Parallel Approximations of the Tukey g-and-h Likelihoods and Predictions for Non-Gaussian Geostatistics

Sagnik Mondal[1,2], Sameh Abdulah[1,3], Hatem Ltaief[1,3], Ying Sun[1,2], 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 Thuwal 23955-6900, Kingdom of Saudi Arabia*
[2]*Statistics Program, King Abdullah University of Science and Technology*
[3]*Extreme Computing Research Center, King Abdullah University of Science and Technology*

2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS) | 978-1-6654-8106-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/IPDPS53621.2022.00044

***Abstract*—Maximum likelihood estimation is an essential tool in the procedure to impute missing data in climate/weather applications. By defining a particular statistical model, the maximum likelihood estimation can be used to understand the underlying structure of given geospatial data. The Gaussian random field has been widely used to describe geospatial data, as one of the most popular models under the hood of maximum likelihood estimation. Computation of Gaussian log-likelihood demands operations on a dense symmetric positive definite matrix, often parameterized by the Matérn correlation function. This computation of the log-likelihood requires $\mathcal{O}(n^2)$ storage and $\mathcal{O}(n^3)$ operations, which can be a huge task considering that the number of geographical locations, n, now commonly reaches into the millions. However, despite its appealing theoretical properties, the assumptions of Gaussianity may be unrealistic since real data often show signs of skewness or have some extreme values. Herein, we consider the Tukey g-and-h (TGH) random field as an example of a non-Gaussian random field that shows more robustness in modeling geospatial data by including two more parameters to incorporate skewness and heavy tail features in the model. This work provides the first HPC implementation of the TGH random field's inference on parallel hardware architectures. Using task-based programming models associated with dynamic runtime systems, our implementation leverages the high concurrency of current parallel systems. This permits to run the exact log-likelihood evaluation of the Tukey g-and-h (TGH) random fields for a decent number of geospatial locations. To tackle large-scale problems, we provide additionally an implementation of the given model using two different low-rank approximations. We compress the aforementioned positive-definite symmetric matrix for computing the log-likelihood and rely on the Tile Low-Rank (TLR) and the Hierarchical Off-Diagonal Low-Rank (HODLR) matrix approximations. We assess the performance and accuracy of the proposed implementations using synthetic datasets up to $800K$ and a $300K$ precipitation data of Germany to demonstrate the advantage of using non-Gaussian over Gaussian random fields. Moreover, by relying on TLR/HODLR matrix computations, we can now solve for larger matrix sizes while preserving the required accuracy for prediction. We show the performance superiority of TLR over HODLR matrix computations when calculating the TGH likelihoods and predictions. Our TLR-based approximation shows a speedup up to $7.29X$ and $2.96X$ on shared-memory and distributed-memory systems, respectively, compared to the exact implementation.**

I. Introduction

Techniques relying on latent Gaussian models are widely used to make spatial predictions beyond the observed geographical locations. The target modeling process involves obtaining a set of statistical parameters with the aid of the likelihood function through the Maximum Likelihood Estimation (MLE) method in order to impute missing spatial data. MLE operates on a dense covariance matrix with dimension $n \times n$ where n represents the number of geospatial locations. The challenge for high-dimensional problems lies in the computation requirements of the log-likelihood function, which necessitates $\mathcal{O}(n^3)$ computation and $\mathcal{O}(n^2)$ memory space, making its computation prohibitive for large n. Techniques exist to tailor the modeling process to reduce the computing cost by order of magnitude. The low-rank approximation has been shown to be necessary for the closely related problem of the spatial statistics framework [1]–[4]. However, most of the existing studies evaluate the effectiveness of the low-rank approximation with the Gaussian process, which is limited in real-life datasets.

In typical applications, the spatial variables are non-Gaussian, where skewness and heavy tails can be captured. A common approach to deal with this non-Gaussian behavior is to apply a non-linear transformation to the non-Gaussian data such as square-root transform [5]–[7], Box–Cox transform [8] and the log-normal transform [9] which is known as the trans-Gaussian approach. Ideally, transforming non-Gaussian data using one of these techniques should be enough to apply Gaussian modeling directly to the data. However, it is impossible to find a suitable transformation technique that can give the best modeling performance for arbitrary datasets.

In [10], a highly flexible trans-Gaussian model has been proposed under the name Tukey g-and-h (TGH) random fields. This work presents a parallel implementation of the TGH random fields in the context of climate and environmental applications. We rely on a task-based programming model to provide the TGH modeling and prediction operations in the exact form where the dense linear algebra operations

1530-2075/22/$31.00 ©2022 IEEE
DOI 10.1109/IPDPS53621.2022.00044

have been drawn from the Chameleon library [11] and the underlying tasks schedule managed by the StarPU runtime system [12]. We also provide two low-rank implementations of the TGH model: TLR-based, where the compression and linear algebra operations run through the HiCMA library [13], and Hierarchical Off-Diagonal Low-Rank (HODLR)-based approximations [14], where the compression and linear algebra operations run through HODLRLIB [15]. The TLR-based implementation shows a speedup up to $7.29X$ and $2.96X$ on shared-memory and distributed-memory systems, respectively, compared to exact implementation.

Our main contributions are as follows: 1) We propose a parallel implementation of the TGH likelihoods and predictions to model a wide range of real spatial datasets on shared-memory and distributed-memory systems; 2) We provide parallel TLR-based and HODLR-based approximations to the exact predictive model to reduce the prohibitive complexity of the underlying algebra operations of the associated dense covariance matrix; 3) We show the effectiveness of the non-Gaussian scheme provided by our implementation compared with the traditional Gaussian modeling with both synthetic and real datasets; 4) We provide a comparison in the parameters' estimation accuracy to produce the required accuracy for both TLR-based and HODLR-based approximation; 5) We assess the performance of exact/TLR/HODLR non-Gaussian modeling and prediction on shared-memory and distributed-memory systems; and 6) We demonstrate the quality of our implementation in a $300K$ precipitation dataset of Germany to show that we are able to provide a robust predictive model to work in such a non-Gaussian dataset.

The remainder of the paper is organized as follows. Section II covers related work. Section III gives an overview and background of our problem. Section IV describes in detail the algorithm and our parallel implementation. Section V analyzes accuracy and performance using synthetic and real datasets in the context of climate/weather applications. We conclude in Section VI.

II. Related Work

In this section, we discuss some of the popular approaches for constructing non-Gaussian random fields. Use of various non-Gaussian distributions such as skew-Gaussian distribution (Kim and Mallick [16]; Zhang and El-Shaarawi [17]; Genton and Zhang [18], Rimstad and Omre [19]), t-distribution (Røislien and Omre [20]), skew-t distribution (Bevilacqua et al. [21]), log-skew-elliptical distribution (Marchenko and Genton [22]) can create different non-Gaussian random fields. Palacios and Steel [23] and Fonseca and Steel [24] provided non-Gaussian models for spatial data by scale mixing the Gaussian random field. Gräler [25] and Krupskii et al. [26] made non-Gaussian random fields using copula. Wallin and Bolin [27] provided a class of non-Gaussian spatial Matérn fields using stochastic partial differential equations. Another popular technique to model non-Gaussian spatial data, known as trans-Gaussian random field, is applying a non-linear transformation of the original data such that the transformed data become Gaussian. A few well-used non-linear conversions are log-normal (De Oliveira [28]), square-root (Johns et al. [29]), Box-Cox (De Oliveira et al. [30]), and power transformations (Allcroft and Glasbey [31]). However, it may be challenging to find this transformation in some situations, if not impossible. Xu and Genton [10] proposed Tukey g-and-h (TGH) random fields based on the Tukey's g-and-h transformation. It is a more flexible trans-Gaussian random field than the existing trans-Gaussian models. Due to its many appealing statistical properties and its easily interpretable parameterization, we select the TGH model for the large-scale implementation.

The computational challenges for fitting the Gaussian and TGH models are similar. The exact evaluation of the log-likelihood for both models requires dense matrix inversion. The task of this matrix inversion can be very challenging when the size of the matrix is large, i.e., when the number of geographical locations of the problem is high. A parallel computing framework can help us fasten the computation time for these models. The task of spatial prediction (or kriging) has already been implemented for the Gaussian model in a parallel computing framework. For example, the authors in [32], [33], [34] have implemented kriging for Gaussian model in parallel computing frameworks using Message Passing Interface (MPI), OpenMP, Parallel Virtual Machines (PVMs), and/or Graphics Processing Units (GPUs). Abdulah et al. [35] first provided a framework in parallel computing for the exact computation of the log-likelihood of the Gaussian model using dense linear algebra task-based algorithms and dynamic runtime systems.

Another approach to tackle the computation and storage complexity is to use some approximation techniques for the dense matrix. The authors in [36], [37] used a method named covariance tapering for approximation of the large covariance matrix. Other approximation techniques include low-rank approximations. For instance, the authors in [3] and [38] used Tile Low-Rank (TLR) and Hierarchically Off-Diagonal Low-Rank (HODLR) approximations of the covariance matrix for computation of the log-likelihood of the Gaussian model.

III. Background and Overview of the Problem

In this section, we formally define the Tukey g-and-h random field and its log-likelihood function. Subsequently, we provide the kriging equation and an assessment tool for the TGH modeling.

