(\boldsymbol{\theta})$, and the MLE is obtained by maximizing the log-likelihood function $\ell(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$.

However, as we can see from the expression, evaluating the function $\ell(\boldsymbol{\theta})$ requires computing the inverse of the covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$, with the time complexity $O(n^3)$. Although estimating the parameter $\boldsymbol{\theta}$ using the MLE is a classical way to understand the correlation structure of spatial data, the cubic computational complexity associated with the MLE renders its dense computation impractical with large n. This challenge is amplified by the availability of large-scale spatial data, where the location count can soar into millions or even hundreds of millions, particularly in cases involving high-resolution data.

Consequently, recent research has focused on developing advanced approximation methods capable of handling large geospatial datasets while maintaining acceptable accuracy in the modeling process. These methods aim to offer more efficient alternatives for working with extensive spatial data without significantly compromising the quality of the results. For example, the covariance tapering method (Kaufman et al., 2008) sparsifies the covariance matrices by ignoring the correlations between locations with large distances and setting them to zero to accelerate the computation; Cressie and Johannesson (2008) proposed a method that uses several nonstationary covariance functions to perform spatial prediction for large-scale spatial data; Banerjee et al. (2008) proposed the Gaussian Predictive Processes (GPP), which projects the original problem to a subspace containing a set of spatial locations, to reduce the dimensionality of the spatial covariance matrix; the Mixed-Precision (MP) method (Cao et al., 2022) aims to accelerate the computation by keeping the most important values with the highest level of precision, while truncating the rest of the values to lower levels of precision, so that the computation can be faster without affecting the accuracy too much. Some other approximation methods can also be found in the literature, such as Gaussian Markov random field approximations (Rue & Held, 2005 & Rue & Tjelmeland, 2002), Kalman filtering (Sinopoli et al., 2004) and low-rank splines (Kim & Gu, 2004), to name but a few. Sun et al. (2012) provide a comprehensive overview of approximation methods on large-scale spatial data.

Tile Low-Rank (TLR) approximation (Akbudak et al., 2017) is one of the novel approximation methods. It employs low-rank approximation to the covariance matrix and facilitates parallel processing via a task-based parallelism mechanism. This approach significantly accelerates the evaluation of the likelihood function for a large number of locations on modern parallel hardware architectures where these task-based algorithms are optimized. In the TLR framework, low-rank compression is applied on individual tiles rather than the entire matrix. This strategy enables the distribution of both compression and computational tasks across various processing units, for example, CPUs or GPUs, enhancing efficiency and scalability. The TLR approximation approach exploits the data sparsity of a given covariance matrix by compressing the off-diagonal tiles up to a user-defined accuracy threshold. In TLR approximation, the diagonal and off-diagonal tiles of the covariance matrices are stored and processed differently. The maximum rank among all the off-diagonal tiles is decisive on the performance, where smaller ranks lead to faster computations and less memory consumption. Since the approximation is applied at the tile level, ordering the covariance matrix elements can impact the compression level per tile. Therefore, one of the main requirements of TLR approximation is to spatially order the locations so that more correlated locations are stored together to allow better compression at the tile level.

Our work explores the accuracy and performance of geospatial data modeling using TLR approximation under varying orderings of locations. We use the *ExaGeoStat* software (Abdulah et al., 2018a) along with its corresponding *R* version *ExaGeoStatR* (Abdulah et al., 2023), which are developed to enable high-performance computing and analysis of spatial and spatio-temporal data on multicore and manycore systems. The ranks of the off-diagonal tiles can be affected by how we order the locations of the spatial data. Several multi-dimensional ordering methods have been proposed to sort the elements in a given $n \times n$-dimensional covariance matrix in the literature. Herein, we implemented several multi-dimensional ordering algorithms and assessed their quality for different covariance functions through an empirical study. We assess and compare the performance of TLR approximation method in estimating the Matérn covariance parameters using orderings with the Morton curve (Morton, 1966), the Hilbert curve (Hilbert, 1935), the k-dimensional (k-d) tree (Bentley, 1975), as well as the Maximum-Minimum Degree (MMD) Algorithm (Guinness, 2018), the Reversed Cuthill–McKee (RCM) Algorithm (Cuthill & McKee, 1969), and the Minimum Degree Algorithm (George & Liu, 1989), which we will describe in detail.

The rest of this article is organized as follows. Section 2 provides an overview of the TLR approximation method. Section 3 details the various ordering algorithms implemented in *ExaGeoStat*, along with the experiments conducted using them. Section 4 introduces the statistical models under consideration, which describe the parametrization of the covariance matrix and the log-likelihood function. Section 5 presents our experimental results, analyzing the performance of parameter estimation using the TLR algorithm and various ordering algorithms. This analysis focuses on estimation accuracy, ranks of the off-diagonal tiles in the covariance matrices, and execution time for Cholesky factorization on these matrices. An application to soil moisture data is also demonstrated, comparing results obtained with different ordering algorithms. The conclusion and discussion are provided in Section 6.

2 | TILE LOW-RANK (TLR) APPROXIMATION

Parallel processing, an advanced computing paradigm, leverages modern parallel architectures to accelerate computation. This is achieved by employing multiple processing units simultaneously executing different parts of a single algorithm. In most of the existing parallel linear algebra libraries, task-based parallelism is a prevalent approach for enhancing the efficiency of linear solvers. This method involves dividing the computational workload into discrete tasks that can be executed simultaneously. The central strategy is to divide the target matrix into smaller tiles, allowing each processing unit to handle a specific tile independently. The algorithm is thus conceptualized as a directed acyclic graph (DAG), where each node represents an individual task, and the connecting arrows indicate task dependencies. By utilizing runtime system libraries like OmpSs (Duran et al., 2011), StarPU (Augonnet et al., 2009), Charm++ (Kale & Krishnan, 1993), PaRSEC (Bosilca et al., 2013), and Kokkos (Edwards et al., 2014), a scheduler adeptly assigns work to various processing units, ensuring no dependencies are violated during execution.

In Akbudak et al. (2017), the proposal of the tile low-rank (TLR) approach for compressing the covariance matrix specifically in climate/weather applications was presented. This study aimed to synergize fast, parallel processing, task-based linear algebra solvers with approximation techniques on manycore systems. Its application and evaluation in spatial statistics have been discussed in Abdulah et al. (2018b) with accuracy evaluation for synthetic and real datasets.

TLR approximation is based on approximating each off-diagonal tile using a low-rank approximation method. The approach in Akbudak et al. (2017) utilizes Singular Value Decomposition (SVD) to derive two matrices, **U** and **V**, representing the original tile's singular vectors. Each tile is compressed to a specific rank denoted by r, defining one dimension of these matrices, while the other dimension corresponds to the tile size, nb. Practically, the tile size not only impacts the accuracy of the compression but also plays a critical role in the performance of linear solvers during runtime. Figure 1 illustrates an example of compressing a dense tile into the two matrices **U** and **V**.

Rank distribution also depends on how the coordinates in the covariance matrix are ordered. The ordering strategy can significantly influence the effectiveness of the TLR approximation, as it determines the structure and pattern of interactions within the matrix. Different ordering techniques can lead to varying compression efficiencies and ranks in the resulting tiles, thereby impacting the overall performance and accuracy of the TLR approximation. It is crucial to choose an appropriate ordering method that aligns with the specific characteristics and requirements of the data and the computational objectives at hand. In the subsequent section, we summarize various ordering algorithms from the literature studied in this work.

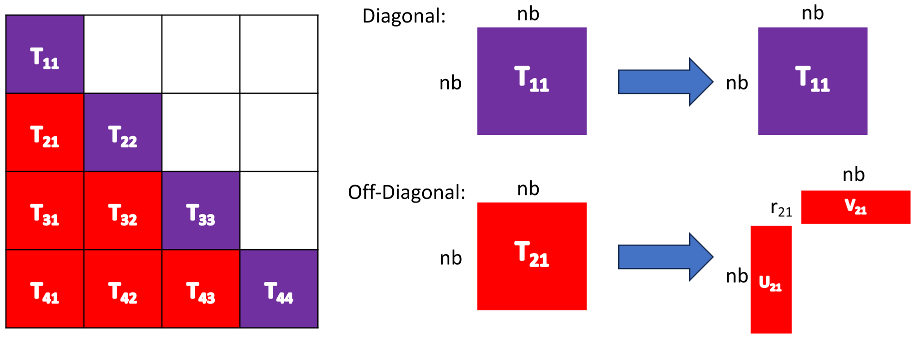

FIGURE 1 An illustration of the tile low-rank compression. The diagonal tiles are kept unchanged, while the off-diagonal tiles are compressed via singular value decomposition and then stored into smaller matrices, **U** and **V**, instead of the original dense matrix tiles of size $nb \times nb$. The rank of the tile T_{21} in this figure is r_{21}.

3 | SPATIAL ORDERING METHODS

Spatial ordering is crucial for ordering spatial data in many fields, including spatial statistics (Cerioli & Riani, 1999), computer science (Asano et al., 1997), and geography (Guo & Gahegan, 2006), to efficiently organize and manage spatial data. The main goal is to transform the spatial relations into a one-dimensional structure while preserving the spatial locality. In one-dimensional coordinate systems, such as those used for time series data, a natural order is dictated by the progression along the real line. This inherent ordering means that adjacent data points in a time series are typically more correlated, following a sequential arrangement based on time. However, this natural ordering does not exist in multi-dimensional coordinate systems. In these systems, coordinates extend across multiple dimensions and lack a straightforward, inherent sequence like their one-dimensional counterparts. Assuming creating a covariance matrix based on these data, large values will be mostly located near the diagonal, and the off-diagonal elements will tend to become closer to zero. However, for spatial data with multi-dimensional coordinates, such property is not granted if we order the locations randomly, and there is a risk of having many large values in the off-diagonal part of the covariance matrix. Therefore, choosing the order of the locations wisely can be a crucial step in optimizing the computation.

The TLR approximation operates based on a user-defined accuracy threshold, which dictates how many singular values are used for off-diagonal tile compression. TLR assumes that diagonal tiles are kept in a dense format. A smaller number of singular values results in more compressed tiles, albeit with a greater loss of information. This chosen number of singular values is termed "rank", denoted by r. Reducing r compresses the matrix, balancing compression and information retention. For the TLR approximation to be effective in terms of memory and computation, assuming that all tiles in the matrix are square and of size $nb \times nb$, the rank r should ideally be less than $2 \times (nb/2)$, to ensure that the number of elements in both $\boldsymbol{U}$ and $\boldsymbol{V}$ are less than $(nb \times nb)/2$.

Given a specific user-defined threshold, the arrangement of elements in the covariance matrix significantly impacts the ranks of off-diagonal tiles when using TLR approximation. Typically, there are two methods to decrease the ranks of the off-diagonal tiles:

- To reorder the n-dimensional locations before generating the covariance matrix. The goal is to ensure that locations adjacent in the final one-dimensional order are also neighboring in the original multi-dimensional space. As a result, the larger values in the covariance matrix will be clustered around the diagonal, leading to lower ranks for the off-diagonal tiles.
- To directly reorganize the covariance matrix based on the value of each entry. This approach will result in lower ranks for the off-diagonal tiles of the covariance matrix.

In this study, we integrated various ordering algorithms into the *ExaGeoStat* software to examine their effects on the accuracy and performance of the TLR approximation. Our evaluation is based on a range of spatial statistics covariance functions. The subsequent subsections provide a concise overview of the different ordering methods employed.

3.1 | Space-filling curves

Space-filling curves are unique in their ability to occupy any given space using a one-dimensional line, regardless of its dimension. This concept has been utilized for decades to transform points in high-dimensional spaces into a one-dimensional arrangement. Here, we consider a set of randomly selected points within a unit hypercube of dimension d, where $d \in \mathbb{Z}^+$. By applying a space-filling curve to encompass this unit hypercube, each point becomes associated with a specific position on the curve. Now, envision straightening this curve into a one-dimensional line. As the curve transforms, the points align along this line, establishing a sequential order. The Morton and Hilbert curves are notable examples of space-filling curves used for this ordering process.

3.1.1 | Morton curve

The Morton curve (Morton, 1966), or the Z-order curve, is a curve that covers all integers in the interior of the d-dimensional hypercube $[0, 2^p)^d$ of size 2^p. The curve is constructed by interleaving the binary representation of the coordinates of each point in the d-dimensional space. Each point is converted to a single scalar value and can be easily sorted using traditional algorithms.

An illustration of the Morton curve in the two-dimensional case is shown in Figure 2, which is a recursively Z-shaped curve linking the integer-valued coordinates in the two-dimensional plane. Higher-dimensional Morton curves are designed similarly.

As shown in Figure 2, it is evident that most adjacent points on the original two-dimensional grid are similarly positioned nearby along the Morton curve. This alignment underscores the objective of maintaining spatial ordering in the transformation process. Figure 2 also illustrates that the 'Z' patterns divide the space into 2×2 grids in the two-dimensional scenario. For any given location in this two-dimensional space, its corresponding one-dimensional index on the Morton curve can be determined by interleaving the binary values of its coordinates. This technique is adaptable for implementing Morton ordering in various dimensional spaces.

