2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS) | 979-8-3503-8711-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/IPDPS57955.2024.00031

Parallel Approximations for High-Dimensional Multivariate Normal Probability Computation in Confidence Region Detection Applications

Xiran Zhang[1,2], Sameh Abdulah[1,3], Jian Cao[4], Hatem Ltaief[1,3], Ying Sun[1,2,3], Marc G. Genton[1,2,3], and David E. Keyes[1,3]

[1]*Computer, Electrical, and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology,*
[2]*Statistics Program, King Abdullah University of Science and Technology,*
[3]*Extreme Computing Research Center, King Abdullah University of Science and Technology,*
Thuwal 23955-6900, Saudi Arabia.
[4]*Department of Mathematics, The University of Houston, Houston, Texas, USA.*

***Abstract*—Addressing the statistical challenge of computing the multivariate normal (MVN) probability in high dimensions holds significant potential for enhancing various applications. For example, the critical task of detecting confidence regions where a process probability surpasses a specific threshold is essential in diverse applications, such as pinpointing tumor locations in magnetic resonance imaging (MRI) scan images, determining hydraulic parameters in groundwater flow issues, and forecasting regional wind power to optimize wind turbine placement, among numerous others. One common way to compute high-dimensional MVN probabilities is the Separation-of-Variables (SOV) algorithm. This algorithm is known for its high computational complexity of $O(n^3)$ and space complexity of $O(n^2)$, mainly due to a Cholesky factorization operation for an $n \times n$ covariance matrix, where n represents the dimensionality of the MVN problem. This work proposes a high-performance computing framework that allows scaling the SOV algorithm and, subsequently, the confidence region detection algorithm. The framework leverages parallel linear algebra algorithms with a task-based programming model to achieve performance scalability in computing process probabilities, especially on large-scale systems. In addition, we enhance our implementation by incorporating Tile Low-Rank (TLR) approximation techniques to reduce algorithmic complexity without compromising the necessary accuracy. To evaluate the performance and accuracy of our framework, we conduct assessments using simulated data and a wind speed dataset. Our proposed implementation effectively handles high-dimensional multivariate normal (MVN) probability computations on shared and distributed-memory systems using finite precision arithmetics and TLR approximation computation. Performance results show a significant speedup of up to 20X in solving the MVN problem using TLR approximation compared to the reference dense solution without sacrificing the application's accuracy. The qualitative results on synthetic and real datasets demonstrate how we maintain high accuracy in detecting confidence regions even when relying on TLR approximation to perform the underlying linear algebra operations.**

***Index Terms*—Cholesky factorization, Confidence region detection, Excursion Set, Multivariate normal probability, Separation-of-Variables algorithm, Tile low-rank.**

I. Introduction

The multivariate normal distribution extends the concept of the univariate normal distribution to encompass higher dimensions. In the context of probability, it characterizes the distribution of a random vector featuring multiple components, each potentially correlated with the others. The applications of multivariate normal probability span diverse fields, such as its use in spatial statistics for climate modeling [1] and confidence region detection [2], its application in machine learning through Gaussian Mixture Models (GMMs) [3] and Principal Component Analysis (PCA) [4], and its role in economic studies, particularly in macroeconomic modeling [5].

Confidence region detection is a well-established problem primarily focused on identifying spatial areas where a given process exceeds a certain threshold with a given probability. For example, in applications related to air pollution, identifying areas where pollution levels exceed a specific value, posing a potential risk to human health, and ensuring accurate assessment and management of environmental risks [6]. Another example involves brain imaging applications that pinpoint specific brain regions exhibiting particular characteristics or responses [7]. The usage of confidence region detection can be extended to applications in spatial statistics, astrophysics [8], machine learning [9], environmental science [10], and many others. Mathematically, given a latent stochastic field $X(\boldsymbol{s})$ at spatial location $\boldsymbol{s}$ and a set of observations denoted as $\mathbf{y}$, the goal is to identify a region, denoted as D, where the condition $X(\boldsymbol{s}) > u$ holds true for $\boldsymbol{s}$ within D with a probability of at least $1 - \alpha$. Here, u is the threshold value, and $1 - \alpha$ is the confidence level.

A common strategy for addressing this statistical challenge involves computing the multivariate normal (MVN) probability, which involves solving an integration problem in high dimensions. A Monte Carlo (MC) method can be used to solve this integration problem by simulating a large number of random samples and averaging the integral function over

1530-2075/24/$31.00 ©2024 IEEE
DOI 10.1109/IPDPS57955.2024.00031

these samples [11]. Nevertheless, using this MC method in high dimensions is practically prohibitive when accuracy is essential [2]. In the literature, the Separation-of-Variables (SOV) algorithm has been used to transform the integration problem into a solvable format by transforming the integration region to a unit hypercube. Then, a quasi-MC method can be used to evaluate the new integration problem [2], [12]. The SOV algorithm is known for its computational complexity of $O(n^3)$ and space complexity of $O(n^2)$, with n representing the dimensionality of the MVN problem. The high complexity arises due to the need for solving a Cholesky factorization problem for a given covariance matrix $\mathbf{\Sigma}$. With the substantial increase in the volume of spatial data originating from various sources, it becomes imperative to explore other approaches for solving the SOV problem in large dimensions.

The availability of efficient algorithms for performing linear algebra operations on parallel architectures has enabled significant improvements in existing algorithms, making the handling of substantial scientific data across diverse domains feasible. Parallel tile-based linear solvers, finely tuned for modern hardware architectures, play a pivotal role in optimizing the execution of many applications, addressing previously unsolved problems. Notable parallel linear algebra library examples include Chameleon [13], DPLASMA [14], and HiCMA [15], [16]. The two formers support dense linear algebra matrix operations on shared and distributed-memory systems. The latter integrates support for Tile Low-Rank (TLR) matrix approximations to deal with large-scale scientific problems. Furthermore, leveraging dynamic runtime systems within these libraries contributes to the meticulous tuning of their execution on modern architectures, enabling higher rates of floating-point operations per second (flops) and efficient time-to-solution.

This study proposes a parallelized version of the SOV algorithm, effectively calculating the MVN probability in high-dimensional spaces. Our approach leverages Chameleon, which relies on the StarPU dynamic runtime system for efficiently managing large datasets using fine-grained computations on manycore systems. Additionally, we enhance our implementation to accommodate the TLR approximation for the underlying matrix operations via the HiCMA library. We show that our TLR implementation can reduce the complexity of the SOV algorithm and address challenges posed by high-dimensional MVN problems.

To assess the efficacy of our implementation, we focus on the confidence region detection applications that mainly rely on the MVN probability algorithm. Our results demonstrate that TLR efficiently replaces dense by TLR computations with acceptable accuracy loss. Through a comprehensive evaluation, we analyze the accuracy trade-offs associated with this approximation and compare them with dense computations using synthetic and real wind speed datasets.

II. Contributions

We position our paper against existing works [2], [12], [17] and summarize our contributions as follows:

- We leverage the sequential design for solving the Multivariate Normal (MVN) probability problem in high dimensions from [2]. We develop an MVN parallel implementation targeting the SOV algorithm relying on the state-of-the-art parallel task-based linear algebra library Chameleon [13] powered by the StarPU dynamic runtime system [18].
- We enhance the SOV algorithm in two respects. Initially, we incorporate an optimized tile-based Cholesky factorization implementation into the algorithm, allowing faster execution on parallel architectures. Subsequently, we parallelize the computation of probabilities for different QMC samples by dividing the problem into independent tasks and implementing this process through the StarPU dynamic runtime system. Our proposed high-performance implementation contrasts with the *R* implementation in [12], [17] that has shown limitation in parallel efficiency due to the bulk synchronous programming model.
- We extend our implementation to support parallel TLR approximation of the SOV algorithm, accommodating various compression accuracy levels to reduce the complexity of the SOV algorithm through the use of the HiCMA library [15], [16] and StarPU. Compared to [12], [17], using the StarPU dynamic scheduler to orchestrate task parallelism enables us to mitigate the overhead of load imbalance.
- We evaluate the performance and scalability of our proposed dense and TLR implementations on both shared- and distributed-memory systems. In this study, we process a problem size of approximately 500K and 760K in dense and TLR formats, respectively, which is a significant advancement compared to the roughly 16K by [12].
- We improve the confidence region detection algorithm in [2] by incorporating the high-performance MVN probability implementation. We further deploy it across diverse synthetic and real datasets.
- We evaluate the reduction in accuracy when depending on TLR approximation in contrast to the dense solution, using synthetic datasets that depict varying levels of spatial correlation.
- We use our new high-performance confidence region detection implementation on a spatial wind speed dataset in the Middle East to identify optimal locations for establishing wind farms for energy production. The TLR-based algorithm demonstrates comparable accuracy in region detection compared to the dense execution.