A. Definition of Tukey g-and-h Random Fields

Tukey's g-and-h transformation

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

is a monotonic function of z for $g \in \mathbb{R}$ and $h \geq 0$. Here and from now on, the values of any quantities involving g at $g = 0$ are defined as their limit when $g \to 0$.

The Tukey g-and-h (TGH) random field with location parameter $\xi \in \mathbb{R}$ and scale parameter $\omega > 0$ is defined as

$$T(\boldsymbol{s}) = \xi + \omega\tau_{g,h}\{Z(\boldsymbol{s})\}, \tag{2}$$

where, $Z(\boldsymbol{s})$, $\boldsymbol{s} \in \mathbb{R}^d$, $d \geq 1$ is a standard Gaussian random field, i.e. $\mathbb{E}\{Z(\boldsymbol{s})\} = 0$ and $\mathbb{V}\text{ar}\{Z(\boldsymbol{s})\} = 1$, with some correlation function $\text{corr}\{Z(\boldsymbol{s}_1), Z(\boldsymbol{s}_2)\} = \rho_Z(\boldsymbol{s}_1, \boldsymbol{s}_2)$. The two parameters g and h dictate the extent of skewness and kurtosis in the marginal distribution and the sign of g directs the sign of the skewness of the marginal distribution producing a family of random fields with very flexible marginal distribution. The TGH random field includes a large family of trans-Gaussian random fields. For example, when $g = h = 0$, $T(\boldsymbol{s})$ becomes a Gaussian random field, for $g > 0$ and $h = 0$, $T(\boldsymbol{s})$ becomes a shifted log-Gaussian random field and for $g = 0$ and $h > 0$, $T(\boldsymbol{s})$ is a random field with a Pareto-like marginal distribution.

B. Log-likelihood of Tukey g-and-h Random Fields

Let $\boldsymbol{\theta}_1 = (\xi, \omega, g, h)^\top$ and $\boldsymbol{\theta}_2$ be the parameter vector corresponding to $\rho_Z(\boldsymbol{s}_1, \boldsymbol{s}_2)$, the correlation function of $Z(\boldsymbol{s})$ in (2), and let $\boldsymbol{\theta} = (\boldsymbol{\theta}_1^\top, \boldsymbol{\theta}_2^\top)^\top$. Consider a dataset $\mathcal{D} = \{t(\boldsymbol{s}_1), \ldots, t(\boldsymbol{s}_n)\}$ collected from the TGH random field, $T(\boldsymbol{s})$, at locations $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$. The log-likelihood function of $\boldsymbol{\theta}$ given the dataset $\mathcal{D}$ is

$$\begin{aligned} L(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 | \mathcal{D}) \propto & -\frac{1}{2}\{\boldsymbol{Z}_{\boldsymbol{\theta}_1}^\top (\boldsymbol{R}_{\boldsymbol{\theta}_2}^{-1} + h\boldsymbol{I}_n)\boldsymbol{Z}_{\boldsymbol{\theta}_1} + \log|\boldsymbol{R}_{\boldsymbol{\theta}_2}|\} \\ & - \sum_{i=1}^{n} \log[\exp(g z_{\boldsymbol{\theta}_1, \boldsymbol{s}_i}) + g^{-1}\{\exp(g z_{\boldsymbol{\theta}_1, \boldsymbol{s}_i}) \\ & - 1\} h z_{\boldsymbol{\theta}_1, \boldsymbol{s}_i}] - n \log \omega, \end{aligned} \tag{3}$$

where $z_{\boldsymbol{\theta}_1, \boldsymbol{s}_i} = \tau_{g,h}^{-1}\left\{\dfrac{t(\boldsymbol{s}_i) - \xi}{\omega}\right\}$, $\boldsymbol{Z}_{\boldsymbol{\theta}_1} = (z_{\boldsymbol{\theta}_1, \boldsymbol{s}_1}, \ldots, z_{\boldsymbol{\theta}_1, \boldsymbol{s}_n})^\top$ and $(\boldsymbol{R}_{\boldsymbol{\theta}_2})_{i,j} = \rho_Z(\boldsymbol{s}_i, \boldsymbol{s}_j)$, i, j=1, …, n. We estimate $\boldsymbol{\theta}$ by maximizing the log-likelihood in (3).

C. Kriging with Tukey g-and-h Random Fields

One of the most widely used geostatistical techniques is kriging. In kriging, the objective is to find an optimal point estimator of the process under consideration at a new location by minimizing some loss function. For TGH random fields, the best kriging predictor of $T(\boldsymbol{s}_0)$ under squared error loss function is

$$\begin{aligned} \widehat{T}(\boldsymbol{s}_0) = & \widehat{\xi} + \frac{\widehat{\omega}}{\widehat{g}\sqrt{1 - \widehat{h}\tilde{\sigma}^2}} \exp\left\{\frac{\widehat{h}\tilde{\mu}^2}{2(1 - \widehat{h}\tilde{\sigma}^2)}\right\} \\ & \times \left[\exp\left\{\frac{\widehat{g}^2\tilde{\sigma}^2 + 2\widehat{g}\tilde{\mu}}{2(1 - \widehat{h}\tilde{\sigma}^2)}\right\} - 1\right], \end{aligned} \tag{4}$$

where $\tilde{\mu} = \boldsymbol{r}_{\widehat{\boldsymbol{\theta}}_2}^\top \boldsymbol{R}_{\widehat{\boldsymbol{\theta}}_2}^{-1} \boldsymbol{Z}_{\widehat{\boldsymbol{\theta}}_1}$, $\tilde{\sigma}^2 = 1 - \boldsymbol{r}_{\widehat{\boldsymbol{\theta}}_2}^\top \boldsymbol{R}_{\widehat{\boldsymbol{\theta}}_2}^{-1} \boldsymbol{r}_{\widehat{\boldsymbol{\theta}}_2}$ and $\boldsymbol{r}_{\widehat{\boldsymbol{\theta}}_2} = \{\rho_Z(\boldsymbol{s}_0, \boldsymbol{s}_1), \ldots, \rho_Z(\boldsymbol{s}_0, \boldsymbol{s}_n)\}^\top$ and $\widehat{\boldsymbol{\theta}} = (\widehat{\boldsymbol{\theta}}_1^\top, \widehat{\boldsymbol{\theta}}_2^\top)^\top$ is the MLE of $\boldsymbol{\theta}$.

D. Matérn Correlation Function

For constructing the correlation matrix of $Z(\boldsymbol{s})$ in (2), we need some valid correlation function to ensure the symmetric positive-definite structure of the correlation matrix. We use the Matérn correlation function here because of its high flexibility. The Matérn correlation function is defined as

$$\rho_Z(h) = \frac{1}{\Gamma(\nu) 2^{\nu-1}} \left(4\sqrt{2\nu}\frac{h}{\phi}\right)^{\nu} \mathcal{K}_\nu \left(4\sqrt{2\nu}\frac{h}{\phi}\right), \tag{5}$$

where $h = \|\boldsymbol{s}_1 - \boldsymbol{s}_2\|$ is the distance between locations $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, $\nu > 0$ is the smoothness parameter, $\phi > 0$ is the range parameter, $\Gamma(\cdot)$ is the gamma function, and $\mathcal{K}_\nu(\cdot)$ is the modified Bessel function of the second kind of order ν. The smoothness parameter ν, as the name suggests, dictates the smoothness of the random field and the range parameter ϕ controls how quickly the correlation of the random field decreases with distance. Many popular correlation functions come under the Matérn correlation family. For example, when $\nu = 0.5$, the Matérn correlation function becomes the exponential correlation function $\rho(h) = \exp(-4\sqrt{2\nu}h/\phi)$, when $\nu = 1$, it becomes the Whittle correlation function $\rho(h) = \left(4\sqrt{2\nu}h/\phi\right)\mathcal{K}_1\left(4\sqrt{2\nu}h/\phi\right)$, when $\nu = \infty$ it becomes the Gaussian correlation function $\rho(h) = \exp\{-(4\sqrt{2\nu}h/\phi)^2\}$.

E. Assessment of Fitted TGH Model Using PIT

The probability integral transformation (PIT) can be a useful tool to assess whether the data are originally emulating a TGH random field or not. Suppose we fit a TGH model to a dataset $\mathcal{D} = \{t(\boldsymbol{s}_1), \ldots, t(\boldsymbol{s}_n)\}$, assuming it is collected from a TGH random field $T(\boldsymbol{s})$, defined in (2), at locations $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$. Then, thanks to Theorem 4 in [10], the estimated conditional distribution function of $T(\boldsymbol{s}_0)$ given $\mathcal{D}$ is

$$\widehat{F}_{\boldsymbol{s}_0}(t|\mathcal{D}) = \Phi\left(\frac{z - \tilde{\mu}}{\tilde{\sigma}}\right), \tag{6}$$

where $z = \tau_{\widehat{g},\widehat{h}}^{-1}\{(t - \widehat{\xi})/\widehat{\omega}\}$, $\Phi(\cdot)$ is the distribution function of a standard Gaussian distribution, and $\tilde{\mu}$ and $\tilde{\sigma}^2$ are defined in (4). If the dataset is derived from a TGH random field, the probability integral transform of $T(\boldsymbol{s}_0)$, i.e., $F_{\boldsymbol{s}_0}\{T(\boldsymbol{s}_0)\}$ will be an observation from a uniform distribution over $(0, 1)$. The validation procedure can be done by dividing the data into training and testing. If the data are really coming from a TGH random field, the probability integral transforms of the testing data, transformed by the estimated distribution function with the training data, should have an approximate uniform distribution over $(0, 1)$.

