TECHNOMETRICS
2025, VOL. 67, NO. 3, 546–558
https://doi.org/10.1080/00401706.2025.2475784

Taylor & Francis
Taylor & Francis Group

Block Vecchia Approximation for Scalable and Efficient Gaussian Process Computations

Qilong Pan[a], Sameh Abdulah[b], Marc G. Genton[a], and Ying Sun[a]

[a]Statistics Program, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia; [b]Applied Mathematics and Computational Sciences Program, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia

ABSTRACT

Gaussian Processes (GPs) are vital for modeling and predicting irregularly-spaced, large geospatial datasets. However, their computations often pose significant challenges in large-scale applications. One popular method to approximate GPs is the Vecchia approximation, which approximates the full likelihood via a series of conditional probabilities. The classical Vecchia approximation uses univariate conditional distributions, which leads to redundant evaluations and memory burdens. To address this challenge, our study introduces block Vecchia, which evaluates each multivariate conditional distribution of a block of observations, with blocks formed using the K-means algorithm. The proposed GPU framework for the block Vecchia uses varying batched linear algebra operations to compute multivariate conditional distributions concurrently, notably diminishing the frequent likelihood evaluations. Diving into the factor affecting the accuracy of the block Vecchia, the neighbor selection criterion is investigated, where we found that the random ordering markedly enhances the approximated quality as the block count becomes large. To verify the scalability and efficiency of the algorithm, we conduct a series of numerical studies and simulations, demonstrating their practical utility and effectiveness compared to the exact GP. Moreover, we tackle large-scale real datasets using the block Vecchia method, that is, high-resolution 3D profile wind speed with a million points.
(Code: *https://github.com/kaust-es/ParallelBlockVecchiaGP*)

ARTICLE HISTORY
Received May 2024
Accepted February 2025

KEYWORDS
Clustering; GPU acceleration; Large-scale geospatial data; Likelihood approximation; Nearest neighbors; Vecchia algorithm

1. Introduction

Gaussian Process (GP) is an essential tool in spatial statistics, widely used for modeling and predicting geospatial data. Nonetheless, there is a major challenge when handling large datasets from irregularly distributed locations.

Indeed, assume there are n locations $\mathbf{s}_1, \ldots, \mathbf{s}_n \in \mathbb{R}^d$, where d is the dimension of locations. The GPs are specified by a mean function and a covariance function, $GP(\mu(\mathbf{s}), C_{\boldsymbol{\theta}}(\cdot,\cdot))$, where $C_{\boldsymbol{\theta}}$ is usually assumed to have a parametric form and $\boldsymbol{\theta} \in \mathbb{R}^p$. Let $y_i := y(\mathbf{s}_i) \in \mathbb{R}$ represent single observation at location $\mathbf{s}_i$ and denote the observed data vector by $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$. The data $\boldsymbol{y}$ can be modeled using GPs as $\boldsymbol{y} \sim \mathcal{N}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$, where $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ is a covariance matrix with (i,j) entry determined by a given covariance function $C_{\boldsymbol{\theta}}(\mathbf{s}_i, \mathbf{s}_j)$.

Without loss of generality, we assume that $\boldsymbol{y}$ has a mean of zero. Statistical inference about $\boldsymbol{\theta}$ often relies on the Gaussian log-likelihood function

$$\ell(\boldsymbol{\theta}; \boldsymbol{y}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}_{\boldsymbol{\theta}}| - \frac{1}{2}\boldsymbol{y}^\top \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}\boldsymbol{y}, \quad (1)$$

where the Cholesky decomposition of $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ imposes memory burden $\mathcal{O}(n^2)$ and computational cost $\mathcal{O}(n^3)$, for example, the covariance matrix of 200K locations requires 160GB memory and 2.6 Petabyte (PB) flops.

Various studies have focused on the computational and memory challenges of modeling and predicting using large-scale GPs. These efforts primarily explore two strategies: sparse approximation and low-rank approximation of the covariance matrix (Furrer, Genton, and Nychka 2006; Kaufman, Schervish, and Nychka 2008; Bevilacqua et al. 2016). For instance, sparse approximation techniques such as covariance tapering have been widely studied. This method applies a tapering function that diminishes to zero with increasing distance between two points to the covariance function. This function transforms the original dense covariance matrix into a sparse format, thus, reducing the computations and memory burden.

In low-rank approximation, the full covariance matrix is approximated with two matrices of lower rank, which is achieved by $\mathbf{A} \approx \mathbf{U}\mathbf{V}^\top$, where $\mathbf{U}$ and $\mathbf{V}$ are matrices with fewer columns than $\mathbf{A}$. This reduces the rank of the matrix to the number of columns in $\mathbf{U}$ and $\mathbf{V}$, thus, significantly reducing the size of the matrix operations involved (Huang and Sun 2018; Mondal et al. 2023). Additionally, modern hardware technologies capable of low-precision calculations, such as NVIDIA GPUs, have enhanced the optimization of the sparse

CONTACT Qilong Pan *qilong.pan@kaust.edu.sa* Statistics Program, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia.

Supplementary materials for this article are available online. Please go to *www.tandfonline.com/r/TECH*.

covariance matrix. This improvement is achieved by assigning different precision levels to various sections of the dense covariance matrix, thus, reducing computational complexity rather than completely omitting these sections (Abdulah et al. 2019, 2021; Cao et al. 2022). For low-rank approximations, various methods are employed to enable faster computations and reduced memory compared to the original dense matrix (Katzfuss and Cressie 2011; Huang and Sun 2018; Abdulah et al. 2018b; Mondal et al. 2022).

The Vecchia approximation is among the earliest statistical methods for approximating GPs. It approximates the joint distribution of a GP by decomposing it into a product of independent univariate conditional distributions (Vecchia 1988). This approach reduces computational demands and memory burden by using only a limited number of neighboring points in each univariate conditional distribution, improving speed and lowering memory requirements. The approximated log-likelihood has a computational complexity of $\mathcal{O}(nm^3)$ and memory complexity of $\mathcal{O}(nm^2)$, compared to the exact $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$. Here, n is the total number of spatial locations, and m is the number of neighbors included in the conditioning set ($m \ll n$). Nonetheless, the scalability of the Vecchia approximation encounters two primary hurdles. First, the decomposition in univariate conditional distributions may lead to the redundant computation of multiple conditional log-likelihoods, particularly when they involve common neighbors. This redundancy not only increases computational overhead but also decreases the efficiency of processing. Second, the main computations in the Vecchia approximation are small matrix operations, which are better suited for execution on CPUs than Graphics Processing Units (GPUs). More importantly, the performance of GPUs cannot be fully used for conducting the small matrix operations (Pan et al. 2024). GPUs are designed for handling computationally intensive tasks that benefit from extensive parallelization, such as matrix-to-matrix multiplications. However, the tasks arising from the Vecchia approximation which is characterized by numerous small and discrete operations are not inherently compatible with the GPU architecture's strengths. For a detailed review of the Vecchia approximation, refer to (Guinness 2018, 2021; Katzfuss and Guinness 2021; Zhang, Tang, and Banerjee 2021; Zhang and Katzfuss 2022; Katzfuss, Guinness, and Lawrence 2022; Jimenez and Katzfuss 2023; Pan et al. 2024).

In this work, we introduce a block version of Vecchia approximation using the batched GPU framework, that is, block Vecchia algorithm, where the full likelihood is approximated by the product of a series of multivariate conditional probabilities represented by blocks and their neighbors, and modern GPU architectures are used to accelerate the proposed algorithm. These blocks are created by the K-means algorithm and the classic Vecchia can be viewed as a special case when each block size is 1. Technically, built upon the MAGMA library (Dong et al. 2016), the enhancement in our GPU framework lies in applying varying batched matrix operations, which execute the computationally light tasks, for example, the Cholesky decomposition of the small covariance matrix, in parallel on a single GPU by creating multiple block threads. The batched operations accelerate the algorithm and the consideration of multivariate conditional probability creates computationally intensive tasks, which makes the compute node in GPU more efficient compared to univariate conditional cases in the classic Vecchia. Additionally, further investigation establishes that the block Vecchia approximation algorithm significantly enhances the capability to manage larger problem sizes, outperforming the classic Vecchia algorithm's scalability. In the end, through comprehensive evaluations encompassing numerical studies, simulation experiments, and analyses of real datasets in the context of the Matérn covariance function considered in the GP modeling, key findings have been identified, underscoring the efficiency and effectiveness of the block Vecchia method:

1. **Choice of block count and conditioning size**: In general, the modeling capabilities, including parameter estimation and prediction, improve with an increase in both the block count and the conditioning size in block Vecchia approximations. This insight is crucial for optimizing the approximation process, ensuring both efficiency and precision.
2. **Importance of ordering**: The sequence of blocks plays a critical role in the accuracy of the approximation, that is, the random ordering outperforms others as the number of blocks increases. This finding highlights the need for smart ordering to enhance the accuracy of the block Vecchia approximation.
3. **Performance enhancements and scalability**: Remarkably, the block Vecchia method demonstrates an approximately 80X speedup and 40X larger problem size compared to the classic Vecchia algorithm. This scalability enables the algorithm to leverage modern computational resources effectively, making it a powerful tool for addressing large-scale statistical modeling challenges.
4. **Efficient and accurate prediction**: Similar to the parameter estimation, the prediction accuracy, such as mean square error and standard deviation, can be improved along with the increase of block count and conditioning size.

The article is structured as follows: Section 2 offers a detailed explanation of our proposed implementation. Section 3 presents the evaluation of our implementation from a numerical study and simulation experiments compared to the exact GP. Section 4 provides results on the real dataset in 3D million-level data points and we conclude in Section 5.

2. Block Vecchia Framework

This section provides an overview of the proposed framework for the block Vecchia algorithm. It begins with an explanation of the preprocessing steps, including location clustering, block reordering, and nearest neighbor selection for each block centroid. Following this, the memory requirements for the block Vecchia algorithm are discussed in detail, alongside a thorough description of the proposed implementation. Next, the prediction process using the block Vecchia algorithm is outlined. Finally, a comparison is presented, highlighting the expected memory usage and computational complexity of the block Vecchia algorithm relative to the classic Vecchia algorithm.