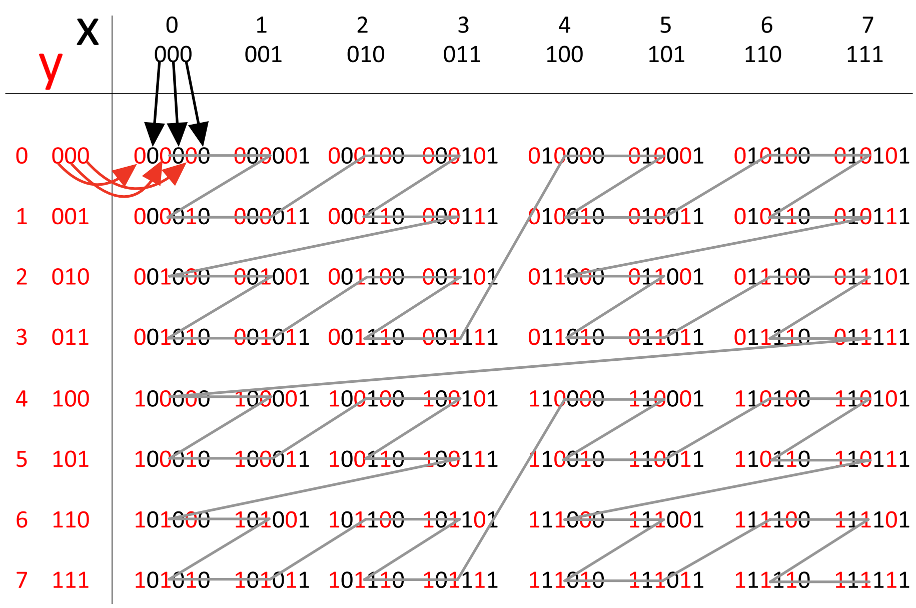

FIGURE 2 An illustration of the 2-d Morton curve that covers the interior of the two-dimensional hypercube $[0, 2^p)^2$, with $p = 3$, as well as its encoding with binary numbers. In our implementation, for each 2-d location, we find its corresponding 1-d index on the Morton curve by interleaving the binary digits of its 2-d coordinates, as shown above.

3.1.2 | Hilbert curve

Similar to the Morton curve, the Hilbert curve (Hilbert, 1935) is another curve that covers all integers in the interior of the d-dimensional hypercube $[0, 2^p)^d$ of size 2^p. The curve starts at one corner of the space grid and snakes through each element in a specific pattern. As the order of the curve increases, it becomes more complex, filling the space more densely. Hilbert curve aims to minimize the distance between the points close to each other in the multi-dimensional space. An illustration of the Hilbert curve in a two-dimensional case is shown in Figure 3.

We have integrated the Morton and Hilbert curve ordering algorithms into the *ExaGeoStat* software. For two-dimensional spatial locations, our implementations involve the following steps:

1. "Encoding": Convert the two-dimensional coordinates into integers. Specifically, if the location coordinates are initially in single-precision floating-point format (16 bits), we convert these into 16-bit unsigned integers. This is achieved by multiplying the coordinates by the maximum value allowable for unsigned integers in the system and then rounding the result to the nearest integer;
2. "Sorting": We compute their respective one-dimensional indices for each two-dimensional coordinate point on either the Morton or Hilbert curve. Following this calculation, we sort these indices using standard sorting algorithms like quicksort;
3. "Decoding": Convert the 1-d coordinates back into 2-d coordinates, then further convert back to floating point values in the unit square by dividing them by the maximum value of unsigned integers.

The time complexity for ordering n spatial locations using both Morton and Hilbert curves is identical. The encoding and decoding steps each have a time complexity of $O(n)$, while the sorting step incurs a time complexity of $O(n \log n)$. Consequently, the total time complexity for Morton and Hilbert orderings is $O(n \log n)$.

Moreover, it is important to acknowledge that when employing Morton and Hilbert orderings, converting floating-point values to integers through rounding and then reconverting them back to floating points results in coordinates that do not precisely match their original values. However, as demonstrated in our experimental findings in the subsequent section, the accuracy of parameter estimation remains largely unaffected by these minor discrepancies.

3.2 | *K*-dimensional (KD) tree

A k-dimensional tree (KD-Tree, Bentley (1975)) is a binary tree data structure that stores k-dimensional data where $k > 1$, similar to a binary search tree for 1-dimensional data. It can also be used for other purposes, and it has been widely applied to many different areas, not only in spatial data analysis. For n spatial locations, both construction and traversal

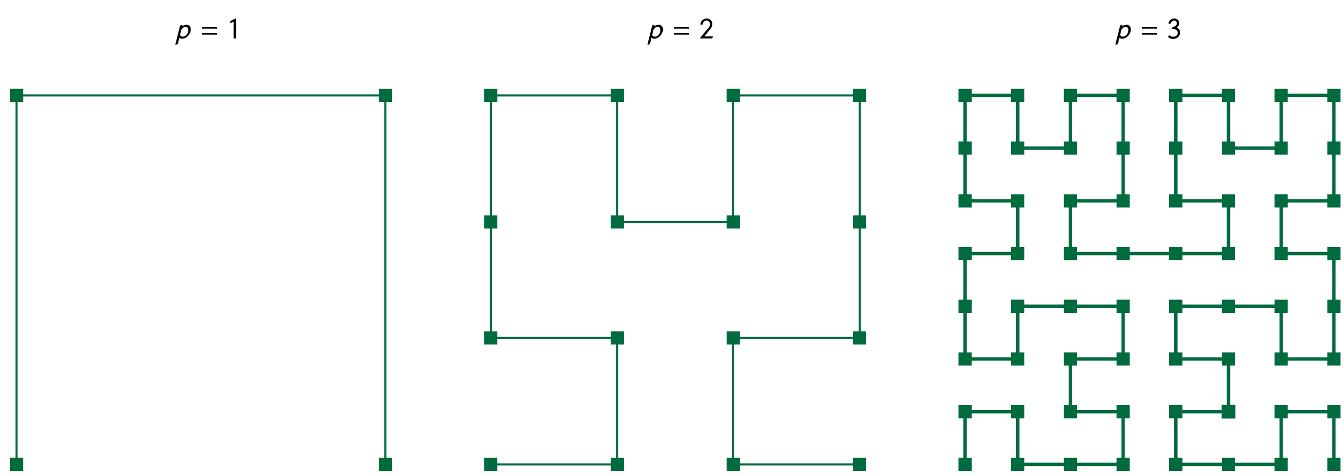

FIGURE 3 An illustration of the 2-d Hilbert curve that covers the interior of the two-dimensional hypercube $[0, 2^p)^2$, for $p = 1, 2, 3$ respectively, adapted from Skilling (2004).

of a KD-Tree are of time complexity $O(n \log n)$. Hence, the overall time complexity for KD-Tree ordering for n spatial locations is $O(n \log n)$.

In a KD-Tree, each non-leaf node stores an index of dimension i and a value m, and it functions as a hyperplane dividing the k-dimensional space into two half-spaces: all inputs with value v smaller than m on the ith dimension will be stored in a leaf in the left subtree of this non-leaf node, while inputs with value v larger than m on the ith dimension will be stored in a leaf in the right subtree of this non-leaf node. Each leaf node stores one input value v. To be specific, for a set of k-dimensional data denoted by $\mathcal{D}$, its KD-Tree is generated as follows:

1. First, denote the root node of the KD-Tree by R, find the range of the data in all of the k dimensions, and record the following two values in R: the index of the dimension which has the largest range, denoted by κ, and the median of the data in this dimension, denoted by med.
2. Next, divide $\mathcal{D}$ into $\mathcal{D}_l$ and $\mathcal{D}_r$ by the dimension and median recorded in the root, such that all the data points whose value on the dimension κ is smaller than or equal to med are in $\mathcal{D}_l$, while the rest are in $\mathcal{D}_r$.
3. Denote the left and right child of node R as R_l and R_r, respectively. To construct the left (right) subtree of R, repeat steps 1 & 2 by regarding $\mathcal{D}_l$ ($\mathcal{D}_r$) as the new $\mathcal{D}$, and R_l (R_r) as the new R.
4. Continue with step 3 recursively. When there is only one single data point in the data set $\mathcal{D}$, instead of recording the dimension index and median in the node R as before, the data point itself is stored in R, which becomes a leaf node of the KD-Tree.

Herein, to order all the k-dimensional locations, we initially create the KD-Tree using these locations. Subsequently, we perform an in-order traversal of the tree. This entails that, upon reaching any node during the traversal, we first record the data from its left subtree, followed by the data of the node itself, and then proceed to capture the data from its right subtree.

3.3 | Other ordering methods

Apart from the previous three ordering methods, which are the main focuses of this article, we also implemented some other ordering methods in *ExaGeoStat*, which we describe as follows.

3.3.1 | Maximum-minimum degree (MMD) ordering

The maximum-minimum degree (MMD, Guinness (2018)) ordering is widely used in solving sparse linear systems. The primary objective of MMD is to rearrange the rows and columns of a given matrix to ensure that each pair of adjacent locations in the final sequence is not excessively close to the original grid layout and to enhance the efficiency of core matrix operations, including matrix-matrix multiplication, factorization, and similar processes. This approach arranges the spatial locations so that each spatial location is followed by its nearest neighbors in the final order. Therefore, in spatial statistics, it is mainly applied in Vecchia approximation (Guinness, 2018), where we need to find the distribution of each spatial location conditioning on its nearest neighbors.

The MMD procedure is as follows: First, the algorithm picks the row or column with the maximum number of nonzero entries (highest degree). Then, it identifies the row or column with the minimum degree among the remaining ones. Finally, the algorithm reorders the matrix such that these rows and columns are moved to a position in the matrix, such as the bottom right corner, where their impact on fill-in is minimized.

The covariance matrix integral to MLE operations is dense, contrasting with the sparse matrices the original maximum-minimum degree (MMD) ordering algorithm addresses. We tailored the MMD algorithm for implementation in the *ExaGeoStat* software to bridge this gap. This adaptation includes defining a user-specified threshold to determine when elements should be treated as nonzero-like or zero-like elements in the original algorithm.

Our preliminary experiments indicate that the modified MMD algorithm is ineffective for the MLE covariance matrix. It results in higher tile ranks than the coordinate-based ordering algorithms like Morton, Hilbert, and KD-Tree. Consequently, due to its limited performance in these initial tests, we have excluded it in the detailed experimental section.

3.3.2 | Graph-based ordering methods

We also consider adopting some existing ordering methods designed for sparse matrices, which can lower the rank of the matrices. These methods are usually based on the corresponding adjacency graph of the sparse matrix. We cannot directly apply these methods because the covariance matrices are always dense, and in these cases, all the nodes in the adjacency graph will be connected. However, we "sparsify" the adjacency graph by setting a threshold on the values in the matrix and letting the edge in the graph be disconnected if its corresponding value in the matrix is smaller than the threshold. In addition, the threshold cannot be too large in order not to lose too much correlation information. Here, we briefly introduce the two graph-based methods that we conducted experiments with within *ExaGeoStat*:

1. Reversed Cuthill–McKee algorithm

   The Reversed Cuthill–McKee (RCM) Algorithm (Cuthill & McKee, 1969) is an algorithm aiming at reducing the bandwidth of a sparse matrix. It works basically in the following steps:

   a. Consider the sparse matrix as an adjacency matrix, then form its corresponding graph and start with an arbitrary node in the graph;
   b. Conduct breadth-first search (BFS) traversal on the graph, starting from the arbitrarily chosen node. For each node in the graph, define its level as its distance from the starting node;
   c. Sort all the nodes based on their levels in descending order while the nodes with the same level are sorted by degree, with nodes having higher degrees appearing first;
   d. Reorder the nodes based on the sorted order and then reverse the order of the reordered nodes to obtain the final ordering.

   Although it cannot be guaranteed that the RCM algorithm can lead to the optimal result, experiments have shown that it can significantly reduce the bandwidth of sparse matrices.

2. Minimum degree algorithm

   Similarly, the minimum degree algorithm (George & Liu, 1989) also aims at reducing the bandwidth of each row of a sparse matrix, which works in the following steps:

   a. Form the graph corresponding to the sparse matrix (considered as an adjacency matrix), then compute the degree of each node in the graph;
   b. Find the node with the minimum degree, then eliminate the selected node by removing it from the graph and updating the degrees of its neighboring nodes;
   c. Update the degrees of the remaining nodes affected by the removal;
   d. Repeat the previous two steps until all nodes have been eliminated.

   The order in which the nodes are eliminated forms the new ordering of the vertices, thus yielding the new ordering of the matrix itself.

Our experiments in *ExaGeoStat* showed that these two algorithms could not help accelerate our computation, with many different choices of the thresholds while "sparsifying" the covariance matrices. In fact, after reordering the covariance matrices using these two algorithms, the ranks of some of the off-diagonal tiles are still too large, such that the TLR approximation cannot even proceed.

4 | SPATIAL MODELS

4.1 | Univariate Matérn model

Most common geostatistical data sets, such as climate and environmental data sets, comprise a collection of locations that are distributed across a specific geographic area, either regularly or irregularly. Each location is linked to a single measurement of a particular climate or environmental variable, such as wind speed, air pressure, soil moisture, or humidity. Such data sets are often modeled as realizations of Gaussian spatial random fields, as formulated in the introduction section. We denote a realization of a Gaussian random field $Z(\boldsymbol{s})$ by $\boldsymbol{Z} = \{Z(\boldsymbol{s}_1), \dots, Z(\boldsymbol{s}_n)\}^\top$, where $\boldsymbol{s}_1, \dots, \boldsymbol{s}_n$ are spatial locations in $\mathbb{R}^d$ for some $d \in \mathbb{Z}^+$. Without loss of generality, we assume the random field $Z(\boldsymbol{s})$ has zero mean and a stationary covariance function which can be parametrized by a vector $\boldsymbol{\theta} \in \mathbb{R}^q$ for some $q \geq 1$:

$$C(\boldsymbol{h};\boldsymbol{\theta}) = \mathrm{Cov}\{Z(\boldsymbol{s}), Z(\boldsymbol{s}+\boldsymbol{h})\}, \tag{3}$$

where $\boldsymbol{h} \in \mathbb{R}^d$ is the spatial lag vector, and C is symmetric with respect to $\boldsymbol{h}$. Herein, the expression (3) is further simplified from (1) due to the stationarity assumption. Therefore, the (i,j)th entry of the covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ equals $\boldsymbol{\Sigma}_{ij}(\boldsymbol{\theta}) = C(\boldsymbol{s}_i - \boldsymbol{s}_j;\boldsymbol{\theta}),\quad i,j = 1, \dots, n$. The log-likelihood function for $\boldsymbol{\theta}$ in this case is formulated in (2).