F. Task-based parallelism and Runtime systems

Matrix operations are the core of the log-likelihood function, where matrix inversion is the most time-consuming operation with cubic complexity. The literature reveals two types of matrix algorithms, i.e., block-based and tile-based, to perform the inversion operation in parallel. The block-based algorithms decompose the target matrix into a successive panel and update computational phases. Updating the matrix has two steps: panel factorization step and update step. LAPACK is an example of a block-based linear algebra library. The matrix is split into a set of tiles with a predetermined tile size in the tile-based algorithms. These algorithms weaken the synchronization points that existed before with block-based algorithms between the panel and update computational phases and allow translating the numerical algorithm into a Directed Acyclic Graph (DAG), where the nodes represent tasks, and the edges define data dependencies. PLASMA [39]

and Chameleon [40] are two examples of tile-based linear algebra libraries.

Representing the numerical operation as a set of tasks allows exploiting the task-based parallelism technique to execute the numeric function in parallel and over a heterogeneous group of distributed resources. In this case, a dynamic runtime system such as OmpSs [41], OpenMP [42], and PaRSEC [43], StarPU [12] can be employed to run the tasks across different hardware resources while ensuring the integrity of data dependencies. In this work, we rely on StarPU, which is preferred for its wide hardware architecture support. StarPU also supports many scheduling techniques that can help in porting it with adequate performance [44].

G. Hierarchical Matrices

Hierarchical matrices have been widely studied in the literature to tackle the challenges related to a large class of applications (e.g., Gaussian processes, electromagnetic integral equations [45], and Bayesian inversion). These applications are related to generating dense matrices with a data sparsity structure. In this paper, we focus on HODLR and TLR for computing MLE. A detailed overview of these methods are given in the following subsections.

1) Hierarchically Off-Diagonal Low-Rank Approximations: Hierarchically Off-Diagonal Low-Rank (HODLR) matrices provide a hierarchical representation of low-rank off-diagonal blocks with recursive definition 2×2 blocks of a given matrix $\boldsymbol{C}$,

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{C}_{11} & \boldsymbol{C}_{12} \\ \boldsymbol{C}_{21} & \boldsymbol{C}_{22} \end{bmatrix}, \tag{7}$$

where $\boldsymbol{C}_{12}$ and $\boldsymbol{C}_{21}$ are in low-rank representation, and $\boldsymbol{C}_{11}$ and $\boldsymbol{C}_{22}$ recursively represent the HODLR matrix. Once the diagonal blocks reach a given leaf size, the recursion stops after t steps, where t is the level of the HODLR matrix. HODLR matrices require representing all off-diagonal blocks in low-rank. Fig. 1 illustrates how to construct a HODLR matrix of level 3. Dense blocks are represented in red, and the low-rank blocks are represented in green. The final structure of the matrix is shown on the right side of the figure.

In this work, we employ HODLRLIB, a parallel HODLR-based library, to perform the required matrix operations in the MLE algorithm. HODLRLIB relies on OpenMP multithreaded BLAS for parallel performance.

2) Tile Low-Rank Approximations: Tile Low-Rank (TLR) approximation [46], [47] is a flat approach that consists in splitting the dense matrix into tiles of similar sizes. Fig. 2 shows an example of compressing an off-diagonal tile $\boldsymbol{T}_{12}$ to two matrices $\boldsymbol{U}_{12}$ and $\boldsymbol{V}_{12}$, where the Singular Value Decomposition (SVD) is used to compress the dense matrix. The most significant k singular values are captured with their associated vectors, which correspond to the rank of the tile. Once each tile is compressed, the tile-based algorithm is expressed in terms of tasks interconnected by their data dependencies. The original algorithm can then be translated into a Directed Acyclic Graph (DAG), where nodes are fine-grained computational tasks, and edges express their data dependencies. As implemented in the HiCMA library [13], the StarPU [12] dynamic runtime system is used to orchestrate the scheduling of tasks with their data dependencies onto processing units. The task-based programming model creates opportunities for look-ahead, which enables to maintain high hardware occupancy.

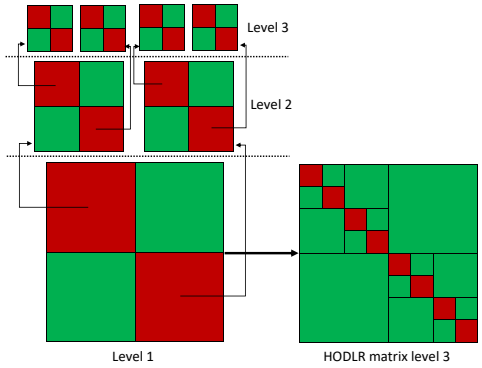


Fig. 1: Recursive construction of HODLR matrix of level 3. The red blocks are represented in dense format at all levels, while the green blocks are represented in low-rank format.

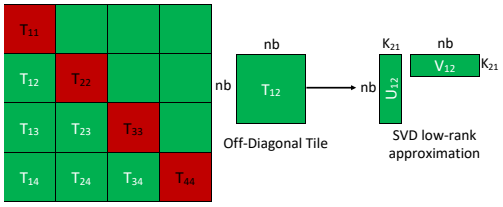


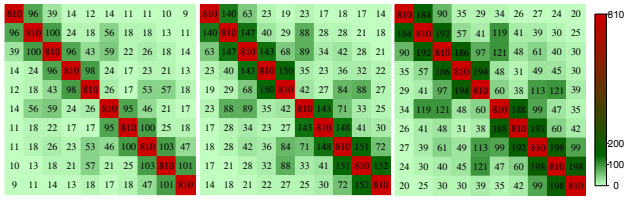
Fig. 2: An example of TLR approximation tile: diagonal tiles (in red) are dense. Off-diagonal tiles (in green) are represented as low-rank approximation.

IV. Parallel TGH Modeling and Prediction

This section explains our proposed parallel implementation of the TGH likelihoods and predictions in exact, tile low-rank (TLR) approximation and hierarchical off-diagonal low-rank (HODLR) approximation format. We show the distribution of the ranks in both TLR and HOLDR matrices associated with TGH random fields compared to the dense matrix.

A. Non-Gaussian Log-Likelihood Estimation

Recalling the TGH random field log-likelihood function in (3), the log-likelihood estimation process involves generating a covariance matrix $\boldsymbol{R}_{\boldsymbol{\theta}_2}$ where $\boldsymbol{\theta}_2$ is the parameters of a given correlation function. Given a set of n geospatial locations, a selected covariance function can be used to build an $n \times n$ covariance matrix. In this work, we use the parametrizable Matérn correlation function with two parameters: ϕ, the spatial range parameter, and ν, the random field smoothness parameter. The other input parameter vector $\boldsymbol{\theta}_1 = (\xi, \omega, g, h)^\top$



(a) 10^{-5}. (b) 10^{-7}. (c) 10^{-9}.

Fig. 3: Rank distributions of a $8,100 \times 8,100$ covariance TLR matrix using nb = 810 with Matérn parameters $\boldsymbol{\theta}_2 = (0.96, 0.5)^\top$ under different accuracy levels.

is used to transform the non-Gaussian measurement vector $\boldsymbol{Z}$ to a Gaussian vector $\boldsymbol{Z}_{\boldsymbol{\theta}_1}$.

Algorithm 1 TGH Log-Likelihood Estimation

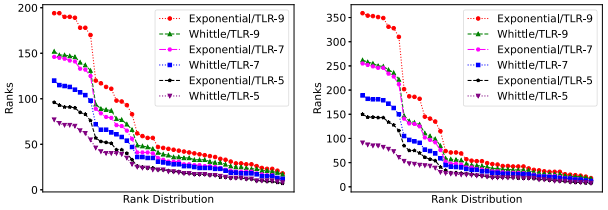
1: **Input:** a set of locations $\boldsymbol{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\}$, associated measurements $t(\boldsymbol{s}_1), \ldots, t(\boldsymbol{s}_n)$, and current parameter vector $\boldsymbol{\theta}_1 = (\xi, \omega, g, h)^\top$.
2: **Output:** the log-likelihood estimation for the current parameter vector $\boldsymbol{\theta}_1$
3: TGH transformation for $z_{\boldsymbol{\theta}_1, \boldsymbol{s}_i} \leftarrow \tau_{g,h}^{-1}\left\{\frac{t(\boldsymbol{s}_i) - \xi}{\omega}\right\} \rightarrow$ eq(1), and $\boldsymbol{Z}_{\boldsymbol{\theta}_1} = (z_{\boldsymbol{\theta}_1, \boldsymbol{s}_1}, \ldots, z_{\boldsymbol{\theta}_1, \boldsymbol{s}_n})^\top$.
4: Generate covariance matrix $\boldsymbol{R}_{\boldsymbol{\theta}_2} \rightarrow$ eq(5)
5: POTRF($\boldsymbol{R}_{\boldsymbol{\theta}_2}$) $\rightarrow$ Cholesky factorization
6: $determinant$ = Det($\boldsymbol{R}_{\boldsymbol{\theta}_2}$)
7: $S = \sum_{i=1}^{n} (\log[\exp(g z_{\boldsymbol{\theta}_1, \boldsymbol{s}_i}) + g^{-1}\{\exp(g z_{\boldsymbol{\theta}_1, \boldsymbol{s}_i}) - 1\} h z_{\boldsymbol{\theta}_1, \boldsymbol{s}_i}] + h z^2_{\boldsymbol{\theta}_1, \boldsymbol{s}_i}) + n \log \omega$,
8: TRSM ($\boldsymbol{R}_{\boldsymbol{\theta}_2}$, $\boldsymbol{Z}_{\boldsymbol{\theta}_1}$) $\rightarrow$ Triangular solve
9: $dotproduct$ = ($\boldsymbol{Z}_{\boldsymbol{\theta}_1} \times \boldsymbol{Z}_{\boldsymbol{\theta}_1}$)
10: $llh = -\frac{1}{2}\{dotproduct + \log(determinant)\}$ - S.