2.1. Clustering and Block Permutation

The first step of the block Vecchia is to discover natural groupings in data based on some similarity measures by clustering methods. Clustering involves using algorithms to organize a set of points into groups (or clusters) where points within the same group are more similar to each other than to those in different groups. Examples include K-means, hierarchical clustering, spectral clustering, and density-based spatial clustering (Xu and Wunsch 2005; Saxena et al. 2017). In the context of the block Vecchia, we use the K-means clustering algorithm, considering its simplicity and portability for large-scale computing. For example, 500 locations are randomly generated in the unit area, and 80 blocks are pre-specified. The blocks are clustered based on the coordinates using K-means, and the results are visualized in Figure 1.

Following the K-means clustering, the block Vecchia algorithm requires reordering the blocks. Considering that the conditional probability in the Vecchia approximation is sensitive to its previous points, the ordering is crucial as it influences the selection of candidates for approximating the conditional probability. Figure 2 presents an illustrative example involving 500 uniformly random locations within $[0, 1] \times [0, 1]$. The example demonstrates four permutations and their 30 nearest neighbors using: Morton reordering (Walker 2018), random reordering (Guinness 2018), KDtree reordering (Bentley 1975) , maxmin reordering (mmd) Guinness (2018) and Hilbert reordering (Hilbert 1935; Chen et al. 2024). Morton reordering, or the Z-order reordering, is a space-filling curve approach that reduces multi-dimensional data to one dimension while preserving the locality of data points. This technique is beneficial in improving memory access patterns and data compression (Walker 2018). Random reordering rearranges the data points randomly, which can be useful for mitigating biases presented in the original ordering and potentially improving model robustness. Maxmin reordering is achieved by reordering the data in a way that maximizes the minimum distance, typically ensuring that the most informative points are considered earlier in the computational process. KDtree reordering is a space-partitioning data structure that helps optimize queries and operations that depend on spatial proximity. Like Morton, Hilbert reordering tends to preserve a better locality than the Z-order curve, which can enhance performance in specific applications (Chen et al. 2022).

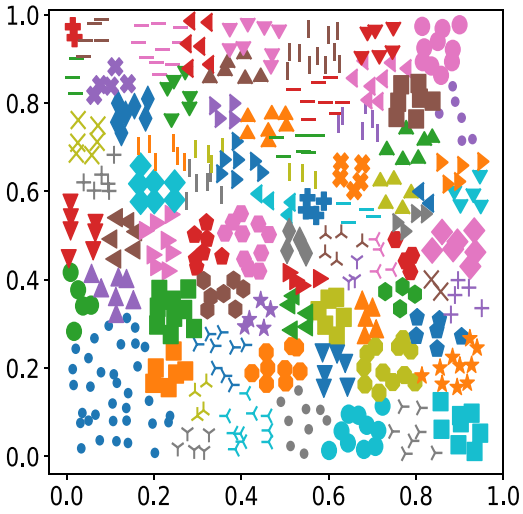


Figure 1. An example illustrating K-means in the block Vecchia, with 500 random locations in $[0, 1] \times [0, 1]$ and 80 blocks. Shape markers represent the blocks.

2.2. Block Vecchia Algorithm

Having a set of geospatial data consisting of n spatial locations and their corresponding observations denoted by $\boldsymbol{y}$, as well as any permutation ζ of the given centroids of blocks, the likelihood, or the joint density, for the observations $\boldsymbol{y}$ can be represented as a product of a series of multivariate conditional densities:

$$
\begin{aligned}
L(\boldsymbol{\theta}; \boldsymbol{y}) &= p_{\boldsymbol{\theta}}\left(y_1, \ldots, y_n\right) \\
&= p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{B_1}\right) \prod_{i=2}^{bc} p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{B_i} \mid \boldsymbol{y}_{B_1}, \ldots, \boldsymbol{y}_{B_{i-1}}\right), \qquad (2) \\
&= p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{B_1^\zeta}\right) \prod_{i=2}^{bc} p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{B_i^\zeta} \mid \boldsymbol{y}_{B_1^\zeta}, \ldots, \boldsymbol{y}_{B_{i-1}^\zeta}\right), \qquad (3)
\end{aligned}
$$

where arbitrary partitions of the observations and permutations of the partitions will not affect the joint density. In (2), $B_1, B_2, \ldots, B_{bc}$ are the partition of $\{1, 2, \ldots, n\}$, for example, an integer set $B_i = \{b_{i1}, \ldots, b_{il_i}\}$ where b_{ij} is the *jth* point in the *ith* block and l_i is the size of the *ith* block, and *bc* means the

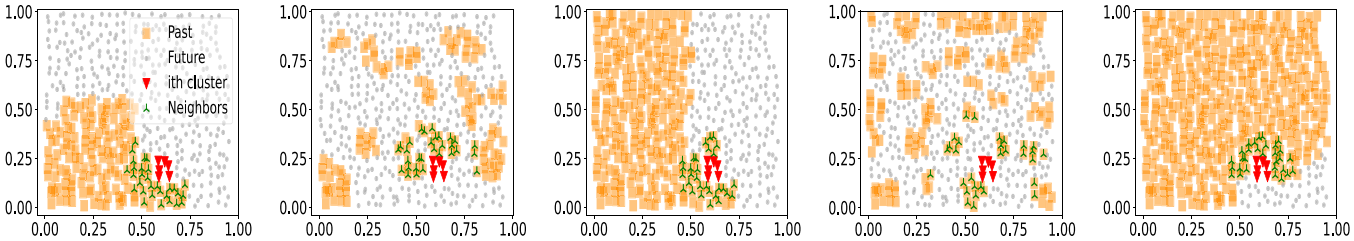


Figure 2. An example illustrating the impact of orderings on the neighbor selection for block Vecchia, where we have 500 uniform random locations in $[0, 1] \times [0, 1]$ and 80 blocks, the square, small circle, triangle-down, and tri-up represents past points, future points, blocks, and neighbors, respectively.

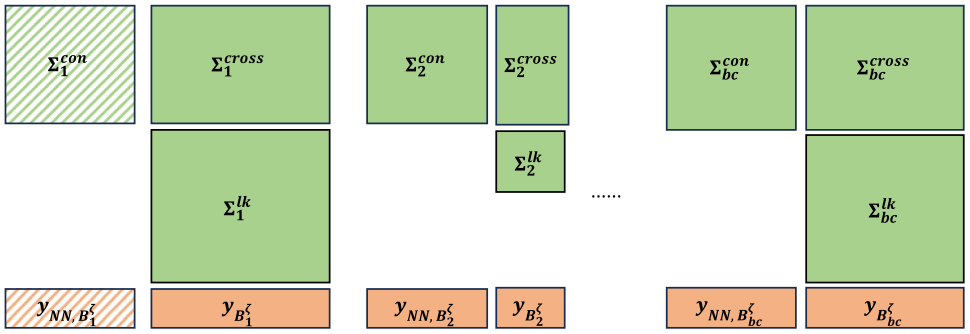


Figure 3. Block Vecchia algorithm (Notations as shown in Algorithm 1 in the supplementary material S1).

block count with $\sum_{i=1}^{bc} l_i = n$. Here ζ in (3) represents the permutation of the blocks. The block Vecchia approximation replaces the complete conditioning vectors $(\boldsymbol{y}_{B_1^\zeta}, \ldots, \boldsymbol{y}_{B_{i-1}^\zeta})$ with a subvector, where the length of subvector is far less than the length of the whole vector. Specifically, a subvector, that is, m_i nearest neighbors $\boldsymbol{y}_{NN,B_i^\zeta} = (y_{j_1}, \ldots, y_{j_{m_i}})^\top$, are selected from the labeling of $B_1^\zeta \cup B_2^\zeta \cup \cdots \cup B_{i-1}^\zeta$. In the definition of $\boldsymbol{y}_{NN,B_i^\zeta}$, m_i represents the size of conditioning set for the block B_i^ζ and $(y_{j_1}, \ldots, y_{j_{m_i}})$ are the selected points for approximating the *ith* cluster conditional probability. Then, we define the approximation as

$$p_{\boldsymbol{\theta},\zeta,NN,B}(y_1, \ldots, y_n) = p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{B_1^\zeta}\right) \prod_{i=2}^{bc} p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{B_i^\zeta} \mid y_{j_1}, \ldots, y_{j_{m_i}}\right)$$
$$= p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{B_1^\zeta}\right) \prod_{i=2}^{bc} p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{B_i^\zeta} \mid \boldsymbol{y}_{NN,B_i^\zeta}\right). \quad (4)$$

Here, *NN* represents the nearest neighbor selection, and $B = \{B_1, \ldots, B_{bc}\}$ is the set of blocks. In the block Vecchia algorithm, the approximation quality relies on the partition B of observations and the permutation ζ of the partition.

To implement the block Vecchia algorithm, for each spatial block, we should compute three covariance matrices, that is, the conditioning covariance matrix constructed by its nearest neighbors $\boldsymbol{\Sigma}_i^{con}$, the cross-covariance matrix between the block and its nearest neighbors $\boldsymbol{\Sigma}_i^{cross}$, and the covariance matrix constructed by points in the ith block $\boldsymbol{\Sigma}_i^{lk}$. Figure 3 depicts the required vector/matrix for each spatial block. Here $\boldsymbol{y}_{B_i^\zeta}$ and $\boldsymbol{y}_{NN,B_i^\zeta}$ are the ith block's and its neighbors' observations, respectively, which exactly match the representation in (4) (the dashed $\boldsymbol{\Sigma}_1^{con}$ and $\boldsymbol{y}_{NN,B_1^\zeta}$ are supplementary position for batched operations). For every pair $(\boldsymbol{y}_{NN,B_i^\zeta}, \boldsymbol{y}_{B_i^\zeta}, \boldsymbol{\Sigma}_i^{con}, \boldsymbol{\Sigma}_i^{cross}, \boldsymbol{\Sigma}_i^{lk})$, their log-likelihoods, $p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{B_i^\zeta} \mid \boldsymbol{y}_{NN,B_i^\zeta}\right)$, are independent of each other. That is to say, the task of computing log-likelihood in (4) can be divided into bc independent small tasks using batched operations, including batchedPOTRF (batched Cholesky decomposition), batchedTRSM (batched triangular linear solver), batchedGEMM (batched matrix-to-matrix multiplication), batchedGEMV (batched matrix-to-vector multiplication), batchedDotProduct (batched inner product). The details of the implementation are summarized in Algorithm 2 in the supplementary material S1, including the clustering, block permutation, nearest neighbor searching, covariance-related matrix construction, and varying batched operations for Vecchia computation (Dong et al. 2016).