In this work, we mainly focus on the Matérn covariance function (Matérn, 1960) without nugget effects, and parametrized by $\boldsymbol{\theta} = (\sigma^2, \beta, \nu)^\top$:

$$C_{\mathcal{M}}(d;\boldsymbol{\theta}) = \frac{\sigma^2}{\Gamma(\nu)2^{\nu-1}}\left(\frac{d}{\beta}\right)^{\nu}\mathcal{K}_\nu\left(\frac{d}{\beta}\right), \tag{4}$$

where $d = \|\boldsymbol{s} - \boldsymbol{s}'\|$ denotes the distance between two locations $\boldsymbol{s}$ and $\boldsymbol{s}'$, σ^2, β, ν are the variance, range and smoothness parameters, respectively, which all take positive values, and $\mathcal{K}_\nu$ is the modified Bessel function of the second kind of order ν.

4.2 | Other spatial models

Apart from the most common univariate Matérn model, we may consider some other common spatial models while dealing with spatial data. In *ExaGeoStat*, we also implemented the bivariate Matérn and non-Gaussian models, which we describe in the following subsections.

4.2.1 | Bivariate Matérn model

The univariate Matérn covariance function can also be generalized to the multivariate case. Here we introduce the bivariate case (Apanasovich et al., 2012 & Gneiting et al., 2010), which is also implemented in *ExaGeoStat*. The parsimonious bivariate Matérn cross-covariance function between variables i and j is given by

$$C_{ij}(d;\boldsymbol{\theta}) = \frac{\rho_{ij}\sigma_{ii}\sigma_{jj}}{\Gamma(\nu_{ij})2^{\nu_{ij}-1}}\left(\frac{d}{a}\right)^{\nu_{ij}}\mathcal{K}_{\nu_{ij}}\left(\frac{d}{a}\right). \tag{5}$$

Similar to (4), here $d = \|\boldsymbol{s} - \boldsymbol{s}'\|$ denotes the distance between two locations $\boldsymbol{s}$ and $\boldsymbol{s}'$. $i,j = 1, 2$ denote the indices of the two components of the bivariate data. The parameter $\boldsymbol{\theta}$ consists of several components. To be specific, $\sigma_{11}^2 > 0$ and $\sigma_{22}^2 > 0$ are the marginal variance parameters of the two components; $a > 0$ is the spatial range parameter; $\nu_{11} > 0$ and $\nu_{22} > 0$ are the marginal smoothness parameters of the two components, while $\nu_{12} = \frac{1}{2}(\nu_{11} + \nu_{22})$ is the cross smoothness; ρ_{ij} is the colocated correlation, defined as

$$\rho_{ij} = \beta_{ij}\frac{\Gamma(\nu_{ii} + d/2)}{\Gamma(\nu_{ii})}\frac{\Gamma(\nu_{jj} + d/2)}{\Gamma(\nu_{jj})}\frac{\Gamma(\nu_{ij})}{\Gamma(\nu_{ij} + d/2)},$$

where $\beta_{ii} = \beta_{jj} = 1$ and $\beta_{ij} = \beta_{ji}$.

For a bivariate spatial data set with n locations, the size of its corresponding covariance matrix is $2n \times 2n$, with the value of each element given by (5).

4.2.2 | Non-Gaussian model

In many practical studies of spatial data analysis, high skewness, and heavy tails can be captured from the data, which makes it important to go beyond the Gaussian random fields while performing statistical inference. In *ExaGeoStat*, we implemented the Tukey g-and-h (TGH) random fields (Xu & Genton, 2017), which is a highly flexible non-Gaussian spatial model. The idea of TGH random fields is to distort Gaussian random fields with two parameters, g and h, to introduce extra skewness and kurtosis. The TGH random field with location parameter $\xi \in \mathbb{R}$ and scale parameter $\omega > 0$ is defined as follows:

$$T(\boldsymbol{s}) = \xi + \omega \tau_{g,h}\{Z(\boldsymbol{s})\},$$

where $\tau_{g,h}$ is the Tukey's g-and-h transformation function

$$\tau_{g,h}(z) = g^{-1}\{\exp(gz) - 1\}\exp(hz^2/2),$$

and $Z(\boldsymbol{s})$ denotes a standard Gaussian random field which is the same as what we describe in Section 4.1. When $g = h = 0$, the TGH random fields degenerate to Gaussian random fields. The details of implementing statistical inference for TGH random fields in *ExaGeoStat* can be found in Mondal et al. (2022).

No matter what spatial model we use, our main target is to find the maximum likelihood estimator (MLE) of $\boldsymbol{\theta}$ by maximizing the log-likelihood function shown in (2). In the next section, we show by numerical experiments how the ordering algorithms applied to the covariance matrices affect this parameter estimation process. Our preliminary experiments show that the effect of the ordering algorithms on spatial data generated from bivariate and non-Gaussian models is similar to those from univariate Matérn models. Therefore, to make it brief, we focus on numerical experiments with spatial data generated from univariate Matérn models in the next section.

5 | NUMERICAL STUDIES

In this section, we assess the performance and accuracy of TLR approximation on the spatial covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ using different ordering algorithms. The experiments on small-scale data are performed using an Intel Xeon Gold 6230 CPU running at 2.10GHz, with memory size equal to 128GB, L1, L2, and L3 cache sizes equal to 32K, 1024K and 36,608K, respectively; while on medium-scale data, we use Intel Xeon E5-2650 v2 CPUs running at 2.60GHz, with memory size equal to 128GB, L1, L2, and L3 cache sizes equal to 32K, 256K and 20,480K, respectively.

This section considers spatial data with locations randomly generated within the two-dimensional unit square.

5.1 | Parameter estimation accuracy assessment using various ordering algorithms

In this section, we demonstrate the effects of various ordering algorithms on the estimation accuracy of the statistical parameters using a set of synthetic spatial data, characterized by a Matérn covariance matrix without nugget effect, as described in equation (4). We first show results from small-scale data ($n = 1600$ locations, with tile size $nb = 320$) experiments performed on a 40-core shared-memory machine and from medium-scale data ($n = 20{,}000$ locations, with tile size $nb = 1000$) experiments performed on Shaheen-II supercomputer, using 4 nodes and 32 CPU cores on each node. Specifically, we report the parameter estimation results using Morton, Hilbert, and KD-Tree ordering methods and compare them with the accuracy of the estimation if no ordering algorithm is applied. We rely on the same optimization algorithm `BOBYQA` (Powell, 2009), which is a bound-constrained algorithm without using derivatives embedded in *ExaGeoStat*.

5.1.1 | Small-scale experiments

This section presents a comparative analysis of the effects of three distinct ordering algorithms – Morton, Hilbert, and KD-Tree – on the estimation accuracy of the TLR approximation method, utilizing a covariance matrix of size 1600×1600. Figure 4 shows a set of boxplots representing the estimated values of three parameters (variance, range, and smoothness) for a Matérn covariance function under various correlation structures: weak, medium, and strong. In addition, the root mean squared errors (RMSEs) of the same estimation results are presented in Table 1. The actual values for these parameters are set at ($\sigma^2 = 1, \beta = 0.03, \nu = 0.5$), ($\sigma^2 = 1, \beta = 0.1, \nu = 0.5$), and ($\sigma^2 = 1, \beta = 0.3, \nu = 0.5$) for each respective structure. The estimation is based on 100 synthetic datasets, each generated using the *ExaGeoStat* software. The estimation employed the TLR method with a compression accuracy set to 10^{-7} and an optimization tolerance of 10^{-9}.

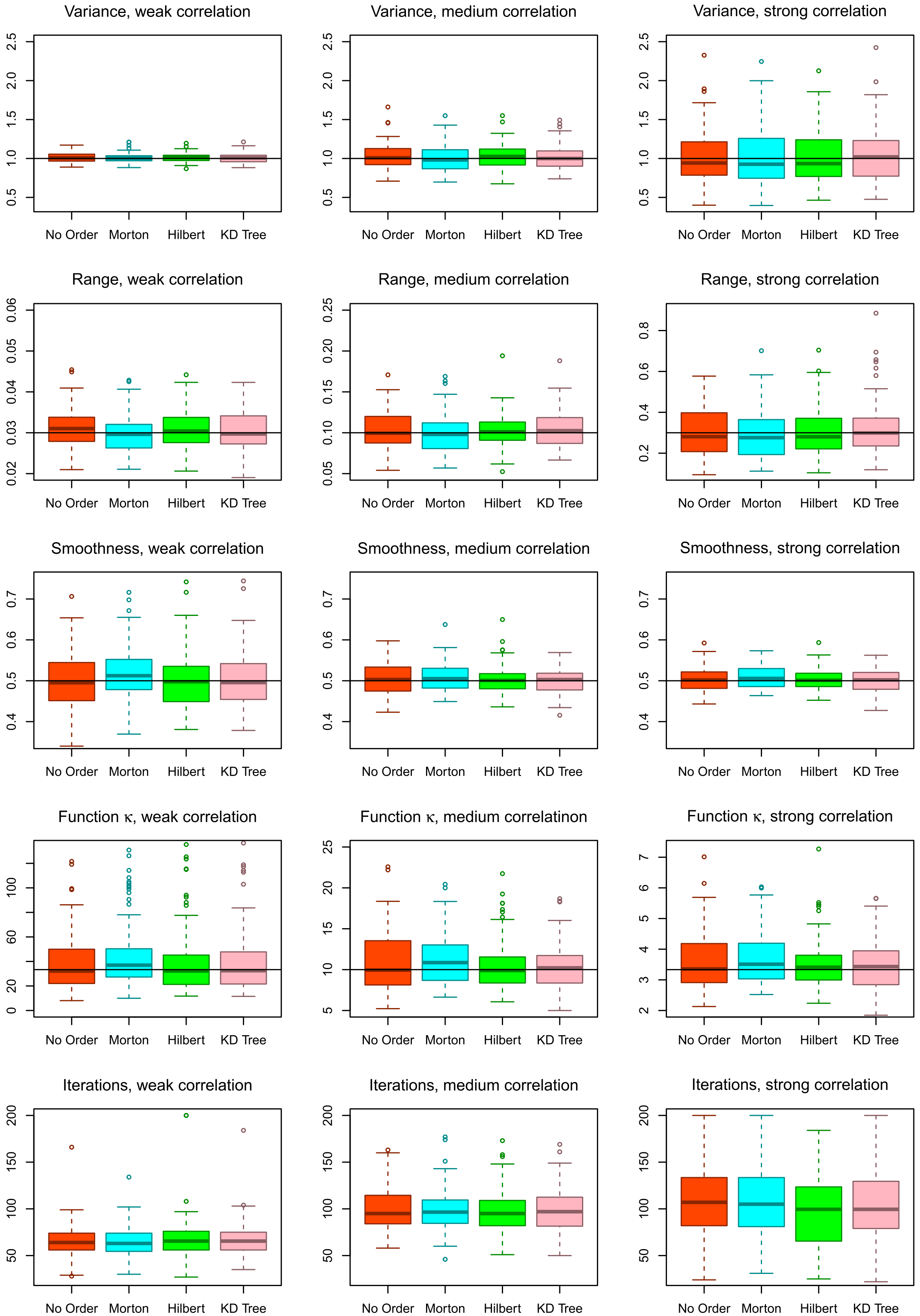

FIGURE 4 BoxPlots of TLR estimation accuracy with $n = 1600$ under either No Order or Morton, Hilbert, KD-Tree orderings. The horizontal lines in the first four rows denote the true values of the corresponding parameters or the function κ in (6).

Additional results under other configurations are available in Appendix A.1. In Figure 4, the initial three rows display the estimated values for variance, range, and smoothness parameters for each setting. The fourth row is dedicated to illustrating the estimated values of the function $\kappa(\sigma^2, \beta, \nu)$ as follows:

$$\kappa(\sigma^2, \beta, \nu) = \sigma^2 \beta^{-2\nu}, \tag{6}$$

which was proved in Zhang (2004) to be identifiable under infill asymptotics. Moreover, in the fifth row, we plot the number of iterations needed for the optimization process in *ExaGeoStat* to converge to the final results.

The results in Figure 4 and Table 1 show that when the correlation structure of spatial data is weak, it is difficult to tell which ordering methods perform better than the others; however, when the correlation becomes stronger, the Hilbert ordering method yields the most consistent and effective estimation results, and demonstrates superior performance in the number of iterations required for convergence. Nevertheless, the accuracy of parameter estimation using various ordering algorithms is minimally affected by the choice of algorithm. This outcome is advantageous, as our objective is to speed up the optimization process without significantly altering the results.

5.1.2 | Medium-scale experiments

In this section, we expand the covariance matrix dimensions to 20,000 × 20,000 to examine the impact of different ordering algorithms on a medium-sized correlation matrix. Figure 5 compares the outcomes of these algorithms through boxplots, and Table 2 presents the corresponding RMSEs. As in the small-scale experiments, we use synthetic spatial data with the Matérn covariance function and weak, medium, and strong correlations under the same settings. Unlike in Figure 4 and Table 1, Figure 5 and Table 2 omit the "no order" results due to the high ranks of individual tiles, which slows down the estimation process. The experiments are repeated 100 times, each with independently generated synthetic datasets for each setting. The first three rows of Figure 5 display the estimation results for the variance, range, and smoothness parameters, while the fourth row depicts the estimated value of the function κ as in (6). Additionally, the fifth row presents the number of iterations required for the optimization process in *ExaGeoStat* to reach convergence.