Algorithm 1 presents the TGH log-likelihood estimation algorithm. The $\boldsymbol{Z}_{\boldsymbol{\theta}_1}$ vector transformation step is performed first (line 1) with a given $\boldsymbol{\theta}_1$ parameter vector. Since the $\tau_{g,h}^{-1}(\cdot)$ has no closed form, we use the Newton-Raphson method to approximate the function for $\boldsymbol{Z}_{\boldsymbol{\theta}_1}$ as follows:

$$\tau_{g,h}^{-1}(z_{\boldsymbol{\theta}_1, \boldsymbol{s}_i}) = \frac{t(\boldsymbol{s}_i) - \xi}{\omega}. \tag{8}$$

The most time-consuming step in Algorithm 1 is the Cholesky factorization of the covariance matrix $\boldsymbol{R}_{\boldsymbol{\theta}_2}$ in line 3. Therefore, we rely on the state-of-the-art dense task-based library, i.e., Chameleon [40], to perform the linear algebra operations that appear in Algorithm 1, including the Cholesky factorization of the covariance matrix. Furthermore, we exploit the data sparsity structure of the covariance matrix and apply low-rank approximation methods to reduce the overall complexity of the underlying linear algebra operations. In particular, we use TLR and HODLR matrix approximations to speed up the computation of the Cholesky factorization heavyweight operation while preserving the accuracy requirement of the application.

1) TLR Approximation of $\boldsymbol{R}_{\boldsymbol{\theta}_2}$: The TLR approximation in the HiCMA library compresses the individual tile using the SVD algorithm, where the ranks of the tiles represent



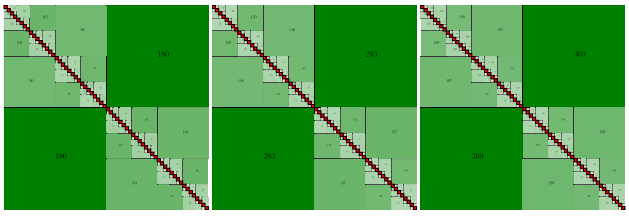
(a) $N = 8,100$ / $ts = 810$. (b) $N = 32,400$ / $ts = 3240$.

Fig. 4: Decay of the ranks of different blocks with TLR approximation when using Matérn kernel with range $\phi = 0.96$ and smoothness $\nu = 0.5$ (Exponential kernel), and $\nu = 1$ (Whittle kernel) and two matrix sizes.

the most significant singular values and vectors in each off-diagonal tile [46]. Therefore, the effectiveness of the TLR mechanism depends on the ranks of the off-diagonal tiles after compression, which in turn depends on the application's accuracy requirements. Thus, we initially validate the potency of the TLR approximation with the TGH modeling by estimating the ranks corresponding to different accuracy levels, namely, TLR-5 (10^{-5}), TLR-7 (10^{-7}), and TLR-9 (10^{-9}). The required accuracy level of our application and the corresponding performance assessment is shown in detail in the performance section. Herein, we just validate the effectiveness of using TLR with our model.

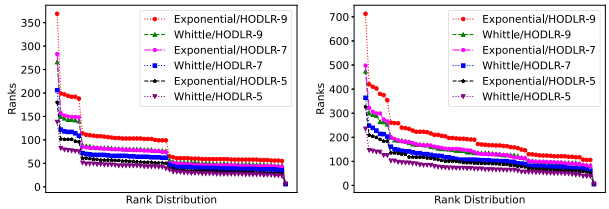
Fig. 3 depicts the rank distribution of a $8,100 \times 8,100$ covariance matrix generated by the Matérn covariance function shown in (5). As can be seen, the ranks of the off-diagonal tiles grow as the tiles get closer to the diagonal with different TLR accuracy levels with a monotonic increase of the ranks with tighter tolerance. However, even with TLR-9, the ranks are still smaller than the full dense tiles in the diagonal. The given example is drawn from a synthetic set of Matérn parameters $\boldsymbol{\theta}_2 = (0.96, 0.5)^\top$, representing a medium correlation dependence between the given spatial locations. We also examine other spatial correlation strengths, and all of them show almost the same ranks with different accuracy levels. We only observe a change in the ranks with different smoothness values. For instance, in Fig. 4, we report the decay of the ranks when using TLR approximation with different datasets with two data sizes, i.e., $8,100$ and $32,400$ using different TLR accuracy levels and a synthetic set of parameters $\boldsymbol{\theta}_2 = (0.96, \nu)^\top$, where ν represents a rough field ($\nu = 0.5$) and a smooth field ($\nu = 1$). As shown, the ranks associated with rough fields have higher ranks than smooth fields across all accuracy levels and the two data sizes. The decaying figure also shows an increase in ranks when tile size is larger but preserving the same decaying rate. Furthermore, we examined the ranks with different data sizes and the same tile sizes. We did not observe any dramatic changes in ranks or ranks decaying. All the above observations show the advantage of using the TLR approximation in TGH modeling from the performance perspective.

2) HODLR Approximation of $\boldsymbol{R}_{\boldsymbol{\theta}_2}$: Herein, we used also three accuracy levels, namely, HODLR-5 (10^{-5}), HODLR-7 (10^{-7}), and HODLR-9 (10^{-9}). Fig. 5 depicts the rank



(a) 10^{-5}. (b) 10^{-7}. (c) 10^{-9}.

Fig. 5: Rank distributions of a $8,100 \times 8,100$ covariance HODLR matrix using leaf $= 100$ with Matérn parameters $\boldsymbol{\theta}_2 = (0.96, 0.5)^\top$ under different accuracy levels.



(a) $N = 8,100$ / $leaf = 100$. (b) $N = 32,400$ / $leaf = 400$.

Fig. 6: Decay of the ranks of different blocks with HODLR approximation when using Matérn kernel with range $\phi = 0.96$ and smoothness $\nu = 0.5$ (Exponential kernel), and $\nu = 1$ (Whittle kernel) and two matrix sizes.

distribution of a $8,100 \times 8,100$ covariance matrix. Herein, we also use the medium correlation case to show the rank distribution. As shown by the figure, the matrix is divided into blocks in a hierarchical structure where the $leaf$ represents the size of matrices at leaf level. It is clearly shown that the smaller blocks have lower ranks, and at the higher structure levels of the matrix, the ranks increase. In red, the diagonal leaf blocks are kept dense. Furthermore, we do not observe any noticeable growth in ranks with larger N when using the HODLR structure. Fig. 6 shows the decay of the ranks in the case of HODLR matrices. The ranks drop faster than the TLR approximation because HODLR has smaller blocks at lower levels. We choose the size of leaf blocks: 100 and 400 with two matrix sizes. With larger leaf block sizes, the ranks increase. As the TLR approximation, increasing the matrix size does not affect the ranks but compresses more blocks. The figure also shows the impact of the smoothness parameter ν, which produces higher ranks in the rough field.

B. TGH Prediction

The optimization process of the likelihood function aims at tuning the TGH parameter vectors, i.e., $\widehat{\boldsymbol{\theta}}_1$ and $\widehat{\boldsymbol{\theta}}_2$. These parameters are used to predict missing measurements at a set of locations using Equation (4). Algorithm 2 shows in detail the numerical steps to predict missing values using the two tuned parameter vectors. Line 4 accounts for most of the algorithmic complexity as it involves the Cholesky-based solver of the covariance matrix. This algorithm performs prediction of the missing data on the location $\boldsymbol{s}_k$.