2.3. Block Vecchia Prediction

Let us consider the task of predicting values at new spatial locations $\boldsymbol{S}_* = \{\mathbf{s}_{n+1}, \mathbf{s}_{n+2}, \ldots, \mathbf{s}_{n+n^*}\}$ based on observed data $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^\top$ at locations $\boldsymbol{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n\}$. In the standard GP framework, the joint distribution of the observed and new data is multivariate normal:

$$\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{y}_* \end{pmatrix} \sim \mathcal{N}_{n+n^*}\left(\begin{pmatrix} \boldsymbol{\mu}_y \\ \boldsymbol{\mu}_{y_*} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{yy} & \boldsymbol{\Sigma}_{yy_*} \\ \boldsymbol{\Sigma}_{y_*y} & \boldsymbol{\Sigma}_{y_*y_*} \end{pmatrix}\right),$$

where $\boldsymbol{y}_* = (y_{n+1}, y_{n+2}, \ldots, y_{n+n^*})^\top$ are the values to be predicted, and $\boldsymbol{\Sigma}_{yy}$, $\boldsymbol{\Sigma}_{yy_*}$, and $\boldsymbol{\Sigma}_{y_*y_*}$ are covariance matrices computed using a specified covariance function parameterized by $\boldsymbol{\theta}$. The exact conditional distribution of $\boldsymbol{y}_*$ given $\boldsymbol{y}$ is then $\boldsymbol{y}_* \mid \boldsymbol{y} \sim \mathcal{N}_{n^*}\left(\boldsymbol{\mu}_{y_*|y}, \boldsymbol{\Sigma}_{y_*|y}\right)$ with $\boldsymbol{\mu}_{y_*|y} = \boldsymbol{\mu}_{y_*} + \boldsymbol{\Sigma}_{y_*y}\boldsymbol{\Sigma}_{yy}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_y)$, $\boldsymbol{\Sigma}_{y_*|y} = \boldsymbol{\Sigma}_{y_*y_*} - \boldsymbol{\Sigma}_{y_*y}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yy_*}$. Computing $\boldsymbol{\Sigma}_{yy}^{-1}$ is computationally expensive for large n. To alleviate this, the block Vecchia approximation approximates the joint conditional probability by limiting the conditioning to a subset of observations. The block Vecchia approximation involves the following steps: (a) Divide the new locations $\boldsymbol{S}_*$ into bc blocks $B_1^*, B_2^*, \ldots, B_{bc}^*$, each containing l nearby locations; (b) For each block B_i^*, select a conditioning set $\boldsymbol{y}_{NN,B_i^*}$ consisting of m nearest neighbor observations from $\boldsymbol{y}$; (c) Approximate the joint conditional probability as

$$p_{\boldsymbol{\theta}}(\boldsymbol{y}_* \mid \boldsymbol{y}) \approx \prod_{i=1}^{bc} p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{B_i^*} \mid \boldsymbol{y}_{NN,B_i^*}\right), \quad (5)$$

where $\boldsymbol{y}_{B_i^*}$ are the predictions for block B_i^*. For each block, the conditional distribution is $\boldsymbol{y}_{B_i^*} \mid \boldsymbol{y}_{NN,B_i^*} \sim \mathcal{N}_{\#B_i^*}\left(\boldsymbol{\mu}_{B_i^*}, \boldsymbol{\Sigma}_{B_i^*}\right)$, with $\boldsymbol{\mu}_{B_i^*} = \boldsymbol{\Sigma}_i^{\text{cross}}\left(\boldsymbol{\Sigma}_i^{\text{con}}\right)^{-1}\boldsymbol{y}_{NN,B_i^*}$, $\boldsymbol{\Sigma}_{B_i^*} = \boldsymbol{\Sigma}_i^{\text{lk}} - \boldsymbol{\Sigma}_i^{\text{cross}}\left(\boldsymbol{\Sigma}_i^{\text{con}}\right)^{-1}\left(\boldsymbol{\Sigma}_i^{\text{cross}}\right)^\top$, where $\boldsymbol{\Sigma}_i^{\text{con}}$ is the covariance matrix of the conditioning set $\boldsymbol{y}_{NN,B_i^*}$, $\boldsymbol{\Sigma}_i^{\text{cross}}$ is the cross-covariance matrix between $\boldsymbol{y}_{B_i^*}$ and $\boldsymbol{y}_{NN,B_i^*}$, $\boldsymbol{\Sigma}_i^{\text{lk}}$ is the covariance matrix among the locations in block B_i^*. This approximation reduces the computational cost from $\mathcal{O}(n^3)$ to $\mathcal{O}(bc \times m^3)$, where $m \ll n$. The prediction using the block Vecchia approximation can be formalized in Algorithm 2 in the supplementary material S1.

The output, $\boldsymbol{y}_*$ and $\{\boldsymbol{\Sigma}_{B_i^*}\}_{i=1}^{bc}$, helps the univariate (and multivariate) conditional simulation and the prediction interval. Considering the simplicity and computational efficiency, the univariate configuration is adopted. The variance $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_{n*})^\top$ is extracted from the diagonal element from the $\{\boldsymbol{\Sigma}_{B_i^*}\}_{i=1}^{bc}$ for the predicted values $\boldsymbol{y}_*$. Then the conditional simulations are conducted by generating each $y_{*j}, j = 1, \ldots n^*$, from $\mathcal{N}(y_{*j}, \sigma_j)$. We calculate the sample mean $\tilde{\mu}_j$ and variance $\tilde{\sigma}_j^2$ and then the 95% confidence interval is $\left(\tilde{\mu}_j - z_{\alpha/2}\tilde{\sigma}_j, \tilde{\mu}_j + z_{\alpha/2}\tilde{\sigma}_j\right)$ with $\alpha = 0.05$. For general applications, we propose the following scheme. First, it is recommended to use a small block count and conditioning size to estimate the parameters. Then, these estimated parameters serve as initial values for accurate settings with larger block counts and conditioning sizes, continuing until a clear convergence trend emerges. This approach eliminates the need to determine the optimal block count and conditioning size explicitly, as real-world applications often vary in problem size and required precision. Employing this automated estimation method provides a more adaptable and practical solution.

2.4. GPU and Batched Linear Algebra

GPUs are specialized hardware designed to handle parallel operations at a massive scale, which makes them particularly well-suited for tasks that involve large-scale numerical computations, for example, GEMM and elementwise operations. Unlike CPUs, GPUs have hundreds of thousands of cores designed for handling multiple operations simultaneously. This parallelism can be leveraged to significantly speedup the computation of complex mathematical models, such as those used in the Vecchia approximation (Pan et al. 2024).

Batched operations involve processing multiple data points or operations simultaneously, thereby reducing the overhead associated with individual computations (Haidar et al. 2015). In the context of the block Vecchia approximation, which consists of abundant independent and computationally light conditional probabilities, batched operations can be employed to improve computational efficiency by processing those covariance matrices and vectors at the same time, for example, lines 17–21 and 24–27 in the Algorithm 1 in the supplementary material S1. In other words, when implemented on a GPU, these matrices and vectors can be processed in parallel batches rather than sequentially as on a CPU, which dramatically reduces the computation overheads. By using GPU acceleration through batched operations, the computational burden of the block Vecchia approximation is significantly reduced, allowing for faster processing times, especially when dealing with large datasets. This not only improves the feasibility of applying such methods in practice but also opens the door to analyzing more complex models and larger datasets that would be computationally prohibitive on traditional CPU-based systems.

2.5. Computational and Memory Complexity

We analyze the memory usage and computational complexity of the block Vecchia implementation, comparing it against the traditional Vecchia approach. The memory footprint of the classic Vecchia algorithm is $\mathcal{O}(nm^2)$ for many small symmetric covariance matrices $\boldsymbol{\Sigma}_i, i = 1, \ldots, n$ and $\mathcal{O}(nm)$ for the conditioning vectors $\boldsymbol{y}_{NN,B_i^\zeta}, i = 1, \ldots, n$ where m and n stands for the conditioning size and observations, respectively. For the block Vecchia algorithm, each block requires three covariance matrices $\boldsymbol{\Sigma}_i^{lk}$, $\boldsymbol{\Sigma}_i^{cross}$, $\boldsymbol{\Sigma}_i^{con}$, $i = 1, \ldots, bc$ where the average block size is approximately n/bc, and two observation vectors $\boldsymbol{y}_{NN,B_i^\zeta}$ and $\boldsymbol{y}_{B_i}$, $i = 1, \ldots, bc$, with memory complexities of $\sim bc(n/bc)^2/2$, $\sim bc \times m(n/bc)/2 \sim bc \times m^2/2$, $\sim bc \times m$, and $\sim bc \times (n/bc)$, respectively. Therefore, the memory for the classic Vecchia is approximately $\sim nm^2/2 + nm$ and for the block Vecchia is $\sim n^2/(2bc) + mn/2 + bc \times m^2/2 + m \times bc + n$. In terms of the arithmetic complexity, for the block Vecchia, the complexity primarily stems from the Cholesky factorization of the covariance matrix, $\sim bc(n/bc)^3/3$ and $\sim bc(m)^3/3$, and matrix-to-matrix multiplication $\sim 2 \times bc \times m(n/bc)^2$. In contrast, with the classic Vecchia approximation, the complexity for the Cholesky factorization operations is $\sim n(m^3/3)$.