III. Background

This section provides an overview of key terms used in this paper, including Multivariate Normal (MVN) probability, Separation-Of-Variables (SOV) algorithm, confidence region detection applications, task-based parallelism, parallel linear algebra libraries powered by dynamic runtime systems, and Tile Low-Rank (TLR) approximation.

A. Multivariate Normal (MVN) Probability

A Multivariate Normal (MVN) probability is a statistical concept related to multivariate analysis, specifically dealing with the distribution of multiple variables. It is widely used in statistical applications, including Bayesian probit models [19], confidence region detection problems [2], and maximum likelihood estimation [12]. It can be defined as a numerical integration problem with high computation complexity in higher dimensions. The MVN probability is defined as follows:

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

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are two n-dimensional vectors representing the lower and upper integration limits, $\boldsymbol{\mu}$ representing the mean, and $\boldsymbol{\Sigma}$ the covariance matrix of a multivariate Gaussian distribution. Here $\boldsymbol{\Sigma}$ is constructed through a predetermined covariance function $\boldsymbol{\Sigma}_{ij} = C(\|\mathbf{h}_{ij}\|; \boldsymbol{\theta})$ where $\|\mathbf{h}_{ij}\|$ represents the distance between spatial locations i and j, and $\boldsymbol{\theta}$ represents the statistical parameters of the underlying statistical field. Without loss of generality, we set $\boldsymbol{\mu} = \mathbf{0}$ in this paper for simplicity. A direct approach to address the integration problem in equation 1 involves employing an MC method, which depends on random sampling to approximate and simulate the integration. This is achieved by generating a large number of random samples and then averaging the integration function values. However, if accuracy is a concern, this approach becomes impractical when dealing with high-dimensional problems, such as the one addressed in this study.

In the literature, computing the MVN probability is a challenging task where quadrature-based algorithms are impractical in higher dimensions. Thus, Genz has proposed a Monte Carlo simulation solution to provide a numerical solution to the high-dimensional MVN probability [20]. The provided solution aims to transform the MVN problem into classic numerical integration problems that can be solved directly using standard integration algorithms. This transformation has been used in the literature under the name "Separation-of-Variables (SOV)" algorithm. Several methods have been proposed in the literature based on the SOV algorithm for efficient calculation of the MVN probability; see [21] for a review. One important research direction is to scale the Monte Carlo solution to higher dimensions by using, for instance, the hierarchical decomposition of the covariance matrix to improve the computation time [12], [22].

B. Separation-Of-Variable (SOV) Algorithm

In [20], Genz has provided in detail the required transformation to equation 1 to be able to compute the MVN probability:

$$\Phi_n(\boldsymbol{a}, \boldsymbol{b}; \mathbf{0}, \boldsymbol{\Sigma}) = \int_{\Phi(a'_1)}^{\Phi(b'_1)} \int_{\Phi(a'_2)}^{\Phi(b'_2)} \cdots \int_{\Phi(a'_n)}^{\Phi(b'_n)} \mathrm{d}\boldsymbol{z}, \quad (2)$$

where $a'_i = \frac{a_i - \Sigma_{j=1}^{i-1} \boldsymbol{L}_{ij} y_j}{\boldsymbol{L}_{ii}}$ and $b'_i = \frac{b_i - \Sigma_{j=1}^{i-1} \boldsymbol{L}_{ij} y_j}{\boldsymbol{L}_{ii}}$. If we define $w_i \overset{i.i.d}{\sim} U(0,1)$ as random numbers from unit uniform distribution, then the transformation includes $\boldsymbol{x} = \boldsymbol{L}\boldsymbol{y}$ and $y_i = \Phi^{-1}[\Phi(a'_i) + \{\Phi(b'_i) - \Phi(a'_i)\}w_i]$. Herein, $\boldsymbol{L}$ is the lower Cholesky factor of $\boldsymbol{\Sigma}$, i.e., $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^\top$ and $\boldsymbol{L}_{ij}$ represents the (i,j) element of $\boldsymbol{L}$. Computing the Cholesky factor $\boldsymbol{L}$ from $\boldsymbol{\Sigma}$ requires $O(n^3)$ operations and $O(n^2)$ memory, where n represents the MVN dimension. The computation of the MVN integration with $\boldsymbol{w}$ is called Monte Carlo sampling [17]. Detailed Monte Carlo algorithms description can be found in [12], [20], [21]. We assume the field has a zero-mean function (i.e., $\boldsymbol{\mu} = \mathbf{0}$).

To apply the SOV results to Monte Carlo algorithms, we have to transform equation 2 into

$$\Phi_n(\boldsymbol{a}, \boldsymbol{b}; \mathbf{0}, \boldsymbol{\Sigma}) = (b'_1 - a'_1) \int_0^1 (b'_2 - a'_2) \cdots \int_0^1 (b'_n - a'_n) \int_0^1 \mathrm{d}\boldsymbol{w}. \quad (3)$$

Then, we can obtain the probabilities in higher dimensions with random numbers in $[0,1]$, the univariate cumulative normal distribution function and its inverse.

C. Confidence Region Detection

In spatial statistics, confidence region (also known as excursion set) detection is the process of identifying areas in a given spatial region where the values exceed a certain level, i.e., threshold u, or expected range with a certain level of confidence $1-\alpha$. Assuming a random field X, we can define the confidence set as shown in [2] by:

$$E^+_{u,\alpha}(X) = \arg\max_D \left\{|D| : \mathbb{P}\left\{D \subseteq A^+_u(X)\right\} \geq 1-\alpha\right\}, \quad (4)$$

where $A^+_u(X) = \{\boldsymbol{s} \in \Omega : X(\boldsymbol{s}) > u\}$ and Ω is the spatial domain. The positive confidence function can be defined as:

$$F^+_u(\boldsymbol{s}) = \sup\{1-\alpha; \boldsymbol{s} \in E^+_{u,\alpha}(X)\}. \quad (5)$$

The confidence region can be easily computed using the confidence function as $\{\boldsymbol{s} : F^+_u(\boldsymbol{s}) \geq 1-\alpha\}$.

This process is used in many applications to locate places where data points exceed predefined limits or exhibit unusual patterns. Some examples are identifying the levels of air pollution in a specific region [6] and recognizing tumors in MRI images [7].

The mathematical definition of the confidence region/excursion set detection problem is as follows: given a latent stochastic field $x(\mathbf{s})$ and a set of observations denoted as $\mathbf{y}$, our goal is to identify the region $E^+_{u,\alpha}$. Here, the threshold value u and confidence level $1-\alpha$ are user-defined values. This $E^+_{u,\alpha}$ can be approximately captured using marginal probabilities as shown in [2], which needs to be more precise to detect the correct regions. Hence, in the work by Bolin and Lindgren [2], an algorithm is proposed, focusing on the computation of Multivariate Normal (MVN) probability for detecting confidence regions in spatial data. However, the complexity of the MVN probability algorithm

poses limitations, rendering it unsuitable for handling large spatial areas, a necessity in numerous applications.

D. Task-based Linear Algebra Libraries and Dynamic Runtime Systems

Task-based parallelism is a parallel computing paradigm that deals with the target problem as a collection of tasks with predetermined dependencies. These tasks can execute concurrently when computing resources are available and as long as there are no violations of the pre-established dependencies. Task-based parallelism offers flexibility and fine-grained computations, making it suitable for various parallel computing applications. Thus, cutting-edge parallel dense linear algebra libraries, like Chameleon [13] and DPLASMA [14], utilize task-based parallelism to deliver efficient and dependable parallel linear solvers using meticulously designed tile-based algorithms. Additionally, harnessing dynamic runtime systems like StarPU [18], PaRSEC [23], QUARK [24], and others can enhance task management and scheduling on available resources, considering workload and resource availability for improved performance. In this work, we rely on the StarPU dynamic runtime system because of its high level of user-productivity achieved via abstraction for expressing parallelism, simplifying the development of parallel applications. It also includes scheduling heuristics for optimizing data movement between different memory hierarchies, which is crucial for performance optimization.