Algorithm 2 TGH Prediction

1: **Input:** a set of locations $\boldsymbol{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\}$ and associated measurements $t(\boldsymbol{s}_1), \ldots, t(\boldsymbol{s}_n)$, and estimated parameter vector $\widehat{\boldsymbol{\theta}}$.
2: **Output:** the predicted values $\widehat{T}(\boldsymbol{s}_{01}), \ldots, \widehat{T}(\boldsymbol{s}_{0n_{new}})$ at new locations $\boldsymbol{s}_{01}, \ldots, \boldsymbol{s}_{0n_{new}}$.
3: TGH transformation for $z_{\widehat{\boldsymbol{\theta}}_1, \boldsymbol{s}_i} \leftarrow \tau^{-1}_{g,h}\left\{\frac{t(\boldsymbol{s}_i) - \widehat{\xi}}{\widehat{\omega}}\right\}$ and $\boldsymbol{Z}_{\widehat{\boldsymbol{\theta}}_1} = (z_{\widehat{\boldsymbol{\theta}}_1, \boldsymbol{s}_1}, \ldots, z_{\widehat{\boldsymbol{\theta}}_1, \boldsymbol{s}_n})^\top$
4: Generate covariance matrix $\boldsymbol{R}_{\widehat{\boldsymbol{\theta}}_2} \rightarrow$ eq(5)
5: Generate covariance vector $\boldsymbol{r}_{\widehat{\boldsymbol{\theta}}_2} \rightarrow$ eq(4)
6: POSV $(\boldsymbol{R}_{\widehat{\boldsymbol{\theta}}_2}, \boldsymbol{Z}_{\widehat{\boldsymbol{\theta}}_1}) \rightarrow$ System of linear equations solver.
7: CPY $(\boldsymbol{r}_{\widehat{\boldsymbol{\theta}}_2}, \boldsymbol{rcpy}_{\widehat{\boldsymbol{\theta}}_2})$
8: TRSM $(\boldsymbol{R}_{\widehat{\boldsymbol{\theta}}_2}, \boldsymbol{r}_{\widehat{\boldsymbol{\theta}}_2}) \rightarrow$ Triangular solve
9: TRSM $(\boldsymbol{R}^\top_{\widehat{\boldsymbol{\theta}}_2}, \boldsymbol{r}_{\widehat{\boldsymbol{\theta}}_2}) \rightarrow$ Triangular solve
10: $\boldsymbol{tmp_1}$ = GEMV$(\boldsymbol{rcpy}_{\widehat{\boldsymbol{\theta}}_2}, \boldsymbol{Z}_{\widehat{\boldsymbol{\theta}}_1}) \rightarrow$ Matrix-vector multiplication
11: $\boldsymbol{tmp_2}$ = GEMM$(\boldsymbol{rcpy}_{\widehat{\boldsymbol{\theta}}_2}, \boldsymbol{r}_{\widehat{\boldsymbol{\theta}}_2}) \rightarrow$ Matrix-matrix multiplication
12: **for** $k = 1$ to n_{new} **do**
13: $\tilde{\mu} = \boldsymbol{tmp_1}[k]$
14: $\tilde{\sigma}^2 = 1 - \boldsymbol{tmp_2}[k]$
15: $\widehat{T}(\boldsymbol{s}_{0k}) = \widehat{\xi} + \frac{\widehat{\omega}}{\widehat{g}\sqrt{1-\widehat{h}\tilde{\sigma}^2}} \exp\{\frac{\widehat{h}\tilde{\mu}^2}{2(1-\widehat{h}\tilde{\sigma}^2)}\} \times [\exp\{\frac{\widehat{g}^2\tilde{\sigma}^2 + 2\widehat{g}\tilde{\mu}}{2(1-\widehat{h}\tilde{\sigma}^2)}\} - 1]$
16: **end for**

Using low-rank approximation in the form of TLR or HODLR for both $\boldsymbol{R}_{\widehat{\boldsymbol{\theta}}_2}$ and $\boldsymbol{r}_{\widehat{\boldsymbol{\theta}}_2}$ in Algorithm 2 can help in reducing the complexity of the TGH prediction algorithm similar to the likelihood estimation operation. In the following section, we give a detailed performance assessment of different proposed TGH implementations.

V. Performance Results

This section assesses the accuracy and performance of different proposed TGH implementations using synthetic and real precipitation datasets with different sizes and characteristics.

A. Testbed and Methodology

The assessment of the exact and approximate implementations of TGH modeling and prediction has been conducted on Intel and AMD chips to highlight our software portability: a 28-core dual-socket Intel Xeon IceLake Gold 6330 CPU running at 2.00 GHz, and a 64-core dual-socket AMD EPYC Milan 7713 CPU running at 2.00 GHz. For the distributed-memory experiments, we use KAUST Shaheen-II, a Cray XC40 system with $6,174$ dual-socket compute nodes based on 16-core Intel Haswell processors running at 2.3 GHz. Each node has 128 GB of DDR4 memory. The system has a total of $197,568$ processor cores and 790 TB of aggregate memory.

Our implementation enables performance portability as long as an optimized BLAS/LAPACK library is available on the target system. We rely on Intel MKL v2020.0.166 as the optimized BLAS/LAPACK library to link against the necessary optimized numerical routines for our implementations on our testbed systems. We compile our exact and TLR code using Chameleon and HiCMA linear algebra libraries with GCC V11.1, HWLOC v2.5.0, StarPU v1.3.8, GSL v2.7, and NLopt v2.6.2 optimization libraries. For HODLR-based implementation, we use HODLRLIB, compiled it with GCC

V11.1, and linked it with GSL v2.7 and NLopt v2.6.2. All computations are carried out in double-precision arithmetic.

The accuracy and qualitative analyses are performed using synthetic and real datasets, i.e., the precipitation dataset of Germany. We use the daily average precipitation of Germany in January of the year 2021. The data is collected by Kaspar et al. [48] and covers the whole of Germany with a spatial resolution of 1 km $\times$ 1 km. The daily average precipitation is given in mm. Because of this high spatial resolution, the total number of locations is $n = 358,303$. To make the data stationary, we remove the mean of the daily average precipitation of January over the year 2000 to 2020 from the data. So, the data can be interpreted as the excess daily average precipitation for January of 2021. The spatial image of the data is given in Fig. 7.

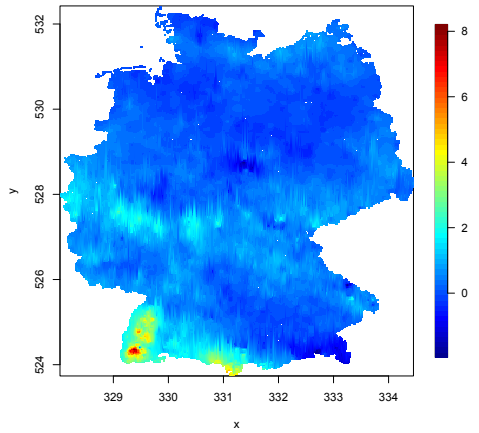


Fig. 7: Spatial image of the excess daily average precipitation (in mm) of January, 2021 on $358,303$ locations over Germany.

B. TLR/HODLR Accuracy Estimation Assessment

In this section, we demonstrate the accuracy of the MLE obtained using synthetic datasets. We generate $20,164$ random observations from a TGH random field defined in (2) on the $[0,1] \times [0,1]$ square in the following four setups, with latent correlation function as in (5):

(a) $\xi = 0$, $\omega = 2$, $g = 0.2$, $h = 0.2$, $\nu = 0.5$, and $\phi = 0.42$, with effective range 0.3;
(b) $\xi = 0$, $\omega = 2$, $g = 0.2$, $h = 0.2$, $\nu = 0.5$, and $\phi = 0.7$, with effective range 0.5;
(c) $\xi = 0$, $\omega = 2$, $g = 0.5$, $h = 0.3$, $\nu = 0.5$, and $\phi = 0.96$, with effective range 0.5;
(d) $\xi = 0$, $\omega = 2$, $g = 0.2$, $h = 0.2$, $\nu = 0.5$, and $\phi = 0.98$, with effective range 0.7.

The effective range is the distance at which the correlation between two points becomes less than 0.05. Given the other parameters, we find ϕ for different setups by changing the effective range. We use the correlation function of the TGH random field, given in [10], for evaluating ϕ in different setups. For each of these four setups, we estimate the parameters by maximizing the likelihood function given in (3). We compute the log-likelihood with TLR and HODLR with different accuracies and maximize them to obtain the MLE. We repeat the procedure 100 times for both TLR and HODLR and their different accuracies. Moreover, we do the same with the exact log-likelihood. We present the boxplots of the estimated parameter values for different setups and different computation techniques in Fig. 8. We used three accuracy levels for both HODLR (i.e., HODLR-5, HODLR-7, and HODLR-9) and TLR (i.e., TLR-5, TLR-7, and TLR-9) compared to the exact estimation (i.e., Exact). The boxplots suggest that the estimation accuracy does not vary much with the different parameter setups. It is also worth noting that the estimation accuracy increases if we increase the accuracy of TLR or HODLR. Our result also shows that TLR works better for TGH model estimation than HODLR over different accuracies. The difference between TLR, HODLR, and exact is negligible for higher accuracy.