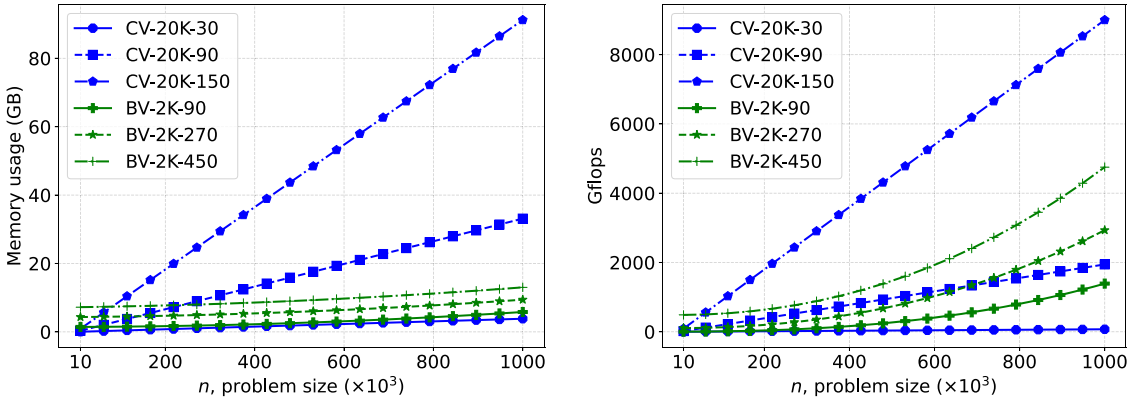
In Figure 4, to have a fair comparison between classic Vecchia and block Vecchia, we set the approximate block size 100, that is, $n/bc \approx 100$ and nearest neighbor 3X larger than the classic Vecchia. Figure 4(a) shows the memory footprint in gigabytes (GB) for increasing problem sizes when employing the classic Vecchia and block Vecchia algorithms. The block Vecchia notably lowers the memory burden. Figure 4(b) demonstrates the floating-point operations (flops) in Gflops when using the block Vecchia and classic algorithms. The figure shows a comparable computation in the number of required flops for the Vecchia algorithm. The observations underscore that the block Vecchia creates computationally intensive tasks compared to the classic Vecchia and then it inspires us to use the modern GPU architecture to accelerate the computations.

3. Numerical Studies

This section assesses the accuracy and computational performance of the proposed block Vecchia algorithm compared to the classical Vecchia algorithm. The block Vecchia code was compiled using GCC version 10.2.0 (or 12.2.0) and CUDA v11.4 (or v11.8), and linked with Intel MKL v2022.2.1, MAGMA v2.6.0 (or v2.7.2), GSL v2.6, and NLopt v2.7.1 optimization libraries. Our machine has 40 CPU cores (Intel(R) Xeon(R) Gold 6230R CPU @ 2.10GHz) and NVIDIA V100 GPU 32GB. Each experiment was repeated five times to ensure repeatability and consistency in the time-to-solution metric. Accuracy was evaluated using the deterministic KL divergence metric alongside qualitative analysis, with simulation studies conducted under various parameter settings. Furthermore, the block Vecchia algorithm was compared with exact GPs in terms of parameter estimation and predictive performance across two real-world applications.

3.1. KL Divergence

In this subsection, we calculate the deterministic metric, KL divergence, to assess the accuracy of the block Vecchia algorithm with the help of *ExaGeoStat* (Abdulah et al. 2018a), on Gaussian random fields with $n = 20{,}000$ spatial locations within $[0, 1] \times [0, 1]$. We also include additional small-scale examples in the



(a) Memory complexity (b) Computational complexity

Figure 4. Comparison of Arithmetic complexity: block Vecchia (BV) versus classic Vecchia (CV) algorithms. The format of the legend is Method-ConditioningSize, for example, CV-20K-30 represents the classic Vecchia with conditioning size 30; BV-2K-90 represents the block Vecchia with block count 2000 and conditioning size 90.

supplementary material S8 and S9 for consumer-grade GPUs, using a dataset size of $n = 8000$.

The KL divergence is defined as

$$D_{\mathrm{KL}}(\mathcal{N}_0 \parallel \mathcal{N}_1) = \frac{1}{2}\left\{\mathrm{tr}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0\right) - n + \log\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_0|}\right\}, \quad (6)$$

where $\mathcal{N}_0, \mathcal{N}_1$ represents two n-dimension Gaussian distributions with zero-mean and covariance matrices $\boldsymbol{\Sigma}_0, \boldsymbol{\Sigma}_1$, representing the exact and Vecchia-approximated covariance matrix. Let $\mathcal{N}_0$ be the exact distribution and $\mathcal{N}_1$ be the approximate distribution by the block Vecchia. Then (6) simplifies as

$$D_{\mathrm{KL}}(\mathcal{N}_0 \parallel \mathcal{N}_1) = \ell_0(\boldsymbol{\theta}; \mathbf{0}) - \ell_a(\boldsymbol{\theta}; \mathbf{0}), \quad (7)$$

where $\ell_0(\boldsymbol{\theta}; \mathbf{0})$ represents the exact log-likelihood at the $\boldsymbol{y} = \mathbf{0}$ which is calculated by *ExaGeoStat* and $\ell_a(\boldsymbol{\theta}; \mathbf{0})$ represents the Vecchia-approximated log-likelihood at $\boldsymbol{y} = \mathbf{0}$. In addition, we rely on the isotropic Matérn covariance function as in (8), and Gneiting (2002) provided other kernels.

$$C_{\boldsymbol{\theta}}(\mathbf{s}_i, \mathbf{s}_j) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\|\mathbf{s}_i - \mathbf{s}_j\|}{\beta}\right)^{\nu} \mathcal{K}_{\nu}\left(\frac{\|\mathbf{s}_i - \mathbf{s}_j\|}{\beta}\right), \quad (8)$$

where $\boldsymbol{\theta} = \left(\sigma^2, \beta, \nu\right)^{\top}$, σ^2 is the variance, $\mathcal{K}_{\nu}(\cdot)$ is the Bessel function of the second kind of order, $\nu\Gamma(\cdot)$ is the gamma function, and $\beta > 0$ and $\nu > 0$ are range and smoothness parameters, respectively. Next, we select the parameter for smoothness ν across values of 0.5, 1.5, and 2.5, representing varying degrees: low, medium, and high, respectively. Furthermore, we adjust the effective range for each smoothness level to 0.1, 0.3, and 0.8, corresponding to low, medium, and high dependency values in the unit square, respectively. Then, the corresponding parameter β is calculated and reported in Table 1. These adjustments are crucial as they influence the correlation within the data (Pan et al. 2024).

In the following experiments, the shorthand format is employed, that is, *block count (bc) - conditioning size (cs) - ordering method*. For instance, *BV-1000-60-Morton* means the Block Vecchia (BV) with $bc = 1000$, $cs = 60$, and Morton ordering; *CV-20K-60-random* means the Classic Vecchia (CV) with $bc = 20{,}000$ and $cs = 60$, and random ordering. In the classic Vecchia algorithm, random ordering is adopted as the baseline due to its superior accuracy within large-scale spatial scenarios (Guinness 2018; Pan et al. 2024). The calculation of the KL divergence is performed as in (7), and the outcomes are visualized in Figures 5–7. In all subfigures, the y-axis represents the logarithm of the KL divergence value. The numerical studies are divided into three parts, presenting some of the main results (please refer to S2, S3, and S4 in the supplementary materials for the comprehensive results).

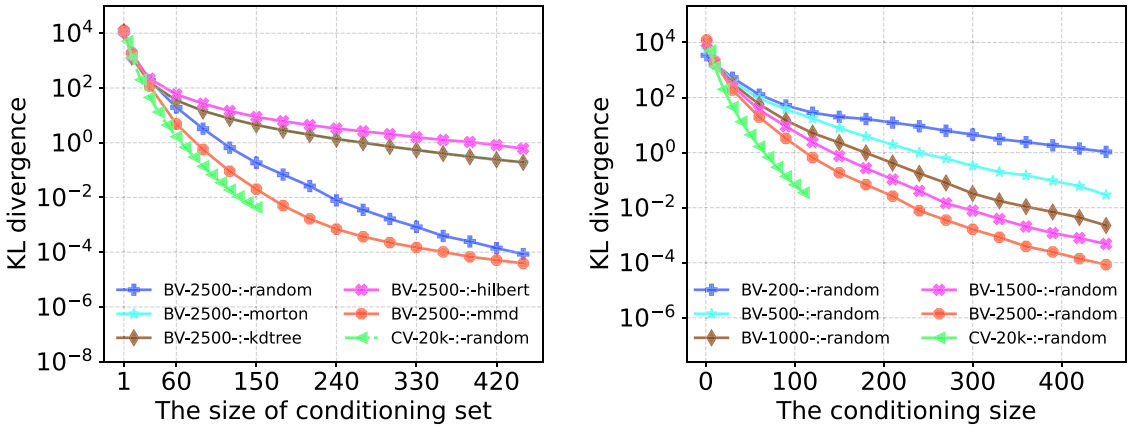
Table 1. The cross combinations of low/medium/high smoothness and low/medium/high effective range.

	$\nu = 0.5$	$\nu = 1.5$	$\nu = 2.5$
effective range=0.1	0.026270	0.017512	0.014290
effective range=0.3	0.078809	0.052537	0.042869
effective range=0.8	0.210158	0.140098	0.114318

NOTE: Each entry in the table represents β, and the 0.1, 0.3, 0.8 are the statistical effective range, that is, the distance at which spatial correlations fall to 5% (Huang et al. 2021).