E. Tile Low-Rank (TLR) Approximation

The low-rank approximation is a prevalent mathematical technique used to estimate a dense matrix by representing it as the product of one or more matrices with lower ranks. Given the widespread use of tile-based algorithms in numerous linear algebra packages, Akbudak et al. have introduced a Tile Low-Rank (TLR) approximation method for manycore systems [15]. This approach enables the separate approximation of each tile using the Singular Value Decomposition (SVD) algorithm. In this method, the ranks of the tiles correspond to the most significant singular values and vectors within each off-diagonal tile. The effectiveness of the TLR technique hinges on the ranks achieved for the off-diagonal tiles after compression, which, in turn, is influenced by the precision requirements of the specific application. The use of TLR approximation has found application in various domains, enabling efficient compression of dense matrices and rapid execution of underlying linear algebra operations with acceptable accuracy. Examples include climate modeling [15], [25], [26], astronomy [27], and seismic computation [28].

IV. Confidence Region Detection Framework

This section introduces the proposed computational framework for the confidence region detection problem. By considering a collection of spatial locations, their corresponding measurements, a user-defined threshold u, and a user-defined confidence level $1-\alpha$, the proposed framework can identify regions where values surpass the defined threshold u. We also demonstrate our contribution in parallelizing the underlying high-dimensional Multivariate Normal (MVN) probability algorithm. For clarity, we summarize all symbols used in this section in Table I to facilitate understanding of the content. We use regular (unbolded) characters to denote scalar values, bold lowercase characters for vectors, and bold uppercase characters for matrices. We also use $\boldsymbol{A}(i,j)$ to denote the (i,j) element in a matrix $\boldsymbol{A}$, and $\boldsymbol{A}_{(i,j)}$ to indicate the (i,j) tile in the matrix $\boldsymbol{A}$.

TABLE I: List of symbols.

Symbol	Definition
$\boldsymbol{geom}$	A set of irregularly distributed spatial locations.
$\hat{\boldsymbol{\theta}}$	Estimated statistical parameter vector.
$1-\alpha$	User-defined confidence level.
u	User-defined threshold.
$\boldsymbol{\Sigma}$	Covariance matrix.
$\Phi(x)$	Univariate normal distribution function $\mathbb{P}(Z \leq x)$.
$\boldsymbol{p}_M$	Marginal probability vector $\mathbb{P}(X_i > u)$.
$\boldsymbol{o}_{\boldsymbol{p}_M}$	Indices of ordered $\boldsymbol{p}_M$.
$\boldsymbol{f}$	Positive confidence function in equation 5.
$\boldsymbol{a}$	Lower limits of the MVN integration.
$\boldsymbol{b}$	Upper limits of the MVN integration.
$\boldsymbol{A}$	Matrix of n set of $\boldsymbol{a}$ vectors.
$\boldsymbol{B}$	Matrix of n set of $\boldsymbol{b}$ vectors.
$\boldsymbol{R}$	Random matrix filled with i.i.d $U(0,1)$ values.

A. Confidence Region Detection Algorithm

Algorithm 1 illustrates how to identify confidence regions relying on the computation of Multivariate Normal (MVN) probabilities. The algorithm takes several inputs, as indicated in line 1, including $\hat{\boldsymbol{\theta}}$ obtained using a specific covariance function of the form $C(\|\mathbf{h}\|;\boldsymbol{\theta})$, where $\mathbf{h} = \mathbf{s}_1 - \mathbf{s}_2 \in \mathbb{R}^d$ and $\|\mathbf{h}\|$ denotes the Euclidean norm. Herein, we employ the Matérn covariance function [29] with the form:

$$C(\|\mathbf{h}\|;\boldsymbol{\theta}) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}\left(\frac{\|\mathbf{h}\|}{a}\right)^{\nu}\mathcal{K}_\nu\left(\frac{\|\mathbf{h}\|}{a}\right), \qquad (6)$$

where $\mathcal{K}_\nu(\cdot)$ represents the modified Bessel function of the second kind with order ν, and $\Gamma(\cdot)$ denotes the gamma function. In this context, $\boldsymbol{\theta}$ contains the marginal variance ($\sigma^2 > 0$), smoothness ($\nu > 0$), and spatial range ($a > 0$). We obtained the Matérn covariance parameters $\hat{\boldsymbol{\theta}}$ for the given dataset using the Maximum Likelihood Estimation (MLE) algorithm employed in the *ExaGeoStat* software [30].

In line 2, the algorithm generates a covariance matrix $\boldsymbol{\Sigma}$ using the estimated parameters $\hat{\boldsymbol{\theta}}$ and the set of locations $\boldsymbol{geom}$. In lines 3-5, the marginal probabilities of the locations denoted as $\boldsymbol{p}_M$, are calculated using the mean μ_i, i.e., μ_i is the mean in location i across time, where u is a user-defined threshold value. In line 6, we store the indices of the locations based on their marginal probabilities in $\boldsymbol{o}_{\boldsymbol{p}_M}$ vector in descending order. In line 7, the algorithm initializes the required data structures that will be used in the computation, i.e., Chameleon/HiCMA descriptors. The descriptors are unique data structures that

Algorithm 1 Confidence Region Detection Algorithm

1: **function** CRD ($\boldsymbol{geom}$: a set of spatial locations, $\boldsymbol{Y}$ a set of measurements of sampled locations, $\hat{\boldsymbol{\theta}}$: Matérn covariance parameters; n: number of spatial locations)
2: Generate a covariance matrix $\boldsymbol{\Sigma}$ using $\hat{\boldsymbol{\theta}}$ and $\boldsymbol{geom}$ or read a given covariance matrix $\boldsymbol{\Sigma}$.
3: **for** $0 \leq i < n$ **do**
4: $\boldsymbol{p}_M[i] \leftarrow 1 - \Phi((u - \mu[i] - \boldsymbol{Y}[i])/\sqrt{\boldsymbol{\Sigma}[i,i]})$
5: **end for**
6: $\boldsymbol{o}_{\boldsymbol{p}_M} \leftarrow \text{descending\_order\_index}(\boldsymbol{p}_M)$
7: pmvn_init()
8: $\boldsymbol{L} \leftarrow \text{dpotrf}(\boldsymbol{\Sigma})$ (a) ▷ Cholesky factorization
9: $\boldsymbol{b} \leftarrow \{\infty, \ldots, \infty\}$
10: **for** $n > i \geq 0$ **do**
11: $\boldsymbol{c} \leftarrow \boldsymbol{o}_{\boldsymbol{p}_M}[1:i]$
12: $\boldsymbol{a} \leftarrow \{-\infty, \ldots, -\infty\}$
13: $\boldsymbol{a}[\boldsymbol{c}] \leftarrow (u - \mu[\boldsymbol{c}] - \boldsymbol{Y}[\boldsymbol{c}])/\sqrt{\boldsymbol{\Sigma}[\boldsymbol{c},\boldsymbol{c}]}$
14: $\boldsymbol{f}[\boldsymbol{c}] \leftarrow \text{PMVN}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{L}, n, N, m)$
15: **end for**
16: **end function**

enable the storage of matrices as a set of tiles managed by one or more computing processes. In the case of HiCMA, the pmvn_init() function also encompasses the compression of the covariance matrix $\boldsymbol{\Sigma}$ into the TLR format. In line 8, a Cholesky factorization operation is performed over $\boldsymbol{\Sigma}$ to obtain the lower triangular matrix $\boldsymbol{L}$, where $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^\top$. In lines 9-15, the locations are extracted according to $\boldsymbol{o}_{\boldsymbol{p}_M}$ and the joint MVN probabilities are computed using the PMVN function in algorithm 2 based on the lower and upper limits, constructing the confidence function $\boldsymbol{f}$.

B. Multivariate Normal Probability (PMVN) Algorithm

Algorithm 2 calculates the Multivariate Normal (MVN) probability. To allow parallel implementation of the algorithm, four matrices – $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{R}$, and a temporary matrix $\boldsymbol{Y}$ – are used. Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ incorporate redundant lower limits vector $\boldsymbol{a}$ and upper limits vector $\boldsymbol{b}$, respectively (lines 2-3). Matrix $\boldsymbol{R}$ is set with values drawn from a unit uniform distribution (line 4). In lines 5-7, the algorithm invokes the QMC() (introduced later in Algorithm 3) to update the set of tiles in the first row of matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{Y}$. This loop can be executed concurrently, with each task handling a single tile. To propagate these changes to all the tiles in the next row, a set of GEMMs operations are concurrently applied, as shown in lines 10-13. The QMC() algorithm is once again invoked for all the tiles in this row, as demonstrated in lines 15-17. These procedures are repeated until the final row is reached, resulting in the $\boldsymbol{p}$ probability vector for each column of tiles. The final MVN probability, p, is computed as the mean of the $\boldsymbol{p}$ vector (line 19).