C. Evaluation of Prediction with TGH Random Fields

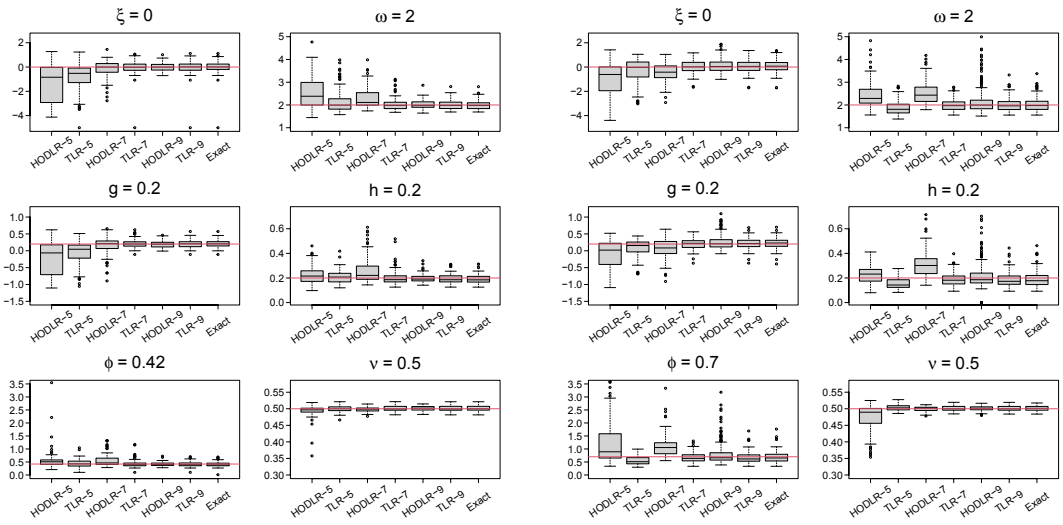
We check the validity of our PIT assessment tool of the TGH model. In this simulation study, we generate observations with a parameter of setup (a) from the previous section and fit both Gaussian and TGH models. We create the PIT histogram using the method discussed in Section V-C. Fig. 9 suggests that the PIT histogram obtained by fitting the Gaussian model is not at all a representation of a uniform distribution. To create a single measure of model adequacy, we introduce the mean divergence distance (MDD) from the PIT histograms. The MDD is defined as the average squared divergence of the length of the histogram bars from the value 1 (uniform density): the smaller the MDD, the better fitted the model. The MDD of both models is reported in Fig. 9. As expected, the TGH model performs better than the Gaussian model for this synthetic data.

D. Precipitation Data Assessment

We fit both Gaussian and TGH models to a subsample of size $300K$ locations. The estimated parameter values are given in Table I. The estimates of the parameters g and h from the TGH model indicate that the data is far from a Gaussian random field. This claim has been justified by our PIT assessment tool. We predict the values for $30K$ locations based on these estimates, separate from the training subsample. We produce the PIT histogram for both the fitted model based on the testing data. The PIT histograms are given in Fig. 10. We can see from Fig. 10 that the PIT histogram obtained from the Gaussian model does not correspond to a uniform distribution. The PIT from the TGH model shows a uniform distribution where the Gaussian model MDD is 0.5434 while TGH model MDD is 0.0290. This observation leads us to conclude that the fitted TGH model is appropriate to explain the variability of the data.

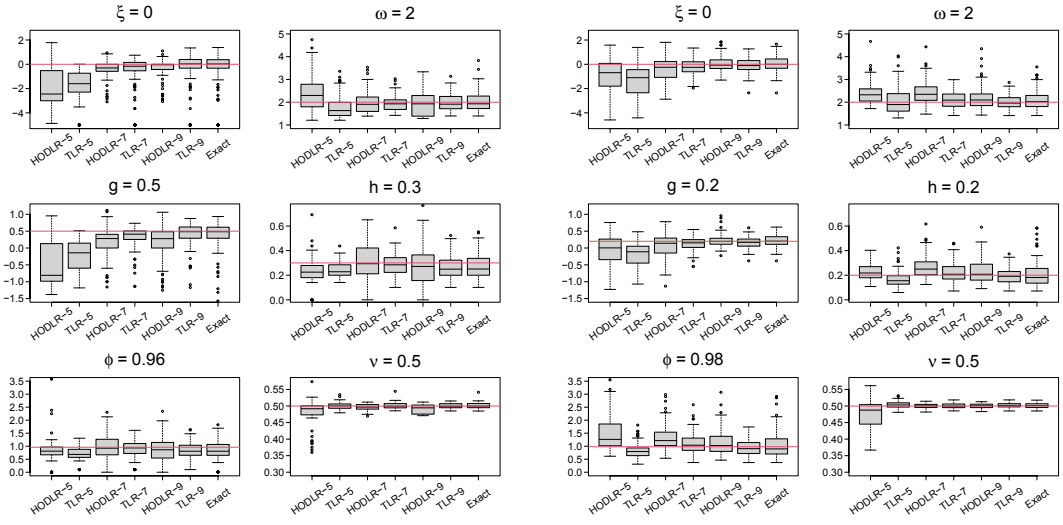
E. TLR/HODLR Matrices Performance Assessment

We provide the TGH modeling and prediction operations in low-rank structures using TLR and HODLR approximations.



(a) (0, 2, 0.2, 0.2, 0.5, 0.42)

(b) (0, 2, 0.2, 0.2, 0.5, 0.7)



(c) (0, 2, 0.5, 0.3, 0.5, 0.96)

(d) (0, 2, 0.2, 0.2, 0.5, 0.98)

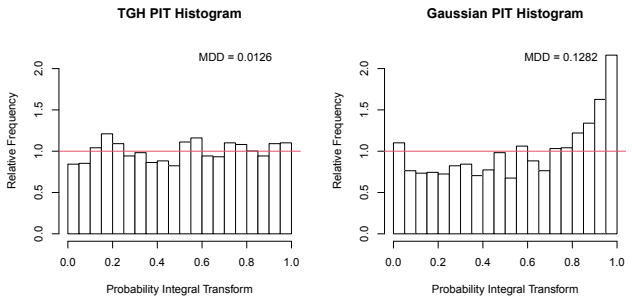
Fig. 8: Boxplots of parameter estimates (ξ, ω, g, h, ϕ, ν) under the HODLR (H), TLR (T), and exact (E) log-likelihood computation. The true parameters are indicated by the red lines.

TABLE I: Parameter estimates for Gaussian and TGH models with the real dataset.

Model	ξ	ω	g	h	ϕ	ν
Gaussian	0.26	1.04	-	-	10.36	0.57
TGH	-0.37	0.85	-1.69	0.65	10.78	0.64

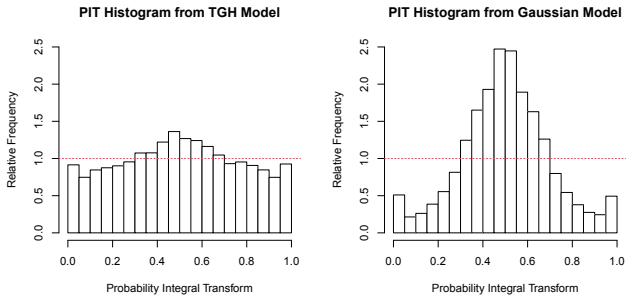
The baseline TLR software is the HiCMA library, which runs on both shared-memory and distributed memory architectures, while the HODLR underlying software is HODLRLIB, which only has a shared-memory implementation through OpenMP multithreaded BLAS. We compare the TGH model using TLR against HODLR approximations on two different shared-memory architectures from two vendors, i.e., Intel and AMD. We assess the performance of both implementations of the TGH likelihood function.

Fig. 11 shows the breakdown of execution times to calculate a single TGH likelihood function using both TLR and HODLR matrices. We report the result of three accuracy levels for each approximation method using five dataset sizes. The execution time is broken down into matrix assembly time (i.e., the time taken to compute the covariance matrix in low-rank structure), matrix factorization time (POTRF), and others (i.e., matrix determinant computing, measurements vector transformation, matrix-vector triangular solve (TRSV)). The figure highlights that the HiCMA-based implementation of the TLR approximation is faster than the HODLRLIB implementation of the HODLR approximation. Moreover, it can be seen that with different accuracy levels, the total execution time difference



(a) TGH. (b) Gaussian.

Fig. 9: PIT histograms for predictive distribution with the same TGH synthetic dataset.



(a) TGH model. (b) Gaussian Model.

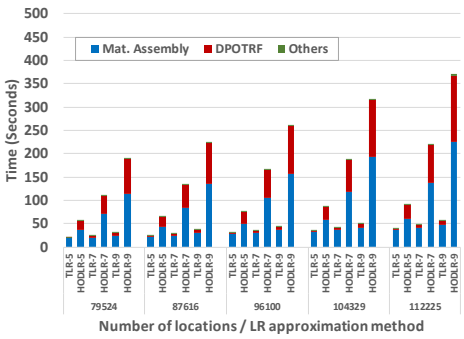
Fig. 10: PIT histograms for both TGH and Gaussian model with the real dataset.

increases between TLR and HODLR. For instance, on Intel IceLake, TLR outperforms HODLR by $2.29X$, $4.66X$, and $6.46X$ with accuracy levels 10^{-5}, 10^{-7}, and 10^{-9}, respectively. We also observed that the number of cores strongly influences the speedup; for instance, on AMD Milan, TLR outperforms HODLR by $6.82X$ with an accuracy level of 10^{-9} since HODLRLIB cannot as efficiently utilize the system resources. This performance difference between HiCMA and HODLRLIB comes mostly from the programming models: task-based with asynchronous executions (i.e., HiCMA) versus bulk-synchronous fork-join paradigm (i.e., HODLRLIB). The figure also shows that assembling the covariance matrix with TLR format accounts for most of the likelihood function estimation. The matrix assembly takes up to 70% and 85% of the time on Intel IceLake and AMD Milan, respectively. In contrast, due to a slower HODLRLIB, the matrix assembly takes about 65% of the total execution time on both machines.