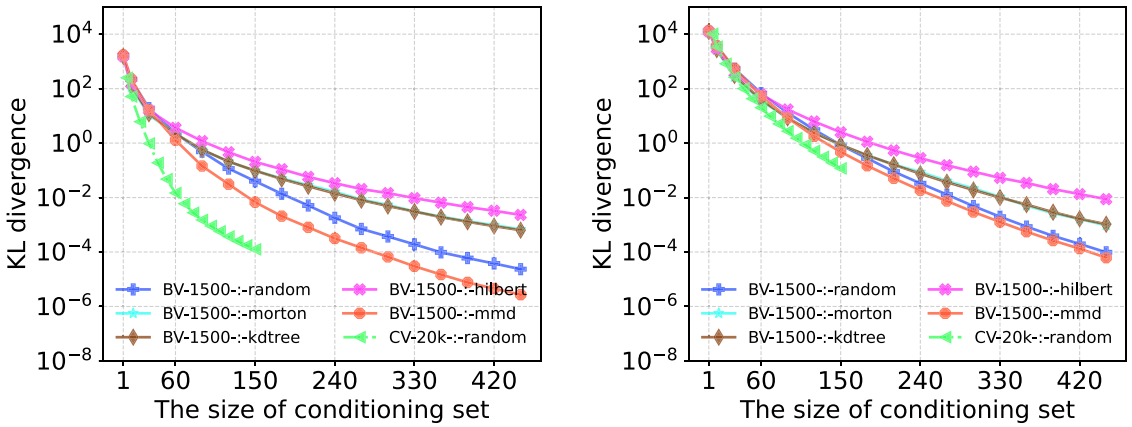
We investigate the impact of permutation and block count of the block Vecchia algorithm. As illustrated in Figure 5(a), the permutation plays an essential role in enhancing the accuracy of the block Vecchia algorithm, where the maxmin permutation (mmd in figures) achieves the highest accuracy, random permutation yields near-optimal accuracy, while other permutations fail to produce promising results. In Figure 5(b), we set random reordering as the default, recognizing that it achieves near-optimal accuracy while demanding fewer computational resources than the maxmin reordering algorithm (Guinness 2018). The KL divergence is plotted against different block counts, consistently confirming that a larger block count leads to more accurate results. This enhancement can be attributed to two factors: first, a smaller block size (or a larger block count) ensures that the neighbors are representative of the points within the block; second, random ordering potentially yields better neighbor candidates for blocks. Figure 5 only presents the case of $\beta = 0.052537$ and $\nu = 1.5$. Refer to supplementary S3 for more parameter settings.

In our analysis of two approximation methodologies, the classic Vecchia exhibits a pronounced decrease in accuracy as smoothness increases. At the same time, the block Vecchia demonstrates considerable robustness, maintaining accuracy



(a) $bc = 2500$ (b) Random reodering

Figure 5. KL divergence and conditioning size along with increasing block count and different permutations under $\beta = 0.052537$, $\nu = 1.5$ and $\log_{10}$ scale.



(a) $\beta = 0.026270, \nu = 0.5$ (b) $\beta = 0.014290, \nu = 2.5$

Figure 6. KL divergence and conditioning size under 20K locations with $\log_{10}$ scale under the low effective range.

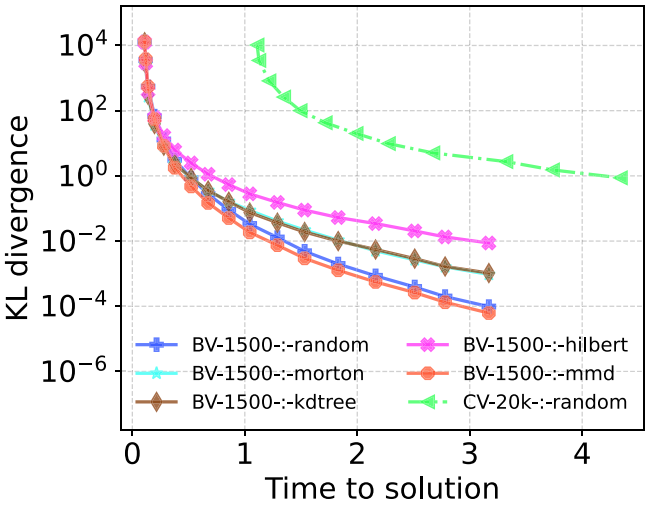


Figure 7. KL divergence and time-to-solution (second) under 20K locations with $\log_{10}$ scale under $\nu = 2.5$, $\beta = 0.014290$.

across a range of smoothness parameters. For the evaluation of the block Vecchia approximation capabilities concerning smoothness parameters, we focus on the case where $bc = 1500$, identified as both relatively efficient and accurate. Figure 6 provides quantitative evidence of this phenomenon: for the *CV-20K-150-random* configuration, KL divergence escalates from 10^{-4} to 10^{-1}, indicating a substantial loss in approximation precision. Conversely, the *BV-1500-450-random* configuration shows only a slight decrease in KL divergence, from $10^{-4.6}$ to 10^{-4}, highlighting the effectiveness of the BV method to manage the complexities associated with high smoothness levels effectively. Furthermore, as ν increases, the gap between the block Vecchia and classic Vecchia methods diminishes. Refer to Figure S3 in the supplementary material S3 for more parameter settings.

The block Vecchia algorithm outperforms the classic Vecchia algorithm in both computational efficiency and accuracy. To evaluate the efficiency and accuracy of the block Vecchia, we focus on the case at $bc = 1500$, which is the same as above. In Figure 7, the time-to-solution is defined as the cumulative duration required to compute the log-likelihood for a single iteration in MLE. This includes the time for matrix generation and batched BLAS operations while explicitly excluding the time spent on nearest neighbor searching and K-means clustering. These latter operations are omitted from consideration due to their one-time execution at the outset of MLE and their minimal impact on the overall computation time across numerous itera-

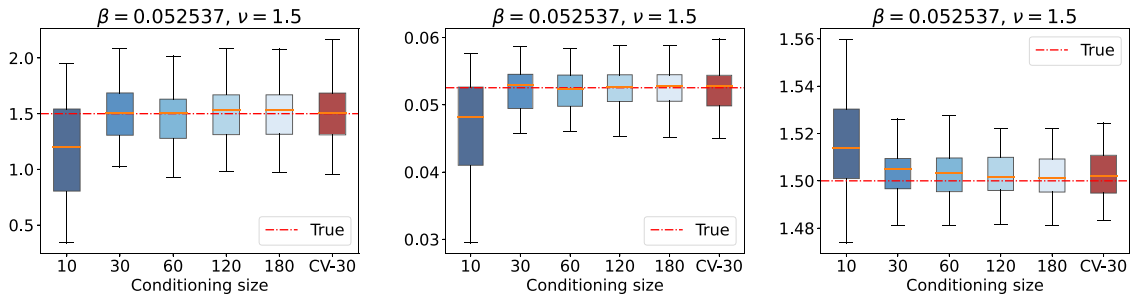


Figure 8. Simulations of $n = 20{,}000$ (50 samples) on parameter estimation at $\nu = 1.5$ and middle effective ranges. The number of the block is set as 1500 for every block Vecchia settings.

tions. The *BV-1500-:-random* achieves equivalent levels of accuracy approximately 5X faster than its classic counterpart, *CV-20K-:-random*. Additionally, it offers a more precise KL approximation. Notably, the block Vecchia provides a more accurate approximation within the same time frame compared to the classic Vecchia, *CV-20K-:-random*. Refer to Figure S4 in the supplementary material S4 for more parameter settings. In addition, we provide the accuracy of block Vecchia approximation along with the increasing number of locations in the supplementary material S5.

3.2. Simulations for Parameter Estimation and Prediction

In this section, we assess the accuracy of statistical parameter estimation and the prediction uncertainty of the block Vecchia. Using the *ExaGeoStat* framework (Abdulah et al. 2018a), a high-performance unified framework for computational geostatistics on many-core systems, we generate 50 datasets on irregular 2D spatial locations within the unit square $[0, 1] \times [0, 1]$ to investigate the parameter uncertainty numerically, which has been studied for exact GP asymptotically (Wang et al. 2023). These datasets are based on Gaussian random fields with a problem size of $n = 20{,}000$ observations and the Matérn covariance function, which adheres to the parameter configuration outlined in Table 1. We estimate the parameters set (σ^2, β, ν) using the BOBYQA optimization algorithm (Powell 2009), which is a gradient-free method and has great power to search the global optimal value. For our approximation methodologies, we select the classic Vecchia approach with a conditioning size of 30 as our baseline, following the recommendation made by (Guinness 2018; Pan et al. 2024). This conditioning size is also chosen based on its demonstrated capability to provide comparable approximation accuracy in terms of KL divergence, as evidenced in Figure 6 for the classic Vecchia algorithm. In contrast, for the block Vecchia approximation, we set the block count, $bc = 1500$, identified as offering a balance between accuracy and computational efficiency relative to the classic approach, with varying conditioning sizes of $(10, 30, 60, 120, 180)$ explored.

Following the parameter estimation, we evaluate the predictive performance of the block Vecchia method. The experimental setup involves the simulated dataset of 20,000 points used in the parameter estimation, where 90% of the data is used for training and 10% for testing. The number of blocks in the block Vecchia method is set to 200. Using the true parameter values

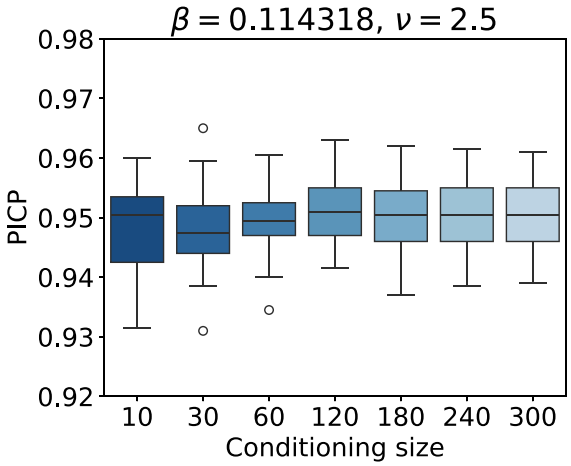


Figure 9. PICP for the simulated datasets.

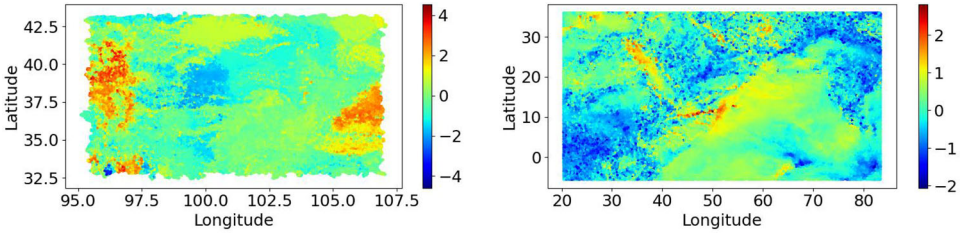
from the model in Table 1, we conduct 1000 rounds of conditional simulation and calculate the 95% confidence interval. Then, we report the Prediction Interval Coverage Probability (PICP) based on these simulations (Zhao et al. 2008; Nag, Sun, and Reich 2023):

$$PICP = \frac{1}{n^*} \sum_{i=1}^{n^*} \mathbf{1}_{\hat{y}_i \in [L_i, U_i]}, \tag{9}$$

where $\hat{y}_i$ is the predicted value at $\boldsymbol{s}_i$, n^* is the total number of new locations, L_i is the lower and U_i is the upper prediction bounds at $\boldsymbol{s}_i$.