Algorithm 2 Multivariate Normal Probability (PMVN) Integration Algorithm

1: **function** PMVN($\boldsymbol{a}$: lower limits; $\boldsymbol{b}$: upper limits; $\boldsymbol{L}$: Cholesky factor; n: dimension; N: QMC sample size; m: tile size)
2: $\boldsymbol{A} \leftarrow [\boldsymbol{a}, \boldsymbol{a}, \ldots, \boldsymbol{a}] \in \mathbb{R}^{n \times N}$
3: $\boldsymbol{B} \leftarrow [\boldsymbol{b}, \boldsymbol{b}, \ldots, \boldsymbol{b}] \in \mathbb{R}^{n \times N}$
4: $\boldsymbol{R} \in \mathbb{R}^{n \times N}; \boldsymbol{R}(i,j) \overset{i.i.d}{\sim} \mathcal{U}(0,1)$
5: **for** $0 \leq k < N/m$ **do** ▷ $\lceil N/m \rceil$ is # tile-by-column
6: $\text{QMC}(\boldsymbol{L}_{(0,0)}, \boldsymbol{R}_{(0,k)}, \boldsymbol{A}_{(0,k)}, \boldsymbol{B}_{(0,k)}, \boldsymbol{p}_{(k)}, \boldsymbol{Y}_{(0,k)})$
7: **end for** (b)
8: **for** $1 \leq r < n/m$ **do** ▷ $\lceil n/m \rceil$ is # tile-by-row
9: **for** $r \leq j < n/m$ **do**
10: **for** $0 \leq k < N/m$ **do**
11: $\boldsymbol{A}_{(j,k)} \leftarrow \boldsymbol{A}_{(j,k)} - \boldsymbol{L}_{(j,r-1)} \cdot \boldsymbol{Y}_{(r-1,k)}$
12: $\boldsymbol{B}_{(j,k)} \leftarrow \boldsymbol{B}_{(j,k)} - \boldsymbol{L}_{(j,r-1)} \cdot \boldsymbol{Y}_{(r-1,k)}$
13: **end for** (c)
14: **end for**
15: **for** $0 \leq k < N/m$ **do**
16: $\text{QMC}(\boldsymbol{L}_{(r,r)}, \boldsymbol{R}_{(r,k)}, \boldsymbol{A}_{(r,k)}, \boldsymbol{B}_{(r,k)}, \boldsymbol{p}_{(k)}, \boldsymbol{Y}_{(r,k)})$
17: **end for** (d)
18: **end for**
19: **return** $\text{mean}(\boldsymbol{p})$
20: **end function**

C. Quasi-Monte Carlo (QMC) Algorithm

Equation 3 shows that the parallelization of the algorithm is feasible only across Monte Carlo (MC) chains. This limitation arises because a'_i, b'_i, and y_i are interdependent in an iterative manner. Assuming tile-based $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{Y}$ matrices within a specific tile, each row relies on the preceding row for the update process.

Algorithm 3 provides a mechanism to update a single tile of $\boldsymbol{p}$ and $\boldsymbol{Y}$ while operating on m MC chains in total. For simplicity, we assume all tiles have the same dimensions. The algorithm requires one tile each for the lower Cholesky factor $\boldsymbol{L}$, the random matrix $\boldsymbol{R}$, the lower and upper limits matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, and the probability vector $\boldsymbol{p}$, the random matrix $\boldsymbol{Y}$. In lines 3-7, we initialize the first row in a given tile to compute a'_1, b'_1, and y_1 in equation 3. In lines 8-15, we simultaneously complete m steps for all the m MC chains. The function does not provide a direct output, as the updates to $\boldsymbol{p}$ and $\boldsymbol{Y}$ occur within the process and are propagated to subsequent tiles during the subsequent steps in Algorithm 2.

In Algorithms 1 and 2, red boxes were added over specific lines to highlight the integration of task-based parallel computation as follows: (a) Cholesky factorization, (b) and (d) parallel QMC computations, and (c) parallel GEMMs. The step (a) can be performed using both dense and TLR computations. However, (b), (c), and (d) are only performed in dense since $\boldsymbol{A}$ and $\boldsymbol{B}$ are non-admissable matrices.

Algorithm 3 QMC for MVN probabilities Algorithm

1: **function** QMC($\boldsymbol{L}$: Cholesky factor tile; $\boldsymbol{R}$: random matrix tile; $\boldsymbol{A}$: lower limits tile; $\boldsymbol{B}$: upper limits tile; $\boldsymbol{p}$: probability vector tile; $\boldsymbol{Y}$: temp matrix tile)
2: $m \leftarrow \text{nrow}(\boldsymbol{A})$ ▷ Tile size
3: **for** $1 \leq j < m$ **do**
4: $\boldsymbol{A}(0,j) \leftarrow \frac{\boldsymbol{A}(0,j)}{\boldsymbol{L}(0,0)}$; $\boldsymbol{B}(0,j) \leftarrow \frac{\boldsymbol{B}(0,j)}{\boldsymbol{L}(0,0)}$
5: $\boldsymbol{Y}(0,j) \leftarrow \Phi^{-1}[\boldsymbol{R}(0,j) \cdot \{\Phi(\boldsymbol{B}(0,j)) - \Phi(\boldsymbol{A}(0,j))\}]$
6: $\boldsymbol{p}_j \leftarrow \boldsymbol{p}_j \cdot \{\Phi(\boldsymbol{B}(0,j)) - \Phi(\boldsymbol{A}(0,j))\}$
7: **end for**
8: **for** $1 \leq i < m$ **do**
9: **for** $1 \leq j < m$ **do**
10: $s \leftarrow \boldsymbol{L}(i, 1:(i-1))\boldsymbol{Y}(1:(i-1), j)$
11: $a' \leftarrow \frac{\boldsymbol{A}(i,j)-s}{\boldsymbol{L}(i,i)}$; $b' \leftarrow \frac{\boldsymbol{B}(i,j)-s}{\boldsymbol{L}(i,i)}$
12: $\boldsymbol{Y}(i,j) \leftarrow \Phi^{-1}[\boldsymbol{R}(i,j) \cdot \{\Phi(b') - \Phi(a')\}]$
13: $\boldsymbol{p}_j \leftarrow \boldsymbol{p}_j \cdot \{\Phi(b') - \Phi(a')\}$
14: **end for**
15: **end for**
16: **end function**

V. Results

This section evaluates the proposed implementation for the MVN probability algorithm under various objectives. Initially, we intend to gauge the performance in terms of time-to-solution of the dense and TLR implementations on shared- and distributed-memory systems. Subsequently, our focus is on assessing the accuracy of the proposed framework in the context of confidence region detection applications, encompassing both synthetic and real datasets. Lastly, our goal is to evaluate the accuracy of TLR approximation in conjunction with confidence region detection applications, considering the compression level necessary to maintain the required precision for the application.

A. Environment Settings

We assess the performance of our proposed framework across various shared-memory architectures: a dual-socket 28-core Intel Icelake 6330 running at 2.00 GHz, a dual-socket 64-core Intel Cascade Lake running at 2.30 GHz, a dual-socket 128-core AMD Milan running at 2.00 GHz, and a dual-socket 64-core AMD Naples running at 2.20 GHz. In the distributed-memory experiments, we rely on a Cray XC40 system, Shaheen-II, at KAUST, with 6,174 dual-socket 16-core Intel Haswell processors operating at 2.3 GHz. Each node in this system is equipped with 128 GB of DDR4 memory. The Shaheen system, boasting a total of 197,568 processor cores, is backed by an aggregate memory of 790 TB.

We compile our software using gcc v10.2.0 and link it with various libraries, including Chameleon, HiCMA, HWLOC v2.8.0, StarPU v1.3.9, Intel MKL v2020.0.166, and NLopt v2.7.0 for optimization.