The reported experimental results show that the Hilbert ordering method no longer gives superior results as in small-scale cases. Instead, the KD-Tree gives the most stable and unbiased estimation results when the correlation structure of the spatial data is weak or medium. The number of iterations needed to converge is similar among the three algorithms. Nevertheless, we can still conclude that the choice of ordering algorithm does not significantly affect the accuracy of parameter estimation, which is what we desire.

TABLE 1 The RMSEs of TLR estimation accuracy with $n = 1600$ under either no order or Morton, Hilbert, KD-Tree orderings, over 100 replicates of the experiments.

Correlation	Parameter	No order	Morton	Hilbert	KD-tree
Weak	Variance (σ^2)	0.0613	0.0586	**0.0547**	0.0628
	Range (β)	**0.0047**	0.0048	0.0049	0.0052
	Smoothness (ν)	**0.0663**	0.0696	0.0703	0.0704
Medium	Variance (σ^2)	0.1614	0.1717	**0.1547**	0.1657
	Range (β)	0.0225	0.0233	**0.0214**	0.0225
	Smoothness (ν)	0.0362	0.0346	0.0341	**0.0323**
Strong	Variance (σ^2)	0.3564	0.3719	**0.3327**	0.3463
	Range (β)	0.1237	0.1174	**0.1118**	0.1317
	Smoothness (ν)	0.0306	**0.0255**	0.0262	0.0264

Note: The lowest value of RMSE is marked in bold for each parameter and each correlation structure.

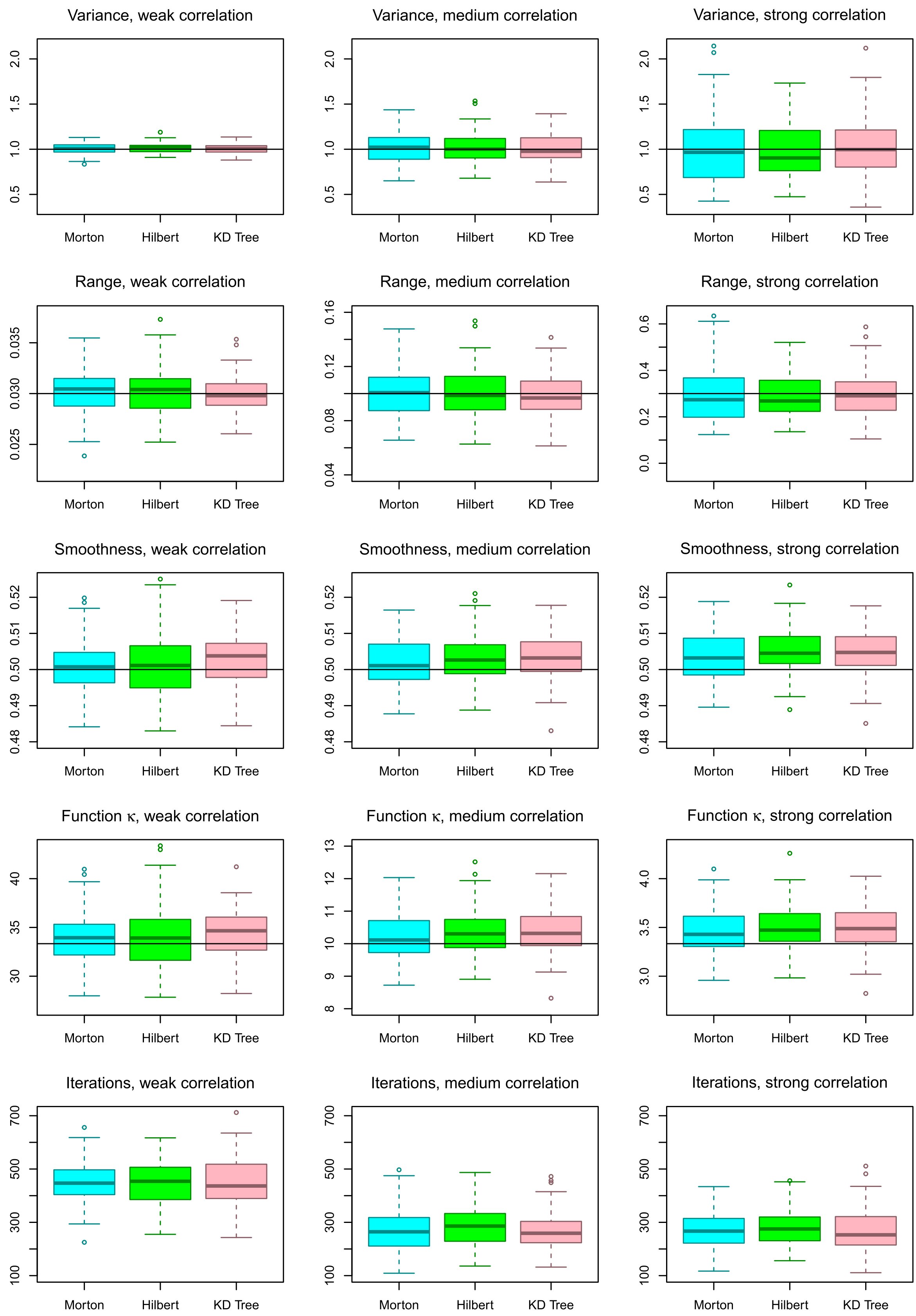

FIGURE 5 BoxPlots of TLR estimation accuracy with $n = 20{,}000$ under Morton, Hilbert, KD-Tree orderings. The horizontal lines in the first four rows denote the true values of the corresponding parameters or the function κ in (6).

TABLE 2 The RMSEs of TLR estimation accuracy with $n = 20,000$ under Morton, Hilbert, KD-Tree orderings, over 100 replicates of the experiments.

Correlation	Parameter	Morton	Hilbert	KD-Tree
Weak	Variance (σ^2)	0.0619	**0.0499**	0.0537
	Range (β)	0.0022	0.0021	**0.0018**
	Smoothness (ν)	0.0077	0.0086	**0.0068**
Medium	Variance (σ^2)	0.1607	0.1734	**0.1594**
	Range (β)	0.0161	0.0186	**0.0156**
	Smoothness (ν)	0.0067	0.0065	**0.0063**
Strong	Variance (σ^2)	0.3742	**0.3282**	0.3586
	Range (β)	0.1090	**0.0985**	0.1038
	Smoothness (ν)	0.0068	**0.0061**	0.0063

Note: The lowest value of RMSE is marked in bold for each parameter and each correlation structure.

5.2 | Tile ranks with different ordering algorithms

The quality of TLR compression on individual tiles significantly influences the accuracy, memory usage, and computational time required for the TLR approximation algorithm. In this section, we evaluate the effectiveness of TLR compression applied to a covariance matrix generated with a Matérn covariance function. This is achieved by determining the ranks of all matrix tiles while employing various ordering algorithms.

We rely on synthetic datasets generated with true values $\sigma^2 = 1$, $\nu = 0.5$ and $\beta = 0.03, 0.1, 0.3$, focusing on a location count of $n = 10{,}000$ and a tile size of $nb = 1000$. Additional results under varied settings are shown in Appendix A.2. Figure 6 shows heatmaps of sample covariance matrices for weak, medium, and strong correlation structures. For each correlation type, we created 100 spatial datasets. We also present boxplots of the off-diagonal tiles' minimum, median, mean, and maximum rank values in the corresponding covariance matrices, as shown in Figure 7. Additionally, Figure 8 presents histograms of these ranks from the 100 datasets. Furthermore, Table 3 outlines the average memory size needed to store the off-diagonal tiles using different ordering methods for the specified correlation structures.

From the figures and the table, we can highlight the performance of the three ordering algorithms (Hilbert, Morton, and KD-Tree) in compressing individual off-diagonal tiles as follows:

- **Lower off-diagonal tile ranks**: All three algorithms successfully reduce off-diagonal tile ranks, which leads to lower memory consumption and higher computation speed.
- **Hilbert's superiority in weak correlation**: Hilbert outperforms Morton and KD-Tree in cases with weak correlation. This can be shown by the distribution of off-diagonal tile ranks (Figure 8), where Morton and KD-Tree show peaks around 150, but Hilbert does not.
- **Memory efficiency**: The memory required to store off-diagonal tiles using Hilbert is smaller than with Morton and KD-Tree in cases of weak or medium correlation (Table 3). However, in strong correlation cases, the memory usage is almost identical across all three algorithms.
- **Effect of correlation strength on tile ranks**: Figure 6 shows that the ranks of off-diagonal tiles become higher when the correlation becomes weaker. At first, this may seem counter-intuitive, but the explanation is that the covariance matrix values change more rapidly toward the off-diagonal direction when the correlation is smaller, which results in higher tile ranks.

5.3 | Computation performance assessment

This section assesses the computation performance of Morton, Hilbert, and KD-Tree ordering algorithms under different correlation scenarios.

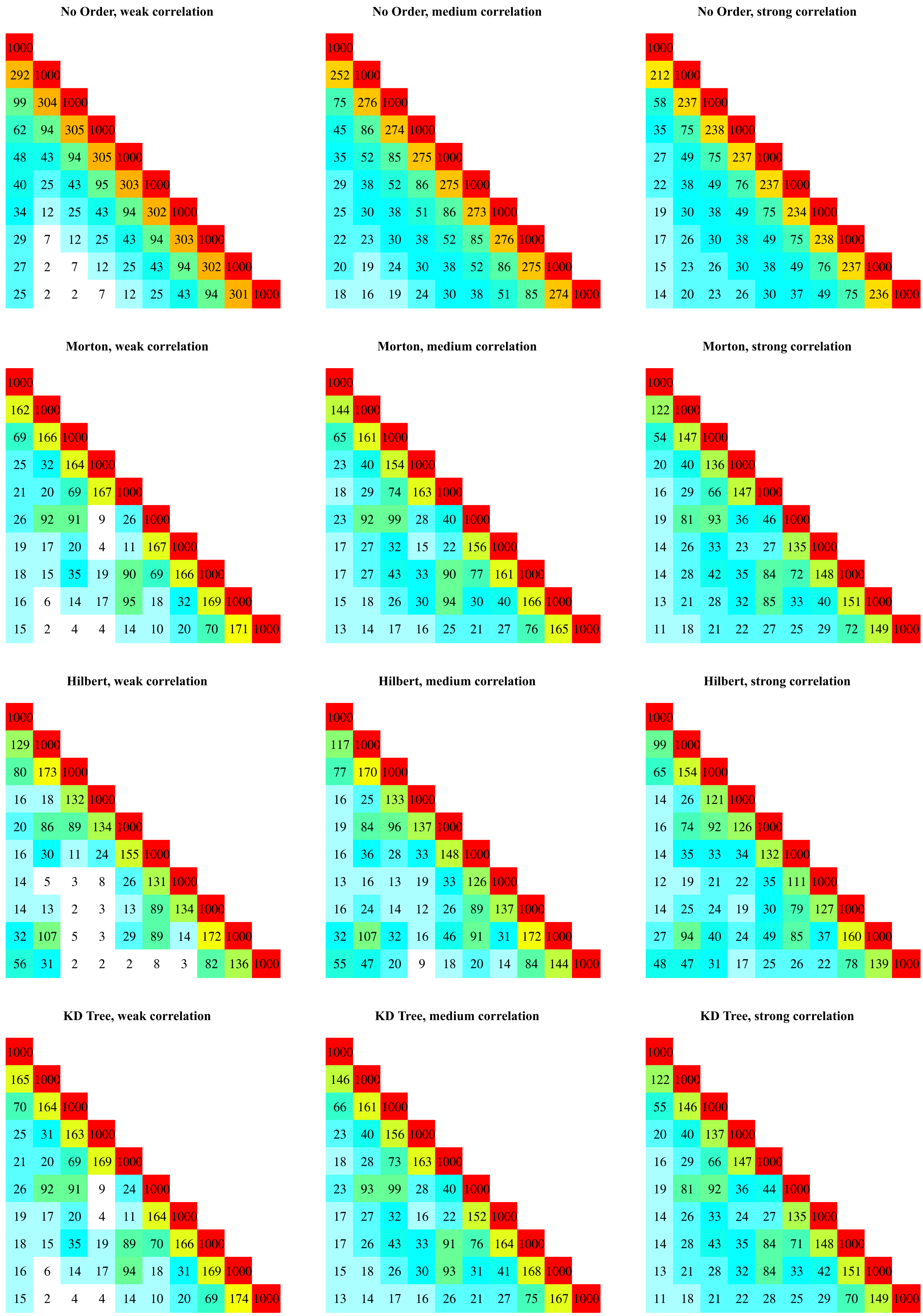

FIGURE 6 Heatmaps of tile ranks for various ordering algorithms for weak, medium, and strong correlation structures. Each small square symbolizes a 1000×1000 tile, annotated with its corresponding rank. Diagonal tiles maintain full rank. Within each heatmap, darker colors indicate higher ranks.