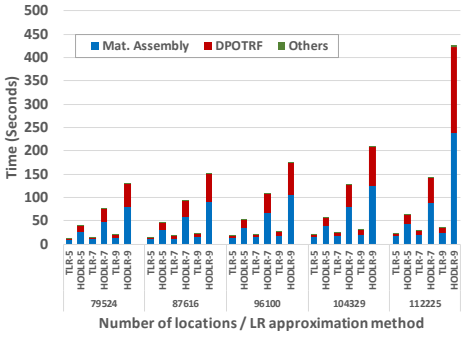
F. TLR Matrices Performance Assessment

In this section, we compare the performance of TLR against exact when computing the non-Gaussian modeling and the prediction operations on shared and distributed-memory architectures.

Performance on Shared-Memory Systems. Fig. 12 shows the performance of the TLR approximation compared to the

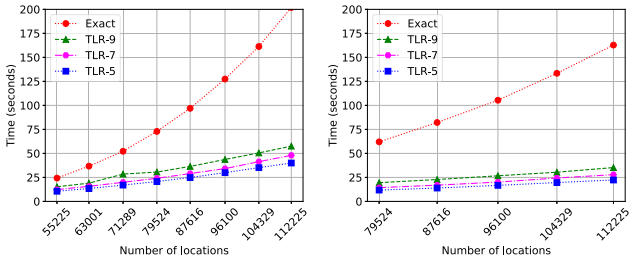


(a) 56-core Intel IceLake.



(b) 128-core AMD Milan.

Fig. 11: Time breakdown of a single TGH likelihood function estimation on shared-memory architectures using TLR / HODLR matrices.

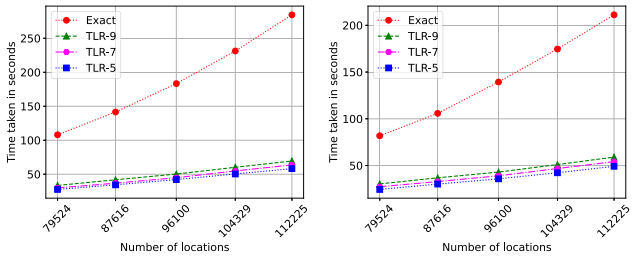


(a) 56-core Intel IceLake. (b) 128-core AMD Milan.

Fig. 12: Performance of a single non-Gaussian MLE iteration on shared-memory architectures using exact / TLR matrices.

exact implementation of the TGH likelihood function on our target two shared-memory systems and using different data sizes. With different accuracy, TLR outperforms the exact version of the likelihood estimation with a different number of locations. TLR shows better performance than the exact implementation, reaching up to $7.29X$ and $5.74X$ on Intel IceLake and AMD Milan, respectively. Furthermore, Figure 13 shows the performance of the prediction operation using exact and TLR. The figure shows the TLR approximation of accuracy 10^{-9} outperforms the exact computation with up to $4.29X$ and $4.88X$ on the given machines.

Performance on Distributed-Memory Systems. We assess the exact and TLR-based approximation performance on the



(a) 56-core Intel IceLake. (b) 128-core AMD Milan.

Fig. 13: Performance of a single TGH prediction operation on shared-memory architectures using exact / TLR matrices.

Shaheen-II Cray XC40 system. Fig. 14 shows the scalability of the exact non-Gaussian MLE on 64, 128, 256, and 512 nodes. The figure shows that the exact MLE implementation scales very well with the same matrix size and a different number of nodes. For instance, with $562,500$ spatial locations, the total execution time for single MLE iteration is 629.44, 380.78, 255.97 seconds using 128, 256, and 512 nodes. We also assess the performance of the TLR-based implementation using 512 nodes on Shaheen-II compared to the exact computation. Fig. 15 shows the execution time of the exact and TLR approximation for MLE with different accuracy levels. As shown, the TLR approximation shows better performance compared to exact computation with a different number of locations up to 800K locations using 512 nodes. The figure shows that the TLR-approximation outperforms the exact computation with up to $2.96X$. We believe the performance of TLR MLE can be further improved by using a runtime system that provides rank-aware data distribution to mitigate the load imbalance [49].

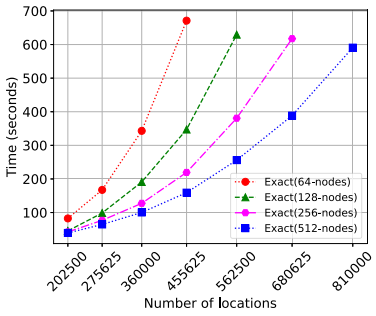


Fig. 14: Scalability of a single non-Gaussian MLE iteration on Shaheen-II with up to 512 nodes using exact matrices.

VI. Conclusion

This paper introduces parallel non-Gaussian modeling and prediction implementation based on the Tukey g-and-h (TGH) random fields in the context of climate/weather applications. We propose a task-based exact computation for both operations with the aid of existing dense linear algebra libraries to tackle the underlying matrix operation with large problem sizes. We also assess low-rank approximations using tile low-rank (TLR) and hierarchically off-diagonal low-rank (HODLR). We rely on HiCMA and HODLRLIB to perform the required matrix

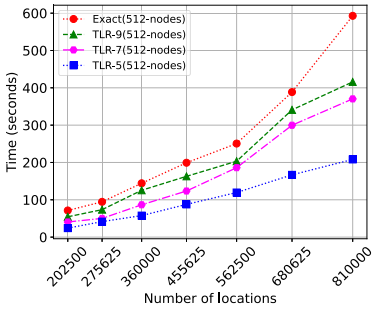


Fig. 15: Performance of a single non-Gaussian MLE iteration on Shaheen-II nodes using exact / TLR matrices.

operations for MLE. The TLR approximation was performed for modeling, and the prediction operations outperformed its HODLR counterpart. Using up to $800K$ geospatial locations, it also shows a speedup up to $7.29X$ and $2.96X$ compared to the exact implementation on shared and distributed-memory systems, respectively. For future work, we would like to consider mixed-precision arithmetics with GPU hardware accelerators to boost the performance of dense linear algebra kernels. We would also like to investigate randomized algorithms to directly generate the compressed matrix and improve the matrix assembly phase.

Acknowledgement

This work is funded and supported by King Abdullah University of Science and Technology (KAUST) through the Office of Sponsored Research (OSR). We would also like to acknowledge Intel for support in the form of an Intel Parallel Computing Center award Cray for the support provided during the Center of Excellence award to the Extreme Computing Research Center at KAUST. This research used the resources of the Extreme Computing Research Center (ECRC) and the KAUST Supercomputing Laboratory, including Cray XC40, Shaheen II supercomputer.

References

[1] N. Cressie and G. Johannesson, "Fixed Rank Kriging for Very Large Spatial Data Sets," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 1, pp. 209–226, 2008.

[2] M. L. Stein, J. Chen, and M. Anitescu, "Stochastic Approximation of Score Functions for Gaussian Processes," *The Annals of Applied Statistics*, pp. 1162–1191, 2013.

[3] S. Abdulah, H. Ltaief, Y. Sun, M. G. Genton, and D. E. Keyes, "Parallel Approximation of the Maximum Likelihood Estimation for the Prediction of Large-Scale Geostatistics Simulations," in *IEEE International Conference on Cluster Computing (CLUSTER)*, 2018, pp. 98–108.

[4] M. L. O. Salvana, S. Abdulah, H. Huang, H. Ltaief, Y. Sun, M. G. Genton, and D. E. Keyes, "High Performance Multivariate Geospatial Statistics on Manycore Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 11, pp. 2719–2733, 2021.

[5] V. J. Berrocal, A. E. Gelfand, and D. M. Holland, "A Bivariate Space-Time Downscaler under Space and Time Misalignment," *The Annals of Applied Statistics*, vol. 4, no. 4, pp. 1942–1975, 2010.

[6] L. Yao, Y. Guan, and A. Wang, "Fault Diagnosis and Minimum Entropy Fault Tolerant Control for Non-Gaussian Singular Stochastic Distribution Systems using Square-Root Approximation," *International Journal of Modelling, Identification and Control*, vol. 24, no. 3, pp. 206–215, 2015.

[7] Y. Yan and M. G. Genton, "Non-Gaussian Autoregressive Processes with Tukey g-and-h Transformations," *Environmetrics*, vol. 30, no. 2, 2019.

[8] V. De Oliveira, B. Kedem, and D. A. Short, "Bayesian Prediction of Transformed Gaussian Random Fields," *Journal of the American Statistical Association*, vol. 92, no. 440, pp. 1422–1433, 1997.