B. Datasets

We use *ExaGeoStat* [30] to generate three spatial synthetic datasets from the exponential kernel with the range parameter equal to 0.033 (weak correlation), 0.1 (medium correlation) and 0.234 (strong correlation). Each dataset contains 40,000 data points. The generated data comprise a set of locations and corresponding measurements recorded at those locations. We follow the synthetic data generation process in [17], where 6,250 samples under the additive noise $\mathcal{N}(0, 0.5^2)$ denoted as $\boldsymbol{y}$, are randomly selected from the original data for the sake of computing the posterior covariance matrix $\boldsymbol{\Sigma}_{\text{post}}$ as follows:

$$\boldsymbol{\Sigma}_{\text{post}} = (\boldsymbol{\Sigma}^{-1} + (1/0.5^2)\boldsymbol{A}^\top \boldsymbol{A})^{-1} \tag{7}$$

where $\boldsymbol{A} \in \mathbb{R}^{6250 \times 40000}$ denotes the indicator matrix, $\boldsymbol{\Sigma}$ is the covariance matrix of $\boldsymbol{x}$, and $\boldsymbol{\mu}_{\text{post}}$ is the posterior mean of $\boldsymbol{x}$ which can be computed as:

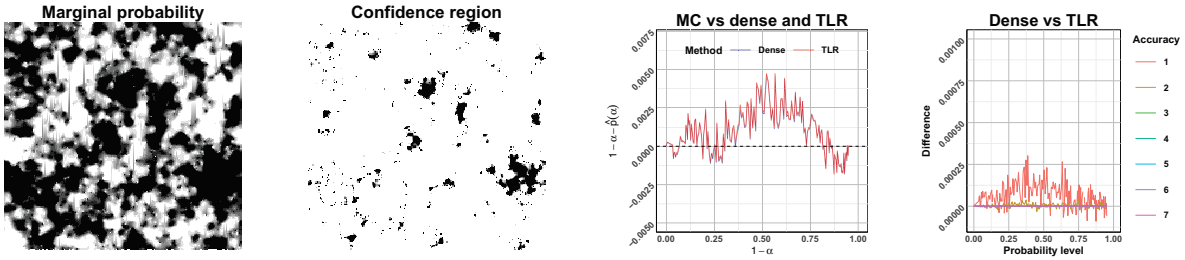
$$\boldsymbol{\mu}_{\text{post}} = \boldsymbol{\mu} + (1/0.5^2)\boldsymbol{\Sigma}_{\text{post}}\boldsymbol{A}^\top(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\mu}) \tag{8}$$

where $\boldsymbol{\mu}$ denotes the mean of $\boldsymbol{x}$. The posterior covariance matrix $\boldsymbol{\Sigma}_{\text{post}}$ can be used as input to Algorithm 1 (line 2) and the posterior mean $\boldsymbol{\mu}_{\text{post}}$ can be used to update the $\boldsymbol{a}$ vector in line 12.

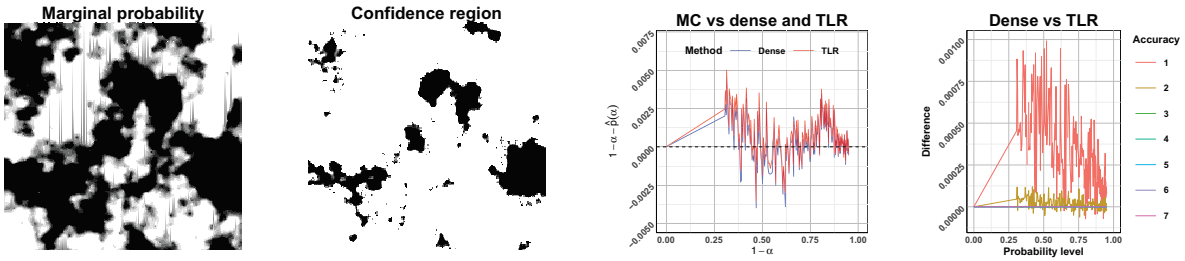
We also present an analysis of wind speed data in Saudi Arabia from 2013 to 2016 by [31]. The dataset consists of hourly measurements aggregated into daily values across 53,362 locations. The focus of the study is to provide a quantitative measure of wind speed on a specific day, specifically July 15, 2015. We performed postprocessing steps before applying the confidence region detection algorithm to the data. First, we computed the mean and standard deviation of the daily wind speed, followed by standardizing the average wind speed on the chosen day. We can then assume the wind field adheres to a stationary Gaussian random field characterized by a zero mean. Second, a Matérn kernel is fitted using *ExaGeoStat* on the transformed field. The data and the estimated parameters are inputs to the confidence region detection Algorithm 1 to delineate regions on the map with a 0.95 probability of experiencing high wind speeds. The results include plots illustrating the original plot, excursion sets derived through the proposed methods, and the marginal probability map. The comparison reveals that the marginal probability map differs significantly from the collective excursion set, highlighting the importance of modeling using multivariate normal probabilities. Additionally, the excursion maps generated by the dense and TLR methods exhibit consistency, with TLR being favored due to its faster computation.

C. Qualitative Assessment

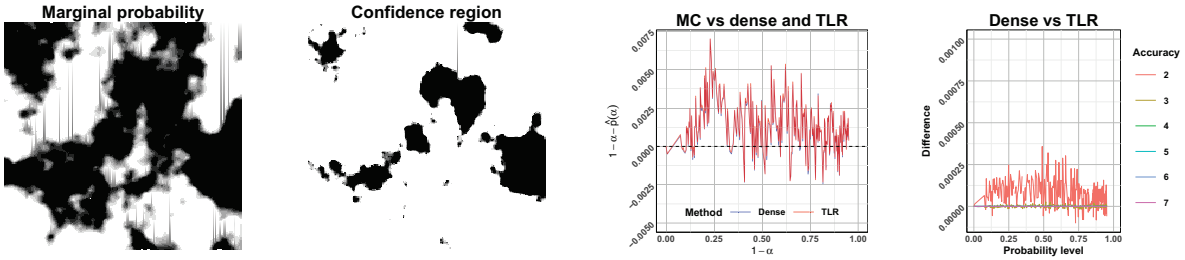
We aim to evaluate the accuracy of identifying confidence regions using our proposed implementation for dense and TLR computations. We employed synthetic and real datasets, as elaborated in the following subsections. All the experiments in this section rely on QMC sample size = 10,000, which consistently yielded higher accuracy than smaller ones. In the performance section, we show the performance of our proposed implementations with different QMC sizes regardless of the obtained accuracy.



(a) Weak correlation (1, 0.033, 0.5).



(b) Medium correlation (1, 0.1, 0.5).



(c) Strong correlation (1, 0.234, 0.5).

Fig. 1: Confidence region detection accuracy assessment using 40K synthetic datasets generated in a regular grid with varying correlation levels. The figure illustrates the accuracy of both dense and TLR results compared to MC results (MC error) and the accuracy of TLR results compared to dense results.

1) Accuracy Assessment on Synthetic Datasets: For synthetic dataset accuracy assessment, we use three datasets with different properties, each consisting of 40,000 data points, characterized by varying correlations, i.e., weak, medium, and strong. The results are depicted in Figure 1. For each correlation level, four images illustrate the detected regions using marginal probability, detected regions employing the confidence region detection algorithm, comparing dense and TLR results with MC results (MC error), and comparing TLR results with the dense results. An MC validation algorithm can be used to validate the accuracy of confidence region detection methods, such as the ones proposed in this paper. This algorithm draws N samples from the fitted distribution, where N_s represents the number of samples exceeding a given threshold. The MC estimate of the confidence probability is then expressed as $\hat{p}(\alpha) = N_s/N$, and it is expected that $\hat{p}(\alpha) \approx 1-\alpha$ if the confidence region is accurately estimated. In the third column of Figure 1, we show the difference $1-\alpha-\hat{p}(\alpha)$ as a function of $1-\alpha$, based on a sample size of $N = 50{,}000$. The small discrepancy observed across all α values is primarily attributed to the error associated with the MC estimation of $\hat{p}(\alpha)$; this error is unrelated to the accuracy of our method. The two left-row graphs reveal notable differences between the marginal probabilities and the confidence region obtained through the MVN algorithm. This underscores the significance of collectively evaluating the multivariate normal (MVN) probabilities. In particular, the confidence regions represent a subset where the joint probabilities exceed $1-\alpha$.

In the two right-row graphs, a comparison is made between probability results obtained from dense (red curves) and TLR (blue curves) methods with results derived from naive MC chains. These results maintain the same level of error as those presented in [17] and are also an order of magnitude lower than the second example in [2], which utilizes an approximate posterior covariance matrix.

Additionally, we note that the difference between the results obtained from dense and TLR methods is minimal.