Results of parameter estimation about the case of $\nu = 1.5$ and the middle effective range are depicted in Figure 8, and results of PICP are shown in Figure 9. The additional outcomes sharing the same trend are detailed in the supplementary materials S6.

As illustrated in Figure 8, there is a clear trend toward convergence of the median values of all estimators to the true parameter values with the increase in the number of nearest neighbors. Concurrently, an observable reduction in the variance of all estimators occurs with the increased neighbor count. In addition, when configured with 30 nearest neighbors, the block Vecchia algorithm attains parameter estimation accuracy comparable to that of the classic Vecchia method with 30 nearest neighbors. Notably, this equivalence in estimation accuracy is achieved with a significant increase in computational efficiency, approximately a 14X speedup. Figure 9 assesses the predictive performance of



(a) Residual of Observations (soil) (b) Residual of Observations (wind)

Figure 10. Real datasets residuals: soil moisture and wind speed.

the block Vecchia approximation. The ratio of the block count and the number of predicted locations is 1:10, representing a 10X speedup. As we increase the conditioning size, the PICP becomes higher and its variance becomes smaller, other parameter setups have a similar trend; see the supplementary material S6.

3.3. Block Vecchia Performance on Large-Scale Real Datasets

We further compare our block Vecchia algorithm with the exact GP on two subsampled real datasets: a soil moisture dataset from the Mississippi River Basin region and a wind speed dataset from the Middle East region (Pan et al. 2024). The soil moisture data cover the Mississippi River basin in the United States on January 1, 2004, as reported in (Chaney, Metcalfe, and Wood 2016). The dataset, previously used in (Huang and Sun 2018; Abdulah et al. 2018a), involves Gaussian field modeling and contains 2M irregularly distributed locations. To mitigate computational expenses, we randomly selected 250K locations for the training dataset and 25K for the testing dataset. This subsampling enables us to compare the estimated parameters obtained through the Vecchia approximation with those from the exact modeling at an affordable expense, considering that employing all 2 million locations would pose significant computational burdens (Pan et al. 2024). The residuals, shown in Figure 10(a), are fitted using a zero-mean Gaussian process model, which incorporates a Matérn covariance function as in (8). For the block Vecchia methods, using random ordering, seven different conditioning sizes (10, 30, 60, 90, 120, 180, 210) across four block counts (1K, 5K, 15K, 25K) are considered. In the Vecchia approximation and *ExaGeostat*, BOBYQA (Powell 2009) is adopted as the optimization algorithm with the same configuration. Besides, we use *ExaGeostat* (Abdulah et al. 2018a) to estimate the parameters for the exact Gaussian process. Finally, the estimated parameters are used to perform exact spatial prediction, known as kriging, and the Mean Square Prediction Error (MSPE) is calculated (Abdulah et al. 2018a).

The second real dataset consists of a 1M wind speed location with three components: Zonal (U) and Meridional (V) wind, covering the Arabian Peninsula in the Middle East. It was generated using the WRF-ARW (Weather Research and Forecasting - Advanced Research WRF) model (Powers et al. 2008). This dataset is used in our study in both 2D and 3D analyses, where wind speed is computed based on U and V for the 2D dataset (longitude, latitude) and the 3D dataset (longitude, latitude, and pressure level). The WRF-ARW model has a 5 km horizontal grid spacing, spanning 51 vertical levels, with the highest level at 10 hPa. This dataset spans 37 years, providing daily data. Each file records 24 hr of hourly wind speed measurements across 17 atmospheric layers. This study focuses on the dataset on September 1, 2017, starting at 00:00 a.m. Our interest is in wind speed measurements at a height of 10 meters above the ground, corresponding to layer 0 for 2D analysis and all layers for 3D analysis. Distance calculations in the wind speed dataset match those in the soil moisture dataset (Pan et al. 2024). The residuals are modeled in the same way as the previous dataset.

Figure 11 shows the estimated parameters for both datasets, while Figure 12 highlights the MSPE associated with the prediction. We found that the parameter vector $\boldsymbol{\theta}$, as estimated through the block Vecchia approximation, closely aligns with that obtained via *ExaGeoStat* (exact MLE), particularly as the number of conditioning neighbors increases. Figure 11 illustrates that, for both datasets, a conditioning size of 60 with 25,000 block counts is optimal for achieving an estimation close to the exact MLE. Figure 12 further demonstrates that the block Vecchia approximation achieves a prediction error remarkably close to the actual values when predicting missing data.

4. Application to 3D Wind Speed Profiles

Understanding 3D wind speed profiles is vital across multiple disciplines due to their significant impact on various environmental and human activities. For example, in meteorological forecasting, accurate 3D wind profiles are essential for initializing and running numerical weather prediction models. Winds at different altitudes influence the development and movement of weather systems, such as cyclones and anticyclones. Upper-level winds, like the jet stream, steer weather systems and affect their intensity (Kalnay 2003); winds transport heat and moisture vertically and horizontally, affecting temperature distributions and humidity levels. This transport is crucial for predicting phenomena like heatwaves, cold fronts, and precipitation patterns (Holton and Hakim 2012). In aviation safety and efficiency, turbulence, often caused by wind shear and atmospheric instability, poses safety risks and discomfort. Detailed wind profiles enable pilots and flight dispatchers to anticipate and avoid turbulent areas, enhancing passenger safety (Storer, Williams, and Gill 2019). In renewable energy, the efficiency and feasibility of wind turbines depend on wind speeds at different heights. 3D wind profiles help in selecting optimal turbine hub heights and in designing turbines that maximize energy capture (Manwell et al. 2010); detailed vertical wind data allow for precise estima-

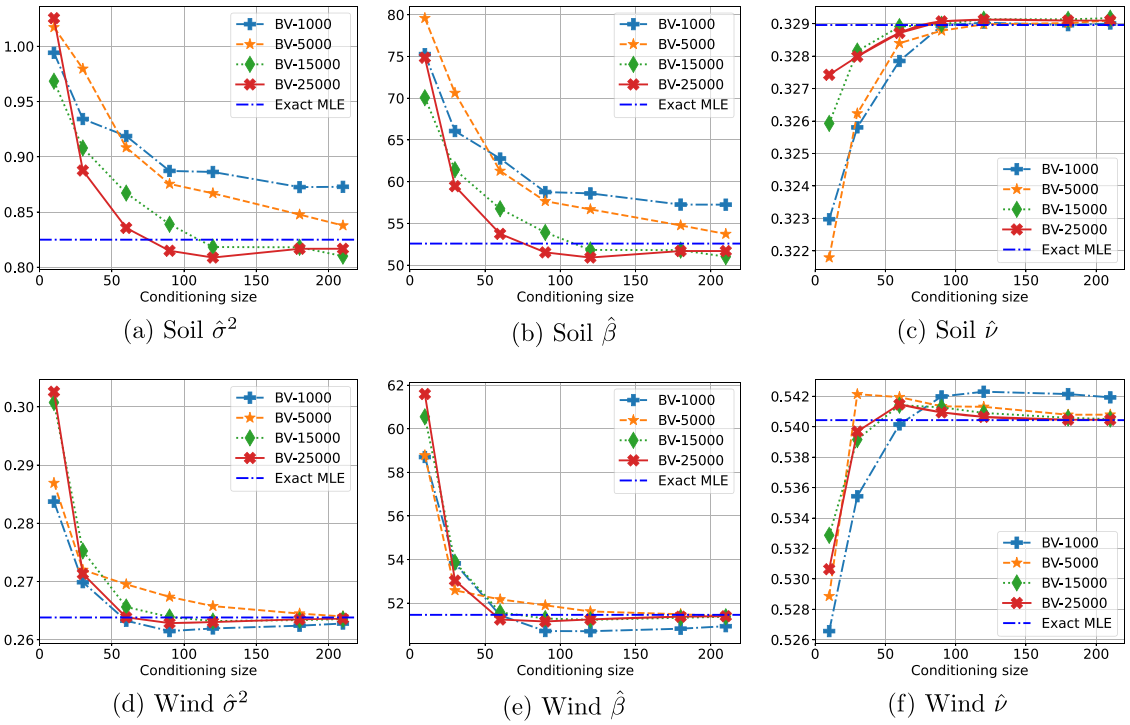


Figure 11. The estimated parameters using block Vecchia with different block counts compared to *ExaGeoStat* (exact MLE). The first row is the parameter vector for soil moisture, and the second for wind speed.

tion of potential energy yields, where investors and engineers use this information for planning and development (Burton et al. 2011). In climate modeling, high-resolution 3D wind data enhance climate models' ability to project future climate scenarios, including temperature and precipitation changes. This information is vital for developing mitigation and adaptation strategies (Masson-Delmotte et al. 2021).

Traditional GP models are computationally infeasible for such large datasets, particularly when handling the vertical dimension in 3D profiles. However, by breaking down the computations into smaller blocks, the block Vecchia approximation enables the processing of large-scale 3D wind profiles, capturing fine vertical and horizontal wind patterns with much greater efficiency and scalability. This scalability is crucial for analyzing vast datasets across multiple altitudes and spatial regions, which were previously inaccessible due to computational constraints. In this section, we focus on modeling high-resolution 3D wind speed profiles. The scalability of the block Vecchia method is evaluated on a single GPU (NVIDIA V100 with 32 GB memory) with problem sizes at the million level. The 17 atmospheric layers of the 3D profile are included as an additional input dimension in the 3D profile wind speed data, for example, Figure S10 in the supplementary material S7 illustrates the residuals, and the dataset description is detailed in Section 3.3. The wind speed measurements across the 17 layers span 37 million irregular locations, posing a significant computational challenge for modern computers. We employ a subsampling technique to address this, randomly select 1 million locations and scale the coordinates to $(x, y, z) \in [0, 1]^3$. The rest of the experimental configuration remains consistent with the previous section, and we then apply the block Vecchia approximation to estimate the parameters.