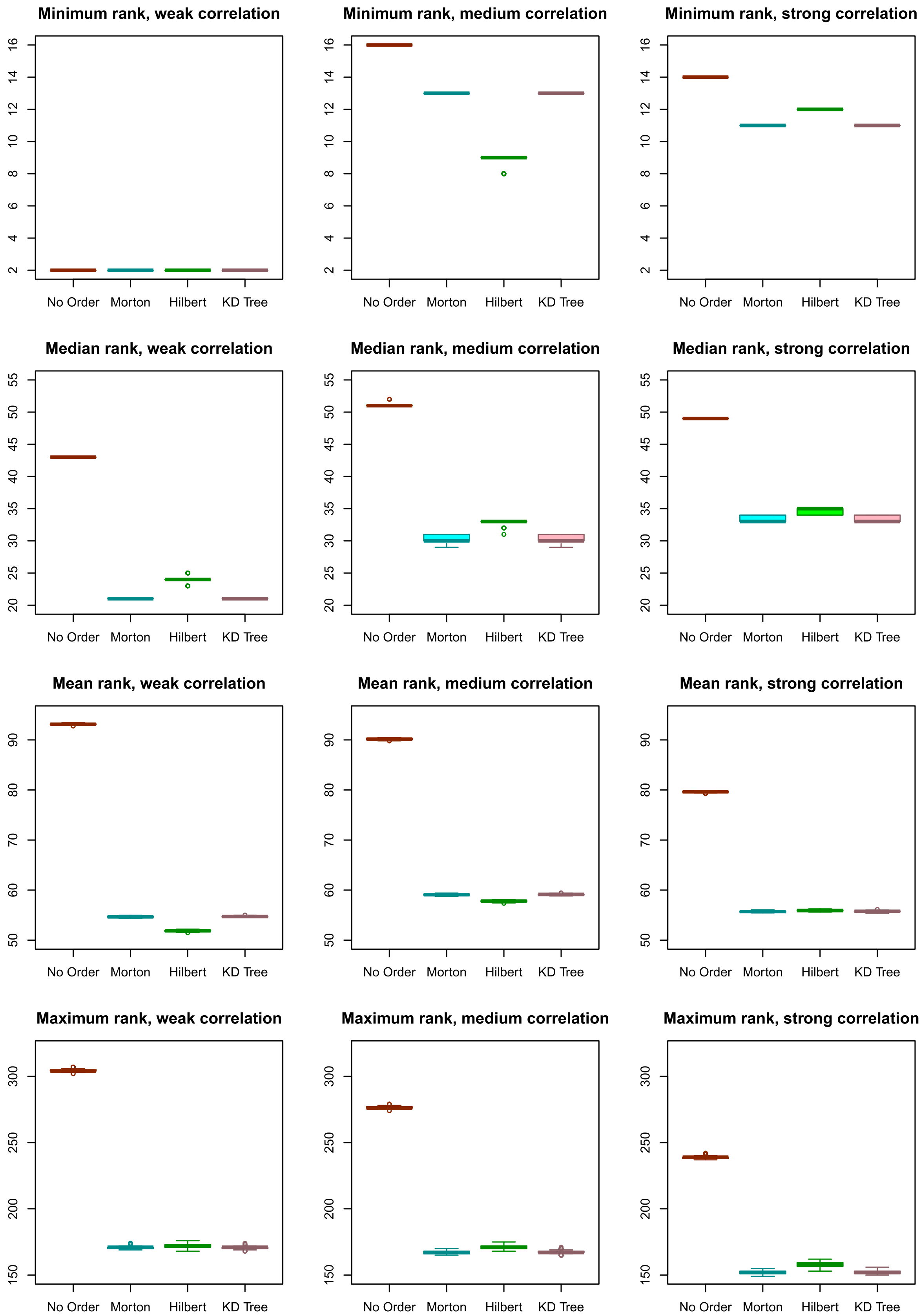

FIGURE 7 BoxPlots of the minimum, median, mean, and maximum of off-diagonal tile ranks with different ordering methods with weak, medium, and strong correlation structures. We generated 100 sets of data with $n = 10{,}000$ locations, and they are all divided into 1000×1000 tiles.

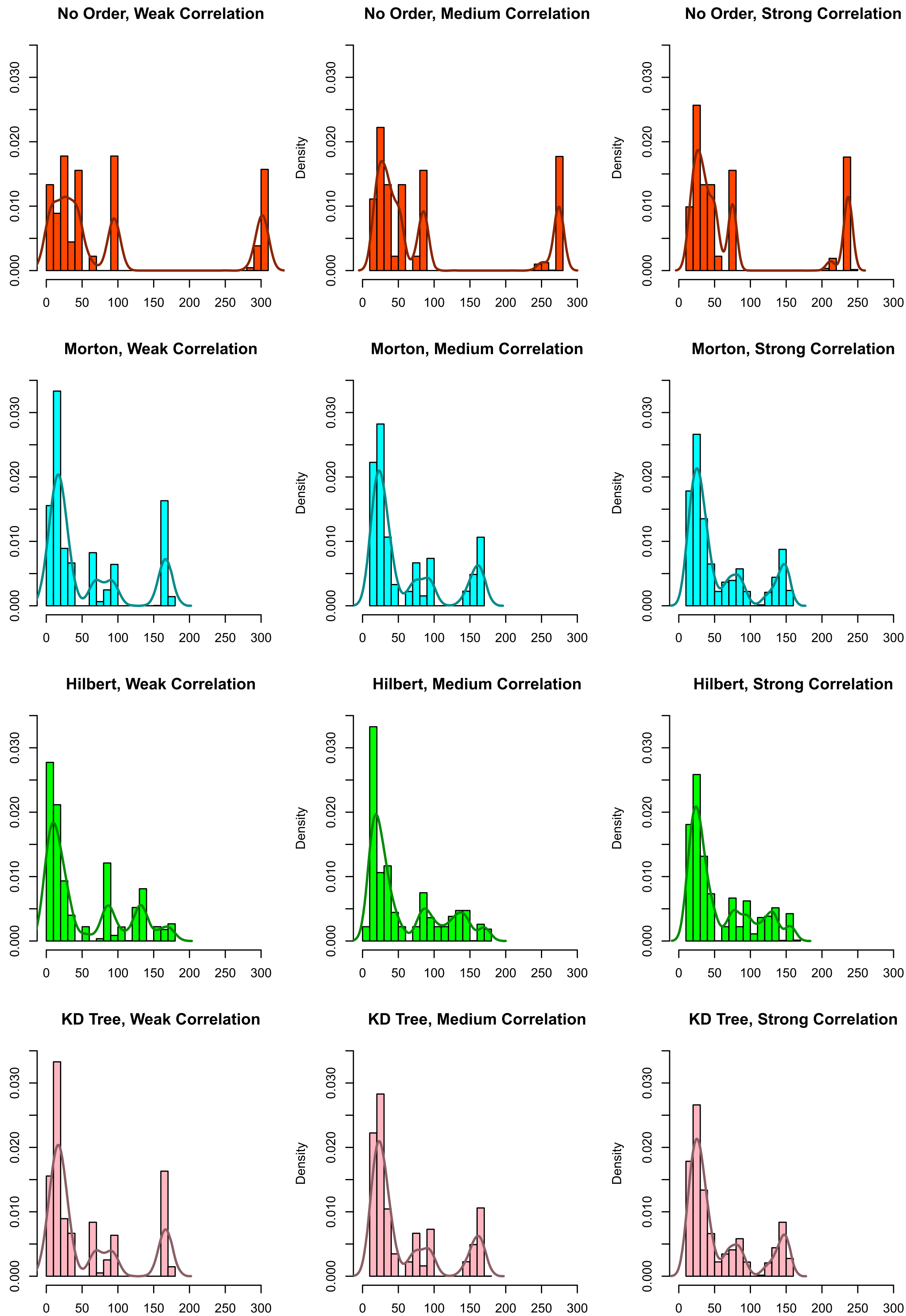

FIGURE 8 Histograms of the proportions and curves of the empirical densities of off-diagonal tile ranks with different ordering methods with weak, medium, and strong correlation structures. We generated 100 sets of data with $n = 10{,}000$ locations, and they are all divided into 1000×1000 tiles.

TABLE 3 The average memory (in MB) for storing all the off-diagonal tiles of a covariance matrix of $n = 10{,}000$ to 40,000 spatial locations divided into 1000×1000 tiles, with different ordering methods and correlation structures.

	Ordering	Weak	Medium	Strong
$n = 10{,}000$	Dense	360		
	No order	67	65	57
	Morton	39	43	**40**
	Hilbert	**37**	**42**	**40**
	KD-tree	39	43	**40**
$n = 20{,}000$	Dense	1520		
	No order	256	250	227
	Morton	105	123	122
	Hilbert	**91**	**110**	**113**
	KD-tree	100	116	115
$n = 30{,}000$	Dense	3480		
	No order	556	549	501
	Morton	191	233	237
	Hilbert	**161**	**203**	**211**
	K D-tree	189	229	232
$n = 40{,}000$	Dense	6240		
	No order	NA	955	870
	Morton	251	316	**328**
	Hilbert	**240**	**312**	331
	KD-tree	251	316	**328**

Note: Here, we also compare with the theoretical storage needed without using TLR approximation, denoted by "dense", where the storage is the same for different correlation structures if n is fixed, and we can see the storage is much larger than using TLR approximation. The lowest storage required is marked in bold for each number n and each correlation structure. The "NA" in this table means that when $n = 40{,}000$ with weak correlation, if we use no order, then some of the off-diagonal tiles of the covariance matrix will be too dense for the TLR approximation to proceed.

5.3.1 | Cholesky factorization performance

Recalling the formula of the log-likelihood in (2), the Cholesky factorization of the covariance matrix $\mathbf{\Sigma}$ is the most time-consuming operation while calculating the log-likelihood in each iteration during the optimization process to find the MLE. Hence, this process is a key indicator of the overall performance.

To make it brief here, in Figure 9, we compare the execution time of a single TLR Cholesky factorization of a compressed covariance matrix using the three ordering algorithms under different settings. The corresponding TLR accuracy is 10^{-7}, and the tile size is 1000. We consider the cases where the number of locations n equals to 3600, 6400, 10,000, 14,400, 22,500, 40,000, 62,500, and 90,000, and the corresponding results are shown in each subfigure of Figure 9 from left to right in the x-axis. A more complete summary of the experiment results can be found in Appendix A.3.

The results indicate that TLR Cholesky factorization using Hilbert ordering outperforms Morton or KD-Tree ordering in most cases. As the data size increases, the advantage of Hilbert ordering over the other two methods becomes more consistent in percentage terms, indicating that Hilbert ordering saves more computation time as data size grows. However, as depicted in the figure, there are two instances where Hilbert ordering does not exhibit the expected linear performance compared to Morton ordering, that is, at 10,000 and 40,000 locations (although it remains faster in most cases). Upon further analysis of these two cases, we found that Hilbert ordering still assists in reducing the ranks of the compressed covariance matrix before Cholesky decomposition. However, the ranks may fluctuate after the factorization process. Due to an implementation constraint in the current version of the *ExaGeoStat* software with TLR approximation used in this study, the tile sizes must be allocated upfront to accommodate the largest anticipated rank of each tile during execution.

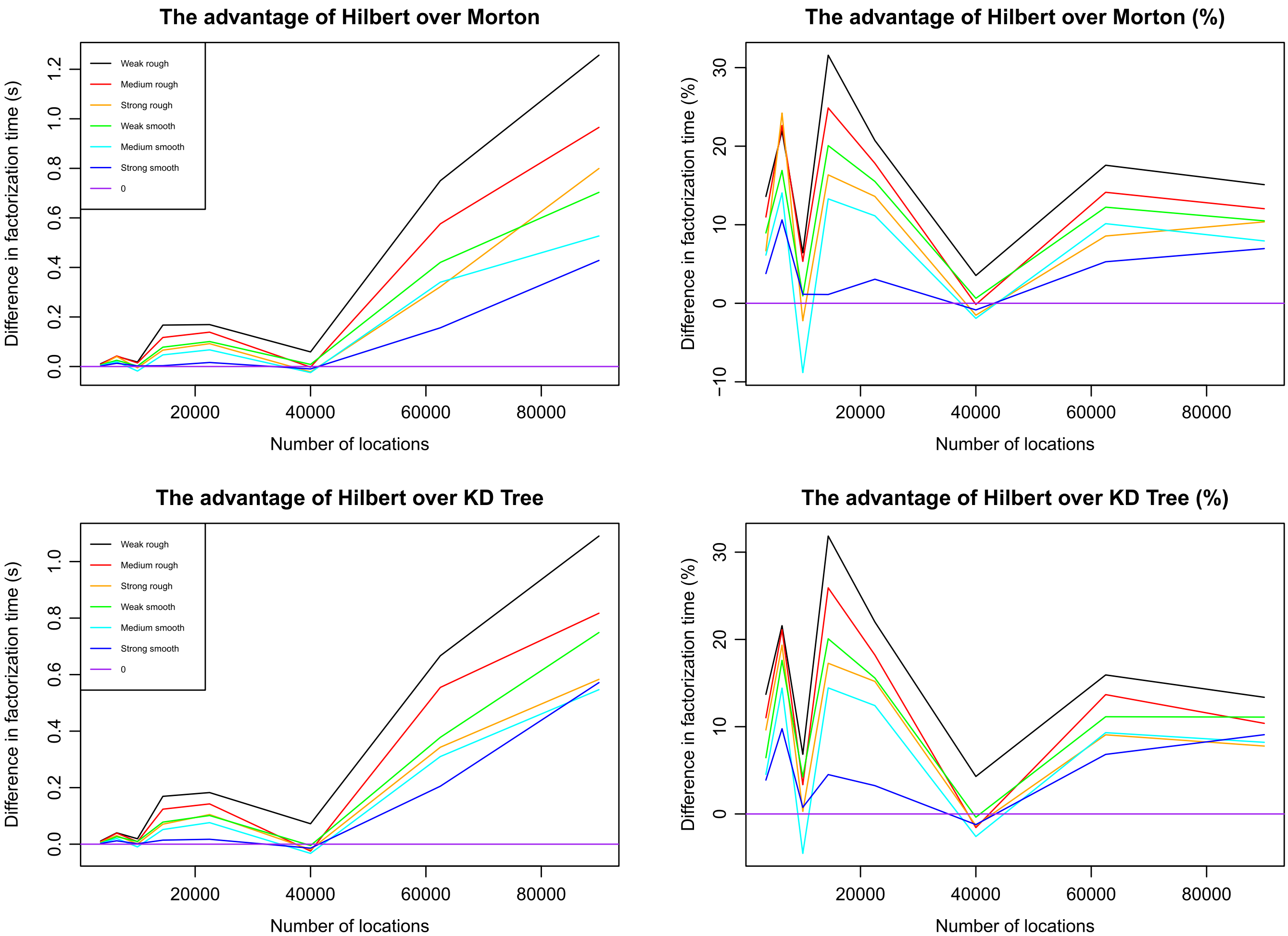

FIGURE 9 TLR Cholesky factorization execution time. Variance set at $\sigma^2 = 1$. The weak, medium, and strong refer to the range parameter β values of $0.03, 0.1, 0.3$, respectively. The words "rough" and "smooth" correspond to the cases where $\nu = 0.5$ and $\nu = 1$, respectively. The left-hand plots show the actual time differences (in seconds) for various orderings, while the right-hand plots present these differences in percentage terms. The legends for the subfigures on the left apply to the subfigures on the right as well.