To showcase these distinctions, we carried out experiments using TLR with varying levels of accuracy. For weak and medium correlations, an accuracy of $1e-1$ is acceptable, with discrepancies smaller than 1×10^{-3}. As accuracy increases, the gap between the two methods steadily diminishes in all instances. When accuracy surpasses $1e-3$, the difference becomes negligible enough to be ignored.

2) Accuracy Assessment on a Real Wind Speed Dataset: We performed some data preprocessing on the wind data before applying our algorithm, wherein we calculated the average daily wind speed for all summer days between 2013 and 2016 in Saudi Arabia. Next, we compute the mean and standard deviation of the summer wind speed over time. The July 15, 2015, wind speed record is then standardized by subtracting the mean and dividing by the standard deviation. The standardized data is subsequently fed into *ExaGeoStat* to estimate the Matérn parameters, yielding the values $(1, 0.005069, 1.43391)$.

In our analysis, we set the threshold value at 4 m/s following [32]. Furthermore, we opt for a confidence level of 0.95. We employ a dense tile size of 320, a TLR tile size of 980, and set a maximum rank of 145, maintaining the TLR accuracy level of $1e-4$. The experiment results are depicted in Figure 2. Figure 2a illustrates the initial distribution of wind speeds on July 15, 2015, highlighting elevated wind speeds in the northern, eastern, and southwestern regions, varying from 2 m/s to approximately 12 m/s. Figure 2b displays the marginal probability of wind speeds at various locations. However, the results pose a problem, as a significant portion of Saudi Arabia exhibits a probability greater than 0.8 of experiencing an average wind speed exceeding 4 m/s on that particular day, which is highly unrealistic.

To tackle this issue, we employ Algorithm 1 to identify confidence regions. Figure 2c and Figure 2d depict these confidence regions, focusing mainly on the mountainous areas in the north, east, and west. Notably, the results obtained from the dense and TLR versions exhibit substantial similarity. We plot the differences between the dense and TLR results across various probability levels to emphasize their distinctions. The analysis reveals that the disparity is of the order of 1×10^{-4}, further affirming the reliability of the TLR version as shown in Figure 3.

D. Quantitative Assessment

Herein, we present the performance evaluation of our Multivariate Normal (MVN) probability implementation on shared- and distributed-memory systems. The comparison includes dense and Tile Low-Rank (TLR) approximations with varying levels of accuracy, considering different MVN dimensions and QMC sample sizes.

1) Performance on Shared-Memory Systems: We leverage four distinct shared-memory architectures to evaluate the time-to-solution of our PMVN algorithm (Algorithm 2) as stated in section V-A. We operate on all the cores for each machine. Performance curves on various architectures, with different Multivariate Normal (MVN) problem dimensions and varying Quasi-Monte Carlo (QMC) sample sizes, are shown in Figure 4. The dashed curve shows the performance achieved with TLR approximation across various QMC sample sizes, surpassing dense computation by up to 14X, 19X, 9X, and 20X on Intel Ice Lake, Intel Cascade Lake, AMD Milan, and AMD Naples, respectively, as shown in table II. The table also shows that TLR still can achieve better speedup than the dense version with smaller QMC sample sizes, i.e., 100 and 1000.

System	QMC sample sizes		
	100	1000	10,000
56-core Intel Ice Lake	3X	3X	14X
40-core Intel Cascade Lake	3X	3X	19X
64-core AMD Milan	5X	5X	20X
128-core AMD Naples	2X	2X	9X

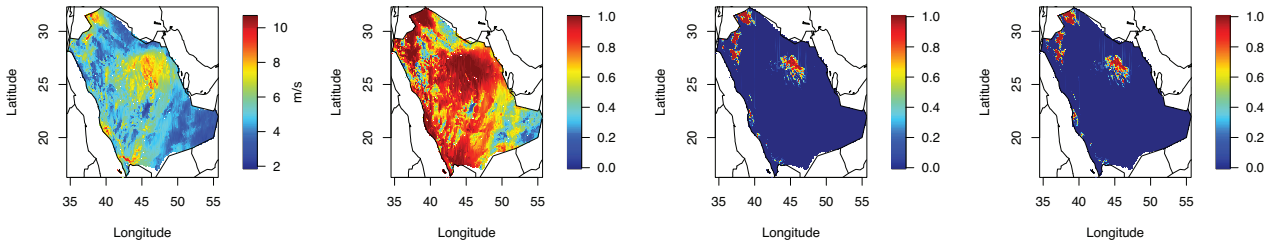
TABLE II: Speedup of TLR to dense implementations with different QMC sample sizes on shared-memory systems.

The notable speedup achieved through the utilization of TLR compared to dense computation comes from the fast execution of the Cholesky factorization in the TLR format. With a compression accuracy requirement of $1e-3$, validated above through accuracy assessments on synthetic and real datasets, the small ranks of various tiles enable expedited Cholesky factorization compared to the dense version.

Figure 5 illustrates the rank distributions of a $19{,}600 \times 19{,}600$ matrix at a $1e-3$ compression accuracy. Most tiles exhibit minimal ranks, facilitating a faster computation of the Cholesky factorization operation. The settings employed align with those detailed in Figure 1. Moreover, the figure shows that the ranks exhibit a more pronounced degradation under strong correlations than weak ones, which helps speed up the execution when the correlation is stronger between different locations.

To validate the accuracy of the detected confidence regions, we use an MC validation algorithm as mentioned in V-C. The execution overhead of the MC validation process on all four machines above is shown in Figure 6 (average over 5 runs). However, because the MC validation is not a complete algorithm to obtain the confidence regions, its execution time should not be considered a part of our algorithm and is unsuitable for comparisons.

2) Performance on Distributed-Memory Systems: To evaluate the effectiveness of our implementation on distributed-memory systems, we rely on up to 512 nodes of a Cray XC40 system. In Figure 7, two sub-figures depict the performance on two sets of nodes: 16, 32, 64, and 128, and 64, 128, 256, and 512. We experiment with varying the problem dimensionality across different numbers of nodes. In the left figure, we observe the efficiency of scalability for dense execution (represented by the red curves) across different node configurations, scaling up to $n = 360{,}000$. The scalability of TLR (indicated by the dashed blue curves) is also acceptable across various node counts, although some performance degradation is shown when utilizing 64 nodes. Table III shows the speedup of TLR compared to dense using different numbers of nodes. The speedup of TLR approximation execution compared to the



(a) Original data. (b) Marginal probability. (c) Confidence regions (Dense). (d) Confidence regions (TLR).

Fig. 2: Results of summer wind speed data (on July 15, 2015) in the Middle East (Saudi Arabia).

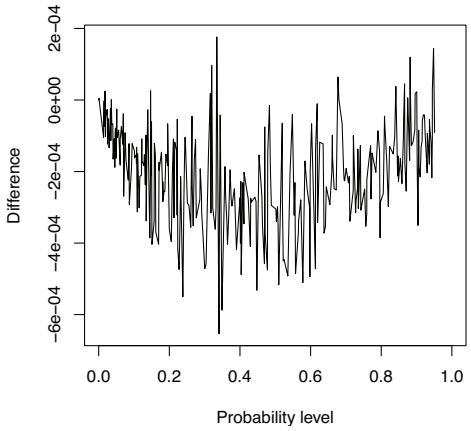
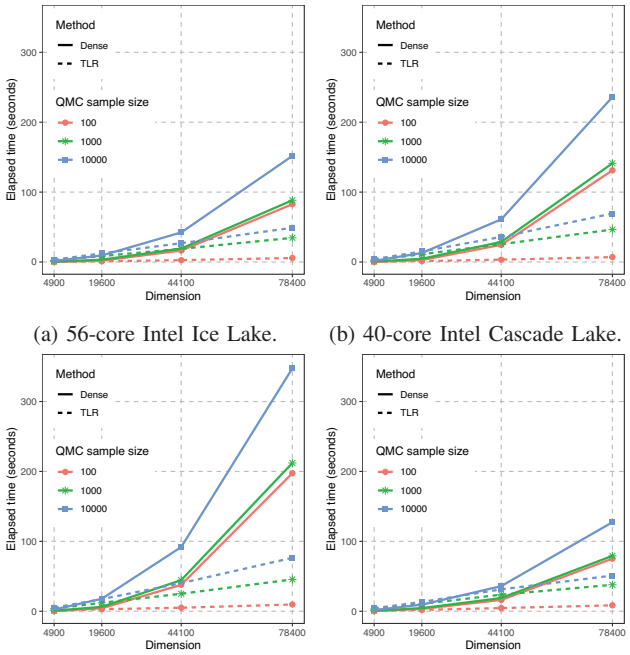


Fig. 3: Difference between dense and TLR results of summer wind speed data (on July 15, 2015) in Saudi Arabia.