Figure 13 presents the results of parameter estimation for modeling the residuals of 3D wind speed in a million-level context and the MSPE. The number of blocks increases (e.g., 10K, 50K, and 100K), reflecting our interest in improving accuracy in the approximation of the block Vecchia method. The findings indicate that: (a) parameter estimation gradually converges to a specific value as the conditioning size increases; (b) the convergence rate improves with a larger block count. The experiment encourages extending the Vecchia approximation to larger-scale problems. These results demonstrate that larger conditioning sizes and block counts enhance the accuracy of parameter estimation. More importantly, the block Vecchia method can handle much larger problem sizes than the classic Vecchia, facilitating large-scale modeling. We also assessed the predictive ability of the block Vecchia method. We plug in the converged estimated parameters $\hat{\theta}$ into the block Vecchia approximation, then conduct 1000 rounds of the conditional simulations, and finally report the MSPE and the standard deviation. Specifically, Figure 13(d) illustrates the MSPE of the block Vecchia approximation for the residuals of 3D wind speed. We have used different block counts (BV-5000, BV-10000, BV-20000, BV-30000) and varied the conditioning size for prediction in the approximated GP framework, where the block Vecchia method is applied for scalable approximations. It is observed that (a) increasing the conditioning size improves the prediction accuracy, and using a larger block count (10,000–30,000) yields more stable results even at smaller conditioning sizes; (b) the gains in prediction accuracy increases at a certain level of conditioning size for all types of block counts. In addition, we provide a full dataset application on soil moisture with 2M points in the supplementary material S7.

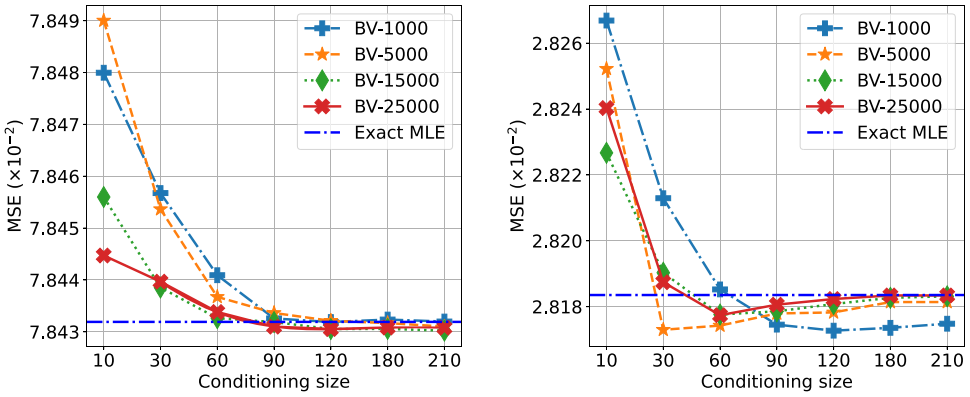


Figure 12. The MSPE of block Vecchia with different block counts compared to *ExaGeoStat* (exact MLE). The first is the MSPE for soil moisture, and the second for wind speed.

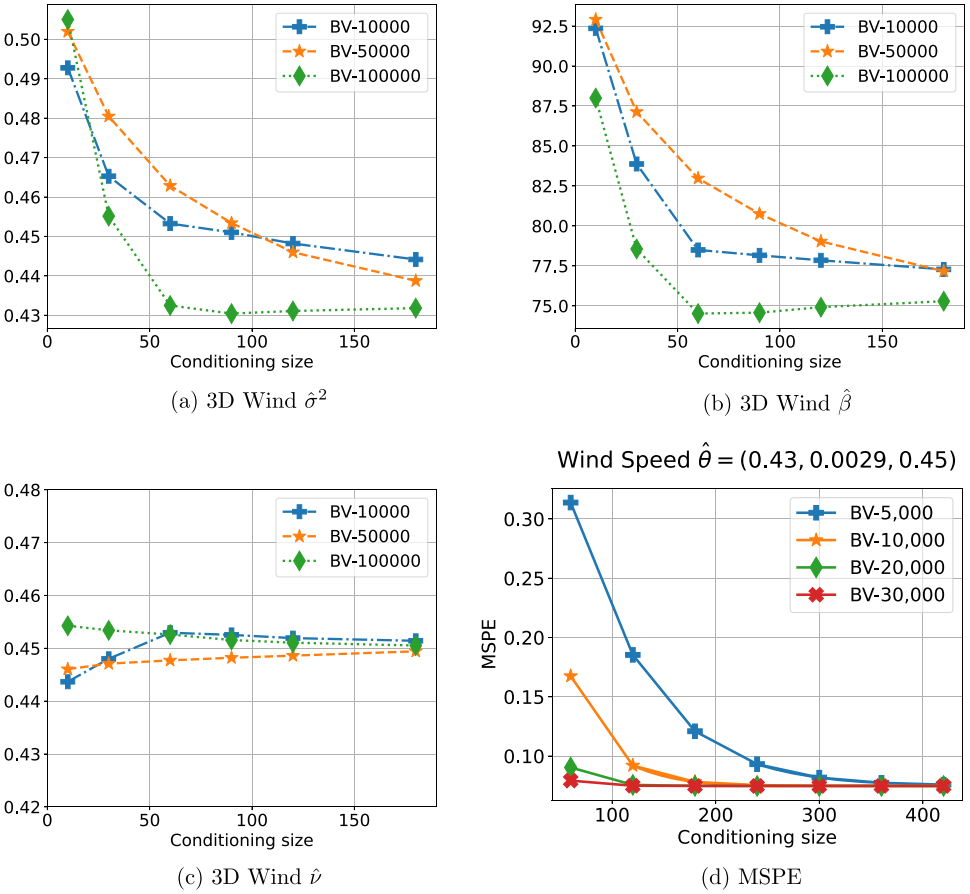


Figure 13. The estimated parameters using block Vecchia with different block counts (the range parameters in (b) are scaled); and the MSPE of block Vecchia with different block counts for the residuals of 3D wind speed in (d).

5. Conclusion

This study introduces the block Vecchia algorithm and its GPU framework based on batched operations provided by the MAGMA linear algebra library. The algorithm evaluates in batch, simultaneously processing multiple operations or data points, and simultaneously computes the multivariate conditional likelihood of all location blocks to improve efficiency, reduce storage requirements, and enhance scalability in large-scale scenarios. Our numerical study and real datasets analysis provide a deep insight into the block Vecchia algorithm: (a) The analysis reveals that a larger conditioning size and block count can improve modeling accuracy compared to the classic Vecchia algorithm, ensuring efficiency and precision; (b) The sequence of blocks plays a critical role in the accuracy of the approximation, that is, the random ordering markedly enhances the approximation as the number of blocks increases; (c) The block Vecchia method demonstrates an approximately 80X speedup compared to the classic Vecchia algorithm without compromising the accuracy of the approximations. This significant enhancement in computational speed, coupled

with improved accuracy, represents a substantial advancement in applying Gaussian process models; and (d) scalability: The block Vecchia method allows handling problem sizes 40X larger than those accommodated by the classic Vecchia algorithm. This scalability enables the algorithm to leverage existing GPUs effectively, making it a powerful tool for addressing large-scale statistical modeling challenges. In a recent study (Hazra et al. 2024), the Vecchia approximation (represented by the GpGp package (Guinness, Katzfuss, and Fahmy 2021)) was shown to outperform many popular methods for fitting Gaussian Processes across five large spatial datasets under various parameter settings. This confirms the superiority of the classic Vecchia approach. Accordingly, we applied our proposed block Vecchia algorithm to all datasets from this study, evaluating its performance against the results reported in the study and the local approximation Gaussian Process (laGP, Gramacy 2016). Our results, presented in Table S1 in the supplementary material 10, demonstrate that the block Vecchia algorithm is as accurate as GpGp/GpGp0, while our GPU implementation achieves an average of 1.5X speedup in modeling the five given datasets. According to the asymptotic properties of Vecchia approximation (Zhang, Tang, and Banerjee 2021; Kang et al. 2024), the asymptotic properties of the MLEs from the block Vecchia method will be investigated in our future research. Besides, considering the promising results of the classic Vecchia approximation on high-dimension problems (Katzfuss, Guinness, and Lawrence 2022; Jimenez and Katzfuss 2023), it is worth studying computer experiments with block Vecchia.

Additionally, we apply the block Vecchia method to large-scale real datasets, 3D wind speed profiles, consisting of millions of data points not addressed in previous studies using other GP methods. In parameter estimation and prediction, the block Vecchia method demonstrates an efficient and accurate approach, with parameter convergence to consistent values and high prediction accuracy across different block counts. The results further show that the block Vecchia approximation, which decomposes the likelihood function into smaller, manageable conditional distributions for efficient parallel processing on modern GPUs, facilitates the use of high-resolution 3D geospatial data in applications such as agriculture, urban planning, and environmental monitoring.

Supplementary Materials

The supplementary material consists of two components, (a) algorithm description and experimental results. In Section S1, we detail the estimation and prediction steps of the proposed batched implementation of the block Vecchia algorithm. Sections S2, S3, S4, and S5 present additional experimental results, examining the impact of key factors on the proposed block Vecchia algorithm, such as increasing block count, varying smoothness levels, accuracy versus computational time, and the number of locations. Section S6 includes further simulation experiments, exploring additional settings for smoothness and range parameters, along with prediction simulations. Section S7 provides a visualization of residuals from the 3D wind-speed dataset and an extended analysis of a large-scale 2D soil moisture dataset containing 2 million points. Sections S8 and S9 present smaller-scale KL divergence results, an easier reproduction in this article. Section S10 includes more results on large-scale spatial competition, reinforcing the conclusions in the main article. (b) Code Supplement. The code archive provides details on the software dependencies and installation instructions. Additionally, it outlines the computational time required to reproduce each figure in the article.

Disclosure Statement

No potential conflict of interest was reported by the author(s).

Funding

This research was supported by King Abdullah University of Science and Technology (KAUST). We gratefully acknowledge the funding and resources provided by KAUST, which made this work possible and has facilitated the research presented in this study.