[9] G. Rios and F. Tobar, "Learning Non-Gaussian Time Series using the Box-Cox Gaussian Process," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[10] G. Xu and M. G. Genton, "Tukey g-and-h Random Fields," *Journal of the American Statistical Association*, vol. 112, no. 519, pp. 1236–1249, 2017.

[11] E. Agullo, O. Aumage, M. Faverge, N. Furmento, F. Pruvost, M. Sergent, and S. P. Thibault, "Achieving High Performance on Supercomputers with a Sequential Task-based Programming Model," *IEEE Transactions on Parallel and Distributed Systems*, 2017.

[12] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier, "StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 2, pp. 187–198, 2011.

[13] 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*, 2022.

[14] "HODLRLIB: A Library Consisting of Fast Matrix Operations for Matrices based on the Hierarchical Off-Diagonal Low-Rank (HODLR) Structure," Feb. 2022, available at https://github.com/sivaramambikasaran/hodlr.

[15] S. Ambikasaran and E. Darve, "An $O(N \log N)$ Fast Direct Solver for Partial Hierarchically Semi-Separable Matrices," *Journal of Scientific Computing*, vol. 57, no. 3, pp. 477–501, 2013.

[16] H.-M. Kim and B. K. Mallick, "A Bayesian Prediction using the Skew Gaussian Distribution," *Journal of Statistical Planning and Inference*, vol. 120, no. 1-2, pp. 85–101, 2004.

[17] H. Zhang and A. El-Shaarawi, "On Spatial Skew-Gaussian Processes and Applications," *Environmetrics: The official journal of the International Environmetrics Society*, vol. 21, no. 1, pp. 33–47, 2010.

[18] M. G. Genton and H. Zhang, "Identifiability Problems in some Non-Gaussian Spatial Random Fields," *Chilean Journal of Statistics*, vol. 3, no. 2, pp. 171–179, 2012.

[19] K. Rimstad and H. Omre, "Skew-Gaussian Random Fields," *Spatial Statistics*, vol. 10, pp. 43–62, 2014.

[20] J. Røislien and H. Omre, "T-Distributed Random Fields: A Parametric Model for Heavy-Tailed Well-log Data," *Mathematical Geology*, vol. 38, no. 7, pp. 821–849, 2006.

[21] M. Bevilacqua, C. Caamaño-Carrillo, R. B. Arellano-Valle, and V. Morales-Oñate, "Non-Gaussian Geostatistical Modeling using (Skew) t Processes," *Scandinavian Journal of Statistics*, vol. 48, no. 1, pp. 212–245, 2021.

[22] Y. V. Marchenko and M. G. Genton, "Multivariate Log-Skew-Elliptical Distributions with Applications to Precipitation Data," *Environmetrics: The official journal of the International Environmetrics Society*, vol. 21, no. 3-4, pp. 318–340, 2010.

[23] M. B. Palacios and M. F. J. Steel, "Non-Gaussian Bayesian Geostatistical Modeling," *Journal of the American Statistical Association*, vol. 101, no. 474, pp. 604–618, 2006.

[24] T. C. Fonseca and M. F. Steel, "Non-Gaussian Spatiotemporal Modelling through Scale Mixing," *Biometrika*, vol. 98, no. 4, pp. 761–774, 2011.

[25] B. Gräler, "Modelling Skewed Spatial Random Fields through the Spatial Vine Copula," *Spatial Statistics*, vol. 10, pp. 87–102, 2014.

[26] P. Krupskii, R. Huser, and M. G. Genton, "Factor Copula Models for Replicated Spatial Data," *Journal of the American Statistical Association*, vol. 113, no. 521, pp. 467–479, 2018.

[27] J. Wallin and D. Bolin, "Geostatistical Modelling using Non-Gaussian Matérn Fields," *Scandinavian Journal of Statistics*, vol. 42, no. 3, pp. 872–890, 2015.

[28] V. De Oliveira, "On Optimal Point and Block Prediction in log-Gaussian Random Fields," *Scandinavian Journal of Statistics*, vol. 33, no. 3, pp. 523–540, 2006.

[29] C. J. Johns, D. Nychka, T. G. F. Kittel, and C. Daly, "Infilling Sparse Records of Spatial Fields," *Journal of the American Statistical Association*, vol. 98, no. 464, pp. 796–806, 2003.

[30] V. De Oliveira, B. Kedem, and D. A. Short, "Bayesian Prediction of Transformed Gaussian Random Fields," *Journal of the American Statistical Association*, vol. 92, no. 440, pp. 1422–1433, 1997.

[31] D. J. Allcroft and C. A. Glasbey, "A Latent Gaussian Markov Random-Field Model for Spatiotemporal Rainfall Disaggregation," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 52, no. 4, pp. 487–498, 2003.

[32] T. Goulart, L. G. Tavaresa, M. Loboscoa, R. W. dos Santosa, and F. de Oliveira Chavesb, "A Parallel Implementation of the Ordinary Kriging Algorithm for Heterogeneous Computing Environments," *Journal of Computational Interdisciplinary Sciences*, vol. 8, no. 3, pp. 143–152, 2017.

[33] T. Cheng, "Accelerating Universal Kriging Interpolation Algorithm using CUDA-enabled GPU," *Computers and Geosciences*, vol. 54, pp. 178–183, 2013.

[34] P. Tahmasebi, M. Sahimi, G. Mariethoz, and A. Hezarkhani, "Accelerating Geostatistical Simulations using Graphics Processing Units (GPU)," *Computers and Geosciences*, vol. 46, pp. 51–59, 2012.

[35] 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.

[36] R. Furrer, M. G. Genton, and D. Nychka, "Covariance Tapering for Interpolation of Large Spatial Datasets," *Journal of Computational and Graphical Statistics*, vol. 15, no. 3, pp. 502–523, 2006.

[37] H. Sang and J. Z. Huang, "A Full Scale Approximation of Covariance Functions for Large Spatial Data Sets," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 74, no. 1, pp. 111–132, 2012.

[38] C. J. Geoga, M. Anitescu, and M. L. Stein, "Scalable Gaussian Process Computations using Hierarchical Matrices," *Journal of Computational and Graphical Statistics*, vol. 29, no. 2, pp. 227–237, 2020.

[39] J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, P. Wu, I. Yamazaki, A. YarKhan, M. Abalenkovs, N. Bagherpour *et al.*, "PLASMA: Parallel Linear Algebra Software for Multicore Using OpenMP," *ACM Transactions on Mathematical Software (TOMS)*, vol. 45, no. 2, pp. 1–35, 2019.

[40] "The Chameleon Project," Feb. 2022, https://project.inria.fr/chameleon/.

[41] A. Duran, E. Ayguadé, R. M. Badia, J. Labarta, L. Martinell, X. Martorell, and J. Planas, "OmpSs: A Proposal for Programming Heterogeneous Multi-Core Architectures," *Parallel Processing Letters*, vol. 21, no. 02, pp. 173–193, 2011.

[42] R. Chandra, L. Dagum, D. Kohr, R. Menon, D. Maydan, and J. McDonald, *Parallel Programming in OpenMP*. Morgan kaufmann, 2001.

[43] G. Bosilca, A. Bouteiller, A. Danalis, T. Herault, P. Lemarinier, and J. Dongarra, "DAGuE: A Generic Distributed DAG Engine for High Performance Computing," *Parallel Computing*, vol. 38, no. 1-2, pp. 37–51, 2012.

[44] G. Tzanos, V. Soni, C. Prouveur, M. Haefele, S. Zouzoula, L. Papadopoulos, S. Thibault, N. Vandenbergen, D. Pleiter, and D. Soudris, "Applying StarPU Runtime System to Scientific Applications: Experiences and Lessons Learned," in *POMCO 2020-2nd International Workshop on Parallel Optimization using/for Multi-and Many-core High Performance Computing*, 2020.

[45] H. Guo, J. Hu, H. Shao, and Z. Nie, "Hierarchical Matrices Method and its Application in Electromagnetic Integral Equations," *International Journal of Antennas and Propagation*, vol. 2012, 2012.

[46] K. Akbudak, H. Ltaief, A. Mikhalev, and D. Keyes, "Tile Low Rank Cholesky Factorization for Climate/Weather Modeling Applications on Manycore Architectures," in *International Supercomputing Conference*. Springer, 2017, pp. 22–40.

[47] P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker, "Improving Multifrontal Methods by Means of Block Low-Rank Representations," *SIAM Journal on Scientific Computing*, 2015.

[48] F. Kaspar, G. Müller-Westermeier, E. Penda, H. Mächel, K. Zimmermann, A. Kaiser-Weiss, and T. Deutschländer, "Monitoring of Climate Change in Germany–data, products and services of Germany's National Climate Data Centre," *Advances in Science and Research*, vol. 10, no. 1, pp. 99–106, 2013.

[49] S. Abdulah, Q. Cao, Y. Pei, G. Bosilca, J. Dongarra, M. G. Genton, D. Keyes, H. Ltaief, and Y. Sun, "Accelerating Geostatistical Modeling and Prediction With Mixed-Precision Computations: A High-Productivity Approach With PaRSEC," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 964–976, 2022.