Number of nodes	Speedup using QMC sample size = 10,000
16	1.8X
32	1.8X
64	1.4X
128	1.7X
256	1.3X
512	1.5X

TABLE III: Speedup of TLR to dense implementations on a Cray XC40 system with different number of nodes. The most accurate QMC sample size is used, i.e., 10,000.



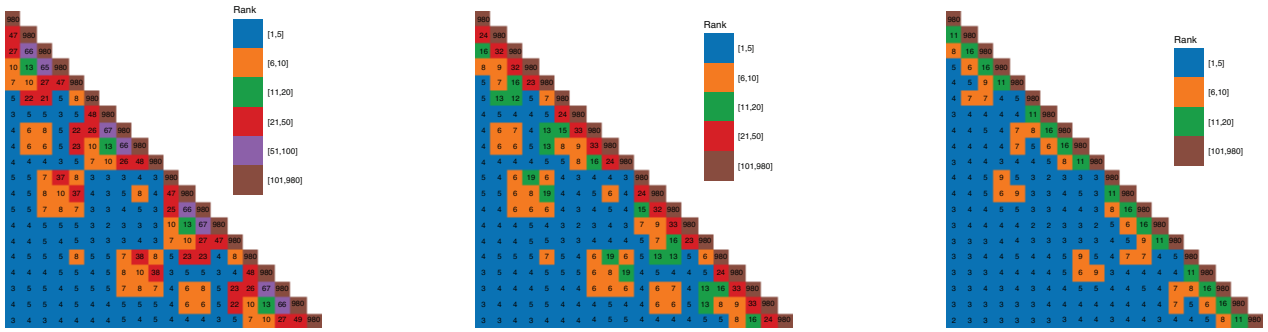
(a) 56-core Intel Ice Lake. (b) 40-core Intel Cascade Lake.

(c) 64-core AMD Milan. (d) 128-core AMD Naples.

Fig. 4: Performance of one MVN integration operation on multiple shared-memory architectures using dense and TLR approximation.

dense computation is up to 1.8X. In the sub-figure on the right, performance results for up to 512 nodes and 760,384 data points are presented. The figure demonstrates scalability for both dense and TLR executions, except for the 256 nodes configuration, which exhibits some performance issues at two points. The maximum speedup achieved by TLR approximation execution compared to the dense execution in this sub-figure is 1.7X.

Comparing the performance with shared-memory computations, the dense computation in the MVN probability Algorithm 2 takes more time than the Cholesky factorization operation, which is the only part that can be computed in either dense or low-rank format. This is why the differential impact on performance is observed in the TLR version of the algorithm on distributed-memory systems. According to our experiments, the TLR Cholesky factorization within the MVN probability algorithm only can achieve speedups of 5.2X, 4.5X, 2.6X, 3.1X, 1.9X, and 2.6X on 16, 32, 64, 128, 256, and 512 nodes, respectively, when compared to the dense version.



(a) Weak correlation (1, 0.033, 0.5). (b) Medium correlation (1, 0.1, 0.5). (c) Strong correlation (1, 0.234, 0.5).

Fig. 5: Rank distributions of a 19600 × 19600 covariance matrix using a 980 tile size with Matérn covariance function under three different settings when compressing the matrix using TLR approximation with accuracy 1e-3 (preserves the application accuracy).

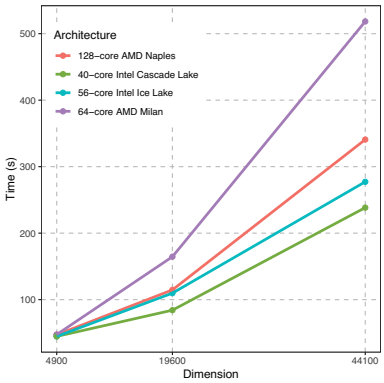


Fig. 6: Performance of the MC validation process on various dimensional sizes using shared-memory architectures.

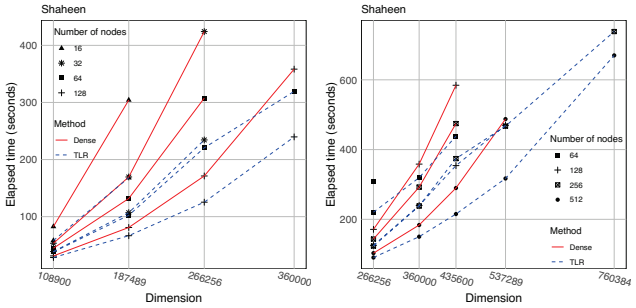


Fig. 7: Performance of one MVN integration operation on a Cray XC40 distributed-memory system using dense and TLR approximation.

VI. Related Work

Multivariate normal (MVN) probability frequently appears in statistical applications, including the probability density functions of several skew-normal [33], Bayesian probit models [19], and confidence regions detection problems [2], and maximum likelihood estimation [12]. It can be defined as a numerical integration problem with n dimensions, which has a prohibitive computation when n is large. Computing the MVN probability is a challenging task where quadrature-based algorithms are impractical in higher dimensions. In 1992, Genz et al. [20] proposed a Monte Carlo-based solution to reduce the complexity of computing the MVN probability by applying a set of transformations to the original problem. The proposed algorithm has $O(n^3)$ complexity to compute the Cholesky factorization of $n \times n$ covariance matrix and $O(n^2)$ to process a single MC sample. In the literature, many solutions have been provided. For instance, Genton et al. [22] have proposed a hierarchical quasi-Monte Carlo (QMC) method to improve the underlying linear algebra operations to compute the MVN probability. The provided optimization includes compressing the covariance matrix using a hierarchical low-rank approximation to reduce the complexity of processing a single MC sample from $O(n^2)$ to $O(mn + kn\log(n/m))$, i.e., k is the rank of off-diagonal matrix blocks and m is the inadmissible matrix blocks. In [34], the minimax tilting method has been proposed as an efficient approximation to the MVN problem, accurately estimating the required probability. The minimax tilting method can improve the convergence rate but requires an optimization step to $O(n)$ parameters. In [35], a two-step method has been proposed where the original MVN probability is decomposed into a low-dimensional and high-dimensional residual. However, this method can be used in the case of orthant MVN problems with constant upper and lower integration limits. The Multivariate Normal (MVN) probability finds utility in various applications, including detecting confidence regions. In such applications, the key objective is identifying regions surpassing a specified threshold within a given confidence level. These applications are prevalent in medical [7], [9] and environmental sciences [36], [37].

VII. Conclusion

This study presents a novel mitigation of the problem of the high complexity of utilizing MVN by proposing a parallel

implementation of the SOV algorithm designed for manycore systems. This implementation enables rapid computation of MVN probability using dense and Tile Low-Rank (TLR) approximation methods by relying on the StarPU dynamic runtime system and advanced linear algebra libraries, i.e., Chameleon and HiCMA. Our proposed implementation has been applied to the context of confidence region detection applications, demonstrating its effectiveness in handling large spatial regions encompassing up to 700K locations. Furthermore, our proposed TLR-based implementations exhibit a notable speedup compared to the dense solution, achieving up to 20X improvement on shared-memory systems. These enhancements come with high accuracy achievements compared to dense solutions. We assess the accuracy of our implementation through confidence region detection applications using synthetic and wind speed datasets.

In future work, we aim to incorporate multi- and mixed-precision executions to enhance support for the MVN probability algorithm. The results obtained in this paper indicate that the algorithm maintains the necessary accuracy even at very low levels of TLR compression accuracy. Consequently, we anticipate that lower-precision executions can expedite the computation process with minimal impact on the required accuracy. In such a scenario, the natural extension of this work would involve GPU support, leveraging tensor core execution for the underlying linear algebra operations.

VIII. Acknowledgment

Financial support and backing for this study were provided by King Abdullah University of Science and Technology (KAUST) via the Office of Sponsored Research (OSR). The research utilized the facilities of the Extreme Computing Research Center (ECRC) and the KAUST Supercomputing Laboratory (KSL). Key resources employed in this study included the Cray XC40 and the Shaheen II supercomputer, which are pivotal assets of the KSL.

References

[1] A. J. Cannon, "Multivariate quantile mapping bias correction: an n-dimensional probability density function transform for climate model simulations of multiple variables," *Climate dynamics*, vol. 50, pp. 31–49, 2018.