Consequently, we observed that in these two cases, the Hilbert-based covariance matrix exhibits higher tile ranks after factorization than expected but they are still lower than the ranks of the Morton order.

We also note that the difference in the factorization time using different ordering methods is more significant when the correlation among the spatial data is weaker, which is in line with our finding in Section 5.2 that when the correlation is weaker, Hilbert ordering can reduce the ranks of off-diagonal tiles as well as the memory size needed to store the covariance matrices. In addition, when the smoothness $\nu = 0.5$, we observe a larger difference between Hilbert's performance and other orderings, compared with the case where $\nu = 1$.

5.4 | Application on soil moisture data

We apply our proposed methods on a daily soil moisture data set from a numerical model, which was organized by Litvinenko et al. (2019), and can be downloaded from https://github.com/litvinen/HLIBCov.git. Soil moisture plays a crucial role in assessing the condition of hydrological processes and finds widespread applications in weather forecasting, crop yield prediction, and early detection of flood and drought events. Improving the characterization of soil moisture has demonstrated a significant enhancement in weather forecasting. However, the high spatial resolution needed often results in large datasets from numerical models, making the computation of many statistical inference methods impractical. In

TABLE 4 Experimental results on soil moisture data for various numbers of locations, n.

n	Ordering	$\hat{\sigma}^2$	$\hat{\beta}$	$\hat{\nu}$	Number of iterations	Time to solution (s)	Time per iteration (s)
2000	Morton	1.1541	0.2335	0.2655	416	80	0.19
	Hilbert	1.1549	0.2339	0.2655	383	70	0.18
	KD-tree	1.1552	0.2340	0.2655	435	83	0.19
4000	Morton	1.0602	0.2640	0.2350	397	135	0.34
	Hilbert	1.0602	0.2640	0.2350	355	118	0.33
	KD-tree	1.0600	0.2639	0.2350	310	103	0.33
8000	Morton	1.0637	0.2318	0.2390	566	1807	3.19
	Hilbert	1.0638	0.2318	0.2390	531	1707	3.21
	KD-tree	1.0636	0.2318	0.2389	493	1558	3.16
16,000	Morton	1.0725	0.2517	0.2353	203	2242	11.04
	Hilbert	1.0678	0.2493	0.2353	282	3085	10.94
	KD-tree	1.0680	0.2492	0.2354	293	3175	10.84
32,000	Morton	1.0245	0.1577	0.2610	764	20,524	26.86
	Hilbert	1.0261	0.1582	0.2610	605	15,550	25.70
	KD-tree	1.0257	0.1581	0.2610	519	13,101	25.24

Note: Parameter estimation results are shown in the columns with titles $\hat{\sigma}^2$, $\hat{\beta}$, and $\hat{\nu}$, respectively, and in the last three columns, we show the number of iterations needed until the optimization process converges, as well as the average execution time for each iteration and the total execution time of the optimization, both in seconds. We do not present the results using no order because, in that case, some off-diagonal tiles will be too dense for the TLR approximation to proceed.

this experiment, we examine high-resolution soil moisture data on January 1, 2014, in the upper soil layer of the Mississippi River basin in the United States. The spatial resolution of the data is 0.0083 degrees, and the distance of one-degree difference in this region is approximately 87.5 km. The dataset encompasses 2,000, 000 locations, and it does not conform to a regular grid. In our experiment, subsets of the data with numbers of locations $n = 2000, 4000, 8000, 16{,}000, 32{,}000$ are randomly selected. We apply Morton, Hilbert, and KD-Tree orderings with TLR approximation to perform estimation on the variance parameter σ^2, range β and smoothness ν assuming a Gaussian random field with Matérn covariance function as defined in (4).

In Table 4, we show the estimation results for the parameters σ^2, β, and ν, as well as the number of iterations needed until the optimization process converges, the average execution time for each iteration, and the total execution time of the optimization, which are shown in the last three columns. Different ordering methods have almost no influence on the parameter estimation results. For the time it takes to complete each iteration, there is also no big difference among these three ordering algorithms. This makes sense since, from the parameter estimation results, we can see the dependence structure of this data set is between medium and strong, closer to the strong side. As we demonstrated in Table 3 and Figure 8, the difference in reduction of off-diagonal tiles among different ordering algorithms is not that much in this case.

6 | CONCLUSION AND DISCUSSION

Maximum likelihood estimation (MLE) is commonly used to estimate the statistical parameters of covariance functions to model spatial data. It relies on a dense covariance matrix of a size matching the number of spatial locations, hence making large-scale manipulations complex and often prohibitive. Consequently, approximation methods are commonly employed to simplify this complexity. These approximation methods can become more effective when paired with parallel processing. Our study focused on the Tile Low-rank (TLR) approximation, which approximates the matrix tiles in low-rank rather than the entire matrix. Yet, a major challenge with TLR approximation lies in how the ordering of elements within the covariance matrix can significantly affect the quality of the approximation.

In this work, we implemented several ordering algorithms in the *ExaGeoStat* framework to reorder the locations of spatial data before generating the spatial covariance matrices and performing TLR approximation. Some ordering methods, like Morton, Hilbert, and KD-Tree, can largely reduce computation time and storage while performing parameter estimation on the data using MLE. We conducted numerical experiments with data generated from Gaussian random fields with the Matérn covariance function. To be specific, we examined the accuracy of parameter estimation and the convergence rate (Figures 4 and 5), the ranks of off-diagonal tiles in TLR estimation (Figures 6–8) as well as the storage required (Table 3), and the execution time of Cholesky factorization of the covariance matrix (Figure 9). In addition, we applied our methods to a soil moisture data set (Table 4).

From our conducted numerical experiments, as we desire, the ordering algorithms do not significantly affect the parameter estimation accuracy. However, some differences can still be observed: in small-scale cases, Hilbert slightly outperforms the other ordering methods when the correlation structure of the spatial data is medium or strong, for example, in the medium scenario, the RMSE of estimation results of the variance using Hilbert is 4.3%, 11.0%, and 7.1% lower than applying no order, Morton and KD-Tree, respectively; while in medium-scale cases, different ordering algorithms achieve the highest accuracy in different scenarios, for example, in the medium scenario, the RMSE of estimation results of the variance using KD-Tree is 0.8% and 8.8% lower than using Morton and Hilbert, and in the strong scenario, the RMSE of estimation results of the variance using Hilbert is 14.0% and 9.3% lower than using Morton and KD-Tree. We also find that the computation time and storage of the estimation procedure varies depending on the ordering algorithms employed, especially when the correlation structure is weak; Hilbert can reduce the ranks of off-diagonal tiles to the largest extent and, therefore, reduce the computation time. For example, when the weakly correlated spatial dataset contains $n = 10{,}000$ locations, and we divide the correlation matrix into 1000×1000 tiles, the average tile ranks using Hilbert is 5.4% and 5.5% lower than using other ordering methods, namely Morton and KD-Tree, and it is 79.6% lower than using no order. However, when the correlation among the spatial data gets stronger and stronger, the advantage of the Hilbert method gradually vanishes. Under the scenario of a strong correlated spatial dataset, still with $n = 10{,}000$ locations and the correlation matrix divided into 1000×1000 tiles, the average tile ranks using Hilbert is still 42.4% lower than using no order, but 0.4% and 0.3% higher than using Morton and KD-Tree. This observation was also verified by our experiments on real data.

ACKNOWLEDGMENTS

This study was supported by the King Abdullah University of Science and Technology. We thank the staff at the KAUST Supercomputing Laboratory (KSL) and the Extreme Computing Research Center (ECRC) for supplying the critical computational resources necessary for the experiments in this research.

ORCID

Sihan Chen https://orcid.org/0000-0003-1598-8341

REFERENCES

Abdulah, S., Li, Y., Cao, J., Ltaief, H., Keyes, D. E., Genton, M. G., & Sun, Y. (2023). Large-scale environmental data science with ExaGeoStatR. *Environmetrics*, *34*, e2770.

Abdulah, S., Ltaief, H., Sun, Y., Genton, M. G., & Keyes, D. E. (2018a). ExaGeoStat: A high performance unified software for geostatistics on manycore systems. *IEEE Transactions on Parallel and Distributed Systems*, *29*, 2771–2784.

Abdulah, S., Ltaief, H., Sun, Y., Genton, M. G., & Keyes, D. E. (2018b). *Parallel approximation of the maximum likelihood estimation for the prediction of large-scale geostatistics simulations*. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)* (pp. 98–108). IEEE.

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

Apanasovich, T. V., Genton, M. G., & Sun, Y. (2012). A valid Matérn class of cross-covariance functions for multivariate random fields with any number of components. *Journal of the American Statistical Association*, *107*, 180–193.

Asano, T., Ranjan, D., Roos, T., Welzl, E., & Widmayer, P. (1997). Space-filling curves and their use in the design of geometric data structures. *Theoretical Computer Science*, *181*, 3–15.

Augonnet, C., Thibault, S., Namyst, R., & Wacrenier, P.-A. (2009). *Starpu: a unified platform for task scheduling on heterogeneous multicore architectures*. In *Euro-Par 2009 Parallel Processing: 15th International Euro-Par Conference, Delft, The Netherlands, August 25-28, 2009. Proceedings 15* (pp. 863–874). Springer.

Banerjee, S., Gelfand, A. E., Finley, A. O., & Sang, H. (2008). Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society: Series B(Statistical Methodology)*, *70*, 825–848.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, *18*, 509–517.

Bosilca, G., Bouteiller, A., Danalis, A., Faverge, M., Hérault, T., & Dongarra, J. J. (2013). Parsec: Exploiting heterogeneity to enhance scalability. *Computing in Science & Engineering*, *15*, 36–45.

Cao, Q., Abdulah, S., Alomairy, R., Pei, Y., Nag, P., Bosilca, G., Dongarra, J., Genton, M. G., Keyes, D. E., Ltaief, H., & Sun, Y. (2022). *Reshaping geostatistical modeling and prediction for extreme-scale environmental applications*. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis* (pp. 1–12). IEEE.

Cerioli, A., & Riani, M. (1999). The ordering of spatial data and the detection of multiple outliers. *Journal of Computational and Graphical Statistics*, *8*, 239–258.

Chen, W., Genton, M. G., & Sun, Y. (2021). Space-time covariance structures and models. *Annual Review of Statistics and Its Application*, *8*, 191–215.

Cressie, N., & Johannesson, G. (2008). Fixed rank kriging for very large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *70*, 209–226.

Cuthill, E., & McKee, J. (1969). *Reducing the bandwidth of sparse symmetric matrices*. In *Proceedings of the 1969 24th national conference* (pp. 157–172). New York: Association for Computing Machinery.

Duran, A., Ayguadé, E., Badia, R. M., Labarta, J., Martinell, L., Martorell, X., & Planas, J. (2011). OmpSs: a proposal for programming heterogeneous multi-core architectures. *Parallel Processing Letters*, *21*, 173–193.

Edwards, H. C., Trott, C. R., & Sunderland, D. (2014). Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *Journal of Parallel and Distributed Computing*, *74*, 3202–3216.

Elhorst, J. P., Gross, M., & Tereanu, E. (2021). Cross-sectional dependence and spillovers in space and time: Where spatial econometrics and global var models meet. *Journal of Economic Surveys*, *35*, 192–226.

George, A., & Liu, J. W. (1989). The evolution of the minimum degree ordering algorithm. *SIAM Review*, *31*, 1–19.

Gneiting, T., Genton, M. G., & Guttorp, P. (2007). Geostatistical space-time models, stationarity, separability, and full symmetry. *Monographs on Statistics and Applied Probability*, *107*, 151–175.

Gneiting, T., Kleiber, W., & Schlather, M. (2010). Matérn cross-covariance functions for multivariate random fields. *Journal of the American Statistical Association*, *105*, 1167–1177.

Guinness, J. (2018). Permutation and grouping methods for sharpening Gaussian process approximations. *Technometrics*, *60*, 415–429.

Guo, D., & Gahegan, M. (2006). Spatial ordering and encoding for geographic data mining and visualization. *Journal of Intelligent Information Systems*, *27*, 243–266.