References

Abdulah, S., Cao, Q., Pei, Y., Bosilca, G., Dongarra, J., Genton, M. G., Keyes, D. E., Ltaief, H., and Sun, Y. (2021), "Accelerating Geostatistical Modeling and Prediction with Mixed-Precision Computations: A High-Productivity Approach with PaRSEC," *IEEE Transactions on Parallel and Distributed Systems*, 33, 964–976. [547]

Abdulah, S., Ltaief, H., Sun, Y., Genton, M. G., and Keyes, D. E. (2018a), "ExaGeoStat: A High Performance Unified Software for Geostatistics on Manycore Systems," *IEEE Transactions on Parallel and Distributed Systems*, 29, 12, 2771–2784. [550,553,554]

——— (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. [547]

——— (2019), "Geostatistical Modeling and Prediction Using Mixed Precision Tile Cholesky Factorization," in *2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, pp. 152–162, IEEE. [547]

Bentley, J. L. (1975), "Multidimensional Binary Search Trees Used for Associative Searching," *Communications of the ACM*, 18, 509–517. [548]

Bevilacqua, M., Fassò, A., Gaetan, C., Porcu, E., and Velandia, D. (2016), "Covariance Tapering for Multivariate Gaussian Random Fields Estimation," *Statistical Methods & Applications*, 25, 21–37. [546]

Burton, T., Jenkins, N., Sharpe, D., and Bossanyi, E. (2011), *Wind Energy Handbook*, Hoboken, NJ: Wiley. [555]

Cao, Q., Abdulah, S., Alomairy, R., Pei, Y., Nag, P., Bosilca, G., Dongarra, J., Genton, M. G., Keyes, D. E., Ltaief, H., et al. (2022), "Reshaping Geostatistical Modeling and Prediction for Extreme-Scale Environmental Applications," in *2022 SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 13–24, IEEE Computer Society. [547]

Chaney, N. W., Metcalfe, P., and Wood, E. F. (2016), "HydroBlocks: A Field-Scale Resolving Land Surface Model for Application Over Continental Extents," *Hydrological Processes*, 30, 3543–3559. [554]

Chen, J., Yu, L., and Wang, W. (2022), "Hilbert Space Filling Curve based Scan-Order for Point Cloud Attribute Compression," *IEEE Transactions on Image Processing*, 31, 4609–4621. [548]

Chen, S., Abdulah, S., Sun, Y., and Genton, M. G. (2024), "On the Impact of Spatial Covariance Matrix Ordering on Tile Low-Rank Estimation of Matérn Parameters," *Environmetrics*, 35, e2868. [548]

Dong, T., Haidar, A., Luszczek, P., Tomov, S., Abdelfattah, A., and Dongarra, J. (2016), "MAGMA Batched: A Batched BLAS Approach for Small Matrix Factorizations and Applications on GPUs," Technical Report. [547,549]

Furrer, R., Genton, M. G., and Nychka, D. (2006), "Covariance Tapering for Interpolation of Large Spatial Datasets," *Journal of Computational and Graphical Statistics*, 15, 502–523. [546]

Gneiting, T. (2002), "Nonseparable, Stationary Covariance Functions for Space–Time Data," *Journal of the American Statistical Association*, 97, 590–600. [551]

Gramacy, R. B. (2016), "laGP: Large-Scale Spatial Modeling via Local Approximate Gaussian Processes in R," *Journal of Statistical Software*, 72, 1–46. [557]

Guinness, J. (2018), "Permutation and Grouping Methods for Sharpening Gaussian Process Approximations," *Technometrics*, 60, 415–429. [547,548,551,553]

——— (2021), "Gaussian Process Learning via Fisher Scoring of Vecchia's Approximation," *Statistics and Computing*, 31, 25. [547]

Guinness, J., Katzfuss, M., and Fahmy, Y. (2021), "GpGp: Fast Gaussian Process Computation Using Vecchia's Approximation," *R package version 0.4. 0*. [557]

Haidar, A., Dong, T., Luszczek, P., Tomov, S., and Dongarra, J. (2015), "Batched Matrix Computations on Hardware Accelerators based on GPUs," *The International Journal of High Performance Computing Applications*, 29, 193–208. [550]

Hazra, A., Nag, P., Yadav, R., and Sun, Y. (2024), "Exploring the Efficacy of Statistical and Deep Learning Methods for Large Spatial Datasets: A Case Study," *Journal of Agricultural, Biological and Environmental Statistics*, 30, 231–254. [557]

Hilbert, D. (1935), "Über die stetige Abbildung einer Linie auf ein Flächenstück," *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes: Nebst Einer Lebensgeschichte*, pp. 1–2. [548]

Holton, J. R., and Hakim, G. J. (2012), *An Introduction to Dynamic Meteorology*, New York: Academic Press. [554]

Huang, H., Abdulah, S., Sun, Y., Ltaief, H., Keyes, D. E., and Genton, M. G. (2021), "Competition on Spatial Statistics for Large Datasets," *Journal of Agricultural, Biological and Environmental Statistics*, 26, 580–595. [551]

Huang, H., and Sun, Y. (2018), "Hierarchical Low Rank Approximation of Likelihoods for Large Spatial Datasets," *Journal of Computational and Graphical Statistics*, 27, 110–118. [546,547,554]

Jimenez, F., and Katzfuss, M. (2023), "Scalable Bayesian Optimization Using Vecchia Approximations of Gaussian Processes," in *International Conference on Artificial Intelligence and Statistics*, pp. 1492–1512, PMLR. [547,557]

Kalnay, E. (2003), *Atmospheric Modeling, Data Assimilation and Predictability*, Cambridge: Cambridge University Press. [554]

Kang, M., Schäfer, F., Guinness, J., and Katzfuss, M. (2024), "Asymptotic Properties of Vecchia Approximation for Gaussian Processes," arXiv preprint arXiv:2401.15813. [557]

Katzfuss, M., and Cressie, N. (2011), "Spatio-Temporal Smoothing and EM Estimation for Massive Remote-Sensing Data Sets," *Journal of Time Series Analysis*, 32, 430–446. [547]

Katzfuss, M., and Guinness, J. (2021), "A General Framework for Vecchia Approximations of Gaussian Processes," *Statistical Science*, 36, 124–141. [547]

Katzfuss, M., Guinness, J., and Lawrence, E. (2022), "Scaled Vecchia Approximation for Fast Computer-Model Emulation," *SIAM/ASA Journal on Uncertainty Quantification*, 10, 537–554. [547,557]

Kaufman, C. G., Schervish, M. J., and Nychka, D. W. (2008), "Covariance Tapering for Likelihood-based Estimation in Large Spatial Data Sets," *Journal of the American Statistical Association*, 103, 1545–1555. [546]

Manwell, J. F., McGowan, J. G., and Rogers, A. L. (2010), *Wind Energy Explained: Theory, Design and Application*, Chichester: Wiley. [554]

Masson-Delmotte, V., Zhai, P., Pirani, A., Connors, S. L., Péan, C., Berger, S., Caud, N., Chen, Y., Goldfarb, L., Gomis, M. I., et al. (2021), *Climate Change 2021: The Physical Science Basis* (Vol. 2), Cambridge, UK: Cambridge University Press. [555]

Mondal, S., Abdulah, S., Ltaief, H., Sun, Y., Genton, M. G., and 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. [547]

——— (2023), "Tile Low-Rank Approximations of Non-Gaussian Space and Space-Time Tukey g-and-h Random Field Likelihoods and Predictions on Large-Scale Systems," *Journal of Parallel and Distributed Computing*, 180, 104715. [546]

Nag, P., Sun, Y., and Reich, B. J. (2023), "Bivariate DeepKriging for Large-Scale Spatial Interpolation of Wind Fields," arXiv preprint arXiv:2307.08038. [553]

Pan, Q., Abdulah, S., Genton, M. G., Keyes, D. E., Ltaief, H., and Sun, Y. (2024), "GPU-Accelerated Vecchia Approximations of Gaussian Processes for Geospatial Data using Batched Matrix Computations," in *ISC High Performance 2024 Research Paper Proceedings (39th International Conference)*, 1–12. Prometeus GmbH. [547,550,551,553,554]

Powell, M. J. D. (2009), "The BOBYQA Algorithm for Bound Constrained Optimization Without Derivatives," *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, 26. [553,554]

Powers, J. G., Huang, X.-Y., Klemp, B., Skamarock, C., Dudhia, J., and Gill, O. (2008), "A Description of the Advanced Research WRF Version 2," *NCAR tech*, 15. [554]

Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O. P., Tiwari, A., Er, M. J., Ding, W., and Lin, C.-T. (2017), "A Review of Clustering Techniques and Developments," *Neurocomputing*, 267, 664–681. [548]

Storer, L. N., Williams, P. D., and Gill, P. G. (2019), "Aviation Turbulence: Dynamics, Forecasting, and Response to Climate Change," *Pure and Applied Geophysics*, 176, 2081–2095. [554]

Vecchia, A. V. (1988), "Estimation and Model Identification for Continuous Spatial Processes," *Journal of the Royal Statistical Society*, Series B, 50, 297–312. [547]

Walker, D. W. (2018), "Morton Ordering of 2D Arrays for Efficient Access to Hierarchical Memory," *The International Journal of High Performance Computing Applications*, 32, 189–203. [548]

Wang, K., Abdulah, S., Sun, Y., and Genton, M. G. (2023), "Which Parameterization of the Matérn Covariance Function?," *Spatial Statistics*, 58, 100787. [553]

Xu, R., and Wunsch, D. (2005), "Survey of Clustering Algorithms," *IEEE Transactions on Neural Networks*, 16, 645–678. [548]

Zhang, J., and Katzfuss, M. (2022), "Multi-Scale Vecchia Approximations of Gaussian Processes," *Journal of Agricultural, Biological and Environmental Statistics*, 27, 440–460. [547]

Zhang, L., Tang, W., and Banerjee, S. (2021), "Fixed-Domain Asymptotics Under Vecchia's Approximation of Spatial Process Likelihoods," arXiv preprint arXiv:2101.08861. [547,557]

Zhao, J. H., Dong, Z. Y., Xu, Z., and Wong, K. P. (2008), "A Statistical Approach for Interval Forecasting of the Electricity Price," *IEEE Transactions on Power Systems*, 23, 267–276. [553]