[2] D. Bolin and F. Lindgren, "Excursion and contour uncertainty regions for latent Gaussian models," *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, pp. 85–106, 2015.

[3] E. Pignat and S. Calinon, "Bayesian Gaussian mixture model for robotic policy imitation," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4452–4458, 2019.

[4] H. Sun, H. V. Burton, and H. Huang, "Machine learning applications for building structural design and performance assessment: State-of-the-art review," *Journal of Building Engineering*, vol. 33, p. 101816, 2021.

[5] G. Amisano and J. Geweke, "Prediction using several macroeconomic models," *Review of Economics and Statistics*, vol. 99, no. 5, pp. 912–925, 2017.

[6] M. Cameletti, F. Lindgren, D. Simpson, and H. Rue, "Spatio-temporal modeling of particulate matter concentration through the spde approach," *AStA Advances in Statistical Analysis*, vol. 97, pp. 109–131, 2013.

[7] K. Ejaz, M. Arif, M. S. M. Rahim, D. Izdrui, D. M. Craciun, and O. Geman, "Confidence region identification and contour detection in MRI image," *Computational Intelligence and Neuroscience*, vol. 2022, 2022.

[8] H. Bevins, A. Fialkov, E. de Lera Acedo, W. Handley, S. Singh, R. Subrahmanyan, and R. Barkana, "Astrophysical constraints from the saras 3 non-detection of the cosmic dawn sky-averaged 21-cm signal," *Nature Astronomy*, vol. 6, no. 12, pp. 1473–1483, 2022.

[9] K. Ejaz, M. S. M. Rahim, U. I. Bajwa, H. Chaudhry, A. Rehman, and F. Ejaz, "Hybrid segmentation method with confidence region detection for tumor identification," *IEEE Access*, vol. 9, pp. 35 256–35 278, 2020.

[10] M. Sommerfeld, S. Sain, and A. Schwartzman, "Confidence regions for spatial excursion sets from repeated random field observations, with an application to climate," *Journal of the American Statistical Association*, vol. 113, no. 523, pp. 1327–1340, 2018.

[11] R. E. Caflisch, "Monte carlo and quasi-monte carlo methods," *Acta numerica*, vol. 7, pp. 1–49, 1998.

[12] J. Cao, M. G. Genton, D. E. Keyes, and G. M. Turkiyyah, "Exploiting low-rank covariance structures for computing high-dimensional normal and student-t probabilities," *Statistics and Computing*, vol. 31, pp. 1–16, 2021.

[13] E. Agullo, C. Augonnet, J. Dongarra, H. Ltaief, R. Namyst, S. Thibault, and S. Tomov, "A hybridization methodology for high-performance linear algebra software for GPUs," in *GPU Computing Gems Jade Edition*. Elsevier, 2012, pp. 473–484.

[14] G. Bosilca, A. Bouteiller, A. Danalis, M. Faverge, A. Haidar, T. Herault, J. Kurzak, J. Langou, P. Lemarinier, H. Ltaief *et al.*, "Flexible development of dense linear algebra algorithms on massively parallel architectures with DPLASMA," in *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*. IEEE, 2011, pp. 1432–1441.

[15] K. Akbudak, H. Ltaief, A. Mikhalev, and D. Keyes, "Tile low rank cholesky factorization for climate/weather modeling applications on manycore architectures," in *International Conference on High Performance Computing*. Springer, 2017, pp. 22–40.

[16] S. Abdulah, K. Akbudak, W. Boukaram, A. Charara, D. Keyes, H. Ltaief, A. Mikhalev, D. Sukkari, and G. Turkiyyah, "Hierarchical computations on manycore architectures (HiCMA)," *See http://github. com/ecrc/hicma*, 2019.

[17] J. Cao, M. G. Genton, D. E. Keyes, and G. M. Turkiyyah, "tlrmvnmvt: Computing high-dimensional multivariate normal and student-t probabilities with low-rank methods in r," *Journal of Statistical Software*, vol. 101, pp. 1–25, 2022.

[18] C. Augonnet, S. Thibault, R. Namyst, and P. Wacrenier, "StarPU: A unified platform for task scheduling on heterogeneous multicore architectures," *Concurrency Computat. Pract. Exper.*, vol. 23, pp. 187–198, 2011.

[19] N. Anceschi, A. Fasano, D. Durante, and G. Zanella, "Bayesian conjugacy in probit, tobit, multinomial probit and extensions: A review and new results," *Journal of the American Statistical Association*, no. just-accepted, pp. 1–64, 2023.

[20] A. Genz, "Numerical computation of multivariate normal probabilities," *Journal of computational and graphical statistics*, vol. 1, no. 2, pp. 141–149, 1992.

[21] A. Genz and F. Bretz, *Computation of multivariate normal and t probabilities*. Springer Science & Business Media, 2009, vol. 195.

[22] M. G. Genton, D. E. Keyes, and G. Turkiyyah, "Hierarchical decompositions for the computation of high-dimensional multivariate normal probabilities," *Journal of Computational and Graphical Statistics*, vol. 27, no. 2, pp. 268–277, 2018.

[23] G. Bosilca, A. Bouteiller, A. Danalis, M. Faverge, T. Herault, and J. Dongarra, "PaRSEC: Exploiting heterogeneity to enhance scalability," *Computing in Science Engineering*, vol. 15, no. 6, pp. 36–45, Nov 2013.

[24] A. Yarkhan, J. Kurzak, and J. Dongarra, "Quark users' guide," *Electrical Engineering and Computer Science, Innovative Computing Laboratory, University of Tennessee*, vol. 268, 2011.

[25] S. Abdulah, H. Ltaief, Y. Sun, M. G. Genton, and D. E. Keyes, "Tile low-rank approximation of large-scale maximum likelihood estimation on manycore architectures," 2018.

[26] S. Mondal, S. Abdulah, H. Ltaief, Y. Sun, M. G. Genton, and D. E. Keyes, "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*, vol. 180, p. 104715, 2023.

[27] H. Ltaief, J. Cranney, D. Gratadour, Y. Hong, L. Gatineau, and D. Keyes, "Meeting the real-time challenges of ground-based telescopes using low-rank matrix computations," in *Proceedings of the International*

Conference for High Performance Computing, Networking, Storage and Analysis, 2021, pp. 1–16.

[28] H. Ltaief, Y. Hong, L. Wilson, M. Jacquelin, M. Ravasi, and D. E. Keyes, "Scaling the "Memory Wall" for Multi-Dimensional Seismic Processing with Algebraic Compression on Cerebras CS-2 Systems." ACM/IEEE, 2023.

[29] T. Gneiting, W. Kleiber, and M. Schlather, "Matérn cross-covariance functions for multivariate random fields," *Journal of the American Statistical Association*, vol. 105, no. 491, pp. 1167–1177, 2010.

[30] S. Abdulah, H. Ltaief, Y. Sun, M. G. Genton, and D. E. Keyes, "Exageostat: A high performance unified software for geostatistics on manycore systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 12, pp. 2771–2784, 2018.

[31] P. Giani, F. Tagle, M. G. Genton, S. Castruccio, and P. Crippa, "Closing the gap between wind energy targets and implementation for emerging countries," *Applied Energy*, vol. 269, p. 115085, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261920305973

[32] W. Chen, S. Castruccio, M. G. Genton, and P. Crippa, "Current and future estimates of wind energy potential over saudi arabia," *Journal of Geophysical Research: Atmospheres*, vol. 123, no. 12, pp. 6443–6459, 2018. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2017JD028212

[33] E. González-Estrada and W. Cosmes, "Shapiro–wilk test for skew normal distributions based on data transformations," *Journal of Statistical Computation and Simulation*, vol. 89, no. 17, pp. 3258–3272, 2019.

[34] Z. I. Botev, "The normal law under linear restrictions: simulation and estimation via minimax tilting," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, pp. 125–148, 2017.

[35] D. Azzimonti and D. Ginsbourger, "Estimating orthant probabilities of high-dimensional Gaussian vectors with an application to set estimation," *Journal of Computational and Graphical Statistics*, vol. 27, no. 2, pp. 255–267, 2018.

[36] T. P. Barnett, D. W. Pierce, and R. Schnur, "Detection of anthropogenic climate change in the world's oceans," *Science*, vol. 292, no. 5515, pp. 270–274, 2001.

[37] S. Marsili-Libelli, S. Guerrizio, and N. Checchi, "Confidence regions of estimated parameters for ecological systems," *Ecological Modelling*, vol. 165, no. 2-3, pp. 127–146, 2003.