Hilbert, D. (1935). *Über die stetige abbildung einer linie auf ein flächenstück*. In *Dritter Band: Analysis·Grundlagen der Mathematik·Physik Verschiedenes* (pp. 1–2). Springer.

Kale, L. V., & Krishnan, S. (1993). *Charm++ a portable concurrent object oriented system based on C++*. In *Proceedings of the Eighth Annual Conference on Object-Oriented Programming Systems, Languages, and Applications* (pp. 91–108). New York: Association for Computing Machinery.

Kaufman, C. G., Schervish, M. J., & Nychka, D. W. (2008). Covariance tapering for likelihood-based estimation in large spatial data sets. *Journal of the American Statistical Association*, *103*, 1545–1555.

Kim, K., Oh, H.-S., & Park, M. (2023). Principal component analysis for river network data: Use of spatiotemporal correlation and heterogeneous covariance structure. *Environmetrics*, *34*, e2753.

Kim, Y.-J., & Gu, C. (2004). Smoothing spline Gaussian regression: more scalable computation via efficient approximation. *Journal of The Royal Statistical Society Series B: Statistical Methodology*, *66*, 337–356.

Litvinenko, A., Sun, Y., Genton, M. G., & Keyes, D. E. (2019). Likelihood approximation with hierarchical matrices for large spatial datasets. *Computational Statistics & Data Analysis*, *137*, 115–132.

Matérn, B. (1960) Spatial variation. stochastic models and their application to some problems in forest surveys and other sampling investigations.

Mondal, S., Abdulah, S., Ltaief, H., Sun, Y., Genton, M. G., & Keyes, D. E. (2022). *Parallel approximations of the Tukey g-and-h likelihoods and predictions for non-Gaussian geostatistics*. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (pp. 379–389). IEEE.

Moraga, P., & Montes, F. (2011). Detection of spatial disease clusters with Lisa functions. *Statistics in Medicine*, *30*, 1057–1071.

Morton, G. (1966). *A computer oriented geodetic data base and a new technique in file sequencing*. IBM Ltd.

Ombao, H., Shao, X., Rykhlevskaia, E., Fabiani, M., & Gratton, G. (2008). Spatio-spectral analysis of brain signals. *Statistica Sinica*, 18(4),1465–1482.

Porcu, E., & White, P. A. (2022). Random fields on the hypertorus: covariance modeling and applications. *Environmetrics*, *33*, e2701.

Powell, M. J. (2009). *The BOBYQA algorithm for bound constrained optimization without derivatives. Cambridge NA Report NA2009/06*, Vol. 26. Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge.

Rue, H., & Held, L. (2005). *Gaussian Markov Random fields: Theory and applications*. CRC press.

Rue, H., & Tjelmeland, H. (2002). Fitting Gaussian Markov random fields to Gaussian fields. *Scandinavian Journal of Statistics*, *29*, 31–49.

Sinopoli, B., Schenato, L., Franceschetti, M., Poolla, K., Jordan, M. I., & Sastry, S. S. (2004). Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, *49*, 1453–1464.

Skilling, J. (2004). *Programming the Hilbert curve*. In *AIP Conference Proceedings* (Vol. *707*, pp. 381–387). American Institute of Physics.

Sun, Y., Bowman, K. P., Genton, M. G., & Tokay, A. (2015). A Matérn model of the spatial covariance structure of point rain rates. *Stochastic Environmental Research and Risk Assessment*, *29*, 411–416.

Sun, Y., Li, B., & Genton, M. G. (2012). *Geostatistics for large datasets*. In *Advances and Challenges in Space-time Modelling of Natural Events* (pp. 55–77). Springer.

Wiens, A., Nychka, D., & Kleiber, W. (2020). Modeling spatial data using local likelihood estimation and a Matérn to spatial autoregressive translation. *Environmetrics*, *31*, e2652.

Xu, G., & Genton, M. G. (2017). Tukey g-and-h random fields. *Journal of the American Statistical Association*, *112*, 1236–1249.

Zhang, H. (2004). Inconsistent estimation and asymptotically equal interpolations in model-based geostatistics. *Journal of the American Statistical Association*, *99*, 250–261.

How to cite this article: Chen, S., Abdulah, S., Sun, Y., & Genton, M. G. (2024). On the impact of spatial covariance matrix ordering on tile low-rank estimation of Matérn parameters. *Environmetrics*, *35*(6), e2868. https://doi.org/10.1002/env.2868

APPENDIX . SUPPLEMENTARY EXPERIMENTAL RESULTS

In Sections 5.1 and 5.2, we analyzed the parameter estimation accuracy and the effect of TLR compression to a given covariance matrix generated using a Matérn covariance kernel, with synthetic data sets generated using rough settings (i.e., the smoothness parameter $\nu = 0.5$). In the appendix, we show some results using smooth settings ($\nu = 1$), and in this case, for the weak, medium and strong correlation structures, the true values of the variance, range, and smoothness parameters are set to be ($\sigma^2 = 1, \beta = 0.025, \nu = 1$), ($\sigma^2 = 1, \beta = 0.075, \nu = 1$), and ($\sigma^2 = 1, \beta = 0.2, \nu = 1$), respectively.

A.1 Parameter estimation

In Figure A1, we demonstrate the parameter estimation results of small-scale data ($n = 1600$ locations, with tile size $nb = 320$) experiments performed on a normal machine with 40 CPU cores, with data generated from the smooth settings. As in Section 5.1.1, the experiments are repeated 100 times using different synthetic data sets generated independently each time for each setting. In the first three rows, we show the estimation results of the variance, range, and smoothness parameters in each setting, while in the fourth row, we plot the estimated value of the function κ defined in (6), to give an overall evaluation of the accuracy of the three parameters.

A.2 Tile ranks

As in Section 5.2, we show the heatmaps of some sample covariance matrices with weak, medium, and strong correlation structures in the smooth settings in Figure A2. In addition, for each type of correlation structure, we generated 100 sets of spatial data and made boxplots for the minimum, median, mean, and maximum values of the off-diagonal tiles in the corresponding covariance matrices, which is shown in Figure A3; we also show the histogram of these ranks from the 100 data sets, which we demonstrate in Figure A4.

A.3 Cholesky factorization performance

In Figure 9, we showed the plots comparing the execution time of Cholesky decomposition using Hilbert ordering with using Morton or KD-Tree. Here in Figure A5, we, in addition, show the boxplots of the same experiments to make the experimental results more complete. The conclusion that we can draw from Figure A5 is the same as from Figure 9, that the computation speed using Hilbert is superior to using the other ordering methods in most cases, except in some cases when $N = 10{,}000$ and $N = 40{,}000$.

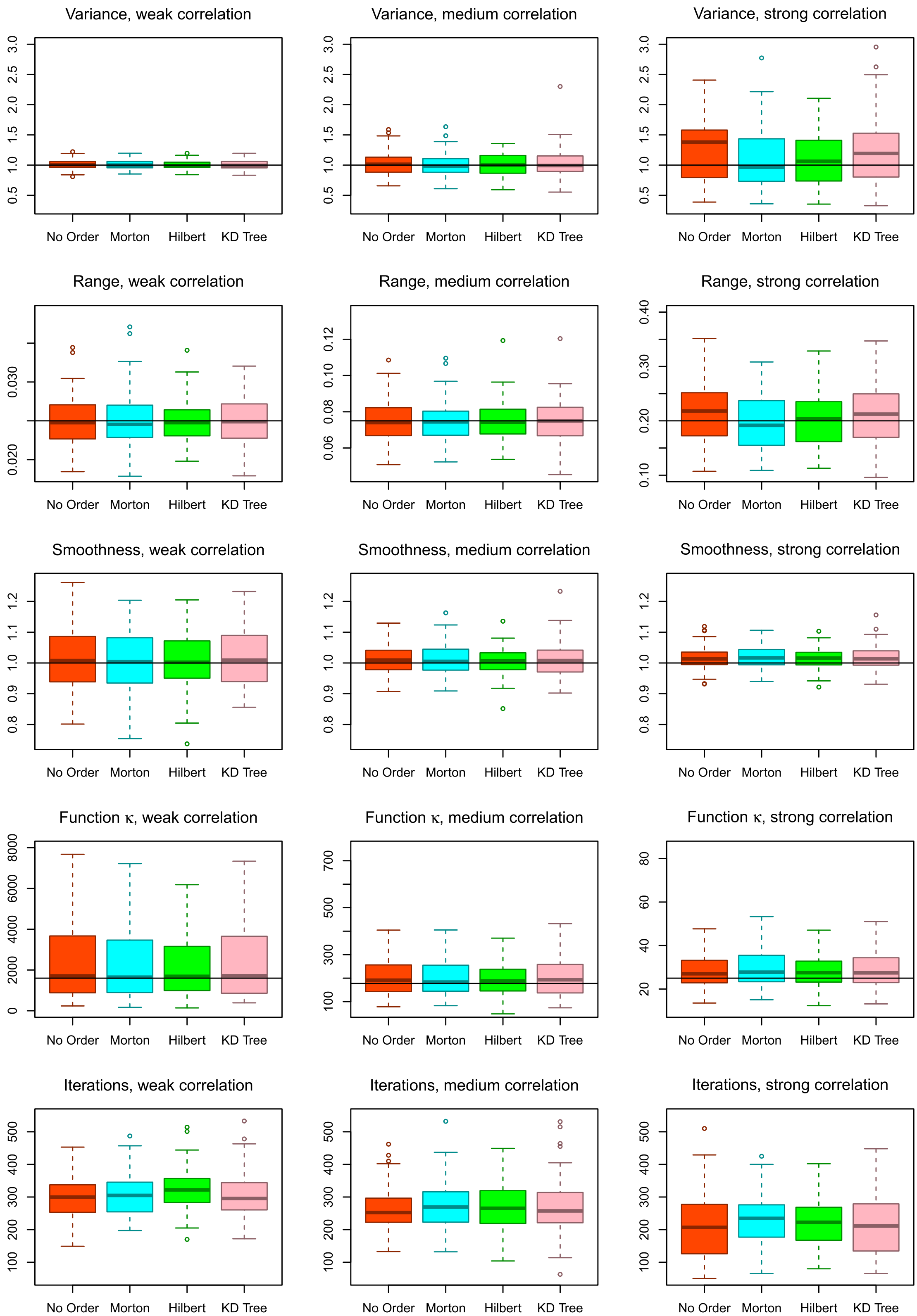

FIGURE A1 BoxPlots of TLR estimation accuracy with $n = 1600$ under either No Order or Morton, Hilbert, KD-Tree orderings. The data are generated from smooth settings. The horizontal lines in the first four rows denote the true values of the corresponding parameters or the function κ in (6).

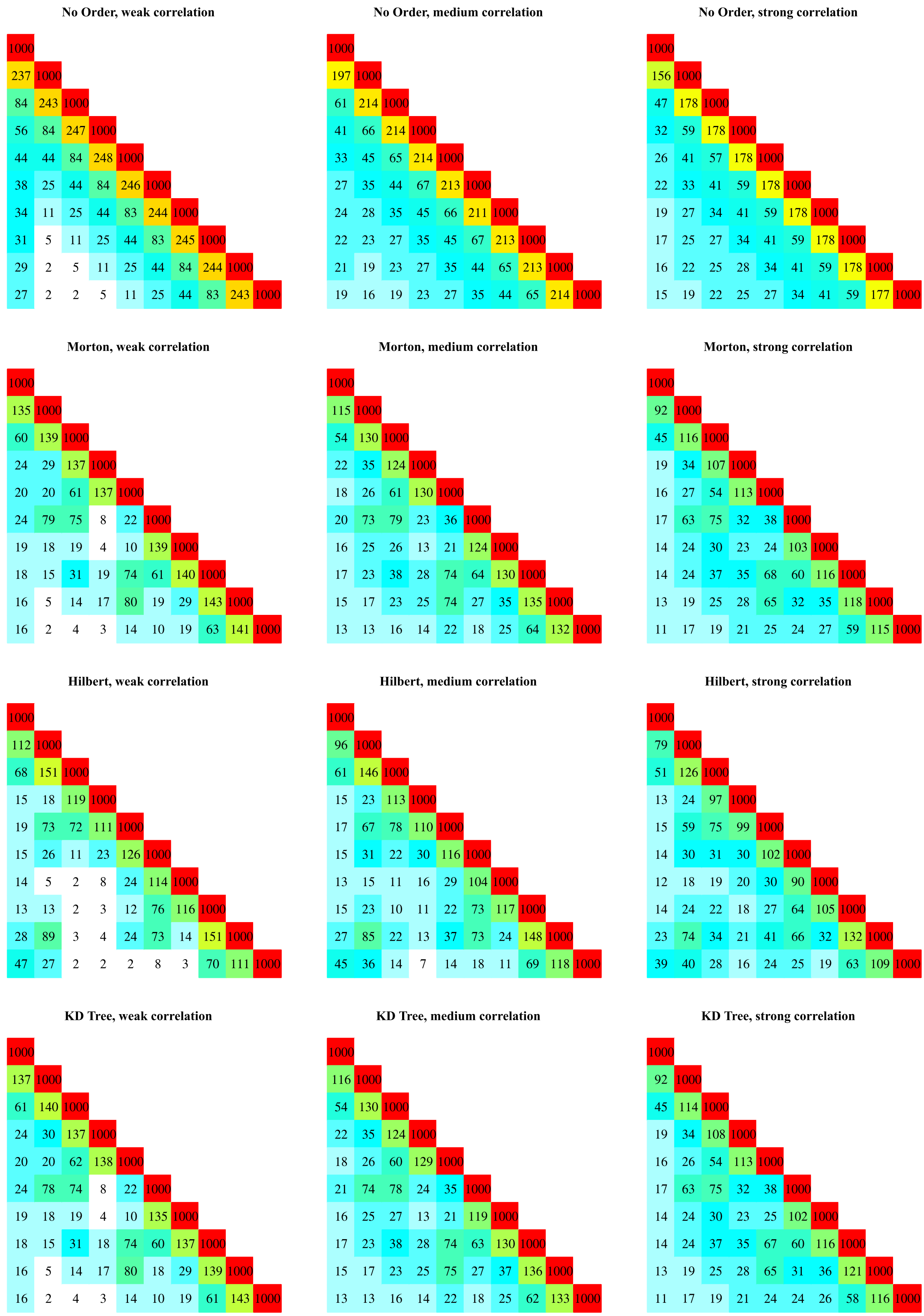

FIGURE A2 Heatmaps of tile ranks for various ordering algorithms for weak, medium, and strong correlation structures in smooth settings. Each small square symbolizes a 1000×1000 tile, annotated with its corresponding rank. Diagonal tiles maintain full rank. Within each heatmap, darker colors indicate higher ranks.

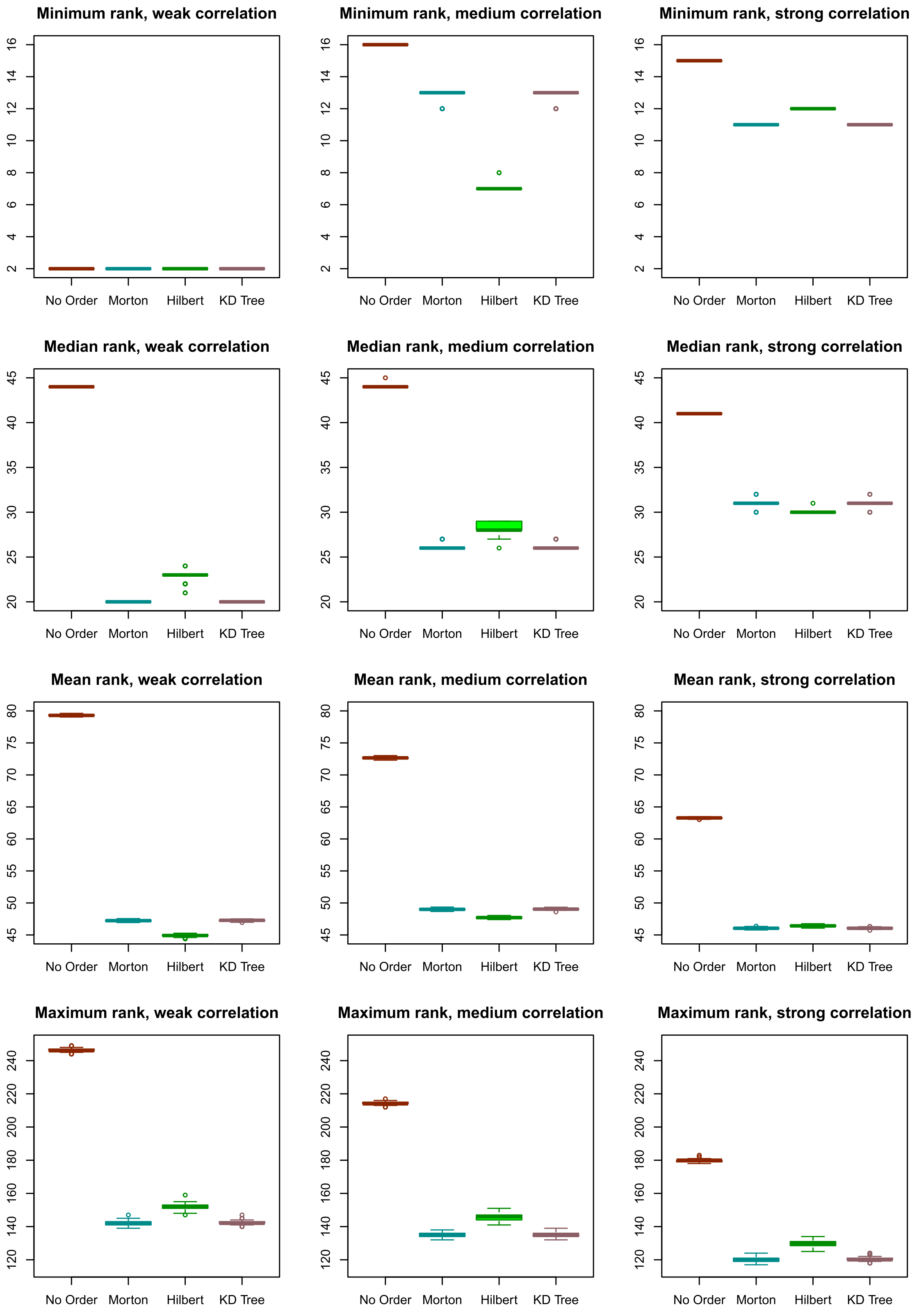

FIGURE A3 BoxPlots of the minimum, median, mean, and maximum of off-diagonal tile ranks with different ordering methods with weak, medium, and strong correlation structures in smooth settings. We generated 100 sets of data with $n = 10{,}000$ locations, and they are all divided into 1000×1000 tiles.

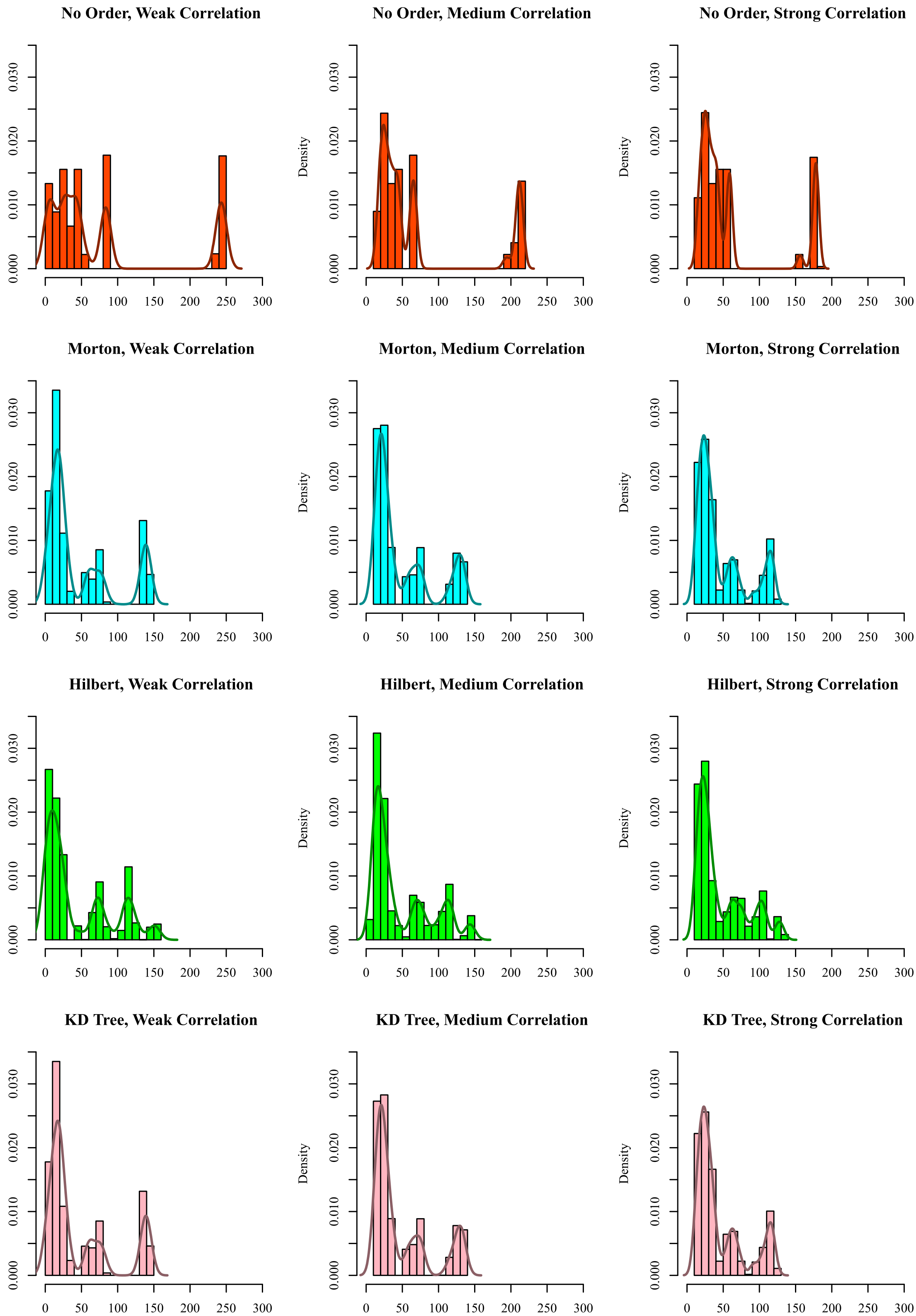

FIGURE A4 Histograms of the proportions and curves of the empirical densities of off-diagonal tile ranks with different ordering methods with weak, medium, and strong correlation structures in the smooth settings. We generated 100 sets of data with $n = 10{,}000$ locations, and they are all divided into 1000×1000 tiles.

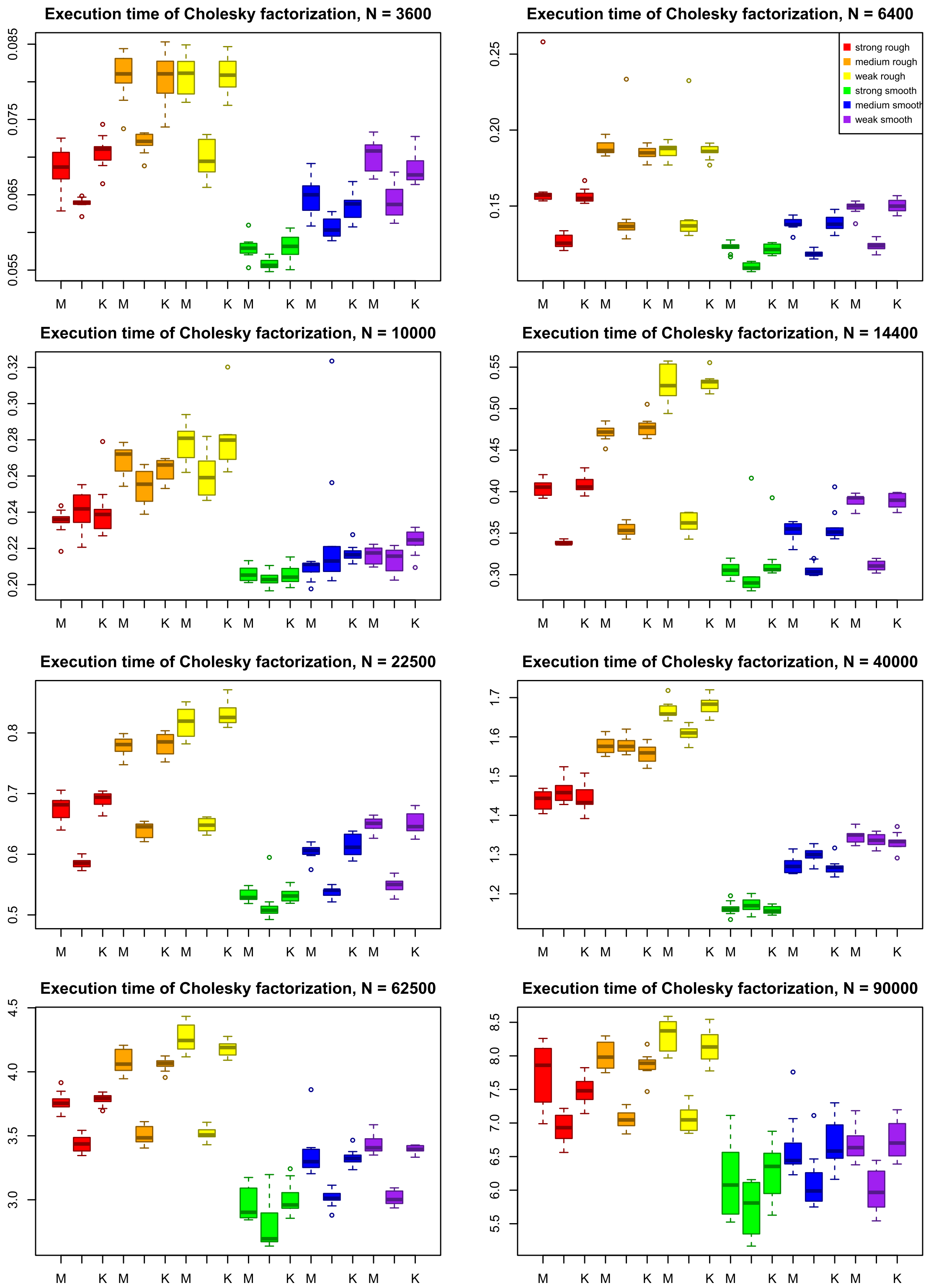

FIGURE A5 TLR Cholesky factorization execution time. Variance set at $\sigma^2 = 1$. The weak, medium, and strong refer to the range parameter β values of 0.03, 0.1, 0.3, respectively. The words "rough" and "smooth" correspond to the cases where $\nu = 0.5$ and $\nu = 1$, respectively. The total six cases of dependence structures are represented by six different colors in the boxplots, as described in the legends. For the x-axes, to make it brief, we use the initials to denote the corresponding ordering algorithms, namely "M" for Morton, "H" for Hilbert, and "K" for KD-Tree.